Київський національний університет технологій та дизайну

Кафедра механічної інженерії

Кафедра комп'ютерних наук

МАТЕМАТИЧНИЙ ТА КОМП'ЮТЕРНИЙ АНАЛІЗ СИСТЕМ І ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

Том 2

Щербань В.Ю.
Колиско О.З.
Щербань Ю.Ю.
Воляник О.Ю.
Чупринка Н.В.
Мельник Г.В.
Гольдберг М.І.
Кириченко А.М.
Калашник В.Ю.

МАТЕМАТИЧНИЙ ТА КОМП'ЮТЕРНИЙ АНАЛІЗ СИСТЕМ І ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

Том 2

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА МЕХАНІЧНОЇ ІНЖЕНЕРІЇ КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЩЕРБАНЬ В.Ю., КОЛИСКО О.З., ЩЕРБАНЬ Ю.Ю., ВОЛЯНИК О.Ю., ЧУПРИНКА Н.В., МЕЛЬНИК Г.В., ГОЛЬДБЕРГ М.І., КИРИЧЕНКО А.М., КАЛАШНИК В.Ю.

МАТЕМАТИЧНИЙ ТА КОМП'ЮТЕРНИЙ АНАЛІЗ СИСТЕМ І ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

Том 2

Математичне та програмне забезпечення для аналізу механічних систем та прикладних питань математичних моделей

Монографія

Київ – 2024

УДК 677.024.3
ББК 65.9(4Укр)306.4-6
Щ 610

Призначена для широкого кола викладачів, науковців, аспірантів, магістрів та студентів профільних вищих навчальних закладів, інженерно-технічних працівників швейної та текстильної промисловості.

Колектив авторів:

ЩЕРБАНЬ В. Ю. – лауреат Державної премії України в галузі науки і техніки, академік Міжнародної академії комп'ютерних наук та систем, доктор технічних наук, професор кафедри механічної інженерії Київського національного університету технологій та дизайну;

КОЛИСКО О.З. - кандидат технічних наук, доцент кафедри комп'ютерних наук Київського національного університету технологій та дизайну;

ЩЕРБАНЬ Ю.Ю. – лауреат Державної премії України в галузі науки і техніки, доктор технічних наук, професор, заступник директора Київського фахового коледжу прикладних наук;

ВОЛЯНИК О.Ю. - кандидат технічних наук, доцент, завідувач кафедри механічної інженерії Київського національного університету технологій та дизайну;

ЧУПРИНКА Н.В. - кандидат технічних наук, доцент, завідувачка кафедри комп'ютерних наук Київського національного університету технологій та дизайну;

МЕЛЬНИК Г.В. - кандидат технічних наук, доцент кафедри комп'ютерних наук Київського національного університету технологій та дизайну;

ГОЛЬДБЕРГ М.І. - кандидат технічних наук, доцент кафедри комп'ютерних наук Київського національного університету технологій та дизайну;

КИРИЧЕНКО А.М. - доктор філософії, асистент кафедри комп'ютерних наук Київського національного університету технологій та дизайну;

КАЛАШНИК В.Ю. - кандидат технічних наук, асистент кафедри комп'ютерних наук Київського національного університету технологій та дизайну.

Рецензенти:

ОПАНАСЕНКО В.М. – лауреат Державної премії України в галузі науки і техніки,, д.т.н., професор, провідний науковий співробітник, Інститут кібернетики НАН України;

ЗАЙЦЕВ С.В. – д.т.н., професор, кафедра інформаційних та комп'ютерних систем, Чернігівський національний технологічний університет;

ЩУЦЬКА Г. В. – д.т.н., директор коледжу, Київський фаховий коледж прикладних наук.

Щ 610 Щербань В.Ю. Математичний та комп'ютерний аналіз систем і технологічних процесів. Т2: Математичне та програмне забезпечення для аналізу механічних систем та прикладних питань математичних моделей : монографія : в 2 т. / В.Ю. Щербань, О.З. Колиско, Ю.Ю. Щербань, О.ЮВоляник, Н.В. Чупринка, Г.В. Мельник, М.І. Гольдберг, А.М. Кириченко, В.Ю. Калашник. – К.: ТОВ Фастбінд Україна 2024. – 702 с.

ISBN 978-617-8237-32-5

Наведені результати досліджень з питань розробки математичних, алгоритмічних, та комп'ютерних компонентів систем автоматизованого проектування, спрямованих на розвиток наукових засад та методології створення нових, ефективних технологій та обладнання для виготовлення широкого асортименту виробів як побутового, так і спеціального призначення. Отримані нові алгоритми та програмне забезпечення для удосконалення технологічних процесів та обладнання текстильної, трикотажної та взуттєвої галузі.

ISBN 978-617-8237-32-5

УДК 677.024.3
ББК 65.9(4Укр)306.4-6

ЗМІСТ

1. УЗАГАЛЬНЕНА ТЕОРІЯ МЕХАНІКИ НИТОК ЗДАТНИХ ДО ДЕФОРМАЦІЇ В ПЕРЕТИНІ …….....4
1.1. ОСНОВНІ ДОПУЩЕННЯ, ПРИЙНЯТІ ПРИ ПОБУДОВІ НИТОК-МОДЕЛЕЙ І ЇХ КЛАСИФІКАЦІЯ..... 5
1.2. ВИЗНАЧЕННЯ ГЕОМЕТРИЧНИХ ХАРАКТЕРИСТИК ОСІ ДЕФОРМОВАНИХ НИТОК..... 20
1.3 ВИЗНАЧЕННЯ ШВИДКОСТЕЙ І ПРИСКОРЕНЬ ТОЧОК ОСІ ДЕФОРМОВАНОЇ НИТКИ 33
2. СТРУКТУРА КОМП'ЮТЕРНОЇ ПРОГРАМИ КІНЕМАТИЧНОГО ТА КІНЕТОСТАТИЧНОГО АНАЛІЗУ ПЛОСКИХ МЕХАНІЗМІВ.....75
ЛІСТИНГ ПРОГРАМ ДЛЯ СТРУКТУРНОГО СИНТЕЗУ СИСТЕМИ ПОДАЧІ НИТКИ..... 149
ЛІСТИНГИ ПРОГРАМ РЕАЛІЗАЦІЇ АЛГОРИТМУ ДЕЙКСТРИ ПРИ ВИЗНАЧЕННІ ФОРМИ ЗАПРАВКИ НИТКИ ……..…215
ЛІСТИНГИ ПРОГРАМ ДЛЯ ВИЗНАЧЕННЯ НАПРУЖЕНОСТІ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ …… 285
ЛІСТИНГ ПРОГРАМ ДЛЯ КІНЕМАТИЧНОГО ТА КІНЕТОСТАТИЧНОГО АНАЛІЗУ МЕХАНІЗМІВ ……..337
ЛІСТИНГ ПРОГРАМИ ДЛЯ КІНЕМАТИЧНОГО АНАЛІЗУ І СИНТЕЗУ МЕХАНІЗМІВ І МАШИН ЛЕГКОЇ ПРОМИСЛОВОСТІ ….422
ЛІСТИНГ ПРОГРАМ ПРИКЛАДНИХ ПРОГРАМ …….485

1. УЗАГАЛЬНЕНА ТЕОРІЯ МЕХАНІКИ НИТОК ЗДАТНИХ ДО ДЕФОРМАЦІЇ В ПЕРЕТИНІ

Удосконалення багатьох технологічних процесів текстильної і легкої промисловості повинне базуватися на теоретичному і експериментальному дослідженні взаємодії ниток з робочими органами технологічного устаткування. Під робочими органами тут матимемо на увазі не тільки голки, платини, спрямовувачі ниток гребінок і ін. - на в'язальних машинах і отвори ремізних рамок, зуби берда - на ткацьких верстатах, але і різного роду спрямовувачів нитки, пристрої для натягу ниток, компенсатори натягу.

Процес формування елементу тканини і трикотажу не закінчується на останній приєднаній уточині і в’язаній петлі, він продовжується і при відході від зони формування тканини або від відбійної площини для трикотажу. Це пояснюється взаємодією ниток між собою, при цьому стабілізація положення ниток в тканині і трикотажі наступить тоді, коли виконуватиметься умова рівноваги між окремими силовими елементами, складових тканини і трикотажу на даній ділянці. З цього виходить, що вивчення взаємодії ниток в робочій зоні доповнюватиме комплекс досліджень процесу переробки ниток на технологічному устаткуванні текстильної і легкої промисловості.

Проведення теоретичних досліджень повинне сприяти широкому освітленню процесів взаємодії ниток з направляючими поверхнями великої кривизни з урахуванням зминання як самих ниток, так і що направляють, що дозволить оптимізувати ці процеси з погляду зниження обривності. Для цього на попередньому етапі необхідно провести докладну класифікацію ниток, визначити необхідні допущення для побудови ниток-моделей [1, 14-31], які використовуються при теоретичному дослідженні різних технологічних процесів легкої і текстильної промисловості.

1.1. Основні допущення, прийняті при побудові ниток-моделей і їх класифікація

При теоретичному дослідженні взаємодії реальних текстильних ниток з направляючими неможливо обійтися без певних допущень, ухвалення яких дозволяє отримати рішення задачі з необхідним ступенем точності. Отже, вибір моделі нитки з урахуванням прийнятих допущень ще на початковій стадії визначає величину помилки, що отримується після проведення теоретичних досліджень, що дозволить використовувати результати, залежно від їх значення, як для якісного, так і для кількісного аналізу процесу, що вивчається.

Побудова нитки-моделі повинна базуватися на докладній класифікації всього різноманіття існуючих ниток [1,14-31].

Найбільш докладна класифікація за фізико - механічними показниками, до теперішнього часу, була запропонована проф. Мігушовим І.І. [1], яка є логічним продовженням досліджень проф. Мінакова А.П. Приведена на рис. 1.1 структурна класифікація ниток складена нами з урахуванням, як форми поперечного перетину реальних ниток, так і наявність двох модифікацій ниток-моделей. Дана схема дозволяє прослідкувати взаємозв'язок між реальними нитками і моделями, встановити порядок при виборі певної моделі нитки, виходячи з конкретних характеристик реального об'єкту.

Проводити класифікацію необхідно по структурі, формі поперечного перетину ниток і по фізико - механічних властивостях.

По структурі реальні нитки можна розділити (див. рис. 1.2) на комплексні нитки, пряжу, мононитки і вироби спеціального призначення [1, 10, 12]. Комплексні нитки утворюються шляхом скручування або склеювання окремих елементарних ниток і мають більш рівномірну

структуру, ніж пряжа, яка складається з окремих елементарних волокон, з'єднаних в процесі прядіння.

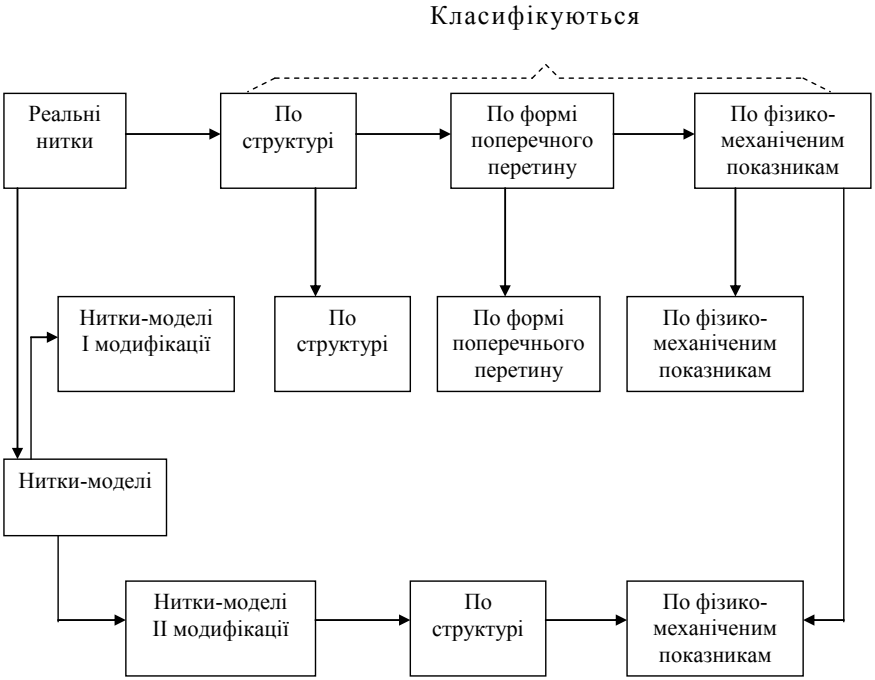


Рис. 1.1. Структурна класифікація ниток

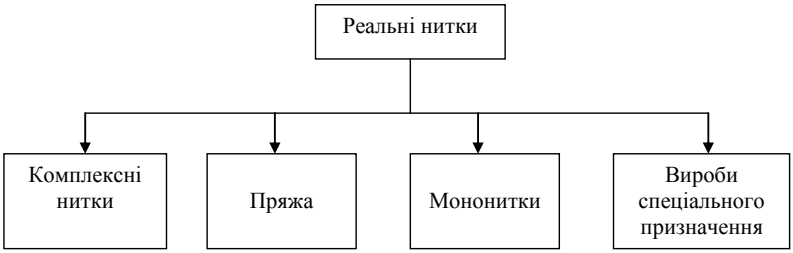


Рис. 1.2. Класифікація реальних ниток по структурі

Пряжа, отримана при кардній або гребінній системі прядіння, відрізняється більш орієнтованим розташуванням окремих волокон, чим пряжа, отримана за апаратним способом.

Мононитками є "... одиночна нитка, що не ділиться в подовжньому напрямі без руйнування..." [1]. До мононитоk відносяться різного роду волосіні, металеві нитки і тому подібне, які можна безпосередньо використовувати в трикотажних і ткацьких технологічних процесах.

До виробів спеціального призначення можна віднести комбіновані гнучкі елементи різних машин і механізмів, дроти, антени, тобто об'єкти, що складаються з поєднання різних матеріалів, що мають складну внутрішню структуру.

Для побудови математичних моделей необхідно від складних реальних об'єктів, шляхом ухвалення необхідних допущень, переходити до моделей, які для свого опису не вимагають застосування складного математичного апарату і, в той же час, дозволяють з достатнім ступенем точність описувати поведінку реальних ниток.

На рис. 1.3 показані різні нитки-моделі, які розділені на дві модифікації. До першої модифікації відносяться моделі, які найповніше описують поведінку реальних ниток за різних умов вантаження і по своїй структурі вельми наближені до останніх. Так комплексні нитки і пряжа (див. рис.1.3, а) представляються у вигляді окремих не скручених філаментів, розташованих один над одним. Як мононитки, так комплексні нитки і пряжа можуть бути представлені у вигляді складних геометрично правильних тіл. На рис. 1.3 б представлений один з випадків, коли реальна нитка представляється циліндровим тілом.

Необхідно відзначити, що моделі І-й модифікації набули найбільш широкого поширення при дослідженні переробки ниток. У тих випадках, коли реальні нитки піддаються дії у вузькому спектрі зміни

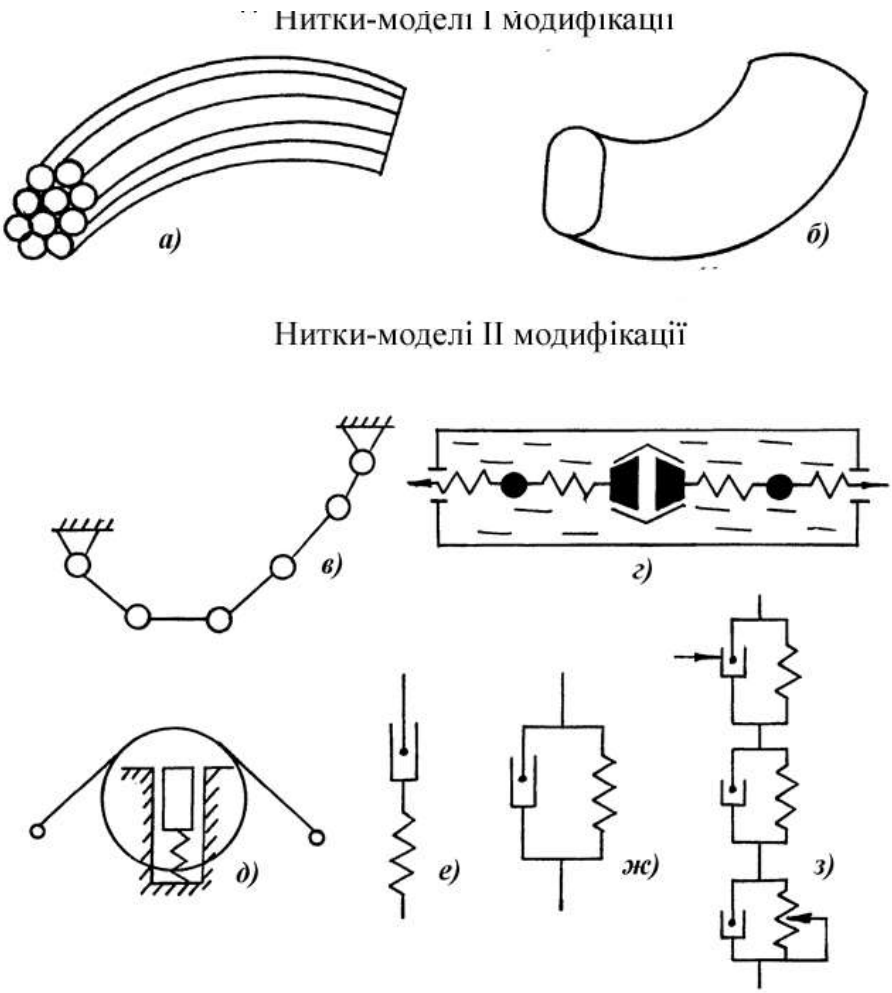


Рис.1.3. Приклади ниток-моделей різної структури

силового поля, характеру навантаження або необхідно досліджувати фізико - механічні показники реальних ниток, умови їх рівноваги тільки при розтягуванні, стисненні в подовжньому і поперечному напрямі, зминання в зоні контакту, застосовують моделі II-й модифікації.

На рис. 1.3 в, г, д, е, ж, з показані окремі з них. Дані моделі представляють сукупність окремих елементів, утворюючих механічні, гідро механічні і ін. системи різного ступеня складності залежно від того, які властивості реальних ниток вони описують.

На мал. 1.3,в показана модель нитки, що провисає в полі сили тяжіння і представляє сукупність сполучених між собою стрижнів. Для опису процесу м'ятої нитки, яка виконує функцію направляючої поверхні, використовується шків з пружиною та повзуном, розташованим в тих, що вертикальних направляють (див. рис. 1.3, д) [1].

Для опису процесу розтягування ниток запропонований ряд моделей, які необхідно вибирати залежно від умов проведення експерименту і кінцевої мети. Так, найширше застосовують моделі Максвела (рис. 1.3, е), Кельвіна-Фойгта (рис. 1.3, же), Кукіна-Соловьева (рис. 1.3, з). Вони утворені послідовним або паралельним з'єднанням пружних і демпфуючих елементів, характеристики яких відображають фізико - механічні властивості реальних ниток. Для опису процесу розтягування ниток використовують і складніші моделі (рис. 1.3 г). У їх конструкцію входять різні елементи: заклинюючи елементи, ланцюжки зосереджених мас, сполучення пружних елементів.

Цілком очевидно, що вибір тієї або іншої моделі нитки (див. рис. 1.1) залежить від структури реальної нитки, як показано на схемі. Критерієм вибору служить, в першу чергу, найбільша відповідність моделі реальному об'єкту, а також характер руху і силового вантаження нитки. Одним з важливих показників при класифікації ниток є форма поперечного перетину останньої.

Це обумовлює вибір форми поперечного перетину моделі нитки.

Реальні текстильні нитки (рис. 1.4 а - к) мають найрізноманітнішу форму поперечного перетину, яка утворюється під впливом зовнішніх силових чинників і взаємного тиску окремих філаментів [1]. Так нитки у вільному стані (що не контактують з направляючими поверхнями) мають різну форму. При невисокому скручуванні (див. рис. 1.4 а) вона невизначена. При середньому скручуванні (див. рис. 1.4, б) має еліптичну форму. При скручуванні близькому до критичного (див. рис. 1.4 в) - коло. Це пояснюється тим, що із збільшенням скручування відбувається зростання деформації окремих філаментів (особливо розташованих на периферії перетину нитки) і, як наслідок, зростає натягнення і питомий тиск. Структура нитки стає більш орієнтованою.

При взаємодії з направляючими поверхнями відбувається зминання нитки в зоні контакту, що приводить до зміни форми поперечника. У ниток невисокого скручування (див. рис. 1.4, г) форма поперечного перетину представляє сплюснутий еліпс. При збільшенні скручування, за рахунок тиску з боку опорної поверхні, форма близька до еліптичної з плоским майданчиком (див. рис. 1.4 д) [1,10], а при скручуванні, близькому до критичного, вона представляє коло з плоским майданчиком контакту (див. рис. 1.4 е).

Мононитки мають поперечний перетин, форма якого визначається формою отворів філь'єр, через яких продавлюється полімер, і іншими специфічними особливостями технології їх виготовлення. Їх форма різноманітна: кругла, бобовидна (див. рис.1.4 ж); гантелевидна (див. рис. 1.4 з); з радіальними порами (див. рис. 1.4 і); з внутрішньою порожниною (див. рис. 1.4 к).

Таке різноманіття форм потребує розробки цілого ряду форм поперечника ниток-моделей І-й модифікації. Цілком очевидно, що вибір тієї або іншої форми залежить від умов, в яких знаходиться нитка: у

вільному стані або контактує з направляючою поверхнею; під дією прикладеної системи сил чи ні.

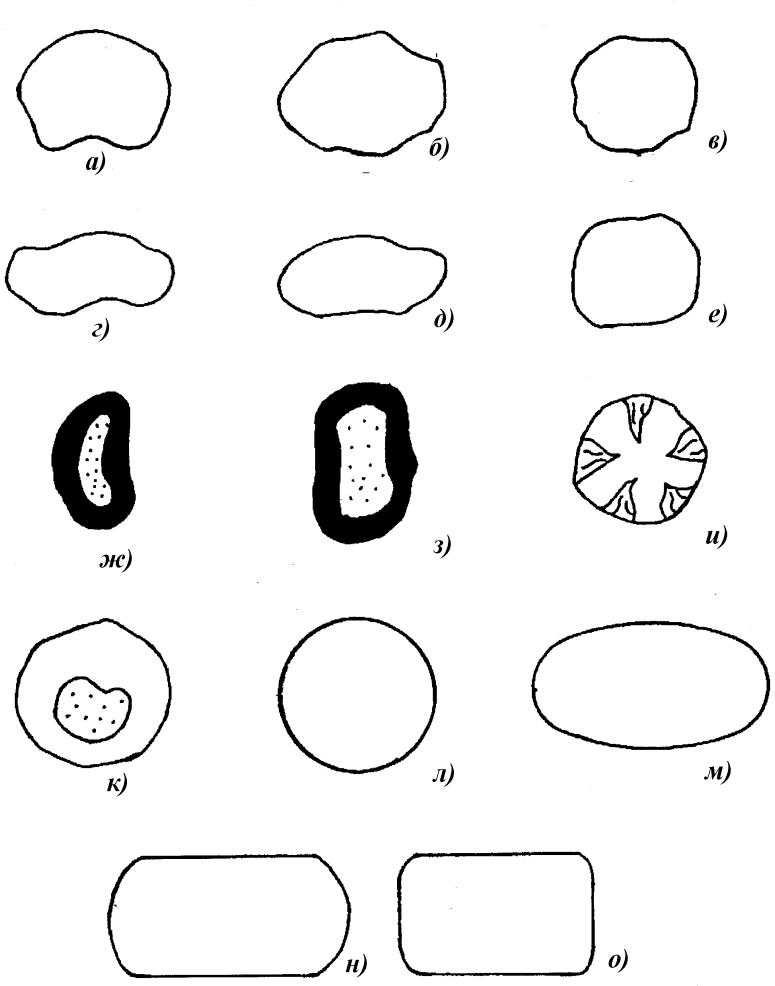


Рис.1.4. Форма поперечного перетину реальних ниток та моделей

До теперішнього часу найширше застосовуються наступні основні форми: кругла (рис. 1.4 л), еліптична (рис.1.4 м), стадіону подібна (мал. 1.4 н), прямокутна (флірет) (рис. 1.4 о).

Як випливає із структурної класифікації ниток (мал. 1.1) вибір форми поперечного перетину можна проводити тільки для моделей І-й модифікації, для моделей ІІ-й модифікації, що представляють сукупність окремих елементів, він не підходить, що і відбите на схемі.

Зважаючи на специфіку дослідження, що проводиться, можна вважати, що найбільш прийнятними за формою поперечного перетину будуть циліндрові, еліптичні і стадіону подібна моделі.

Перейдемо до класифікації ниток і моделей по фізико - механічних властивостях (див. рис. 1.1). У загальному випадку всі текстильні нитки, що переробляються на в'язальних машинах і ткацьких верстатах, під дією прикладених зовнішніх сил випробовують складне вантаження [1,2,10,12]. Як показали проведені дослідження, в реальних нитках, при їх вантаженні, в будь-якому довільному перетині виникає напруга: нормальні [1,10] (наявність яких пояснюється розтягуванням нитки або її окремих складових філаментів); дотичні (наявність яких пояснюється вигином і крученням нитки). При контакті з направляючою поверхнею реальні нитки (включаючи мононитки і металеві нитки) більшою чи меншою мірою піддаються зминанню [2,6], що приводить до виникнення додаткової напруги в поперечному напрямі, яка збільшується від периферії до зони контакту. Декілька слів необхідно сказати про анізотропію тертя текстильних матеріалів, особливо при їх взаємодії з направляючими поверхнями великої кривизни [1,10, 18-22]. Річ у тому, що при проведенні досліджень було виявлено відмінність в значеннях коефіцієнта тертя при подовжньому і поперечному русі нитки. Це важливу обставину необхідно враховувати при дослідженні технологічних процесів, при

експериментальному визначенні фрикційних властивостей реальних ниток і при побудові ниток-моделей.

Декілька слів необхідно сказати про нерівномірний розподіл маси в реальних текстильних нитках по їх довжині і про різні геометричні розміри поперечного перетину. Ці дві важливі характеристики можуть істотним чином зробити свій вплив на величину напруги при вантаженні нитки за рахунок зміни розмірів поперечного перетину (площі).

Як наголошувалося вище, при класифікації ниток по структурі, існує відмінність в будові комплексних ниток і пряжі від мононіток (однорідних тіл), для яких можна застосовувати апарат визначення напруги опору матеріалів. Як відзначав проф. Мігушов І. І. [1], найбільш раціональним шляхом буде використання моделей І-й модифікації (див. рис. 1.1), в якій комплексні нитки і пряжа будуть представлені однорідними суцільними тілами. Проте, не можна повністю скидати з рахунків і комбіновані моделі (див. рис. 1.3, а), в яких комплексні нитки і пряжа представляються як сукупність окремих філаментів, не скручених між собою. Такий підхід, запропонований проф. Щербаковим В.П., дозволяє більш глибоко досліджувати поведінку ниток при їх русі по тій, що направляє, проте виникають значні складнощі при вирішенні конкретних інженерних завдань.

Таким чином, основними фізико - механічними показниками, по яких може бути проведена класифікація реальних ниток, є: здатність до розтягування, опір вигину і крученню, зминання, анізотропія тертя, нерівномірний розподіл маси по довжині нитки і різні геометричні форми поперечника нитки.

Вельми істотним, на наш погляд, є аналіз залежностей "навантаження - деформація" для реальних ниток, який дозволяє поглибити класифікацію по фізико - механічних властивостях з погляду нелінійної і лінійної залежності між вказаними чинниками.

На рис. 1.5 представлені графічні залежності "навантаження - деформація", отримані експериментально. Так при розтягуванні ниток (рис. 1.5, а) залежність між силою розтягування P і деформацією ε може мати прямолінійний характер (пряма 1), криволінійний (криві 2 і 3) і складний характер (крива 4) [1]. Текстильні нитки при переробці на технологічному устаткуванні мають відносну деформацію порядку 2-5% [1,10], тобто працюють в зоні пружних деформацій.

При вигині реальних ниток (рис. 1.5 б), при невеликих значеннях кривизни напрямної, відбувається слабкий вигин. Залежність між моментом Mi, що вигинає, і кривизною $До$ носить лінійний характер (пряма 1). Збільшення кривизни осі нитки при її вигині приводить до зростання значення моменту, що вигинає, в зоні пружно пластичності (крива 2). При подальшому збільшенні кривизни частина роботи витрачається на здійснення пластичної деформації. Тут необхідно відзначити, що комплексні нитки і пряжа при вигині поводяться відмінно від мононіток. Збільшення скручування підвищує значення коефіцієнта жорсткості [35-38].

Аналогічні залежності виходять при експериментальному дослідженні скручування ниток. Момент $Мк$ (рис. 1.5, в), що крутить, може змінюватися пропорційно куту повороту перетину нитки ψ (пряма 1) або ця залежність носитиме складніший характер (криві 2, 3).

При зминанні ниток нормальний тиск в зоні контакту N (рис. 1.5 г) може залежати від відносної деформації поперечника δ як лінійно (пряма 1), так і по складнішому закону (криві 2, 3). Цілком очевидно, що при невеликому значенні натягу нитки і малій кривизні деформація поперечного перетину буде незначною. Для комплексних ниток і пряжі процес зминання перетину може носити і незворотній характер (за рахунок зміни положення окремих филаментов в перетині) на відміну від мононіток.

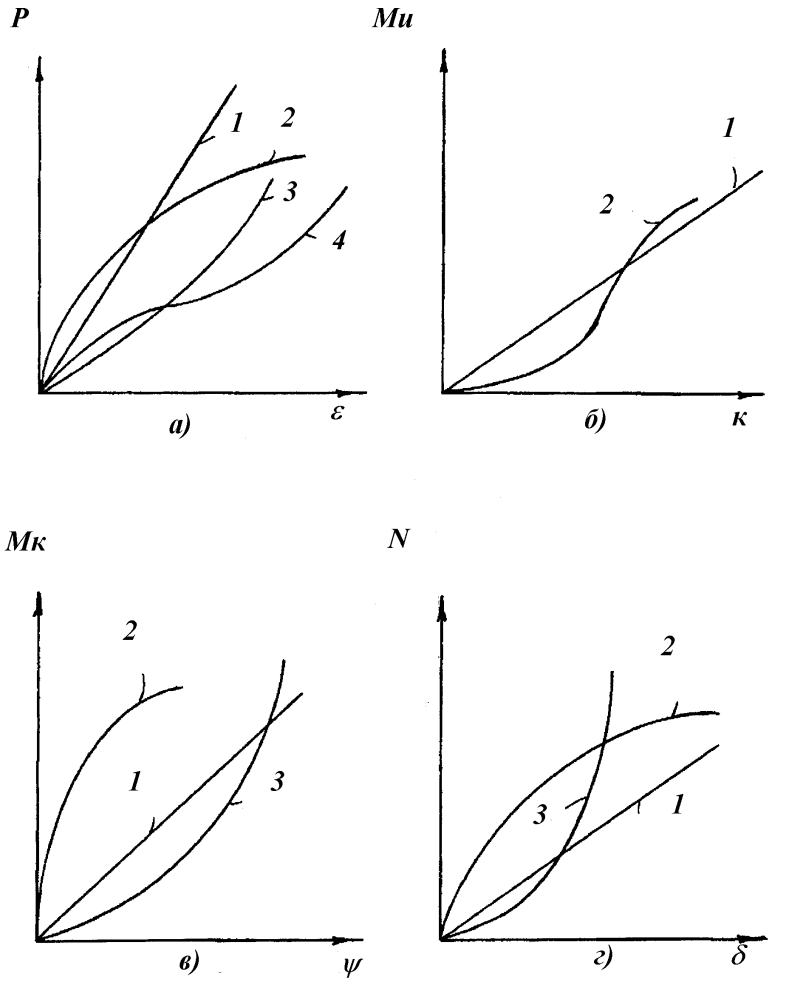


Рис.1.5. Залежності "навантаження - деформація" при розтягуванні, вигині, крутінні та зминанні

Аналізуючи представлені криві, можна зробити висновок про те, що залежність "навантаження - деформація" для різних схем вантаження реальних ниток може носити різний характер. У загальному випадку, для переважної більшості ниток, може застосовуватися модель пружно-в'язкого тіла, що лінійно-деформується

$$A=v\Sigma+\Delta\Sigma \qquad (1.1)$$

або узагальнена модель пружно-в'язкого тіла, що лінійно-деформується

$$A + t_p A' = V\Sigma + V_O t_p \Sigma', \qquad (1.2)$$

де A - напруга, що виникає в нитці відповідно при розтягуванні, вигині, крученні і зминанні; A' - швидкість зміни відповідної напруги; V - коефіцієнт, що характеризує пружні властивості матеріалу (при розтягуванні $V = E$ - поточний модуль жорсткості; при вигині $V = B$ - поточний коефіцієнт жорсткості; при крученні $V = C$ - поточний коефіцієнт крутильної жорсткості; при зминанні $V = E_1$ - поточний модуль жорсткості); Σ, Σ' - відносна деформація і швидкість зміни останньої для відповідних геометричних параметрів (при розтягуванні $\Sigma=\varepsilon$, $\Sigma' = \varepsilon'$ - зміна довжини осі нитки; при вигині $\Sigma = K$, $\Sigma' = K'$ - зміна кривизни осі нитки; при крученні $\Sigma=\Delta$, $\Sigma' = \Delta'$ - зміна геометричного і деформаційного скручування нитки; при зминанні $\Sigma=\delta$, $\Sigma' = \delta'$ - зміна величини поперечного перетину ниток); t_p - час релаксації відповідної напруги; V_0 - миттєвий модуль пружності; Δ - поточне значення коефіцієнтів, що визначають в'язкі властивості матеріалу при кожному конкретному випадку навантаження.

Очевидно, що, помноживши рівняння (1.1) або (1.2) на відповідне значення або площі поперечного перетину нитки, або площі контакту з направляючою і тому подібне, можна отримати залежність зміни натягу нитки, моментів, що вигинають і крутять, сили деформації перетину як функції часу [1,10].

Необхідно відзначити, що нелінійну залежність "навантаження-деформація" зручно представляти у вигляді функціональних рядів, обмежуючись певним числом членів ряду.

Якщо в довільному перетині реальної нитки всі внутрішні сили і вектори моментів спроектувати на осі натурального тригранника, то на дотичну спроектуються вектора натягу ***P*** і моменту ***Мк***, що крутить, на нормаль - момент що вигинає ***Ми2*** і перерізуюча сила ***Q2***, на бінормаль - момент що вигинає ***Ми3*** і перерізуюча сила ***Q3***, ($\vec{M}_u = \vec{M}_{u2} + \vec{M}_{u3}$, $\vec{Q} = \vec{Q}_2 + \vec{Q}_3$).

Якщо у виразі (1.1) вважати, що коефіцієнт ***Λ***, що характеризує в'язкі властивості матеріалу, рівний нулю, то отримаємо модель пружного тіла.

Залежно від значень ***V***, який характеризує пружні властивості матеріалу у формулах (1.1) і (1.2), можна при наближенні його до нуля, отримати ідеально гнучкі на вигин і крутіння, що не чинять опір розтягуванню нитки. При збільшенні ***V*** відбувається зростання жорсткості. Очевидно, що значення даних коефіцієнтів визначається їх структурою.

Як наголошувалося вище, реальні нитки володіють физико-механическими властивостями, про які вже згадувалося. Для конкретних розрахунків, при побудові ниток-моделей, можна не враховувати деякі властивості зважаючи на незначні значення відповідної напруги або сил. Так, для більшості текстильних ниток, з урахуванням умов їх переробки на технологічному устаткуванні, можна нехтувати розтягуванням їх осі. Помилка при цьому не перевищуватиме 5% [35-37]. Більшість комплексних ниток і пряжі невисокого скручування можна вважати абсолютно гнучкими при вигині і крутінні. Мононитки і сталевий дріт можуть вважатися немнучкими в зоні контакту з направляючою поверхнею.

При побудові моделі нитки I і II модифікації по фізико - механічним властивостях необхідно враховувати реальні умови взаємодії нитки з направляючими, конкретну схему силового вантаження. Виходячи з цього, моделі будь-якої модифікації приписуються ті або інші характеристики. Побудова конкретної моделі вимагає значення физико-механических характеристик реальної нитки. Різні поєднання з таких основних характеристик як розтяжність (не розтяжність), опір вигину і крученню, зминання (незминання), анізотропія (ізотропія) тертя, рівномірний (нерівномірний) розподіл маси по довжині нитки, різні (постійні) геометричні розміри перетину нитки, її поточну вагу (або нитка вважається невагомою) дозволяють отримати до 125970 різних моделей нитки. Найчастіше, при описі технологічних процесів, використовують 10-20 моделей. У таблиці 1.1 приведені найосновніші з них.

Розтягуваність характеризується коефіцієнтом

$$f = \frac{\partial S}{\partial S_0} = 1 + \varepsilon ,$$

(для нитки що не розтягується $\boldsymbol{\varepsilon = 0}$ і $\boldsymbol{f = 1}$); зминання – коефіцієнтом

$$\boldsymbol{\delta = \frac{r - r_x}{r}} ,$$

(для нитки що не зминається $\boldsymbol{r = r_x}$ і $\boldsymbol{\delta = 0}$).

Через $\boldsymbol{R_0}$ позначимо головний вектор внутрішніх сил, приведений до центру тяжіння довільного перетину.

Проведена докладна класифікація реальних ниток і моделей I і II модифікацій дозволяє сформулювати основні допущення при побудові останніх:

1) нитками-моделями (надалі просто нитки) є однорідні тіла циліндричної, еліптичної, стадіону подібної форми, або механічні системи, що складаються з окремих циліндричних тіл, замінюючи філаменти (для моделей I модифікації);

Таблиця 1.1.Классификация ниток по їх фізико – механічних властивостях

№ п/п	Геометричні параметри	Силові чинники	Характеристика нитки
I.	$f\neq1,\ \delta=0$	$R_0\neq0,\ Mk=0,\ Mu=0$	Розтяжна, немнучка, ідеально гнучка нитка
2.	$f\neq1,\ \delta=0$	$R_0\neq0,\ Mk\neq0,\ Mu=0$	Розтяжна, немнучка, жорстка на кручення нитка
3.	$f\neq1,\ \delta=0$	$R_0\neq0,\ Mk=0,\ Mu\neq0$	Розтяжна, немнучка, жорстка на вигин нитка
4.	$f\neq1$ $\delta=0$	$R_0\neq0,\ Mk\neq0,\ Mu\neq0$	Розтяжна, немнучка, жорстка на вигин і кручення нитка
5.	$f=1,\ \delta\neq0$	$R_0\neq0,\ Mk=0,\ Mu=0$	Нерозтяжна, мнучка, ідеально гнучка нитка
6.	$f=1$ $\delta\neq0$	$R_0\neq0,\ Mk\neq0,\ Mu=0$	Нерозтяжна, мнучка, жорстка на кручення нитка
7.	$f=1,\ \delta\neq0$	$R_0\neq0,\ Mk=0,\ Mu\neq0$	Нерозтяжна, мнучка, жорстка на вигин нитка
8.	$f=1$ $\delta\neq0$	$R_0\neq0,\ Mk\neq0,\ Mu\neq0$	Нерозтяжна, мнучка, жорстка на вигин і кручення нитка
9.	$f\neq1,\ \delta\neq0$	$R_0\neq0,\ Mk=0,\ Mu=0$	Розтяжна, мнучка, ідеально гнучка нитка
10.	$f\neq1,\ \delta\neq0$	$R_0\neq0,\ Mk\neq0,\ Mu=0$	Розтяжна, мнучка, жорстка на кручення нитка
11.	$f\neq1$ $\delta\neq0$	$R_0\neq0,\ Mk\neq0,\ Mu\neq0$	Розтяжна, мнучка, жорстка на вигин і кручення нитка
12.	$f\neq1,\ \delta\neq0$	$R_0\neq0,\ Mk=0,\ Mu\neq0$	Розтяжна, мнучка, жорстка на вигин нитка

2) нитка може деформуватися як в подовжньому, так і в поперечному напрямах (зминатися);

3) анізотропія тертя ниток може виявлятися як при подовжньому ковзанні по напрямній, так і при поперечному;

4) матеріал нитки заповнює весь передбачуваний об'єм без порожнеч, рівномірно (у разі нерівномірного розподілу матеріалу по довжині нитки задається закон $\boldsymbol{m=m(S)}$);

5) при деформації нитки нормальний перетин залишається плоским (депланацією перетинів нехтують);

6) складний напружено-деформований стан нитки розглядають по спрощених залежностях для простих видів вигину, кручення, зминання, розтягування;

7) для моделей II модифікації нехтуємо тертям в шарнірних парах, між рідиною і стінками направляючих (рис.1.3, в, г, д).

Дані допущення не є вичерпними і у кожному конкретному випадку, при необхідності, вноситимуться додаткові, що відповідним чином аргументуватиметься.

1.2. Визначення геометричних характеристик осі деформованих ниток

Напруга, що виникає в довільному перетині нитки, можна привести до головного вектору $\boldsymbol{P_0}$ і головному моменту $\boldsymbol{M_0}$, які прикладені в центрі тяжіння перетину. Сукупність нескінченної кількості даних точок для кожного з перетинів утворюють вісь нитки [1]. В процесі переробки на технологічному устаткуванні при взаємодії з направляючими відбувається зміна форми осі нитки [1, 2, 18-31], що приводить до зміни кривизни і кручення останньої. При зминанні відбувається деформація поперечного перетину ниток, що викликає зсув центру тяжіння перетину. Отже, в зоні контакту з направляючою відбувається зміна положення осі нитки по

відношенню до лінії, що визначає форму осі абсолютно жорсткої на зминання нитки. Таке переміщення приводить до зростання або зменшення радіусу кривизни осі нитки залежно від фізико - механічних характеристик направляючої поверхні. Крім того, при складанні диференціальних рівнянь рівноваги нескінченно малого елементу нитки що зминається на направляючій поверхні, необхідно знати закон зміни кривизни осі нитки залежно від деформації перетину.

Для дослідження, як модель нитки, вибираємо нитку-модель I модифікації. Вважатимемо, що матеріал нитки заповнює весь об'єм без порожнеч, рівномірно і нитка може розтягуватися [1, 10]. У початковий момент часу нитка не деформована, тому поперечний перетин може мати круглу, еліптичну і стадіону подібну форму.

Для опису форми осі м'ятої нитки необхідно ввести три координатні системи [1, 14-31]. Дві системи рухомі: натурального тригранника $\boldsymbol{\tau^*\nu^*\beta^*}$ і головного тригранника $\boldsymbol{\tau^*n^*b^*}$. Нерухома координатна система $\boldsymbol{O_1XYZ}$ утворюється перетином осей $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$, напрям яких визначається одиничними ортами $\boldsymbol{i, j, k}$ [1]. Дотична вісь $\boldsymbol{\tau^*}$ натурального тригранника отримується шляхом перетину випрямляючої і дотичної площин [1,10]. Нормальна вісь $\boldsymbol{\nu^*}$ прямує до центру кривизни і отримується шляхом перетину нормальної і дотичної площин. Бінормаль $\boldsymbol{\beta^*}$ отримується шляхом перетину нормальної і випрямляючої площин. Нормальна вісь $\boldsymbol{n^*}$ головного тригранника $\boldsymbol{\tau^*n^*b^*}$ прямує по нормалі до направляючої поверхні в даній точці [1-2]. У загальному випадку нормаль $\boldsymbol{n^*}$ головного тригранника і нормаль $\boldsymbol{\nu^*}$ натурального тригранника не співпадають. Кут між ними $\boldsymbol{\psi}$ називається кутом Сен-Венана [1]. Такий же кут буде між бінормалями головного і натурального тригранників. Положення довільної точки $\boldsymbol{A^*}$ (див. рис. 1.6) осі нитки можна визначати лагранжевою (фізичною) дуговою

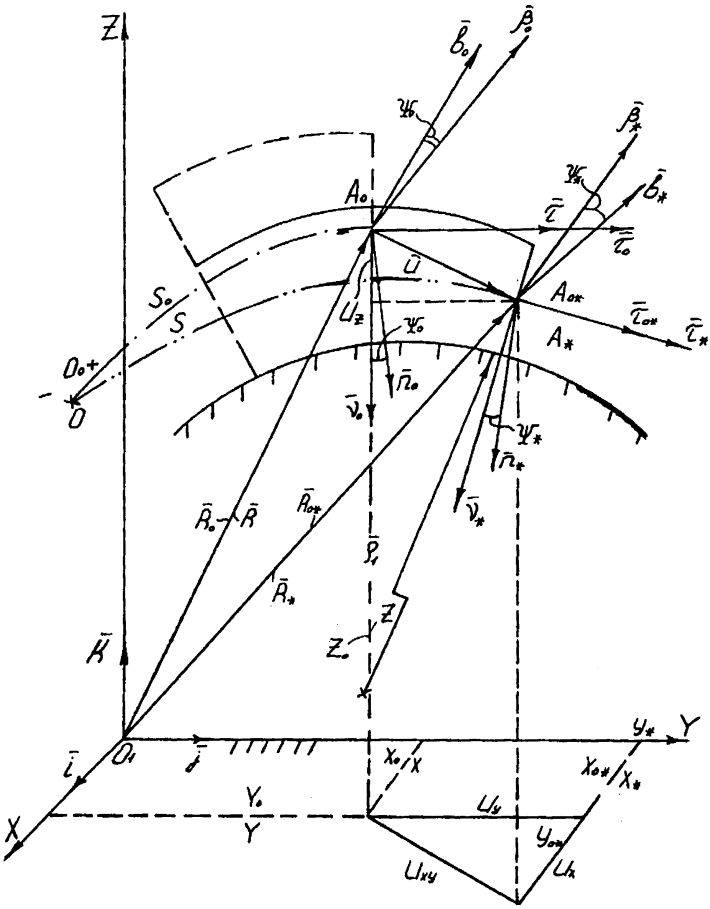


Рис.1.6. Визначення геометричних характеристик форми осі нитки

координатою S_0 і ейлеровою (геометричною) дуговою координатою S [2]. Зв'язок між ними можна представить у вигляді

$$\partial S = (1+\varepsilon)\partial S_0, \tag{1.3}$$

де ε - відносна деформація при розтягуванні.

Положення довільної точки на осі нитки визначається за допомогою відповідних радіус-векторів (див. рис. 1.6), проведених з початку нерухомої координатної системи. Вважатимемо, що точки відліку лагранжевої і ейлерової координат 0 і 0_0 суміщені. Положення точок A_0 і A осі недеформованої нитки визначатиметься координатами S_0 і S, причому, для даних умов $(S=S_0)_{t=0}$.

Радіус-вектор R_0 визначає положення точки A_0 на S_0 - лінії щодо координатної системи O_1XYZ [1]. Радіус-вектор R (див. рис. 1.6) визначає положення точки A_0 на S-лінії щодо нерухомої координатної системи. В результаті при зминанні нитки точка $A(A_0)$ переміститься в нове положення. Новий радіус-вектор R_0^* визначить положення точки A_0^* на S_0-лінії, а радіус-вектор R^* - положення точки A^* на S-лінії. Вектор між точками A і A^* позначимо U . Цей вектор характеризує зсув точок за рахунок деформації перетину нитки.

Векторні вирази для визначення положення точок A_0^* в лагранжевых координатах S_0, t і A^* в ейлеровых координатах S,t матимуть вигляд

$$\overrightarrow{R_{0*}} = \overrightarrow{R_0} + \vec{U}, \ \overrightarrow{R_*} = \overrightarrow{R_*} + \vec{U}. \tag{1.4}$$

Диференціюючи рівняння (1.4) по відповідних дугових координатах, отримаємо

$$\frac{\partial \overrightarrow{R_{0*}}}{\partial S_0} = \frac{\partial \vec{R}_0}{\partial S_0} + \frac{\partial \vec{U}}{\partial S_0}, \ \frac{\partial \overrightarrow{R_*}}{\partial S} = \frac{\partial \vec{R}}{\partial S} + \frac{\partial \vec{U}}{\partial S},$$

де $\frac{\partial R_{0^*}}{\partial S_0} = \overrightarrow{\tau_{0^*}}$ - одиничний орт дотичної рухомого натурального тригранника в лагранжевых координатах S0, t;

$\frac{\partial \vec{R}_0}{\partial S_0} = \overrightarrow{\tau_0}$ - одиничний орт дотичної в точці A_0 недеформованої осі нитки;

$\frac{\partial \vec{R}_*}{\partial S} = \overrightarrow{\tau_*}$ - одиничний орт дотичної рухомого натурального тригранника в ейлеровых координатах S,t;

$\frac{\partial \vec{R}}{\partial S} = \vec{\tau}$ - одиничний орт дотичної в крапці A недеформованій осі нитки.

Отримані рівняння дозволяють встановити залежність між одиничними ортами в різних системах координатних осей. З урахуванням сказаного, останні векторні рівняння приймуть вигляд

$$\overrightarrow{\tau_{0*}} = \overrightarrow{\tau_0} + \frac{\partial \vec{U}}{\partial S_0}, \quad \overrightarrow{\tau_*} = \vec{\tau} + \frac{\partial \vec{U}}{\partial S}. \tag{1.5}$$

З урахуванням виразу (1.3) зв'язок між дотичними ортами в лагранжевих і ейлерових координатах матиме вигляд

$$\overrightarrow{\tau_*} = (1+\varepsilon)\overrightarrow{\tau_{0*}}, \ \vec{\tau} = (1+\varepsilon)\overrightarrow{\tau_0}. \tag{1.6}$$

Вирішуючи спільно рівняння (1.5) і (1.6), отримаємо

$$\overrightarrow{\tau_{0*}} = \frac{\vec{\tau}}{(1+\varepsilon)} + \frac{1}{(1+\varepsilon)}\frac{\partial \vec{U}}{\partial S}, \ \overrightarrow{\tau_*} = (1+\varepsilon)\overrightarrow{\tau_0} + (1+\varepsilon)\frac{\partial \vec{U}}{\partial S_0}.$$

У проекції на осі нерухомої координатної системи O_1XYZ виразу для $\tau 0$ і τ, з урахуванням (1.4), можна представити в наступній формі (див. рис. 1.6)

$$\begin{aligned}
\tau_{0x} &= \frac{\partial x_0}{\partial S_0} = \cos(\vec{\tau}_0;\vec{i}); \tau_x = \frac{\partial x_0}{\partial S} = \cos(\vec{\tau};\vec{i});\\
\tau_{0y} &= \frac{\partial y_0}{\partial S_0} = \cos(\vec{\tau}_0;\vec{j}); \tau_y = \frac{\partial y}{\partial S} = \cos(\vec{\tau};\vec{j});\\
\tau_{0z} &= \frac{\partial z_0}{\partial S_0} = \cos(\overrightarrow{\tau_0};\vec{k}); \tau_z = \frac{\partial z}{\partial S} = \cos(\vec{\tau};\vec{k}).
\end{aligned} \tag{1.7}$$

Відповідно, проектуємо одиничні орты $\tau 0$ і τ^* на осі нерухомої координатної системи, з урахуванням виразу (1.4) можна представити проекції в наступній формі (див. рис. 1.6)

$$\tau_{0^*x}=\frac{\partial x_{0^*}}{\partial S_0}=\frac{\partial\left(x_0+U_x\right)}{\partial S_0}=\frac{\partial x_0}{\partial S_0}+\frac{\partial U_x}{\partial S_0}=\cos(\overrightarrow{\tau_{0^*}};\overrightarrow{i});$$
$$\tau_{0^*y}=\frac{\partial y_{0^*}}{\partial S_0}=\frac{\partial\left(y_0+U_y\right)}{\partial S_0}=\frac{\partial y_0}{\partial S_0}+\frac{\partial U_y}{\partial S_0}=\cos(\overrightarrow{\tau_{0^*}};\overrightarrow{j});$$
$$\tau_{0^*z}=\frac{\partial z_{0^*}}{\partial S_0}=\frac{\partial\left(z_0+U_z\right)}{\partial S_0}=\frac{\partial z_0}{\partial S_0}+\frac{\partial U_z}{\partial S_0}=\cos(\overrightarrow{\tau_{0^*}};\overrightarrow{k});$$
$$\tau_{*x}=\frac{\partial x_*}{\partial S}=\frac{\partial\left(x+U_x\right)}{\partial S}=\frac{\partial x}{\partial S}+\frac{\partial U_x}{\partial S}=\cos(\overrightarrow{\tau_*};\overrightarrow{i});$$
$$\tau_{*y}=\frac{\partial y_*}{\partial S}=\frac{\partial\left(y_0+U_y\right)}{\partial S}=\frac{\partial y}{\partial S}+\frac{\partial U_y}{\partial S}=\cos(\overrightarrow{\tau_*};\overrightarrow{j});\qquad(1.8)$$
$$\tau_{*z}=\frac{\partial z_*}{\partial S}=\frac{\partial\left(z+U_z\right)}{\partial S}=\frac{\partial z}{\partial S}+\frac{\partial U_z}{\partial S}=\cos(\overrightarrow{\tau_*};\overrightarrow{k}).$$

Враховуючи, що вектори ***τ0, τ*** і ***τ0*, τ**** є одиничними ортами, модуль яких рівний одиниці, з урахуванням (1.7) і (1.8), знайдемо зв'язок між відповідними проекціями

$$(\partial x_0)^2+(\partial y_0)^2+(\partial z_0)^2=\frac{(\partial x)^2+(\partial y)^2+(\partial z)^2}{(1+\varepsilon)^2};$$
$$(\partial x_0+\partial U_x)^2+(\partial y_0+\partial U_y)^2+(\partial z_0+\partial U_z)^2=\frac{(\partial x+\partial U_x)^2+(\partial y+\partial U_y)^2+(\partial z+\partial U_z)^2}{(1+\varepsilon)^2}.$$

Для визначення залежності, що визначає одиничний орт головної нормалі залежно від вибору дугових координат, диференціюючи рівняння (1.5) ще раз по відповідних лагранжевих ***S0*** і ейлерових координатах ***S***, отримаємо

$$\frac{\partial\overrightarrow{\tau_{0^*}}}{\partial S_0}=\frac{\partial\overrightarrow{\tau_0}}{\partial S_0}+\frac{\partial^2\overrightarrow{U}}{\partial S_0^2};\ \frac{\partial\overrightarrow{\tau_*}}{\partial S}=\frac{\partial\overrightarrow{\tau}}{\partial S}+\frac{\partial^2\overrightarrow{U}}{\partial S^2}.\qquad(1.9)$$

У векторних рівняннях (1.9) перші похідні від відповідних одиничних ортів дотичних по дугових координатах рівні

$$\frac{\partial\overrightarrow{\tau_{0^*}}}{\partial S_0}=\frac{\overrightarrow{\nu_{0^*}}}{\rho_*};\ \frac{\partial\overrightarrow{\tau_0}}{\partial S_0}=\frac{\overrightarrow{\nu_0}}{\rho_{0^*}};\ \frac{\partial\overrightarrow{\tau_*}}{\partial S}=\frac{\overrightarrow{\nu_*}}{\rho_1};\ \frac{\partial\overrightarrow{\tau}}{\partial S}=\frac{\overrightarrow{\nu}}{\rho_0},\qquad(1.10)$$

де ***ρ*, ρ0**** - відповідно радіуси кривизни осі нитки в точках ***A0*, A0*** ;

ρ1, ρ0 - відповідно радіуси кривизни осі нитки (для ейлерової координати ***S***) в точках ***A*** і ***A****.

Тоді, вирішуючи спільно рівняння (1.9) і (1.10), матимемо

$$\overrightarrow{\nu_{0^*}} = \overrightarrow{\nu_0}\frac{\rho_*}{\rho_{0^*}} + \rho_*\frac{\partial^2\vec{U}}{\partial S_0^2};\ \overrightarrow{\nu_*} = \overrightarrow{\nu}\frac{\rho_1}{\rho_0} + \rho_1\frac{\partial^2\vec{U}}{\partial S^2}. \tag{1.11}$$

Використовуючи відомі співвідношення проф. Мігушова І.І. для одиничних ортов нормалі

$$\overrightarrow{\nu_*} = (1+\varepsilon)\overrightarrow{\nu_{0^*}} + \frac{\partial\varepsilon}{\partial S_0}\rho_*\overrightarrow{\tau_{0^*}};\quad \vec{\nu} = (1+\varepsilon)\overrightarrow{\nu_0} + \frac{\partial\varepsilon}{\partial S_0}\rho_{0^*}\vec{\tau}_0, \tag{1.12}$$

отримаємо співвідношення між одиничними ортами нормалі для лагранжевих і ейлерових координат у відповідних точках

$$\overrightarrow{\nu_*} = (1+\varepsilon)\overrightarrow{\nu_0}\frac{\rho_*}{\rho_{0^*}} + (1+\varepsilon)\rho_*\frac{\partial^2\vec{U}}{\partial S_0^2} + \frac{\partial\varepsilon}{\partial S_0}\rho_*\overrightarrow{\tau_{0^*}}. \tag{1.13}$$

У проекції на осі нерухомої координатної системи вектори одиничних ортов ***ν0*** і ***ν*** приймуть вигляд (з урахуванням (1.7))

$$\begin{aligned}
&\nu_{0x} = \rho_{0^*}\frac{\partial^2 x_0}{\partial S_0^2} = \cos(\overrightarrow{\nu_0};\vec{i});\ \nu_x = \rho_0\frac{\partial^2 x_0}{\partial S^2} = \cos(\overrightarrow{\nu};\vec{i});\\
&\nu_{0y} = \rho_{0^*}\frac{\partial^2 y_0}{\partial S_0^2} = \cos(\overrightarrow{\nu_0};\vec{j});\ \nu_y = \rho_0\frac{\partial^2 y_0}{\partial S^2} = \cos(\overrightarrow{\nu};\vec{j});\\
&\nu_{0z} = \rho_{0^*}\frac{\partial^2 z_0}{\partial S_0^2} = \cos(\overrightarrow{\nu_0};\vec{k});\ \nu_z = \rho_0\frac{\partial^2 z}{\partial S^2} = \cos(\overrightarrow{\nu};\vec{k}).
\end{aligned} \tag{1.14}$$

Для одиничних ортов ***ν0*, ν****

$$\begin{aligned}
&\nu_{0^*x} = \rho_*\frac{\partial^2 x_{0^*}}{\partial S_0^2} = \cos(\overrightarrow{\nu_{0^*}};\vec{i});\ \nu_{*x} = \rho_1\frac{\partial^2 x_*}{\partial S^2} = \cos(\overrightarrow{\nu_*};\vec{i});\\
&\nu_{0^*y} = \rho_*\frac{\partial^2 y_{0^*}}{\partial S_0^2} = \cos(\overrightarrow{\nu_{0^*}};\vec{j});\ \nu_{*y} = \rho_1\frac{\partial^2 y_*}{\partial S^2} = \cos(\overrightarrow{\nu_*};\vec{j});\\
&\nu_{0^*z} = \rho_*\frac{\partial^2 z_{0^*}}{\partial S_0^2} = \cos(\overrightarrow{\nu_{0^*}};\vec{k});\ \nu_{*z} = \rho_1\frac{\partial^2 z_*}{\partial S^2} = \cos(\overrightarrow{\nu_*};\vec{k}).
\end{aligned} \tag{1.15}$$

Вирішуючи спільно системи рівнянь (1.14) і (1.15), можна отримати залежності між відповідними проекціями одиничних ортов нормалі на координатні осі нерухомої системи $\boldsymbol{0_1XYZ}$.

Для визначення виразів між одиничними ортами бінормалі, скористаємося наступною системою векторних рівнянь

$$\begin{aligned}&\overrightarrow{\beta_{0*}}=\overrightarrow{\tau_{0*}}\times\overrightarrow{\nu_{0*}};\quad \overrightarrow{\beta_{*}}=\overrightarrow{\tau_{*}}\times\overrightarrow{\nu_{*}};\\&\overrightarrow{\beta_{0}}=\overrightarrow{\tau_{0}}\times\overrightarrow{\nu_{0}};\qquad \overrightarrow{\beta}=\overrightarrow{\tau}\times\overrightarrow{\nu}.\end{aligned}\tag{1.16}$$

Тоді, з урахуванням (1.5), (1.10) і (1.11) отримаємо

$$\begin{aligned}&\overrightarrow{\beta_{0*}}=\overrightarrow{\beta_0}\frac{\rho_*}{\rho_{0*}}+\frac{\partial\overline{U}}{\partial S_0}\times\overrightarrow{\nu_0}\frac{\rho_*}{\rho_{0*}}+\overline{\tau_0}\times\frac{\partial^2\overline{U}}{\partial S_0^2}\rho_*+\frac{\partial\overline{U}}{\partial S_0}\times\frac{\partial^2\overline{U}}{\partial S_0^2}\rho_*;\\&\overrightarrow{\beta_*}=\overrightarrow{\beta}\frac{\rho_1}{\rho_0}+\frac{\partial\overline{U}}{\partial S}\times\overrightarrow{\nu}\frac{\rho_1}{\rho_0}+\overline{\tau}\times\frac{\partial^2\overline{U}}{\partial S^2}\rho_1+\frac{\partial\overline{U}}{\partial S}\times\frac{\partial^2\overline{U}}{\partial S_0^2}\rho_1.\end{aligned}\tag{1.17}$$

Залежність між одиничними ортами бінормалі у разі лагранжевих і ейлерових координат має вигляд

$$\overrightarrow{\beta_*}=(1+\varepsilon)^2\overrightarrow{\beta_{0*}},$$

або, з урахуванням (1.17)

$$\overrightarrow{\beta_*}=(1+\varepsilon)^2\left[\overrightarrow{\beta_0}\frac{\rho_*}{\rho_{0*}}+\frac{\partial\overline{U}}{\partial S_0}\times\overrightarrow{\nu_0}\frac{\rho_*}{\rho_{0*}}+\overline{\tau_0}\times\frac{\partial^2\overline{U}}{\partial S_0^2}\rho_*+\frac{\partial\overline{U}}{\partial S_0}\times\frac{\partial^2\overline{U}}{\partial S_0^2}\rho_*\right].\tag{1.18}$$

Враховуючи, що деформація нитки в поперечному напрямі відбувається при значно менших навантаженнях, чим деформація в подовжньому напрямі [1,2,8,10] можна рахувати $\boldsymbol{\varepsilon=0}$. Тоді, при фіксованому початку відліку (точки $\boldsymbol{0_0}$ і $\boldsymbol{0}$ співпадають) лагранжевої S_0 і ейлерової S дугових координат $\boldsymbol{dS=dS_0}$.

Вирази для одиничних ортів натуральних тригранників приймуть вигляд

$$\begin{aligned}&\vec{\tau}_*=\overrightarrow{\tau_{0*}};\ \vec{\tau}=\overrightarrow{\tau_0};\ \overrightarrow{\nu_{0*}}=\overrightarrow{\nu_*};\ \vec{\nu}_0=\vec{\nu}\ ;\ \vec{\beta}_{0*}=\vec{\beta}_*;\ \vec{\beta}_0=\vec{\beta};\\&\overrightarrow{\tau_*}=\overrightarrow{\tau_{0*}}=\overrightarrow{\tau_0}+\frac{\partial\vec{U}}{\partial S};\ \overrightarrow{\nu_*}=\overrightarrow{\nu_{0*}}=\overrightarrow{\nu_0}\frac{\rho_1}{\rho_0}+\rho_1\frac{\partial^2\vec{U}}{\partial S^2};\\&\overrightarrow{\beta_*}=\overrightarrow{\beta_0}\frac{\rho_1}{\rho_0}+\frac{\partial\vec{U}}{\partial S}\times\overrightarrow{\nu_0}\frac{\rho_1}{\rho_0}+\vec{\tau}_0\times\frac{\partial^2\vec{U}}{\partial S^2}\rho_1+\frac{\partial\vec{U}}{\partial S}\times\frac{\partial^2\vec{U}}{\partial S^2}\rho_1.\end{aligned}\tag{1.19}$$

Якщо в останній системі рівнянь покласти $\boldsymbol{U=0}$ (нитка не зминається в зоні контакту), то отримаємо очевидну тотожність для немнучкої, нерозтяжної нитки, що взаємодіє з направляючою поверхнею

$$\vec{\tau}_*=\overrightarrow{\tau_0};\ \overrightarrow{\nu_*}=\overrightarrow{\nu_0};\ \overrightarrow{\beta_*}=\overrightarrow{\beta_0}.$$

Перейдемо до визначення кривизни осі зминаємої нитки, для чого одиничні орти головного тригранника в точці ***A**** представимо у вигляді (див. рис. 1.7)

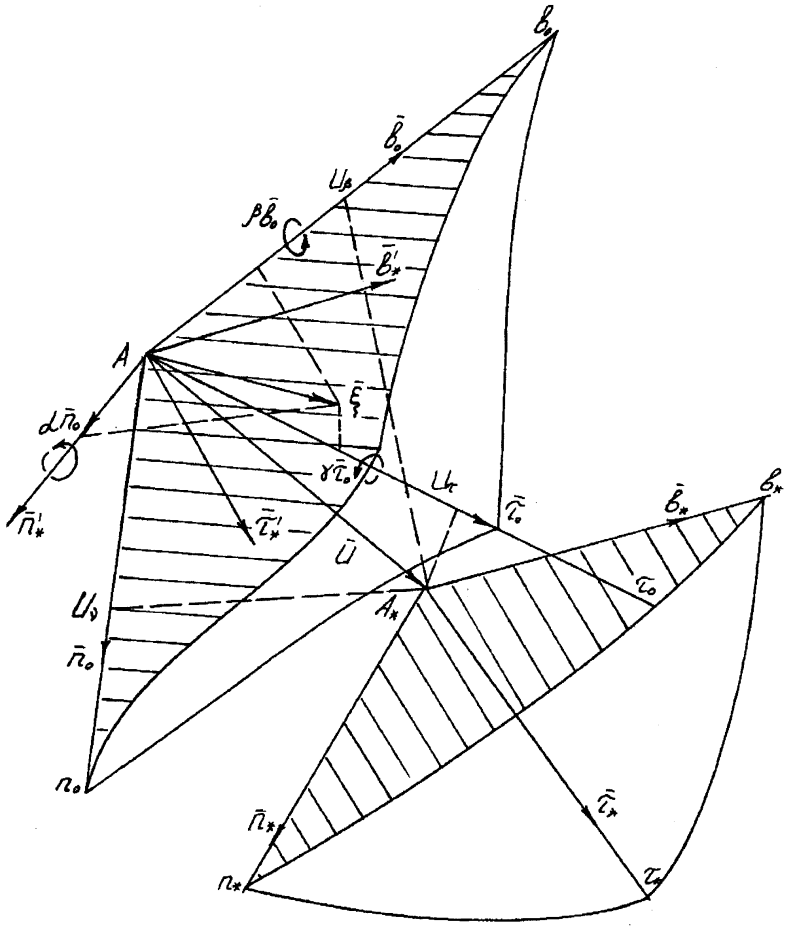


Рис.1.7. Співвідношення між вектором переміщення точок вісі нитки та вектором оберту натурального триграника

$$\begin{aligned}\vec{\tau}_* &= \vec{\tau}_0 - \alpha\vec{b}_0 + \beta\vec{n}_0;\\ \vec{n}_* &= \vec{n}_0 - \beta\vec{\tau}_0 + \gamma\vec{b}_0;\\ \vec{b}_* &= \vec{b}_0 + \alpha\vec{\tau}_0 - \gamma\vec{n}_0,\end{aligned} \tag{1.20}$$

де ***α, β, γ*** - проекції вектора повороту $\vec{\xi}$ на осі головного тригранника ***Aτ0 n0 b0***.

Вектор $\vec{U}$ переміщення точки осі нитки, эйлеровой координатою ***S***, що характеризується, представимо у вигляді проекцій

$$\vec{U} = U_\tau\vec{\tau}_0 + U_\nu\vec{n}_0 + U_\beta\vec{b}_0, \tag{1.21}$$

де ***Uτ, Uv, Uβ*** - проекції вектора $\vec{U}$ на осі головного тригранника.

Вирішуючи спільно сьоме рівняння системи (1.19) і (1.20), отримаємо

$$\vec{\tau}_* = \vec{\tau}_0 + \frac{\partial U_\tau}{\partial S}\vec{\tau}_0 + \frac{\partial U_\tau}{\partial S}\vec{n}_0 + \frac{\partial U_\beta}{\partial S}\vec{b}_0 + U_\tau\frac{\partial\vec{\tau}_0}{\partial S} + U_\nu\frac{\partial\vec{n}_0}{\partial S} + U_\beta\frac{\partial\vec{b}_0}{\partial S}. \tag{1.22}$$

Перші похідні одиничних ортов по дуговій координаті можна представити як векторні твори

$$\frac{\partial\vec{\tau}_0}{\partial S} = \vec{\Omega}_0\times\vec{\tau}_0;\ \frac{\partial\vec{n}_0}{\partial S} = \vec{\Omega}_0\times\vec{n}_0;\ \frac{\partial\vec{b}_0}{\partial S} = \vec{\Omega}_0\times\vec{b}_0, \tag{1.23}$$

де $\vec{\Omega}$ - вектор повної кривизни, величина якого може бути визначена через головні компоненти кривизни осі нитки в довільній крапці (див. рис. 1.6)

$$\vec{U} = U_\tau\vec{\tau}_0 + U_\nu\vec{n}_0 + U_\beta\vec{b}_0, \tag{1.24}$$

де

$$q_0 = \frac{\cos\Psi_0}{\rho_0};\ p_0 = \frac{\sin\Psi_0}{\rho_0};\ r_0 = \frac{1}{\rho_{01}} + \frac{\partial\Psi_0}{\partial S}; \tag{1.25}$$

ρ01 - радіус кривизни геометричного кручення осі нитки; ***ψ0*** - кут Сен-Венана.

Вирішуємо спільно (1.23) і (1.24)

$$\frac{\partial\vec{\tau}_0}{\partial S} = q_0\vec{n}_0 - p_0\vec{b}_0;\ \frac{\partial\vec{n}}{\partial S} = r_0\vec{b}_0 - q_0\vec{\tau}_0;\ \frac{\partial\vec{b}_0}{\partial S} = p_0\vec{\tau}_0 - r_0\vec{n}_0.$$

Підставляючи отриманий результат у вираз (1.22), отримаємо

$$\begin{aligned}\vec{\tau}_* = (1 + \frac{\partial U_\tau}{\partial S} - q_0U_\nu + p_0U_\beta)\vec{\tau}_0 + (\frac{\partial U_\nu}{\partial S} + q_0U_\tau - r_0U_\beta)\vec{n}_0 +\\ + (\frac{\partial U_\beta}{\partial S} - p_0U_\tau + p_0U_\nu)\vec{b}_0\end{aligned} \tag{1.26}$$

Прирівнюючи праві частини рівності (1.26) і першого рівняння системи (1.20), спроектуємо отриманий вираз на осі головного тригранника, для чого послідовно скалярно помножимо його на одиничні орти ***τ0***, ***n0*** і ***b0***

$$\begin{aligned}
&\frac{\partial U_\tau}{\partial S} - q_0 U_\nu + p_0 U_\beta = 0;\\
&\frac{\partial U_\nu}{\partial S} + q_0 U_\tau - r_0 U_\beta = \beta;\\
&\frac{\partial U_\beta}{\partial S} - p_0 U_\tau + p_0 U_\nu = -\alpha.
\end{aligned} \tag{1.27}$$

Отримана система рівнянь (1.27) названа нами умовою на переміщеннях (по аналогії з умовами на швидкостях і прискореннях).

Продиференціюємо векторне рівняння (1.26) по дуговій координаті ***S*** (приймаємо кут Сен-Венана рівним нулю, тобто нормаль співпадає з головною нормаллю до поверхні)

$$\begin{aligned}
&\frac{\vec{n}_*}{\rho_1} = (\frac{\partial^2 U_\tau}{\partial S^2} - q_0 \frac{\partial U_\nu}{\partial S} + p_0 \frac{\partial U_\beta}{\partial S})\vec{\tau}_0 + (\frac{\partial U_\tau}{\partial S} - q_0 U_\nu + p_0 U_\beta)(q_0 \vec{n}_0 - p_0 \vec{b}_0) +\\
&+ (\frac{\partial^2 U_\nu}{\partial S^2} - q_0 \frac{\partial U_\tau}{\partial S} + r_0 \frac{\partial U_\beta}{\partial S})\vec{n}_0 + (\frac{\partial U_\nu}{\partial S} + q_0 U_\tau - r_0 U_\beta)(r_0 \vec{b}_0 - q_0 \vec{\tau}_0) +\\
&+ (\frac{\partial^2 U_\beta}{\partial S^2} - p_0 \frac{\partial U_\tau}{\partial S} + r_0 \frac{\partial U_\nu}{\partial S})\vec{b}_0 + (\frac{\partial U_\beta}{\partial S} - p_0 U_\tau + p_0 U_\nu)(p_0 \vec{\tau}_0 - r_0 \vec{n}_0).
\end{aligned} \tag{1.28}$$

Прирівнюємо ліві частини отриманого рівняння (1.28) і другого рівняння системи (1.20). Скалярно умножаємо ліву і праву частини на одиничний орт ***n0***

$$\frac{(\vec{n}_0 - \beta\vec{\tau}_0 + \gamma\vec{b}_0)\vec{n}_0}{\rho_1} = \frac{1}{\rho_1} = \frac{1}{\rho_0} + \frac{2\partial U_\tau}{\rho_0 \partial S} - \frac{2\partial U_\beta}{\rho_{01}\partial S} + \frac{\partial^2 U_\nu}{\partial S^2} - U_\nu(\frac{1}{\rho_0^2} + \frac{1}{\rho_{01}^2}). \tag{1.29}$$

Отриманий вираз визначає собою величину кривизни осі нитки з урахуванням деформації поперечного перетину в зоні контакту з направляючою. Аналіз отриманого виразу показує, що кривизна осі нитки складається з двох складових: $\boldsymbol{\rho 0^{-1}}$ - геометрична кривизна (без урахування зминання) і складова кривизни, обумовлена деформацією поперечного перетину нитки. На цю обставину звертав увагу проф. Мігушов І.І.[1].

Якщо нехтувати проекцією вектора $\vec{U}$ на дотичну вісь (***Uτ=0***) і вважати, що нитка розташовується в дотичній площині (***Uβ=0***), то тоді

$$\frac{1}{\rho_1} = \frac{1}{\rho_0} + \frac{\partial^2 U_\nu}{\partial S^2} - \frac{U_\nu}{\rho_0^2}. \tag{1.30}$$

Для визначення радіусу кручення ***ρ*1*** осі нитки скористаємося залежністю для головного компоненту кривизни і кручення осі нитки, з урахуванням формул (1.25), отримаємо

$$\begin{aligned} q_1 &= q_0 + \frac{\partial \beta}{\partial S} - \rho_0 \gamma + r_0 \alpha; \\ p_1 &= p_0 + \frac{\partial \alpha}{\partial S} - r_0 \beta + q_0 \gamma; \\ r &= r_0 + \frac{\partial \gamma}{\partial S} - q_0 \alpha + p_0 \beta. \end{aligned} \tag{1.31}$$

Вирішуючи третє рівняння системи (1.31) сумісно з (1.25) і (1.27), матимемо

$$\frac{1}{\rho_{*1}} + \frac{\partial \Psi_*}{\partial S} = \frac{1}{\rho_{01}} + \frac{\partial \Psi_0}{\partial S} + \frac{1}{\rho_0}\left(\frac{\partial U_\beta}{\partial S} - \frac{\sin \Psi_0}{\rho_0} U_\tau + \frac{U_\nu}{\rho_{01}} + U_\nu \frac{\partial \Psi_0}{\partial S}\right).$$

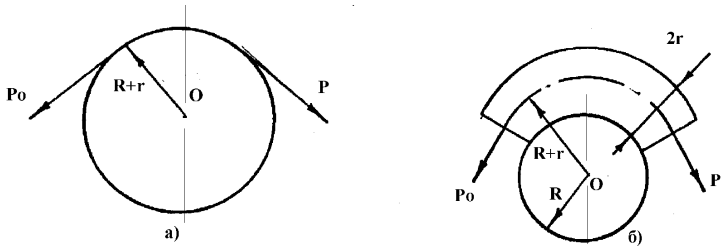
Якщо в останній рівності вважати кут Сен-Венана рівним нулю, тоді

$$\frac{1}{\rho_{*1}} = \frac{1}{\rho_{01}} + \frac{\partial U_\beta}{\rho_0 \partial S} - \frac{U_\nu}{\rho_0 \rho_{01}}. \tag{1.32}$$

Якщо нитка розташовується в дотичній площині, отримаємо

$$\frac{1}{\rho_{*1}} = \frac{1}{\rho_{01}}\left(1 + \frac{U_\nu}{\rho_0}\right). \tag{1.33}$$

Розглянемо приклади визначення кривизни осі нитки для різних випадків взаємодії з направляючою поверхнею (див. рис. 1.8). У першому випадку (рис. 1.8 а) нитка рухається по направляючій поверхні малої кривизни, що деформується; у другому (див. рис. 1.8 б) зминаєма нитка рухається по направляючій поверхні, що не деформується; у третьому випадку і нитка і напрямна деформуються. Початкові радіуси кривизни осі нитки рівні R+r. Закон зміни положення точок осі нитки від величини S має вигляд



Ріс.1.8. Взаємодія нитки направляючими малою(а) і великою(б) кривизни

$$U_v = \{\frac{p_0(R+r)r}{p_0 r} + E_1 b(R+r)^2\}\exp[\frac{\mu s}{(R+r)}],$$

де ***E1*** - модуль пружності при поперечному стисненні; ***μ*** - коефіцієнт тертя між ниткою і що направляє; ***b*** - середня величина сліду контакту.

Результати розрахунку по формулі (1.30) приведені в таблиці 1.2. Розрахунок виконувався при наступних початкових даних: E_1= 500 сН/мм2, ***Po*** = 10 сН (натягнення веденої гілки), ***μ*** = 0,15, ***R***= І мм, ***r*** = 0,2 мм, ***b*** = 0,01 мм.

Аналіз отриманих даних дозволив встановити, що найбільше зменшення кривизни відбувається при взаємодії нитки з направляючою малої кривизни, що деформується.

Таблиця 1.2. Залежність радіусу кривизни осі нитки від дугової координати ***S***

Випадки руху нитки	***S***, мм					
	0,4	0,5	0,6	0,7	0,8	0,9
1	0,2980	0,2913	0,2845	0,2776	0,2706	0,2635
2	0,7030	0,7023	0,7007	0,6990	0,6973	0,6956
3	0,5746	0,5713	0,5680	0,5647	0,5614	0,5579

1.3. Визначення швидкостей і прискорень точок осі деформованої нитки

Теоретичне дослідження процесу руху зминаємої нитки, з погляду визначення швидкостей і прискорень, має велике значення для вирішення ряду конкретних прикладних завдань. Отримані результати можна буде використовувати для вивчення різних технологічних процесів швейною, трикотажною, текстильною галузей, де має місце рух нитки по направляючій поверхні великої кривизни [1,2,10].

Перейдемо до визначення швидкостей і прискорень точок осі м'ятої нитки. На рис. 1.9 представлена розрахункова схема. Для визначеності вважатимемо, що початок відліку лагранжевої і ейлерової координат співпадають. Враховуючи, що зминання нитки в зоні контакту відбувається при зусиллях, значно менших, ніж зусилля, необхідні для розтягування нитки [1], рахуватимемо нитку нерозтяжною $dS=dS_0$, а $\varepsilon=0$.

Положення точки A^* щодо нерухомої координатної системи 0_1XYZ визначається радіус-вектором R^* . Положення точки A на осі незім'ятої нитки (штрихова лінія на рис. 1.9) визначається радіус-вектором R . При деформації поперечного перетину в зоні контакту крапка A переміститься в положення A^* . Вектор AA^* позначимо через U.

Скористаємося співвідношенням (1.4)

$$\overrightarrow{R_*} = \vec{R} + \vec{U}. \tag{1.34}$$

Диференціюючи векторне рівняння (1.4) за часом, отримаємо

$$\overrightarrow{V_*} = \vec{V} + \frac{\partial \vec{U}}{\partial t}, \tag{1.35}$$

де V^* - швидкість точки A^*; V - швидкість точки A.

Для визначення проекцій вектора швидкості V^* на осі нерухомої координатної системи 0_1XYZ [1,2,18-31] необхідно векторне рівняння (1.34) скалярно помножити на відповідні одиничні орти i, j, k. З урахуванням (1.7) отримаємо

$$V_{x*} = V_x + \frac{\partial U_x}{\partial t}; V_{y*} = V_y + \frac{\partial U_y}{\partial t}; V_{z*} = V_z + \frac{\partial U_z}{\partial t},$$

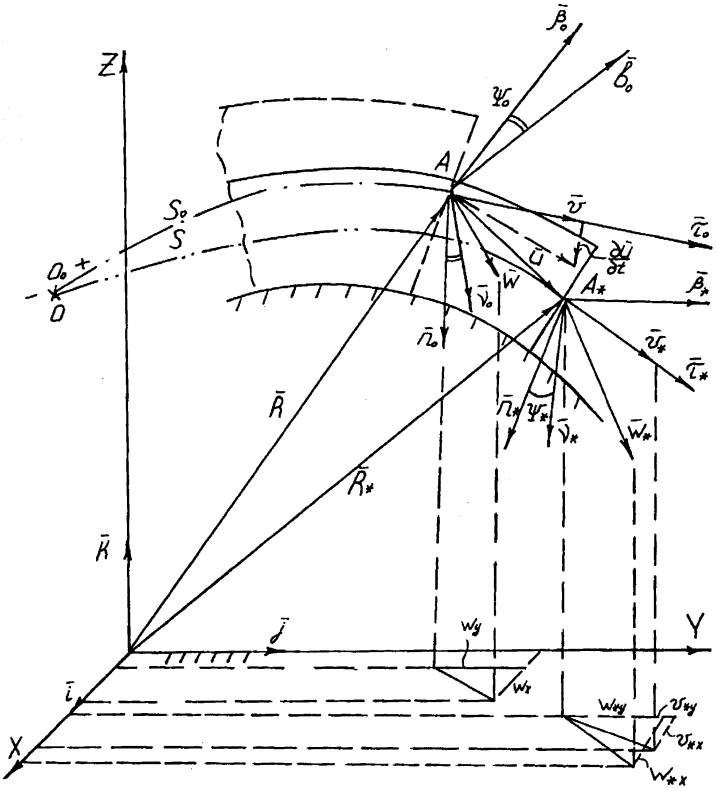


Рис.1.9. Розрахункова схема для визначення швидкостей та прискорень точок вісі зминаємої нитки

де

$$V_x = \frac{\partial x}{\partial t}; V_y = \frac{\partial y}{\partial t}; V_z = \frac{\partial z}{\partial t}.$$

Для визначення закону розподілу швидкостей і прискорень точок нитки, зазвичай вводять в розгляд незалежний вектор $\boldsymbol{P}$ [1]. Якщо даний вектор незмінно пов'язаний з осями $\boldsymbol{\tau, n, b}$ головного тригранника, то

приватні похідні за часом і дуговій координаті будуть рівні (з урахуванням(1.23) -(1.25))

$$\frac{\partial \overline{P}}{\partial S} = \overline{\Omega} \times \overline{P};$$
$$\frac{\partial \overline{P}}{\partial t} = \overline{\omega_e} \times \overline{P}, \tag{1.36}$$

а локальні похідні при цьому рівні нулю.

Якщо вектор $\boldsymbol{P}$, який може мати довільну фізичну природу, міняє свою орієнтацію щодо осей головного тригранника (див. мал. 1.9), то тоді вирази (1.36) приймуть вигляд

$$\frac{\partial \vec{P}}{\partial S} = \vec{\Omega} \times \vec{P} + \frac{\partial \widetilde{P}}{\partial S}; \frac{\partial \vec{P}}{\partial t} = \vec{\omega}_e \times \vec{P} + \frac{\partial \widetilde{P}}{\partial t}, \tag{1.37}$$

де $\frac{\partial \widetilde{P}}{\partial S}, \frac{\partial \widetilde{P}}{\partial t}$ - відповідні локальні похідні.

Тоді, з урахуванням (1.24) -(1.25), (1.34) -(1.35), система рівнянь (1.37), для швидкостей точок осі зминаємої нитки, матиме наступні вирази

$$\left(\frac{\partial \vec{V}_x}{\partial S}\right)_\tau = \frac{\partial V_{x\tau}}{\partial S} + V_{xb}p_1 - V_{xn}q_1;$$
$$\left(\frac{\partial \vec{V}_x}{\partial S}\right)_n = \frac{\partial V_{xn}}{\partial S} - V_{xb}r_1 + V_{x\tau}q_1;$$
$$\left(\frac{\partial \vec{V}_x}{\partial S}\right)_b = \frac{\partial V_{xb}}{\partial S} + V_{xn}r_1 - V_{x\tau}p_1; \tag{1.38}$$
$$\left(\frac{\partial \vec{V}_x}{\partial t}\right)_\tau = \frac{\partial V_{x\tau}}{\partial t} + \omega_2 V_{*b} - \omega_3 V_{*n};$$
$$\left(\frac{\partial \vec{V}_x}{\partial t}\right)_n = \frac{\partial V_{xn}}{\partial t} - \omega_1 V_{*b} + \omega_3 V_{*\tau}; \tag{1.39}$$
$$\left(\frac{\partial \vec{V}_x}{\partial t}\right)_b = \frac{\partial V_{xb}}{\partial t} + \omega_1 V_{*n} + \omega_2 V_{*\tau},$$

де

$r_1 = \frac{1}{\rho_{*1}} + \frac{\partial \Psi_*}{\partial S}, p_1 = \frac{sin\Psi_*}{\rho_1}, q_1 = \frac{cos\Psi_*}{\rho_1}$ - складові вектора повної кривизни, які визначаються з урахуванням виразів (1.29) -(1.33); ψ^* - значення кута Сен-Венана в точці A^*; $V^*\tau$, V^*n, V^*b - відповідно проекції вектора

швидкості V^* на осі τ^*, n^*, b^* головного тригранника; $\omega 1, \omega 2, \omega 3$ - відповідно проекції вектора кутової швидкості ωe елементу нитки на осі головного тригранника (з урахуванням кутової швидкості деформаційного кручення [1]).

Система диференціальних рівнянь (1.38) дозволяє визначити закон зміни швидкості V^* залежно від дугової координати S.

Зв'язок між вектором повної кривизни Ω і вектором абсолютної кутової швидкості ωe визначиться з диференціального рівняння

$$\frac{\partial \vec{\omega}_e}{\partial S} - \frac{\partial \vec{\Omega}}{\partial t} = \vec{\Omega} \times \overrightarrow{\varpi_e}. \tag{1.40}$$

Враховуючи, що

$$\vec{\omega}_e = \omega_1 \vec{\tau}_* + \omega_2 \vec{n}_* + \omega_3 \vec{b}_*, \tag{1.41}$$

а

$$\vec{\Omega}_e = r_1 \vec{\tau}_* + p_1 \vec{n}_* + q_1 \vec{b}_*, \tag{1.42}$$

у проекціях на осі головного тригранника вираз (1.40), з урахуванням (1.41) -(1.42), прийме вигляд

$$\begin{aligned} &\frac{\partial \omega_1}{\partial S} - \frac{\partial r_1}{\partial t} = \omega_2 q_1 - \omega_3 p_1; \\ &\frac{\partial \omega_{21}}{\partial S} - \frac{\partial p_1}{\partial t} = -\omega_1 q_1 + \omega_3 r_1; \\ &\frac{\partial \omega_{31}}{\partial S} - \frac{\partial q_1}{\partial t} = \omega_1 p_1 - \omega_2 r_1. \end{aligned} \tag{1.43}$$

Для визначення залежності між проекціями вектора швидкості V^* і вектора абсолютної кутової швидкості ωe скористаємося наступними співвідношеннями

$$\frac{\partial \vec{V}_e}{\partial S} = \frac{\partial}{\partial S}\frac{\partial \vec{R}_*}{\partial t} = \frac{\partial}{\partial t}\frac{\partial \vec{R}_*}{\partial S} = \frac{\partial \vec{\tau}_*}{\partial t} = \vec{\omega}_e \times \vec{\tau}_*.$$

Вирішуючи останнє векторне рівняння, отримаємо

$$\frac{\partial \vec{V}_*}{\partial S} = -\omega_2 \vec{b}_* + \omega_3 \vec{n}_*. \tag{1.44}$$

Скалярний умножаючи праву частину рівності (1.44) на одиничні орти τ^*, n^*, b^* і прирівнюючи праві частини відповідних рівнянь (1.38),

матимемо систему диференціальних рівнянь ("умов на швидкостях"), що описує рух ниток з урахуванням м'ятої в зоні контакту.

$$\begin{aligned}
&\frac{\partial V_{*\tau}}{\partial S} + V_{*b}\frac{\sin\Psi_*}{\rho_1} - V_{*n}\frac{\cos\Psi_*}{\rho_1} = 0;\\
&\frac{\partial V_{*n}}{\partial S} - V_{*b}\left(\frac{1}{\rho_{*1}} + \frac{\partial\Psi_*}{\partial S}\right) + V_{*r}\frac{\cos\Psi_*}{\rho_1} = \omega_3;\\
&\frac{\partial V_{*b}}{\partial S} + V_{*n}\left(\frac{1}{\rho_{*1}} + \frac{\partial\Psi_*}{\partial S}\right) - V_{*r}\frac{\sin\Psi_*}{\rho_1} = -\omega_2.
\end{aligned} \qquad (1.45)$$

Вирішення системи диференціальних рівнянь (1.45) необхідно проводити з урахуванням (1.28).(1.33). Інтегрування даної системи рівнянь значно полегшується шляхом застосування чисельних методів при його рішенні на ЕОМ.

Останній результат можна отримати при диференціюванні за часом системи рівнянь (1.27) ("умов на переміщеннях"). При цьому, компоненти повної кривизни повинні відповідати її значенню в точці A^*.

Представляє інтерес визначення швидкості в довільній точці осі м'ятої нитки, як функції вектора $\vec{U}$ який визначає ступінь зминання нитки в зоні контакту.

З урахуванням (1.35), вираз для швидкості V^* точки A^* матиме вигляд

$$\vec{V}_* = V_\tau\vec{\tau}_* + V_n\vec{n}_* + V_b\vec{b}_* + U'_\tau\vec{\tau}_* + U'_n\vec{n}_* + U'_b\vec{b}_*, \qquad (1.46)$$

де $V\tau, Vn, Vb$ - відповідно проекції вектора швидкості V точки A на осі головного тригранника в припущенні, що зминання відсутнє; $U\tau', Un', Ub'$ - проекції вектора швидкості і U' переміщення крапки A за рахунок зминання в зоні контакту.

Розподіл швидкостей точок осі зминаємої нитки отримаємо шляхом диференціювання виразу (1.46) по дуговій координаті S

$$\frac{\partial \vec{V}_*}{\partial S} = \left[\frac{\partial (V_\tau + U'_\tau\ )}{\partial S} + (V_b + U'_b) p_1 - (V_n + U'_n) q_1 \right] \vec{\tau}_* +$$
$$+ \left[\frac{\partial (V_n + U'_n\ )}{\partial S} + (V_\tau + U'_\tau) q_1 - (V_b + U'_b) r_1 \right] \vec{n}_* +$$
$$+ \left[\frac{\partial (V_b + U'_b\ )}{\partial S} + (V_n + U'_n) r_1 - (V_\tau + U'_\tau) p_1 \right] \vec{b}_*.$$

Диференціюючи по дуговій координаті рівність (1.35), отримаємо

$$\frac{\partial \vec{V}_*}{\partial S} = \vec{\omega}_e \times \vec{\tau}_* = (\vec{\omega}_0 + \vec{\omega}_u) \times \vec{\tau}_*; \qquad (1.47)$$
$$\vec{\omega}_e = (\vec{\omega}_0 + \vec{\omega}_u),$$

де $\omega 0$ - кутова швидкість обертання головного тригранника $\tau 0 n 0 b 0$ (без урахування зминання); ωu - кутова швидкість обертання головного тригранника, обумовлена переміщенням точок осі нитки за рахунок м'ятої.

Прирівнюючи праві частини останньої системи рівнянь, які описують розподіл швидкостей точок осі м'ятої нитки і векторної рівності (1.47), в проекції на осі головного тригранника $\tau^* n^* b^*$, отримаємо

$$\frac{\partial (V_\tau + U'_\tau)}{\partial S} + (V_b + U'_b) p_1 - (V_n + U'_n) q_1 = 0;$$
$$\frac{\partial (V_n + U'_n\ )}{\partial S} + (V_\tau + U'_\tau) q_1 - (V_b + U'_b) r = \omega_{0b} + \omega_{ub}; \qquad (1.48)$$
$$\frac{\partial (V_b + U'_b\ )}{\partial S} + (V_n + U'_n) r_1 - (V_\tau + U'_\tau) p_1 = -(\omega_{0n} + \omega_{un}),$$

де $\omega 0b, \omega ub$ - проекції векторів кутових швидкостей $\omega 0$, ωu на бінормаль головного тригранника; $\omega 0n, \omega un$ - проекції векторів кутової швидкості елементу нитки, що обертається, на нормальну вісь.

Якщо в системі диференціальних рівнянь (1.48) нехтувати зминанням поперечного перетину ($\vec{U} = 0$), то отримаємо систему рівнянь [1], що визначають умови на швидкостях для немнучких ниток

$$\frac{\partial V_\tau}{\partial S} + V_b p_1 - V_n q_1 = 0;$$
$$\frac{\partial V_n}{\partial S} + V_\tau q_1 - V_b r_1 = \omega_{0b}; \qquad (1.49)$$
$$\frac{\partial V_b}{\partial S} + V_n r_1 - V_\tau p_1 = -\omega_{0n}.$$

Вважаючи, що в системі рівнянь (1.49) кут Сен-Венана рівний нулю $\psi 0=\psi^*= 0$ і враховуючи рівняння (1.24), матимемо відомі співвідношення для швидкостей для ідеальної нитки [1,10]

$$\frac{\partial V_\tau}{\partial S}-\frac{V_n}{\rho_0}=0;\frac{\partial V_n}{\partial S}+\frac{V_\tau}{\rho_0}-\frac{V_b}{\rho_{01}}=\omega_{e3};\frac{\partial V_b}{\partial S}+\frac{V_n}{\rho_{01}}=-\omega_{e2}.$$

Система рівнянь (1.39) служить для визначення проекцій вектора прискорення на координатні осі τ_*, n_*, b_*

$$\begin{aligned}(\vec{W}_*)_\tau&=\frac{\partial V_{*\tau}}{\partial S}+\omega_2 V_{*b}-\omega_3 V_{*n};\\(\vec{W}_*)_n&=\frac{\partial V_{*n}}{\partial S}-\omega_1 V_{*b}+\omega_3 V_{*\tau};\\(\vec{W}_*)_b&=\frac{\partial V_{*b}}{\partial S}+\omega_1 V_{*n}-\omega_1 V_{*\tau},\end{aligned}\qquad(1.50)$$

де $W^*\phi$, W^*n, W^*b - проекції вектора прискорення на осі головного тригранника.

Для визначення залежності між проекціями вектора прискорення від дугової координати, продиференціюємо вираз (1.50) по S, отримаємо

$$\begin{aligned}\left(\frac{\partial\vec{W}_*}{\partial S}\right)_\tau&=\frac{\partial W_{*\tau}}{\partial S}+\frac{\partial\omega_2}{\partial S}V_{*b}+\omega_2\frac{\partial V_{*b}}{\partial S}-\frac{\partial\omega_3}{\partial S}V_{*n}-\omega_3\frac{\partial V_{*n}}{\partial S};\\\left(\frac{\partial\vec{W}_*}{\partial S}\right)_n&=\frac{\partial W_{*n}}{\partial S}+\frac{\partial\omega_3}{\partial S}V_{*\tau}+\omega_3\frac{\partial V_{*\tau}}{\partial S}-\frac{\partial\omega_1}{\partial S}V_{*b}-\omega_1\frac{\partial V_{*b}}{\partial S};\\\left(\frac{\partial\vec{W}_*}{\partial S}\right)_b&=\frac{\partial W_{*b}}{\partial S}+\frac{\partial\omega_1}{\partial S}V_{*n}+\omega_1\frac{\partial V_{*n}}{\partial S}-\frac{\partial\omega_2}{\partial S}V_{*\tau}-\omega_2\frac{\partial V_{*\tau}}{\partial S},\end{aligned}\qquad(1.51)$$

де $\left(\frac{\partial\vec{W}_*}{\partial S}\right)_\tau,\left(\frac{\partial\vec{W}_*}{\partial S}\right)_n,\left(\frac{\partial\vec{W}_*}{\partial S}\right)_b$ - проекції вектора першої похідної прискорення точки A^* по дуговій координаті на осі головного тригранника.

Першу похідну вектора прискорення по дуговій координаті можна представити як

$$\frac{\partial\vec{W}_*}{\partial S}=\frac{\partial}{\partial t}\frac{\partial\vec{V}_*}{\partial S}=\frac{\partial}{\partial t}(\vec{\omega}_e\times\vec{\tau}_*)=\frac{\partial\vec{\omega}_e}{\partial t}\times\vec{\tau}_*+\vec{\omega}_e\times\frac{\partial\vec{\tau}_*}{\partial t}=\frac{\partial\vec{\omega}_e}{\partial t}\times\vec{\tau}_*+\vec{\omega}_e\times(\vec{\omega}_e\times\vec{\tau}_*).\qquad(1.52)$$

З урахуванням виразу (1.41) і

$$\vec{\varepsilon}_e = \frac{\partial \vec{\omega}_e}{\partial t} = \varepsilon_1 \vec{\tau}_* + \varepsilon_2 \vec{n}_* + \varepsilon_3 \vec{b}_*, \tag{1.53}$$

де **e1, e2, e3** - відповідно проекції вектора кутового прискорення на осі ***ф*,n*, b****,* перетворимо формулу (1.52)

$$\frac{\partial \vec{W}_*}{\partial S} = (-\omega_2^2 - \omega_3^2)\vec{\tau}_* + (\omega_1\omega_2 + \varepsilon_3)\vec{n}_* + (\omega_1\omega_3 - \varepsilon_2)\vec{b}_*. \tag{1.54}$$

Прирівнюючи відповідні проекції векторного рівняння (1.54) до правих частин системи (1.51), отримаємо

$$\left.\begin{aligned}
&\frac{\partial W_{*\tau}}{\partial S} + \frac{\partial \omega_2}{\partial S} V_{*b} + \omega_2 \frac{\partial V_{*b}}{\partial S} - \frac{\partial \omega_3}{\partial S} V_{*n} - \omega_3 \frac{\partial V_{*n}}{\partial S} = -\omega_2^2 - \omega_3^2 \overline{\tau_*};\\
&\frac{\partial W_{*n}}{\partial S} + \frac{\partial \omega_3}{\partial S} V_{*\tau} + \omega_3 \frac{\partial V_{*\tau}}{\partial S} - \frac{\partial \omega_1}{\partial S} V_{*b} - \omega_1 \frac{\partial V_{*b}}{\partial S} = \omega_1\omega_2 + \varepsilon;\\
&\frac{\partial W_{*b}}{\partial S} + \frac{\partial \omega_1}{\partial S} V_{*n} + \omega_1 \frac{\partial V_{*n}}{\partial S} - \frac{\partial \omega_2}{\partial S} V_{*\tau} - \omega_2 \frac{\partial V_{*\tau}}{\partial S} = \omega_1\omega_3 - \varepsilon.
\end{aligned}\right\}$$

Систему рівнянь (1.50) можна представити в розгорненому вигляді через складові, які визначають ступінь зминання нитки в зоні контакту. З урахуванням рівнянь (1.46) -(1.48) система (1.50) прийме вигляд

$$\left.\begin{aligned}
&\left(\vec{W}_*\right)_\tau = \frac{\partial (V_\tau + U'_\tau)}{\partial t} + (\omega_{0n} + \omega_{un})(V_b + U'_b) - (\omega_{0b} + \omega_{ub})(V_n + U'_n);\\
&\left(\vec{W}_*\right)_n = \frac{\partial (V_n + U'_n)}{\partial t} - (\omega_{0\tau} + \omega_{u\tau})(V_b + U'_b) + (\omega_{0b} + \omega_{ub})(V_\tau + U'_\tau);\\
&\left(\vec{W}_*\right)_b = \frac{\partial (V_b + U'_b)}{\partial t} + (\omega_{0\tau} + \omega_{u\tau})(V_n + U'_n) - (\omega_{0n} + \omega_{un})(V_\tau + U'_\tau).
\end{aligned}\right\} \tag{1.55}$$

Якщо нитка рухається в площині (наприклад ***τ*,A*, n****), то тоді система рівнянь (1.55) перетвориться

$$\left.\begin{aligned}
&\left(\vec{W}_*\right)_\tau = \frac{\partial (V_\tau + U'_\tau)}{\partial t} - (\omega_{0b} + \omega_{ub})(V_n + U'_n);\\
&\left(\vec{W}_*\right)_n = \frac{\partial (V_n + U'_n)}{\partial t} + (\omega_{0b} + \omega_{ub})(V_\tau + U'_\tau);\\
&\left(\vec{W}_*\right)_b = (\omega_{0\tau} + \omega_{u\tau})(V_n + U'_n) - (\omega_{0n} + \omega_{un})(V_\tau + U'_\tau) = 0.
\end{aligned}\right\} \tag{1.56}$$

У разі прямолінійного руху нитки система (1.56) перетвориться в систему тотожності

$$\left(\vec{W}_*\right)_\tau = \frac{\partial (V_\tau + U'_\tau)}{\partial t}; \left(\vec{W}_*\right)_n = 0; \left(\vec{W}_*\right)_b = 0.$$

Зазвичай, в розрахунках, нехтують подовжнім зсувом при зминанні в зоні контакту і враховують тільки поперечну складову. Тоді, система рівнянь (1.55) спроститься

$$\left(\vec{W}_*\right)_\tau = \frac{\partial V_\tau}{\partial t} - (\omega_{0b} + \omega_{ub})(V_n + U_n');$$
$$\left(\vec{W}_*\right)_n = \frac{\partial (V_n + U_n')}{\partial t} + (\omega_{0b} + \omega_{ub})V_\tau;$$
$$\left(\vec{W}_*\right)_b = (\omega_{0\tau} + \omega_{u\tau})(V_n + U_n') - (\omega_{0n} + \omega_{un})V_\tau = 0.$$

1.4. Основні рівняння динаміки нитки, яка рухається по направляючій великої кривизни

Розробку основ механіки ниток завершимо виведенням рівнянь динаміки, які описують взаємодію ниток з направляючими поверхнями великої кривизни.

До теперішнього часу достатньо детально описаний процес взаємодії з направляючою жорстких на вигин і кручення ниток без урахування зминання в зоні контакту [1]. У роботі [24] робляться спроби описати процес взаємодії рухомої гнучкої нитки з направляючою. Істотне поглиблення дана тема отримала в роботах проф. Ефремова Е.Д. [1].

Відсутність відповідного математичного забезпечення для опису динамічних процесів взаємодії ниток що зминаються з направляючою утрудняє аналіз явищ, що відбуваються при цьому, і, як наслідок, гальмує роботу по оптимізації технологічних процесів і вдосконаленню устаткування.

Підхід до розробки системи диференціальних рівнянь необхідно починати з вибору відповідної моделі нитки I або II модифікації (див. рис. 1.1). Слід зазначити, що математичні рівняння, що описують поведінку ниток-моделей II модифікації, простіші [10]. Проте, як наголошувалося вище, вони використовуються в тих випадках, коли реальні нитки піддаються дії у вузькому спектрі силового поля.

На рис. 1.10, а, б, в представлені різні розрахункові схеми для виведення диференціальних рівнянь руху нитки. Схема 1.10 а описує процес розтягування невагомої нитки, для якою моделлю служить нитка-модель II модифікації Максвела (див. рис. 1.3, е). Диференціальне рівняння, що зв'язує головний вектор всіх зовнішніх сил $\boldsymbol{R_0}$, відносну деформацію $\boldsymbol{\varepsilon}$ і час $\boldsymbol{t}$, має вигляд

$$\frac{\partial \varepsilon}{\partial t} = \frac{1}{E}\frac{\partial R_0}{\partial t} + \frac{R_0}{\eta},$$

де $\boldsymbol{E}$ - модуль пружності нитки при розтягуванні; $\boldsymbol{\eta}$ - коефіцієнт, що характеризує в'язкі властивості ниток.

Векторна умова рівноваги для даної нитки-моделі II модифікації можна записати як

$$\vec{R}_0 = \vec{P}.$$

Нитки-моделі I модифікації (рис. 1.10 б, в) по своїй будові і вигляду найбільш наближені до реальних. Так на рис. 1.10 б представлена найбільш поширена схема, яка використовується для виведення диференціальних рівнянь руху нитки. Всі зовнішні сили і реакції зв'язків, нитки, що діють на елемент, нескінченно малої довжини $\boldsymbol{dS}$, представимо у вигляді головного вектора $\boldsymbol{R_0}$ і головного моменту $\boldsymbol{M_0}$. Дані вектора приведемо до точки $\boldsymbol{A^*}$ - центру мас виділеного елементу нитки. До даної точки приводиться і результуюча сила інерції $\boldsymbol{Ф^*}$ і вектор $\boldsymbol{Mф}$. Головний тригранник $\boldsymbol{\tau^* n^* b^*}$, з вершиною в точці $\boldsymbol{A^*}$, служить координатною системою, щодо якої розглядається умовна рівновага всіх сил і моментів.

Виділення елементу нескінченне малої товщини $\boldsymbol{BC}$ шляхом перетину нитки двома площинами, перпендикулярними осі нитки в точках $\boldsymbol{B}$ і $\boldsymbol{C}$, дозволяє всі внутрішні силові чинники в кожному з них привести до головного вектора і головного моменту: до точки $\boldsymbol{B}$ ($\boldsymbol{M,R}$) - для лівого перетину; до точки $\boldsymbol{C}$

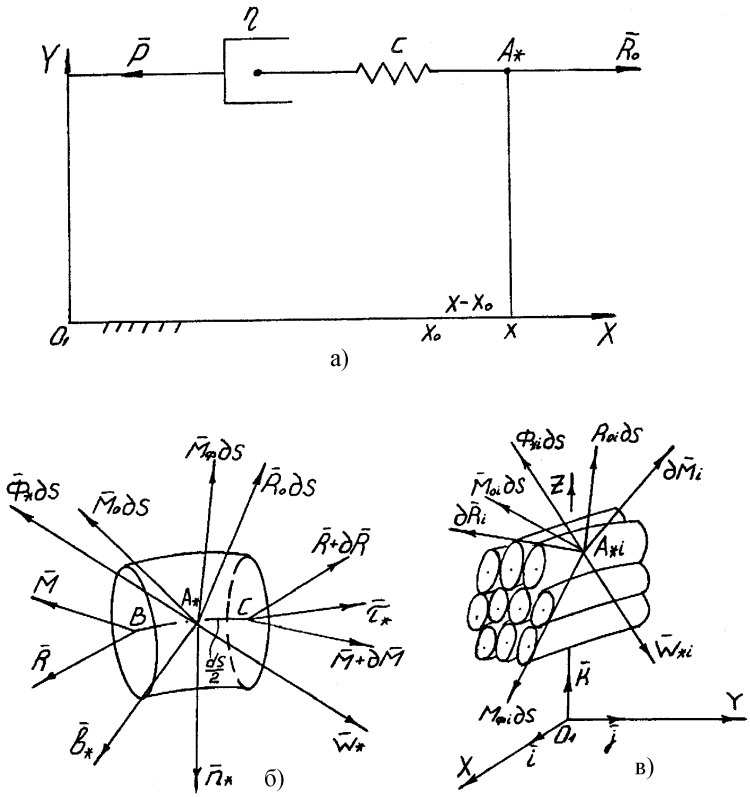


Рис.1.10. Силова схема навантаження елементу нитки

(***R+∂R,M+∂M***) - для правого перетину. При побудові даної схеми вважали, що рух відбувається зліва направо. Векторні рівняння

$$\frac{\partial \vec{R}}{\partial S} + \vec{R}_0 - \vec{\Phi}_* = 0; \frac{\partial \vec{M}}{\partial S} + \vec{M}_0 + \vec{\tau}_* \times \vec{Q} - \vec{M}_\Phi = 0, \qquad (1.57)$$

описують рух нескінченно малого елементу нитки.

При вивченні взаємодії комплексних ниток і пряжі з направляючими поверхнями проф. Щербаковим В. П. була запропонована модель (див. рис.

1.10, в), що складається з окремих, прямолінійно розташованих філаментів (циліндричної форми). Нами вона віднесена до моделей I модифікації.

В даному випадку маємо справу з системою матеріальних тіл. Використовуючи основні принципи методу кінетостатики необхідно для досягнення умовної рівноваги до активно заданих сил і моментів додати відповідні реакції зв'язків і сили інерції, які діють на тіло або систему тіл.

Якщо двома площинами, перпендикулярними осі комплексної нитки або пряжі, відсікти елемент (див. рис. 1.10, в) нескінченно малої довжини ***ds***, що складається з ***n*** елементарних філаментів, то відповідно до приведеного вище методу кінетостатики необхідно до кожного ***i-му*** елементарному филаменту прикласти вказані вище сили.

Центром приведення у кожного ***i*** елементарного волокна буде точка ***A*i*** - центр мас. Всі активно задані сили і відповідні реакції зв'язків приведемо до головного вектора ***R0i*** і головному моменту ***M0i***. Внутрішні зусилля в двох перетинах також приведемо до центру мас. Їх дія на елементарний філамент замінює головний вектор ***dRi*** і головний момент ***dMi***.

Сили інерції приведемо до головного вектора ***Ф*i*** і головному моменту ***Мфi***.

Для ***i*** -го филамента векторні рівняння рівноваги матимуть вигляд

$$\frac{\partial \vec{R}_i}{\partial S} + \vec{R}_{0i} - \vec{\Phi}_{*i} = 0; \frac{\partial \vec{M}_i}{\partial S} + \vec{M}_{0i} - \vec{M}_{\Phi i} = 0.$$

Якщо дані залежності підсумувати для ***n*** елементарних волокон, з яких складається комплексна нитка або пряжа, то при цьому реакції зв'язків (окрім внутрішніх зусиль) знищаться як взаємно урівноважені сили для кожної пари волокон.

У проекції на координатні осі ***X,Y,Z*** отримаємо систему шести рівнянь

$$\sum_{i=1}^{n}\left(\frac{\partial R_{ix}}{\partial S}+\bar{R_{0ix}}-\Phi_{*ix}\right)=0;\ \sum_{i=1}^{n}\left(\frac{\partial M_{ix}}{\partial S}+\bar{M_{0ix}}-M_{\phi ix}\right)=0;$$

$$\sum_{i=1}^{n}\left(\frac{\partial R_{iy}}{\partial S}+\bar{R_{0iy}}-\Phi_{*iy}\right)=0;\ \sum_{i=1}^{n}\left(\frac{\partial M_{iy}}{\partial S}+\bar{M_{0iy}}-M_{\phi iy}\right)=0;$$

$$\sum_{i=1}^{n}\left(\frac{\partial R_{iz}}{\partial S}+\bar{R_{0iz}}-\Phi_{*iz}\right)=0;\ \sum_{i=1}^{n}\left(\frac{\partial M_{iz}}{\partial S}+\bar{M_{0iz}}-M_{\phi iz}\right)=0,$$

де відповідний індекс ***x, y*** або ***z*** указує проекцію на відповідну вісь.

Проте нитки-моделі даного вигляду є досить "грубими". Наприклад при її побудові філаменти вважають розташованими прямолінійно. Насправді, елементарні волокна переплітаються між собою, а це істотним чином міняє характер взаємодії між ними. Крім того, при вирішенні приведеної вище системи рівнянь стикаються з дуже серйозними труднощами. Це стосується складності у визначенні проекцій векторів сил і моментів на відповідні осі координат.

Феноменологічний підхід, який використовувався нами при побудові розрахункової схеми 1.10 б, дозволяє уникнути вказаних вище труднощів при описі руху комплексних ниток і пряжі з урахуванням зминання.

Таким чином, найбільш прийнятною, з погляду простоти отримуваних результатів, є нитка-модель I модифікації, у якої весь об'єм рівномірно заповнений матеріалом.

Для отримання системи диференціальних рівнянь необхідно векторні рівняння (1.57) спроектувати на осі головного тригранника ***τ*n*b**** (див. рис. 1.10 б). Головний вектор ***R*** рівний

$$\vec{R}=P\vec{\tau}_*+Q_2\vec{n}_*+Q_3\vec{b}_*, \qquad (1.58)$$

а головний момент ***M*** рівний

$$\vec{M}=M_k\vec{\tau}_*+M_{u2}\vec{n}_*+M_{u3}\vec{b}_*. \qquad (1.59)$$

Продиференціюємо вирази (1.58) і (1.59) по дуговій координаті

$$\frac{\partial \vec{R}}{\partial S} = \left(\frac{\partial P}{\partial S} - Q_2 q_1 + Q_3 p_1\right)\vec{\tau}_* + \left(\frac{\partial Q_2}{\partial S} + P\, q_1 + Q_3 r_1\right)\vec{n}_* + \left(\frac{\partial Q_3}{\partial S} - P\, p_1 + Q_2 r_1\right)\vec{b}_*,$$

$$\frac{\partial \vec{M}}{\partial S} = \left(\frac{\partial M_k}{\partial S} - M_{u2} q_1 + M_{u3} p_1\right)\vec{\tau}_* + \left(\frac{\partial M_{u2}}{\partial S} + M_k q_1 + M_{u3} r_1\right)\vec{n}_* + \tag{1.60}$$

$$+\left(\frac{\partial M_{u3}}{\partial S} - M_k p_1 + M_{u2} r_1\right)\vec{b}_*,$$

де P - натяг нитки; $Q2, Q3$ - проекції перерізуючої сили на нормаль і бінормаль; Mk - момент, що крутить перетин; $Mu2, Mu3$ - моменти, що вигинають нитку у відповідних перетинах.

Сила інерції, що входить у вираз (1.57), визначиться з урахуванням (1.55) по формулі

$$\vec{\Phi}_* = T\vec{W}_* = T \times$$

$$\times \left\{ \left[\frac{\partial (V_\tau + U'_\tau)}{\partial t} + (\omega_{0n} + \omega_{un})(V_b + U'_b) - (\omega_{0b} + \omega_{ub})(V_n + U'_n) \right] \vec{\tau}_* + \right.$$

$$+ \left[\frac{\partial (V_n + U'_n)}{\partial t} - (\omega_{0\tau} + \omega_{u\tau})(V_b + U'_b) + (\omega_{0b} + \omega_{ub})(V_\tau + U'_\tau) \right] \vec{n}_* + \tag{1.61}$$

$$\left. + \left[\frac{\partial (V_b + U'_b)}{\partial t} + (\omega_{0\tau} + \omega_{u\tau})(V_n + U'_n) - (\omega_{0n} + \omega_{un})(V_\tau + U'_\tau) \right] \vec{b}_* \right\},$$

де T - лінійна щільність нитки.

Для визначення величини $Mф$ скористаємося відомим з теоретичної механіки співвідношенням [10], тоді для елементу нитки

$$\vec{M}_\Phi = \frac{J}{\partial S} \frac{\partial \vec{\omega}}{\partial t}, \tag{1.62}$$

де J - тензор інерції (враховуючи, що осі головного тригранника, розташованого в центрі мас елементу BC, є осями симетрії і головними осями інерції, вважаємо, що відцентрові моменти інерції рівні нулю і значущими будуть тільки розташовані на головній діагоналі матриці тензора інерції).

Складові тензора інерції, розташовані на головній діагоналі матриці, визначаються по формулах

$$J_{\tau^*} = \gamma_н \left[J_{\tau_0} \pm \Delta_\tau(\vec{U}) \right] \partial S,$$
$$J_{n^*} = \gamma_н \left[J_{n_0} \pm \Delta_n(\vec{U}) \right] \partial S, \qquad (1.63)$$
$$J_{b^*} = \gamma_н \left[J_{b_0} \pm \Delta_b(\vec{U}) \right] \partial S,$$

де ***Jф*, Jn*, Jb**** - відповідно моменти інерції виділеного елементу (див. рис. 1.10 б) щодо осей головного тригранника; ***Jτ0, Jn0, Jb0*** - геометричні моменти інерції щодо осей головного тригранника до зминаємої нитки; $\Delta_{\tau(U)}$, $\Delta_{n(U)}$, $\Delta_{b(U)}$ – функціональні коефіцієнти, що визначають зміну геометричних моментів інерції за рахунок зминання в зоні контакту; γ - об'ємна щільність.

З урахуванням залежності (1.53) вираз (1.62) прийме вигляд

$$\vec{M}_\Phi = \gamma_н \left\{ \left[J_{\tau_0} \pm \Delta_\tau(\vec{U}) \right] \varepsilon_1 \vec{\tau}_* + \left[J_{n_0} \pm \Delta_n(\vec{U}) \right] \varepsilon_2 \vec{n}_* + \left[J_{b_0} \pm \Delta_b(\vec{U}) \right] \varepsilon_3 \vec{b}_* \right\}, \qquad (1.64)$$

де ***ε1, ε2, ε3*** - проекції вектора кутового прискорення елементу нитки ***ε**** на осі головного тригранника.

Твір одиничного орта ***τ**** на вектор перерізуючої сили ***Q*** з другого рівняння системи (1.56) буде рівний

$$\vec{\tau}_* \times \vec{Q} = -Q_3 \vec{n}_* + Q_2 \vec{b}_*. \qquad (1.65)$$

Проекції головного вектора всіх зовнішніх сил, для випадку взаємодії нитки з направляючою, можна виразити таким чином

$$\vec{R}_0 = F_\tau \vec{\tau}_* + F_n \vec{n}_* + F_b \vec{b}_*, \qquad (1.66)$$

де ***Fτ, Fn, Fb*** - проекції вектора $\boldsymbol{R_0}$, що становлять, на осі головного тригранника.

Аналогічна залежність буде і для головного моменту зовнішніх сил ***Mo***

$$\vec{M}_0 = M_\tau \vec{\tau}_* + M_n \vec{n}_* + M_b \vec{b}_*, \qquad (1.67)$$

де $\boldsymbol{M_\tau}$, $\boldsymbol{M_n}$, $\boldsymbol{M_b}$ - відповідні проекції на осі головного тригранника.

Результуюча система диференціальних рівнянь, що описують рух ниток що зминаються по направляючій поверхні, була отримана шляхом проектування системи (1.57) з урахуванням (1.24) (1.25) (1.27) (1.29).(1.33), (1.50), (1.56), (1.60).(1.67) на осі головного тригранника

$$\frac{\partial P}{\partial S} - Q_2 q_1 + Q_3 p + F_\tau = T \times$$

$$\times \left[\frac{\partial (V_\tau + U'_\tau)}{\partial t} + (\omega_{0n} + \omega_{un})(V_b + U'_b) - (\omega_{0b} + \omega_{ub})(V_n + U'_n) \right];$$

$$\frac{\partial Q_2}{\partial S} + P\, q_1 + Q_3 r + F_n = T \times$$

$$\times \left[\frac{\partial (V_n + U'_n)}{\partial t} - (\omega_{0\tau} + \omega_{u\tau})(V_b + U'_b) + (\omega_{0b} + \omega_{ub})(V_\tau + U'_\tau) \right];$$

$$\frac{\partial Q_3}{\partial S} - P\, p_1 + Q_2 r_1 + F_b = T \times$$

$$\times \left[\frac{\partial (V_b + U'_b)}{\partial t} + (\omega_{0\tau} + \omega_{u\tau})(V_n + U'_n) - (\omega_{0n} + \omega_{un})(V_\tau + U'_\tau) \right]; \qquad (1.68)$$

$$\frac{\partial M_к}{\partial S} - M_{u2} q_1 + M_{u3} p + M_\tau = \gamma_н \left[J_{\tau_0} \pm \Delta_\tau(\vec{U}) \right] \varepsilon_1;$$

$$\frac{\partial M_{u2}}{\partial S} + M_k q_1 + M_{u3} r - Q_3 + M_n = \gamma_н \left[J_{n_0} \pm \Delta_n(\vec{U}) \right] \varepsilon_2;$$

$$\frac{\partial M_{u3}}{\partial S} - M_k p_1 + M_{u2} r + Q_2 + M_b = \gamma_н \left[J_{b_0} \pm \Delta_b(\vec{U}) \right] \varepsilon_3;$$

$$F_\tau = F_t(A, \mu, N) \cos\varphi_t; F_n = N; F_b = F_t(A, \mu, N) \sin\varphi_t;$$

$$\frac{1}{\rho_0} = \frac{\partial \varphi}{\partial S}; \frac{1}{\rho_{01}} = \frac{\partial \Psi}{\partial S}; \varphi_t = (\vec{\tau}_*; \vec{V}_*); M_b = r_x F_t,$$

де ***Ft(A, м,N)*** - відповідна залежність між нормальним тиском ***N*** в довільній крапці і силою тертя ***Ft*** ; ***μ*** - коефіцієнт тертя; φ_t - кут азимута тертя; ***A*** - коефіцієнт, що характеризує вплив міжмолекулярної взаємодії поверхонь, що труться; r_x - відстань від точки ***A**** осі елементу нитки до тієї, що направляє ($r_x = r - Uv$, де ***r*** - розрахунковий радіус перетину нитки).

Декілька слів необхідно сказати про залежність величини питомого тиску ***N*** від швидкості деформації і величини відносної деформації поперечного перетину нитки.

Як було встановлено проф. Гарбаруком В.Н. [10] натяг нитки із збільшенням швидкості зменшується. При зменшенні розмірів поперечного перетину нитки вплив швидкості її руху на натяг зменшується. Отримані результати справедливі лише для мононіток. Для комплексних ниток і пряжі спостерігається протилежна картина. Як буде показано нижче, при збільшенні швидкості руху нитки її натяг зростає.

Така ж тенденція спостерігається і при збільшенні відносної деформації поперечного перетину нитки. На наш погляд пояснення даного явища необхідно шукати при визначенні швидкості розповсюдження деформацій при зминанні нитки, що взаємодіє з направляючою.

Враховуючи сказане, як модель, що описує залежність між нормальним питомим тиском N і відносною деформацією поперечного перетину δ, прийнята залежність (1.1). Нелінійний зв'язок між вказаними величинами, з урахуванням (1.1), виражається залежностями

$$N = bE_1\delta(1-b_3\delta^{b_4}) + \eta\dot{\delta}^{b_6}(1-b_5\dot{\delta}^{b_6}); \delta = \frac{r-r_x}{r}; \dot{\delta} = \frac{\partial U_v}{\partial t}\frac{1}{r}, \qquad (1.69)$$

де b - ширина сліду контакту; δ' - швидкість відносної деформації поперечного перетину нитки; η - коефіцієнт, що характеризує в'язкі властивості нитки при її поперечній деформації; E_1 - поточний модуль жорсткості; $b3, b4, b5, b6$ - експериментальні коефіцієнти, які визначалися з діаграми "навантаження - деформація".

Аналогічні нелінійні залежності вигляду (1.69) для натягу P, для моменту крутіння $Mк$ і моментів $Mu2$ та $Mu3$, що вигинають нитку представляють фізичні рівняння динамічних деформацій розтягування, кручення і вигину, доповнюють систему диференціальних рівнянь (1.68).

Залежність (1.69) була отримана для випадку деформації нитки у напрямі головної нормалі ($Uv \neq O$). Враховуючи ізотропну властивостей матеріалу нитки при зминанні, можна вважати, що і у напрямі бінормалі ($U\beta \neq 0$) і дотичної ($U\tau \neq 0$) форма рівняння (1.69) не зміниться.

Головні компоненти кривизни визначимо по формулах (1.25), (1.27) і (1.31)

$$q_1 = \frac{\cos\Psi_0}{\rho_0} + \frac{\partial\beta}{\partial S} + \frac{\alpha}{\rho_{01}} + \alpha\frac{\partial\Psi_0}{\partial S};$$
$$p_1 = \frac{\sin\Psi_0}{\rho_0} + \frac{\partial\alpha}{\partial S} - \frac{\beta}{\rho_{01}} - \beta\frac{\partial\Psi_0}{\partial S}; r_1 = \frac{1}{\rho_0} + \frac{\partial\Psi_0}{\partial S} - \alpha\frac{\cos\Psi_0}{\rho_0} + \frac{\sin\Psi_0}{\rho_0}\beta, \qquad (1.70)$$

де

$$\gamma = \frac{\partial U_\tau}{\partial S} - \frac{\cos\Psi_0}{\rho_0} U_\nu + \frac{\sin\Psi_0}{\rho_0} U_\beta ; \beta = \frac{\partial U_\nu}{\partial S} + \frac{\cos\Psi_0}{\rho_0} U_\tau - \frac{U_\beta}{\rho_{01}} - U_\beta \frac{\partial \Psi_0}{\partial S} ;$$

$$\alpha = \frac{\sin\Psi_0}{\rho_0} U_\tau - \frac{\partial U_\beta}{\partial S} - \frac{U_\nu}{\rho_{01\nu}} - U_\nu \frac{\partial \Psi_0}{\partial S}.$$

До отриманих рівнянь необхідно приєднати системи рівнянь (1.43), (1.45), (1.46) і (1.48), які визначають собою умови на швидкостях.

В результаті отримали систему 38 рівнянь, що описує рух нитки-моделі, жорсткої на вигин і кручення, яка зминається в зоні контакту з направляючою поверхнею. Необхідно відзначити, що залежності "навантаження-деформація" носять для пружно вязкопластической нитки нелінійний характер.

Для вирішення системи рівнянь (1.68).(1.70), (1.43), (1.45), (1.46), (1.48) вважаємо спочатку заданими від аргументів ***S*** і ***t*** функції ***ψ0, φt, φ0, ψ*** [1].

Отримана система є замкнутою нелінійною системою з 31 диференціального і 7 рівняннями алгебри. Дана система містить 38 невідомих функцій $P, Q_2, Q_3, M_к, M_{U2}, M_{U3}, F_\tau, F_n, F_b, M_b, F_T, N, r_1, p_1, q_1, V_{*\tau}, V_{*n}, V_{*b}, V_\tau+U_\tau', V_n+U_n', V_b+U_b', \omega_1, \omega_2, \omega_3, \omega_{0\tau}+ \omega_{u\tau}, \omega_{0n}+ \omega_{un}, \omega_{0b}+ \omega_{ub}, \Delta\tau(\bar{U}), \Delta n(\bar{U}), \Delta b(\bar{U}), r_x, \delta, \delta, U_\tau, U_\nu, U_\beta, \rho_0, \rho_{01}$ відносно аргументів ***S*** і ***t***.

Інтегрування даної системи диференціальних рівнянь пов'язане з великими труднощами і можливо тільки із застосуванням ЕОМ. Складність інтегрування полягає у виборі граничних і початкових умов для похідних відповідних ступенів (вище другий).

Зазвичай, при вирішенні конкретних прикладних завдань механіки ниток що зминаються систему рівнянь спрощують, що дозволяє отримати відповідь в кінцевому вигляді.

1.5. Умови взаємодії ниток з направляючою, що деформується

При дослідженні різних технологічних процесів легкої і текстильної промисловості необхідно визначати натяг ниток безпосередньо в зоні переробки (в'язання, формування тканини і ін.). В даному випадку має місце взаємодія нитки з направляючою поверхнею, що деформується [1].

Підхід до рішення даного завдання може базуватися на тих же припущеннях, що і при дослідженні рівноваги або руху стрижня по напрямній що деформується.

При взаємній деформації поверхонь нитки і направляючій в зоні контакту їх геометрична форма змінюється так, щоб при взаємному зсуві виконувався принцип найменшої дії. У даній системі, як і в будь-якій іншій саморегульованій системі, витрата енергії повинна бути мінімальною. Дані особливості саморегульованих механічних систем роблять можливим характеризувати поведінку графічних залежностей їх повної енергії як кривих ліній, що мають властивості максимуму або мінімуму.

Визначення геометричних параметрів осі нитки при її взаємодії з направляючою поверхнею, що деформується, зв'язане з великими труднощами. Це пояснюється тим, що при деформації зона контакту змінює своє положення щодо координатної системи $\boldsymbol{0_1XYZ}$. Переміщення викликане зминанням напрямної і її переносним рухом разом з ниткою (у загальному випадку). Якщо при цьому врахувати зминання самої нитки, то видно, що вона здійснюватиме в просторі складний рух. Результати теоретичного дослідження процесу взаємодії нитки з направляючою, що деформується, служать основою при розробці програмного забезпечення для ЕОМ при вирішенні прикладних завдань.

На рис. 1.11 представлена розрахункова схема для визначення геометричних і кінематичних параметрів точок осі нитки.

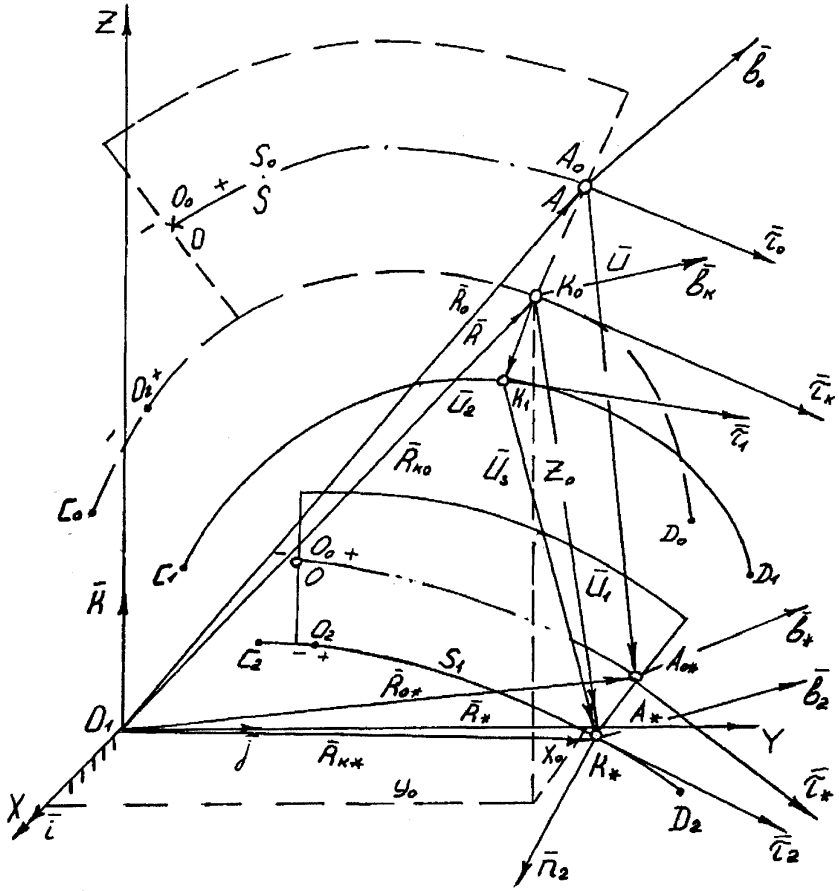


Рис.1.11. Визначення геометричних характеристик точек вісі нитки при її взаємодії з напрямною що деформується

Положення елементу нитки на поверхні, що деформується, визначатимемо щодо нерухомої координатної системи $\boldsymbol{O_1XYZ}$. Для

простоти вважатимемо, що зона контакту представляє деяку лінію. До деформації зона контакту займала положення ***C0D0***.

За рахунок руху направляючої зона контакту, без урахування її зминання, переміститься в положення ***C1D1***. Кінцеве положення зони контакту – ***C2D2***.

Положення точок ***A0, A*** і ***A0*, A**** на осі нитки визначатимемо лагранжевою ***S0*** і ейлеровою ***S*** дуговою координатою. Відносно початку координат нерухомої системи дані точки визначені радіус-векторами ***R0, R*** і ***R0****, ***R****. Початки координат ***0*** і $\boldsymbol{0_0}$ лагранжевої і ейлерової координат нерухомі і розташовані в одній точці.

Положення точок на лінії, що визначає зону контакту, заданою координатою ***S1***. Вона відлічується від деякого нерухомого центру ***O2*** на цій лінії.

Якщо нехтувати розтягуванням нитки, то ***ds=ds0*** і точки ***A0, A*** і ***A0*, A**** співпадуть. Кожній з них, на лінії контакту, відповідатиме точка ***K0*** і ***K**** . Вони отримуються шляхом перетину лінії контакту нормальною площиною до нитки в точках ***A*** і ***A**** відповідно.

Радіус-вектори ***Rk0*** і ***Rk**** визначають положення точок ***K0*** і ***K**** відносно початки нерухомої координатної системи.

У разі, коли поверхня напрямної має малу кривизну і розмірами поперечного перетину нитки можна нехтувати, вісь нитки співпадатиме з лінією контакту.

У проекції на осі головного тригранника $\boldsymbol{\tau_k n_k b_k}$ вектор $\vec{U}_1$ можна представити таким чином

$$\vec{U}_1 = U_{1\tau}\vec{\tau}_k + U_{1n}\vec{n}_n + U_{1b}\vec{b}_b = (U_{2\tau} + U_{3\tau})\vec{\tau}_k + (U_{2n} + U_{3n})\vec{n}_k + (U_{2b} + U_{3b})\vec{b}_k, \qquad (1.71)$$

де ***U1τ, U1n, U1b*** - відповідні проекції вектора $\vec{U}$ на осі головного тригранника; ***U2τ, U3τ, U2n, U3n, U2b, U3b*** - відповідно проекції векторів $\vec{U}_2$ і $\vec{U}_3$ на дотичну ***τk***, нормаль ***nk*** і бінормаль ***bk***.

Позначимо кути між дотичній τk і осями головного тригранника $\tau_0 n_0 b_0$ в точці A осі нитки через $\alpha 1, \beta 1, \gamma 1$. Між нормаллю nk і відповідними осями $\tau 0, n0, b0$ через $\alpha 2, \beta 2, \gamma 2$. Тоді бінормаль bk в точці $K0$ утворює з осями головного тригранника $\tau_0 n_0 b_0$ кути $\alpha 3, \beta 3, \gamma 3$.

Виразимо вектор $\vec{U}$ через відповідні проекції на осі головного тригранника. Для цього спроектуємо векторне рівняння (1.71) на відповідні координатні осі з урахуванням (1.21)

$$\begin{aligned}\vec{U} = U_{\varepsilon\tau}\vec{\tau}_0 + U_{\varepsilon n}\vec{n}_0 + U_{\varepsilon b}\vec{b}_0 = (U_\tau + U_{1\tau}\cos\alpha_1 + U_{1n}\cos\alpha_2 + U_{1b}\cos\alpha_3)\vec{\tau}_0 + \\ + (U_\nu + U_{1\tau}\cos\beta_1 + U_{1n}\cos\beta_2 + U_{1b}\cos\beta_3)\vec{n}_0 + \\ + (U_\beta + U_{1\tau}\cos\gamma_1 + U_{1n}\cos\gamma_2 + U_{1b}\cos\gamma_3)\vec{b}_0,\end{aligned} \tag{1.72}$$

де $U\varepsilon\phi, U\varepsilon n, U\varepsilon b$ - проекції вектора $\vec{U}$ на осі головного тригранника $\tau_0 n_0 b_0$ з урахуванням деформації направляючої поверхні. Співвідношення між одиничними ортами головних тригранників в точках A і A^* знайдемо з системи векторних рівнянь (1.20) з урахуванням (1.71) -(1.72)

$$\begin{aligned}\vec{\tau}_* &= \vec{\tau}_0 - \alpha_\varepsilon \vec{b}_0 + \beta_\varepsilon \vec{n}_0; \\ \vec{n}_* &= \vec{n}_0 - \beta_\varepsilon \vec{\tau}_0 + \gamma_\varepsilon \vec{b}_0; \\ \vec{b}_* &= \vec{b}_0 + \alpha_\varepsilon \vec{\tau}_0 - \gamma_\varepsilon \vec{n}_0,\end{aligned} \tag{1.73}$$

де $\alpha_\varepsilon, \beta_\varepsilon, \gamma_\varepsilon$ - проекції вектора повороту $\vec{U}$ на осі головного тригранника.

Прирівнюючи праві частини першого рівняння системи (1.73) і рівняння (1.26), з урахуванням (1.18) (1.22).(1.25), (1.72), спроектуємо отриманий результат на осі головного тригранника $\tau_0 n_0 b_0$

$$\begin{aligned}\frac{\partial U_{\varepsilon\tau}}{\partial S} - q_0 U_{\varepsilon\nu} + p_0 U_{\varepsilon\beta} &= 0; \\ \frac{\partial U_{\varepsilon\nu}}{\partial S} + q_0 U_{\varepsilon\tau} - r_0 U_{\varepsilon\beta} &= \beta_\varepsilon; \\ \frac{\partial U_{\varepsilon\beta}}{\partial S} - p_0 U_{\varepsilon\tau} + r_0 U_{\varepsilon\nu} &= -\alpha_\varepsilon.\end{aligned} \tag{1.74}$$

Система рівнянь (1.74) визначає собою умови на переміщеннях для нитки, що взаємодіє з направляючою, що деформується.

Використовуючи методику визначення головних компонентів кривизни для ниток що зминаються, а також залежності (1.28).(1.33), для випадку взаємодії нитки з направляючою поверхнею, що деформується, величини ***qε1, pε1, rε1*** будуть рівні

$$
\begin{aligned}
q_{\varepsilon 1} &= \frac{\cos\Psi_0}{\rho_0} + \frac{\partial\beta_\varepsilon}{\partial S} + \frac{\alpha_\varepsilon}{\rho_{01}} + \alpha_\varepsilon \frac{\partial\Psi_0}{\partial S};\\
p_{\varepsilon 1} &= \frac{\sin\Psi_0}{\rho_0} + \frac{\partial\alpha_\varepsilon}{\partial S} - \frac{\beta_\varepsilon}{\rho_{01}} - \beta_\varepsilon \frac{\partial\Psi_0}{\partial S};\\
r_{\varepsilon 1} &= \frac{1}{\rho_0} + \frac{\partial\Psi_0}{\partial S} - \alpha_\varepsilon \frac{\cos\Psi_0}{\rho_0} + \frac{\sin\Psi_0}{\rho_0}\beta_\varepsilon;\\
\gamma_\Sigma &= \frac{\partial U_{\varepsilon\tau}}{\partial S} - \frac{\cos\Psi_0}{\rho_0} U_{\varepsilon\nu} + \frac{\sin\Psi_0}{\rho_0} U_{\varepsilon\beta};\\
\beta_\Sigma &= \frac{\partial U_{\varepsilon\nu}}{\partial S} + \frac{\cos\Psi_0}{\rho_0} U_{\varepsilon\tau} - \frac{U_{\varepsilon\beta}}{\rho_{01}} - U_{\varepsilon\beta}\frac{\partial\Psi_0}{\partial S};\\
\alpha_\Sigma &= \frac{\sin\Psi_0}{\rho_0} U_{\varepsilon\tau} - \frac{\partial U_{\varepsilon\beta}}{\partial S} - \frac{U_{\varepsilon\nu}}{\rho_{01\,\nu}} - U_{\varepsilon\nu}\frac{\partial\Psi_0}{\partial S},
\end{aligned}
\qquad (1.75)
$$

де ***ψ0*** – значення кута Сен-Венана в точці ***A*** осі нитки.

Якщо припустити, що нитка розташовується на направляючій поверхні по геодезичній лінії і кут Сен-Венана рівний нулю, то для нерозтяжної нитки співвідношення між дуговими координатами ***S*** і ***S1***,будут мати вигляд

$$\partial S_1 = \frac{\rho_{k^*}}{(\rho_{k^*} + A_* K_*)}\partial S,$$

де ***ρk**** - радіус кривизни в точці ***K**** лінії контакту.

При ***A*K*→0*** отримаємо ***ds1=ds*** (поперечними розмірами нитки нехтуємо).

Розподіл швидкостей точок осі нитки, що взаємодіє з направляючою, що деформується, отримаємо на підставі рівності (1.38)

$$\left(\frac{\partial \vec{V}_x}{\partial S}\right)_\tau = \frac{\partial V_{x\tau}}{\partial S} + V_{xb}p_{\varepsilon 1} - V_{xn}q_{\varepsilon 1};$$
$$\left(\frac{\partial \vec{V}_x}{\partial S}\right)_n = \frac{\partial V_{xn}}{\partial S} - V_{xb}r_{\varepsilon 1} + V_{x\tau}q_{\varepsilon 1}; \qquad (1.76)$$
$$\left(\frac{\partial \vec{V}_x}{\partial S}\right)_b = \frac{\partial V_{xb}}{\partial S} + V_{xn}r_{\varepsilon 1} - V_{x\tau}p_{\varepsilon 1}.$$

Вирішувати останню систему рівнянь (1.76) необхідно спільно з (1.74).

Аналогічно визначається закон розподілу прискорень точок осі нитки, що взаємодіє з направляючою, що деформується.

На підставі рівності (1.68),(1.69) і (1.74) будується система диференціальних рівнянь, що описують рівновагу елементу нитки на тій, що направляє, що деформується.

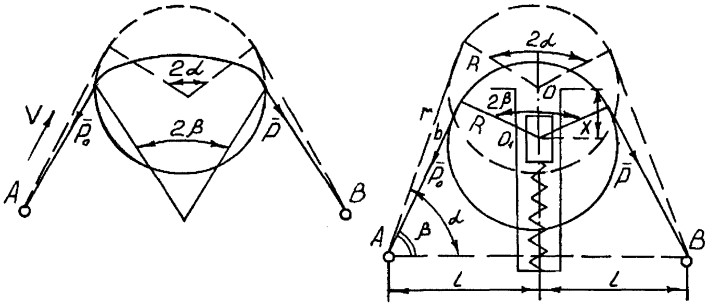
Приведені вище результати показують, що досліджуваний процес є дуже складним з погляду його теоретичного опису. Використання як об'єкт дослідження нитки-моделі I модифікації (див. 1.1) вимагає введення великого числа допущень і обмежень, що знижує точність отримуваних результатів. Все сказане указує на те, що дане питання ще далеке від свого кінцевого рішення, результати якого можна використовувати в інженерних розрахунках.

Нижче приведено дослідження процесу взаємодії нитки з направляючої малої кривизни, що деформується, з використанням нитки-моделі II модифікації.

На рис. 1.12, а приведена розрахункова схема. На лівому рисунку показана реальна картина процесу зминання напрямної і зменшення кута обхвату. Штриховою лінією показана напрямна до деформації.

Під дією нормального тиску напрямна сплющується і змінює форму зовнішньої поверхні, що приводить до зменшення кута обхвату направляючою ниткою. Це, у свою чергу, позначається на величині натяг ведучої галузь нитки.

При складанні розрахункової схеми вважалося, що нерозтяжна нитка, поперечними розмірами якої можна нехтувати, рухається з постійною швидкістю, сили тертя підкоряються закону Амонтона. Жорсткістю на вигин і зминанням нитки можна нехтувати (***Uτ=0=Uн= Uв***). Пружність матеріалу нитки при стисненні характеризується жорсткістю пружини ***C***, яка еквівалентна їй і однорідна за об'ємом. Вважатимемо, що повзун ***0*** переміщається в вертикальних напрямних без тертя, а напрямна жорстко пов'язана з повзуном. Натяг веденої гілки нитки приймемо рівним ***P0***.



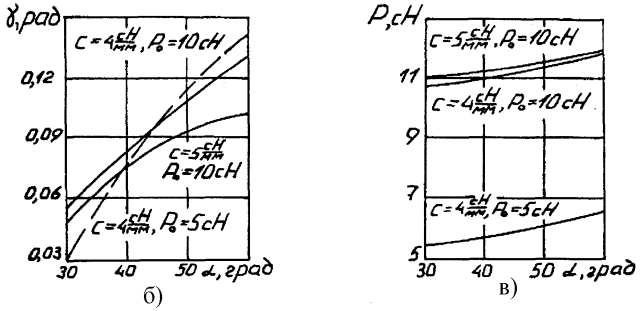


Рис.1.12. Розрахункова схема та основні залежності зміни натягу ведучої гілки нитки як функції кута охоплення направляючої що деформується

При русі нитки по направляючій відбуватиметься деформація пружини. Величину деформації позначимо через ***X.*** Рівновага рухомої направляючої описується наступною системою рівнянь

$$CX = P_0 \sin\beta(1 + e^{1\mu\beta}); \; P = P_0 e^{2\mu\beta}; \; \beta = \alpha - \gamma, \tag{1.77}$$

де β - половина кута обхвату при деформованій пружині; α - половина кута обхвату в початковий момент (до початку деформації); γ - величина, яка визначає зменшення кута обхвату направляючої при зминанні; μ - коефіцієнт тертя.

Для визначення кута обхвату складаємо рівняння, що геометрично зв'язують деформацію X і величину зміни кута обхвату γ

$$X = r\sin\alpha - R\cos\alpha - b\sin\beta + R\cos\beta;$$
$$r = \frac{l - R\sin\alpha}{\cos\alpha}; \; b = \frac{l - R\sin\beta}{\cos\beta}, \tag{1.78}$$

де R - радіус кривизни циліндричної направляючої (відповідає розрахунковому радіусу нитки); l - відстань від точки спрямовувача нитки до вертикальної осі переміщення повзуна.

Вирішуючи спільно рівняння (1.77) і (1.78), нехтуючи членами рівнянь, що містять квадрати і твори малих величин, отримаємо трансцендентне рівняння для визначення величини кута γ як функції деформації направляючої

$$\gamma = \frac{P_0 \sin(\alpha - \gamma)\left[1 + e^{2\mu(\alpha-\gamma)}\right]\cos^2\alpha}{C(l - R\sin\alpha)}. \tag{1.79}$$

Вирішення рівняння (1.79) можна знайти за допомогою ЕОМ, застосовуючи метод дихотомії. На рис. 1.12 б приведені залежності зміни кута γ від величини кута обхвату α. Аналіз показав, що із зменшенням коефіцієнта жорсткості з 5 до 4 сН/мм кут γ збільшується в середньому на 10-16%, що зменшує вихідне натяг. Приведені залежності дозволили

встановити, що при збільшенні сукання ниток зростає і коефіцієнт їх жорсткості при зминанні.

Результати розрахунку натягу ведучої галки нитки по формулі (1.77) представлені нижче.

Таблиця 1.3.Залежність натягу нитки від величини кута обхвату напрямної

Початковий натяг	Натяг ведучої галки нитки ***P***, сН			
	Кут обхвату до деформації, град.			
	30	***40***	***50***	***60***
P0=5 сН, ***C*** = 4 сН/мм	5,52	5,69	5,88	6,07
P0=10сн, ***C***= 5 сН/мм	11,00	11,33	11.67	12,06
P0=10сн, ***З*** = 4 сН/мм	10,98	11,30	11,62	11,90

За наслідками розрахунків побудовані графічні залежності (рис. 1.12 в) зміни натяг ведучій гілки від кута обхвату.

При взаємодії нитки з направляючою що деформується, відбувається зниження натягу (в порівнянні з випадком, коли деформація зминання відсутня) за рахунок зменшення кута обхвату.

Зменшення кута обхвату більшою мірою спостерігається у ниток, що мають мале скручування і використовуються як направляючі поверхні.

1.6. Аналіз причин обривності ниток при їх переробці на технологічному устаткуванні

Вдосконалення технологічного устаткування текстильної і легкої промисловості повинне вестися, шляхом збільшення його продуктивності. Одним з напрямів підвищення продуктивності є зниження часу простою за рахунок ліквідації обривів ниток [1-14].

У роботі [1] приводяться дані розподіли відмов по сумарному часу відновлення і по кількості. При аналізі приведених діаграм видно, що по

кількості відмов обриви ниток складають 77, 6%, а питомий час на їх ліквідацію складає 28,1% від загального часу простою з різних причин.

Обриви ниток, виникаюча поперечна смугастість в період пуску і останову приводять до того, що сотні тонн полотна йдуть на вирізку. Наприклад, за даними Мінлегпрома України, по Мукачевському ПТО умовна вирізка склала приблизно 24 тонни (для різних артикулів трикотажного полотна), по Горлівському ПТО - 145 тонн, по Чернівецькому ПТО - 12 тонн.

Руйнування ниток негативно позначається і на якості трикотажних і тканих полотен. Так в роботі Адамової н.А. і Дибленко в.І. [1] встановлено, що дефекти трикотажного полотна в 52 випадках із ста відбуваються з вини того, що в'яже, коли нитка взаємодіє з направляючими і робочими органами.

У практиці роботи підприємств вивченню обривності приділяється багато уваги. З цією метою проводяться систематичні спостереження за обривністю. Вельми істотним є не тільки визначення кількості обривів на певну довжину одиночної нитки, але і з'ясування причин обриву.

Враховуючи, що обриви ниток є основною причиною останову технологічного устаткування і, як наслідок, зниження продуктивності, поряд дослідників була проведена робота по виявленню основних причин обриву і їх систематизація. У роботі проф. Далідовіча а.С. [1] були виділені чотири основні причини: конструктивні, технологічні, експлуатаційні, старіння. Зважаючи на специфіку дослідження, що проводиться, нас цікавитимуть технологічні і експлуатаційні причини.

Аналіз процесу руйнування ниток при їх переробці необхідно починати з визначення розподіли обривів ниток по глибині і ширині заправки технологічного устаткування. Це необхідно для виявлення умов, в яких працює нитка в зоні руйнування.

На рис. 1.13 і 1.14 представлені технологічні схеми заправки мотальной машини М-150-2, снувальної машини СЛ-250-Ш (для стрічкового снования), ткацького верстата АТТ-120-5М, основовязальной машини "Кокетт",

панчішного автомата ОЗД. Тут же приведені гістограми розподілу обривності (%) і відносного натяг ниток ***P/po*** (де ***Po*** - натяг нитки при виході з балоногасника) залежно від зони технологічної лінії заправки, які були отримані на основі експлуатаційних спостережень і проведення відповідних експериментальних досліджень [8-31]. Пунктиром показана зміна відносного натяг.

Вибір такого переліку устаткування дозволяє широко охопити всю область технологічної переробки ниток, починаючи від підготовчого виробництва і закінчуючи виготовленням безпосередньо тканини і трикотажу. Для мотальной машини М-150-2 (мал. 1.13, а) зона ***А*** відповідає відстані від точки сходу нитки з бобіни до балоногасника, зона ***Б*** включає балоногасник і до приладу для натягу нитки, зона ***У*** включає прилад для натягу нитки і до контрольно-очисного пристрою, зона ***Г*** включає контрольно-очисний пристрій і до точки намотування включно.

Аналіз залежностей показав, що максимальна обривність (60%) і величина відносного натяг (2,8) буде в зоні ***Г*** [19]. Проф. Гордєєв В. А. [1] пояснює це збільшення м'ятою шишок і потовщень при проходженні через нитенатяжной прилад і контрольно-очисний пристрій.

На рис. 1.13 б представлені технологічна схема і гістограми розподілу обривності і відносного натягу ниток для стрічкової снувальної машини СЛ-250-Ш. За даними експериментальних досліджень, проведених на Київській фабриці "Техноткань", найбільша обривність спостерігалася в зоні ***Б*** - від пристрою для натягу нитки до контрольної коробочки (50%). Відносне натяжіння збільшувалося по глибині заправки і в зоні ***Д*** досягало 4,1.

Для ткацького верстата АТТ-120-5М відповідні гістограми і пружна схема заправки приведені на рис. 1.14, а (пунктиром показана зміна відносного натягу). Зона ***А*** включала точку сходу нитки з навоя до скало, зона ***Б*** включала скало і до ламельного розділового приладу, зона ***У*** включала ламельный розділовий прилад до першої ремізи, зона ***Г*** включала ремізні рамки і до зони формування тканини.

Обривність і натяг збільшуються по глибині верстата (наприклад, для капронової тканини СТСЗ-5М, що виробляється на Київській фабриці "Технотканъ", обривність в зонах ***В*** і ***Г*** дорівнювала відповідно 20 і 54%, а відносний

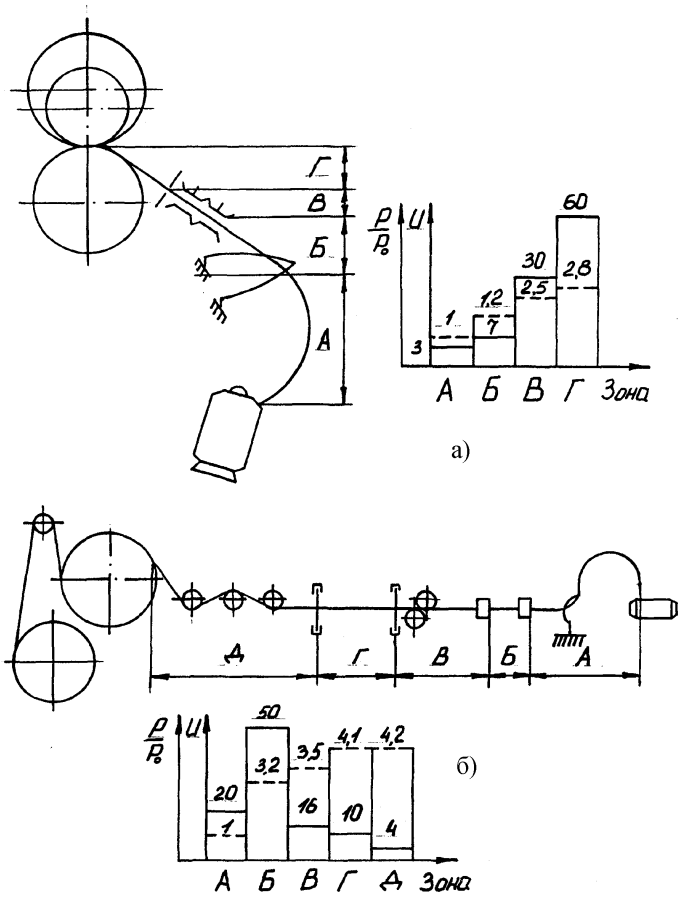


Рис.1.13. Технологічні схеми заправки та номограми вимірів обривності та відносного натягу

натяг - відповідно 1,6 і 2,4). Таке збільшення можна пояснити зростанням натягу ниток основи при їх взаємодії з направляючими великої кривизни (ламелі, отвори галев і ін.).

Д.т.н. Николаєвим с.Д. [1] при дослідженні умов формування тканини арт. 966 "Ліана", з подовжніми смугами полотняного і саржевого 3/1 переплетення, на верстатах АТ-100-5М і АТПР-100-4 встановлено, що відбувається зростання натягу основи при наближенні до зони формування. Умовна напруга в зоні "ламелі - реміз" більше заправного для верстата АТПР в 1,2-1,24 разу, а для верстата АТ в 1,37-1,43 разу.

Форма пружної системи заправки робить вплив на величину натягу і обривності ниток. Найменша обривність спостерігатиметься при співвідношенні довжини основи і тканини 148/85 і 148/104.

По ширині заправки ткацького верстата, як було встановлено проф. Гордєєвим В.А. , більшість обривів качка на верстаті типу АТ відбувається при польоті човника зліва направо і при цьому слід чекати появи місця розриву в зіві з вірогідністю 0,576.

Для основовязальной машини "Кокетт" (рис. 1.14,б) в зоні ***Г*** взаємодії ниток з ушковинами гребінок, де відбувається обрив окремих філаментів і формування шишок і потовщень, обривність досягає 70% [1-14]. При цьому відносний натяг зростає до 3,3.

Дослідження, проведені проф. Волощенко В.П. і к.т.н. Новаком С.Н. [1], показали, що на панчішному автоматі ОЗД в зонах ***В*** і ***Г*** (рис. 1.14,в) обривність зростає відповідно до 40 і 50% при одночасному зростанні відносного натягу до 3,3-3,5. На рис. 1.15,а приведені характерні види руйнування волокон. Унаслідок наявності дефектності структури орієнтованих полімерів розриви концентруються в місцях найбільших дефектів, приводячи до появи мікротріщин. Характер осередків руйнування, що утворюються, залежить від структури волокон. За даними Мортона в.Е. і Херла д.В. [1] сильно напружені і розірвані елементарні волокна мають тріщини на поверхні і в місцях руйнування.

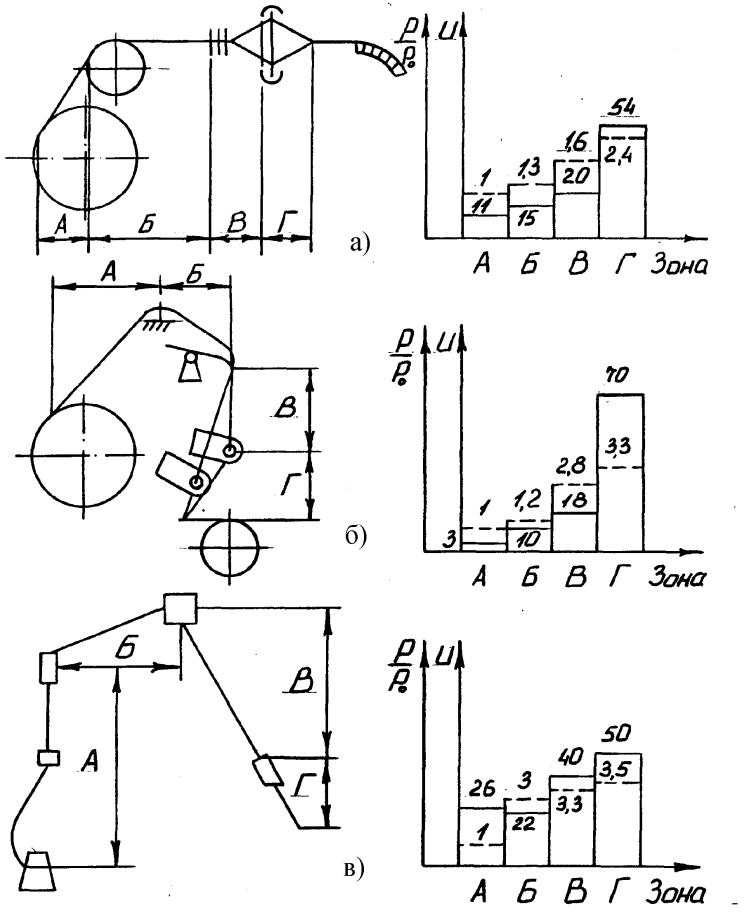


Рис.1.14. Технологічні схеми заправки та номограми вимірів обривності та відносного натягу

З чого автори роблять висновок, що розрив відбувається спочатку в поверхневих шарах, що мають нижче розривне подовження.

При теоретичному дослідженні впливу скручування стеклонитки при вигині на деформованість филаментов Рагоза І.В. встановив, що найбільшу деформацію мають зовнішні волокна. Це приводить до збільшення внутрішньої напруги в останніх і їх руйнування.

Дану обставину має велике значення, коли нитка взаємодіє з направляючою поверхнею. Зусилля, що виникають при стисненні нитки, приводять до виникнення руйнуючої напруги в її поперечному перетині і уздовж волокон.

Специфіка переробки ниток на технологічному устаткуванні приводить до того, що останні випробовують багатократні дії з боку робочих органів. Як відзначав проф. Моїсєєнко ф.А. [1], це приводить до стирання зовнішніх филаментов, їх руйнування. Форма розриву при цьому (рис. 1.15, а) носить явно виражений фібрилізаційний характер. До особливо цього схильні високо анізотропні волокна.

Взаємодія ниток направляючими великої кривизни, при багатократному переміщенні уздовж останніх, приводить до посилення стираючої дії за рахунок збільшення нормального тиску в зоні контакту. У волокон гетерогенної мікроструктури наголошується ступінчастий характер руйнування за рахунок виникнення подовжніх і поперечних тріщин. Міцність в поперечному напрямі визначається руйнуванням міжатомних зв'язків.

Як наголошувалося вище, руйнування волокон концентруються в місцях найбільших дефектів нитки. На рис. 1.15 б представлений процентний розподіл дефектів, наявність яких викликає обрив нитки при її переробці на технологічному устаткуванні [1,10]. До зовнішніх дефектів текстильних ниток відносяться такі, які характеризують чистоту ниток. Необхідно відзначити, що при складанні діаграми не враховувалися причини, що викликають обрив при сході нитки з бобіни (зростаюче натягнення при сході нитки з торця, зліт витків і ін.). Дані чинники впливають на умови переробки нитки на технологічному

устаткуванні. Їх усунення дозволяє понизити обривність і підвищити якість продукту, що випускається.

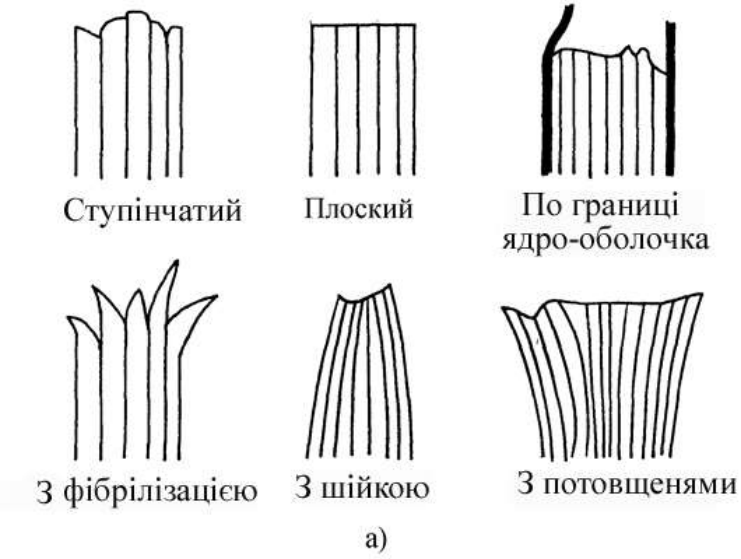


Рис.1.15. Характерні види руйнування волокон та процентний розподіл дефектів нитки

Зважаючи на специфіку дослідження, що проводиться, розглядатимемо вплив тільки тих пороків, яких набуває нитка в процесі свого виготовлення, і які залежать від якості початкового матеріалу.

Аналіз дефектів, що приводять до обриву нитки при переробці, показав, що найбільш вагомими є: слабкі місця на нитці (стоншування) - 32%, шишки і потовщення - 28%, моховитість - 20%. Якщо виникнення перших двох дефектів пояснюється низькою якістю початкової сировини і порушенням технологічного процесу при виробництві ниток (витяжка, кручення), то останній, як наголошувалося вище, виникає безпосередньо при взаємодії ниток з робочими органами технологічного устаткування. За рахунок великої кривизни поверхні останніх значно зростає величина питомого тиску. Це приводить до руйнування окремих филаментов, що знижує міцність нитки на розрив.

При зворотно-поступальних рухах відбувається збиття кінців зруйнованих филаментов і не зароблених при виробництві ниток в шишки і потовщення. Наявність останніх утрудняє рух нитки в нитенатяжных, контрольно-очисних приладах, по робочих органах великої кривизни. Для цього необхідно витрачати додаткову роботу на м'яту шишок і потовщень, що приводить до різких скачок натяг і обриву.

Таким чином, при переробці ниток з дефектами, з одного боку, виникають скачки натягу при проходженні шишок і потовщень через прилади для натягу нитки, по робочих органах великої кривизни, а з іншого боку, в місцях стоншування в нитці виникає напруга, рівна розривної [1-14].

При оцінці міцності ниток в умовах переробки проф. Щербаковим В.П. [1] була запропонована функція руйнування, як основна величина для зв'язку напруги і деформації з урахуванням ступеня накопичення пошкоджень.

Дослідження за визначенням впливу швидкості руху комплексних ниток на обривність елементарних волокон дозволили Матуконісу А.В. і Лазауськасу Ю.Ю. [1] встановити, що для диацетатной і триацетатной ниток обривність із

збільшенням швидкості зростає, а при швидкостях, що перевищують 3,3 м/с, - зменшуються.

Аналізуючи приведену вище інформацію можна зробити вивід про те, що обривність ниток виникає із-за дефектів, які можна розділити на придбані (руйнування окремих филаментов при стиранні і зминанні в зоні контакту нитки з напрямною при переробці, на технологічному устаткуванні) і дефекти, яких набуває нитка в процесі свого виготовлення. Зменшення сумарного опору руху нитки по направляючих і робочих органах дозволить понизити величину напруги в небезпечних перетинах нитки і запобігти її розриву в робочій зоні.

Порівнюючи гістограми обривності і відносного натяг (див. рис. 1.13, 1.14), можна прослідкувати пряму залежність.

Зниження натягу ниток в робочій зоні, як відзначав проф. Окс Б.С., дозволяє зменшити обривність і створює передумови для оптимізації процесів петлеобразования на трикотажних машинах [1, 10].

Представляють інтерес роботи, пов'язані з визначенням ступеня руйнування і обривів ниток при їх взаємодії з макро- і мікронерівностями на поверхні направляючих і робочих органів.

У дослідженні Школи Н.Н. і Антонової А.І. [10] було встановлене зниження міцності капронових ниток при переробці на круглочулочных автоматах на 55%. Наголошується, що при зосередженій дії нитки на певні ділянки отворів фурнітури для спрямування на одноволокнистих нитках з'являються надрізи, глибина яких досягає 60-90 мкм, а в багатоволокнистих нитках відбуваються обриви або затягування елементарних волокон.

Для експерименту на круглочулочном автоматі 34 класи використовували послідовно нові петлеобразующие органи і зношені, із задирками, забоїнами, подряпинами. Дані механічні пошкодження утворюються при аварійних поломках голок, платин. Зниження міцності нитки в процесі того, що в'яже в першому випадку склало 2-3%, а в другому - до 25% .

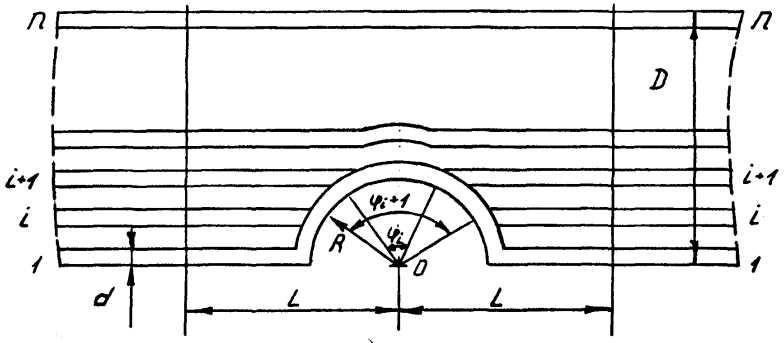
Враховуючи, що комплексні нитки і пряжа складають в загальному об'ємі всіх ниток, що переробляються, велику частину, зупинимося на аналізі причин обривності елементарних волокон при їх взаємодії з макро - і мікронерівностями.

На рис. 1.16, а приведена розрахункова схема. Умовно можна вважати, що макронерівності - це такі нерівності, у яких геометричні розміри соизмеримы з розрахунковим діаметром нитки або пряжі, а мікронерівності - це такі нерівності, у яких геометричні розміри соизмеримы з розмірами поперечного перетину окремого філамента. Зменшення шкідливого впливу взаємодії ниток з мікронерівностями (з погляду зменшення обривності за рахунок руйнування зовнішньої поверхні окремих філаментів) досягається шляхом зменшення шорсткості поверхні робочих органів за рахунок додаткової їх обробки. Складніші процеси відбуваються при взаємодії нитки з макронерівностями. Для простоти аналізу вважатимемо, що макронерівність має форму півкола радіусу R. Кути обхвату окремими філаментами поверхні будуть рівні: $1 - \pi..., i - \varphi, i+1 - \varphi_{i+1...}$. У пропонованій моделі [10] окремі филаменты діаметром d, враховуючи малу величину зони контакту нитки з направляючою, розташовуватимуться по прямих лініях. Велике значення при визначенні напруженості умов взаємодії комплексних ниток і пряжі з макронерівністю має довжина ковзання волокна. Це довжина зони L, в якій волокно зберігає можливість переміщатися щодо інших волокон при зминанні.

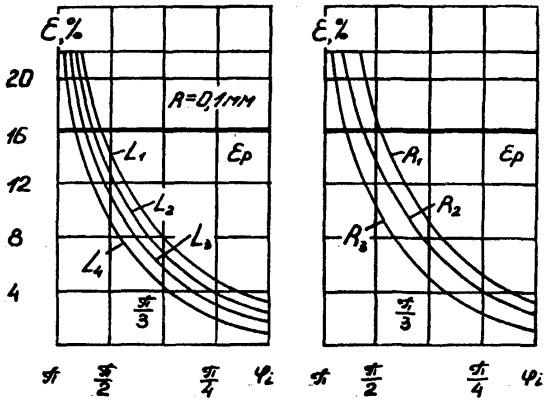
Проведені дослідження показали, що величина L може бути визначена залежністю

$$L = F(f, f_1, R) / K,$$

де $F(f, f1, R)$ - деяка функція, що визначає залежність між суканням нитки і величиной L; f- коефіцієнт тертя між окремими елементарними волокнами; f_1 - коефіцієнт тертя між окремими елементарними волокнами і направляючою поверхнею; K - скручування нитки.



а)



б)

Рис.1.16. Взаємодія ниток з макро- та мікронерівностями поверхні робочих органів

Аналіз виразу показує, що із збільшенням скручування довжина зони ***L*** скорочується. Це пояснюється підвищенням нормального тиску між окремими філаментами.

Абсолютне розривне подовження ***Δl*** для ***i-го*** філамента можна виразити в наступному вигляді

$$\Delta l = l_1 - l_0 = (R + \frac{d}{2})\varphi_i - 2R\sin\frac{\varphi_i}{2}; l_0 = 2L;$$

$$l_i = 2(L - R\sin\frac{\varphi_i}{2}) + (R + \frac{d}{2})\varphi_i.$$

Тоді відносна деформація розтягування ***i-го*** филамента може бути виражена, як

$$\varepsilon = \frac{\left|\left(R + \frac{d}{2}\right)\varphi_i - 2R\sin\frac{\varphi_i}{2}\right|}{2L} 100.$$

Результати розрахунку по формулі приведені на рис. 1.16 б. Початкові дані бралися для капронової комплексної нитки лінійною щільністю 29 текс пологого скручування. Кількість філаментів дорівнювала 80. Розрахунковий діаметр $D = k\sqrt{T}$, де $k=0{,}0386$ [1,10].

Діаметр одного філамента визначали по формулі $d = D / 11 =$ 0,019 мм.

Для даного матеріалу нитки величина розривного подовження ε_P = 16%. Основна умова нормальної взаємодії $\varepsilon < \varepsilon_P$.

Аналіз результатів показав, що із зменшенням сукання (***L1>L2>L3>L4***) величина відносної деформації зменшується, а із збільшенням ***R*** збільшується і наближається до гранично допустимої. З цього можна зробити висновок, що нитки підвищеного скручування більш схильні до руйнувань при взаємодії з макронерівностями. Необхідно відзначити, що значення відносної деформації при $\varphi_i = \pi$ у реальних умовах менше розрахунковою за рахунок жорсткості філаментів на вигин і їх нещільного прилягання до поверхні макронерівності.

Таким чином, всі реальні нитки володіють певними дефектами і пороками. В процесі переробки на технологічному устаткуванні натяг ниток

зростає і досягає своєї максимальної величини в робочій зоні. Обрив нитки наступає тоді, коли за рахунок збільшення натяг (в результаті тертя, зминання в зоні контакту з направляючою) напруга в небезпечному перетині (місці розриву окремих філаментів, стоншування) досягне свого критичного значення.

Отже, зменшення обривності можна досягти двома шляхами: підвищенням якості ниток і пряжі, оптимізацією натягу ниток на всій протяжності пружної системи заправки на основі його мінімізації. Остання мета може бути досягнута на основі комплексних теоретико-експериментальных досліджень процесу взаємодії нитки з направляючими поверхнями з урахуванням м'ятої, жорсткості на вигин, анізотропії фрикційних властивостей.

Основні виводи по розділу

1. Запропонована класифікація ниток-моделей і реальних ниток. Нитки-моделі I і II модифікації можна класифікувати по структурі, по фізико-механічним властивостям. Нитки-моделі першої модифікації, крім того, класифікуються за формою поперечного перетину.

2. З урахуванням зминання нитки в зоні контакту з направляючою визначені геометричні характеристики точок осі ниток що зминаються:

а) співвідношення між одиничними ортами натуральних тригранників в точках A і A^* мають вигляд

$$\vec{\tau}_* = \vec{\tau}_0 + \frac{\partial \vec{U}}{\partial S};\ \vec{\nu}_* = \vec{\nu}_0 \frac{\rho_1}{\rho_0} + \rho_1 \frac{\partial^2 \vec{U}}{\partial S^2};$$

$$\vec{\beta}_* = \vec{\beta}_0 \frac{\rho_1}{\rho_0} + \frac{\partial \vec{U}}{\partial S} \times \vec{\nu}_0 \frac{\rho_1}{\rho_0} + \vec{\tau}_0 \times \frac{\partial^2 \vec{U}}{\partial S^2} \rho_1 + \frac{\partial \vec{U}}{\partial S} \times \frac{\partial^2 \vec{U}}{\partial S_0^2} \rho_1.$$

б) умови на переміщеннях, виражені через складові вектора деформації, для точок осі нитки що зминається

$$\frac{\partial U_\tau}{\partial S} - q_0 U_\nu + p_0 U_\beta = 0;\ \frac{\partial U_\nu}{\partial S} + q_0 U_\tau - r_0 U_\beta = \beta;\ \frac{\partial U_\beta}{\partial S} - p_0 U_\tau + p_0 U_\nu = -\alpha.$$

в) головні компоненти кривизни осі нитки

$$q_1 = \frac{\cos\Psi_0}{\rho_0} + \frac{\partial\beta}{\partial S} + \frac{\alpha}{\rho_{01}} + \alpha\frac{\partial\Psi_0}{\partial S};$$

$$p_1 = \frac{\sin\Psi_0}{\rho_0} + \frac{\partial\alpha}{\partial S} - \frac{\beta}{\rho_{01}} - \beta\frac{\partial\Psi_0}{\partial S};$$

$$r_1 = \frac{1}{\rho_0} + \frac{\partial\Psi_0}{\partial S} - \alpha\frac{\cos\Psi_0}{\rho_0} + \frac{\sin\Psi_0}{\rho_0}\beta.$$

3. Визначені швидкості і прискорення і закони їх розподілу з урахуванням зминання в зоні контакту

$$\vec{V}_* = \vec{V} + \frac{\partial\vec{U}}{\partial t};\ \vec{W}_* = \frac{\partial\vec{V}_*}{\partial t} = \vec{\omega}_e \times \vec{V}_* + \frac{\partial\vec{V}_*}{\partial t};\ \frac{\partial\vec{V}_*}{\partial S} = \vec{\Omega}\times\vec{V_*} + \frac{\partial\vec{V}_*}{\partial S} = -\omega_2\vec{b}_* + \omega_3\vec{n}_*;$$

$$\vec{\Omega}\times(\vec{V}+\vec{U'}) + \frac{\partial(\vec{V}+\vec{U'})}{\partial S} = (\vec{\omega}_0 + \vec{\omega}_u)\times\vec{\tau}_*;$$

$$\frac{\partial\vec{W}_*}{\partial S} = \vec{\Omega}\times\vec{W_*} + \frac{\partial\tilde{W}_*}{\partial S} = \frac{\partial\vec{\omega}_e}{\partial t}\times\vec{\tau}_* + \vec{\omega}_e\times(\vec{\omega}_e\times\vec{\tau}_*).$$

Отримані відповідні проекції даних векторних рівнянь на осі головного тригранника.

4. Складена система диференціальних рівнянь, що описують рух елементу нитки по направляючої великої кривизни з урахуванням зминання, жорсткості на вигин і кручення нитки

$$\frac{\partial\vec{R}}{\partial S} + \overrightarrow{R_0} - \overrightarrow{\Phi_*} = 0;\ \frac{\partial\vec{M}}{\partial S} + \vec{M}_0 + \vec{\tau}_*\times\vec{Q} - \vec{M}_\Phi = 0;\ \vec{\tau}_*\times\vec{Q} = -Q_3\vec{n}_* + Q_2\vec{b}_*;$$

$$\vec{M}_\Phi = \gamma_u\left\{\left[J_{\tau_0} \pm \Delta_\tau(\vec{U})\right]\varepsilon\vec{\tau}_* + \left[J_{n_0} \pm \Delta_n(\vec{U})\right]\varepsilon_2\vec{n}_* + \left[J_{b_0} \pm \Delta_b(\vec{U})\right]\varepsilon_3\vec{b}_*\right\};$$

$$\frac{\partial\vec{R}}{\partial S} = \left(\frac{\partial P}{\partial S} - Q_2 q_1 + Q_3 p_1\right)\vec{\tau}_* + \left(\frac{\partial Q_2}{\partial S} + P\,q_1 + Q_3 r_1\right)\vec{n}_* + \left(\frac{\partial Q_3}{\partial S} - P\,p_1 + Q_2 r_1\right)\vec{b}_*;$$

$$\frac{\partial\vec{M}}{\partial S} = \left(\frac{\partial M\kappa}{\partial S} - M_{u2}q_1 + M_{u3}p_1\right)\vec{\tau}_* + \left(\frac{\partial M_{u2}}{\partial S} + M_k q_1 + M_{u3}r_1\right)\vec{n}_* +$$

$$+\left(\frac{\partial M_{u3}}{\partial S} - M_k p_1 + M_{u2}r_1\right)\vec{b}_*;$$

$$\vec{\Phi}_* = T\vec{W}_* = T\left(\frac{\partial\tilde{V}_*}{\partial t} + \vec{\omega}_e\times\vec{V}_*\right);\ N_i = b_i E_{i1}\delta_i(1 - b_{i3}\delta_i^{\,b_{i4}}) + \eta_i\dot{\delta}_i^{\,b_{i6}}(1 - b_{i\,5}\dot{\delta}_i^{\,b_{i6}});\ i = 1,2,3.$$

5. Проведено теоретичне дослідження процесу взаємодії нитки з направляючою поверхнею з урахуванням деформації останньої. Визначені основні геометричні і кінематичні параметри точок осі нитки.

6. З використанням ниток-моделей II модифікації розглянутий випадок руху по направляючій поверхні малої кривизни. Отриманий вираз

для натягу ведучої гілки нитки з урахуванням зменшення кута обхвату направляючої за рахунок зминання в зоні контакту.

7. Всі реальні нитки володіють певними дефектами і пороками. В процесі переробки на технологічному устаткуванні натяг ниток зростає і досягає своєї максимальної величини в робочій зоні. Обрив нитки наступає тоді, коли за рахунок збільшення натягу (в результаті тертя, зминання в зоні контакту з направляючою) напруга в небезпечному перетині (місці розриву окремих філаментів, стоншування) досягне свого критичного значення.

2. СТРУКТУРА КОМП'ЮТЕРНОЇ ПРОГРАМИ КІНЕМАТИЧНОГО ТА КІНЕТОСТАТИЧНОГО АНАЛІЗУ ПЛОСКИХ МЕХАНІЗМІВ

Кінематичний та динамічний аналіз плоских механізмів виграє значну роль при проектуванні нових механізмів та модернізації існуючих. Результати, які отримуються при проведенні даних досліджень, можна використовувати при розрахунку міцності окремих ланок, їх інерційних характеристик, оптимізації конструктивних параметрів механізмів, мінімізації споживаної енергії.

На рис.2.1 представлена початкова форма TPMForm1 = class(TForm) програми KDAM для кінематичного та кінетостатичного аналізу плоских важільних механізмів. Розробником програми KDAM є лауреат Державної премії України в галузі науки і техніки, доктор технічних наук, професор Щербань В.Ю. Перша та друга версія програми KDAM включали в себе модулі для розрахунку кінематичних параметрів для механізмів, які включали в себе шатунно-повзункову та шатунно-коромислову групи Асура. Третя версія програми KDAM включала в себе додатково кінетостатичний аналіз механізмів, які включали в себе шатунно-повзункову та шатунно-коромислову групи Асура. Четверта версія програми KDAM включала в себе додатково кінематичний аналіз механізмів, які включали в себе кулісну групу. Остання 5.0 версія програми KDAM включала в себе кінематичний та кінетостатичний аналіз механізмів, які включали в себе шатунно-повзункову, шатунно-коромислову та кулісну групи Асура.

На формі TPMForm1 = class(TForm) (рис.2.2) розташовані наступні компоненти PMLabel1: TLabel; PMLabel2: TLabel; PMLabel3: TLabel; PMLabel4: TLabel; PMLabel5: TLabel; PMImage1: TImage; PMButton1: TButton; Label1: TLabel. Компонент PMLabel1: TLabel включає назву програми. Компонент Label1: TLabel включає бренд програми KDAM.

Рис.2.1

Компоненти PMLabel2: TLabel, PMLabel3: TLabel включають інформацію про розробника програми. Компонент PMLabel4: TLabel включає версію програми. Компонент TLabel; PMLabel5 включає інформацію про рік та місце підготовки останньої версії програми KDAM.

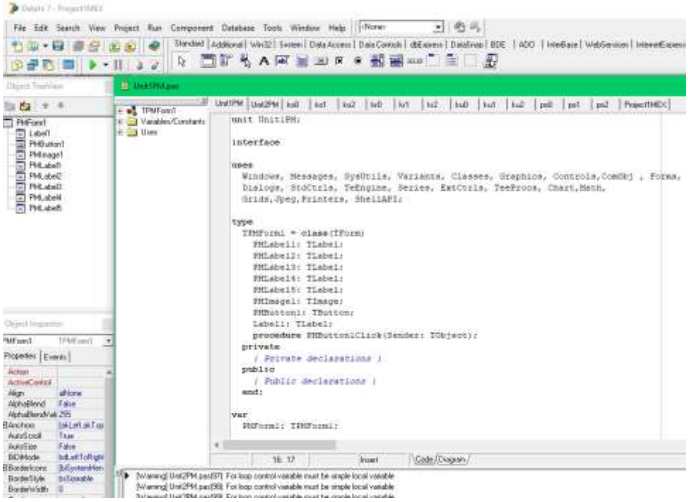


Рис.2.2

Компонент PMButton1: TButton ініціює виконання процедури procedure TPMForm2.PMButton1Click(Sender: TObject) переходу з форми TPMForm1 = class(TForm) до форми TPMForm2 = class(TForm).

Друга форма TPMForm2 = class(TForm) є основною формою програми KDAM і представлена на рис.2.3.

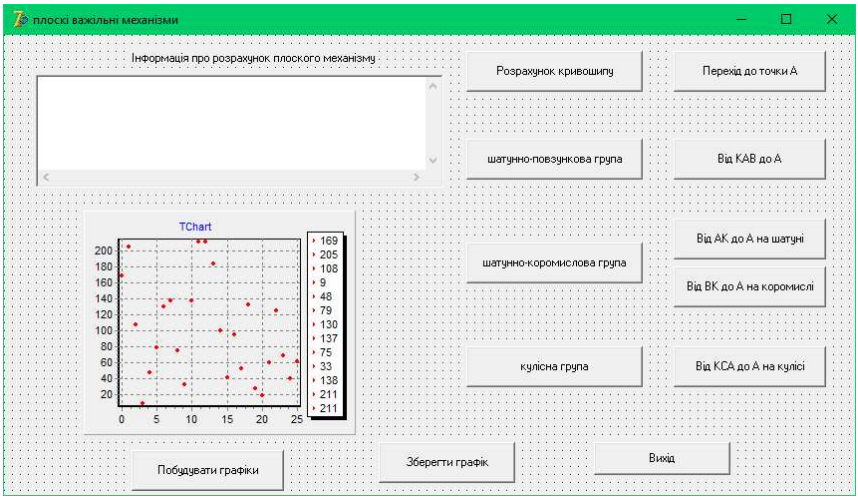


Рис.2.3

Форма TPMForm2 = class(TForm) включає в себе доступ до розрахунку ведучої ланки механізму, розрахунку шатунно-повзункової групи Асура, розрахунку шатунно-коромислової групи Асура, розрахунку кулісної групи Асура. Використання рекурсивного підходу дозволяє послідовне приєднання груп Асура до механізму і виконання кінематичних та динамічних розрахунків.

На формі TPMForm2 = class(TForm) (рис.2.4) розташовані наступні компоненти Button1: TButton, Button2: TButton, Button3: TButton, Memo1: TMemo, Label1: TLabel, Button4: TButton, Button5: TButton, Button6: TButton, Chart1: TChart, Button7: TButton, Series1: TPointSeries, Button8: TButton, Button9: TButton, Button10: TButton, Button11: TButton, Button12: TButton.

Компонент Button2: TButton ініціює виконання процедури procedure Button2Click(Sender: TObject) переходу з форми TPMForm2 = class(TForm) до модуля unit Unit2PM з формою TFormKr0 = class(Tform) для кінематичного та кінетостатичного розрахунку ведучої ланки механізму.

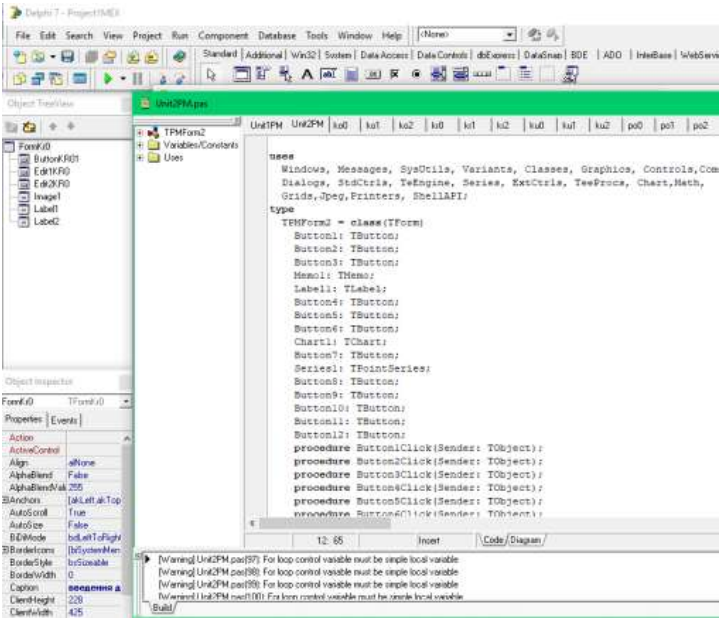


Рис.2.4

Компонент Button6: TButton ініціює виконання процедури procedure Button6Click(Sender: TObject)

```
{
memo2.Lines.Add('Перейменовано кінематичні параметри від 5A до A');
for i:=0 to 360 do mxA[i]:=mx5A[i];
for i:=0 to 360 do myA[i]:=my5A[i];
for i:=0 to 360 do mvAx[i]:=mv5Ax[i];
for i:=0 to 360 do mvAy[i]:=mv5Ay[i];
for i:=0 to 360 do mwAx[i]:=mw5Ax[i];
}
```

присвоювання значень кінематичних та динамічних параметрів точки A5 кінця кривошипу точці A. Компонент Button7: TButton ініціює виконання процедури procedure Button7Click(Sender: TObject)

```
{
begin
 Chart2.SeriesList[0].Clear;
 for i:=0 to 360 do
 begin
 Chart2.SeriesList[0].AddXY(mxA[i],myA[i],'',clRed);
 end;
}
```

побудови графіку траєкторії точки A. Багато строковий компонент для введення / виведення даних Memo1: TMemo призначений для виведення інформації про розрахунки складових плоского механізму. Компонент Button12: TButton ініціює виконання процедури procedure Button12Click(Sender: TObject) зберігання графіку траєкторії точки A.

Група компонентів Button3: TButton, Button4: TButton та Button5: TButton призначена для додавання груп Асура до ведучої ланки. Компонент Button3: TButton ініціює виконання процедури procedure Button3Click(Sender: TObject) переходу до модуля unit po0 додавання шатунно-повзункової групи Асура до механізму. Модуль unit po0 включає форму TFormPo0 = class(TForm) для обрання номера схеми та введення вхідних даних. Компонент Button8: TButton ініціює виконання процедури procedure Button8Click(Sender: TObject) присвоювання значень кінематичних та динамічних параметрів точки KAB на шатуні початковій точці A наступної групи Асура, яка може бути приєднана до механізму.

Компонент Button4: TButton ініціює виконання процедури procedure Button4Click(Sender: TObject) переходу до модуля unit ko0 додавання шатунно-коромислової групи Асура до механізму. Модуль unit ko0

включає форму TFormko0 = class(TForm) для обрання номера схеми та введення вхідних даних для шатунно-коромислової групи. Компонент Button9: TButton ініціює виконання процедури procedure Button9Click(Sender: TObject) присвоювання значень кінематичних та динамічних параметрів точки АК на шатуні початковій точці А наступної групи Асура, яка може бути приєднана до механізму. У випадку приєднання наступної групи Асура до кривошипу використовується компонент Button10: TButton, який ініціює виконання процедури procedure Button10Click(Sender: TObject) присвоювання значень кінематичних та динамічних параметрів точки ВК на коромислі початковій точці А наступної групи Асура.

Компонент Button5: TButton ініціює виконання процедури procedure Button5Click(Sender: TObject) переходу до модуля unit ku0 додавання кулісної групи Асура до механізму. Модуль unit ku0 включає форму TFormku0 = class(TForm) для обрання номера схеми та введення вхідних даних для кулісної групи для розрахунку. Компонент Button11: TButton ініціює виконання процедури procedure Button11Click(Sender: TObject) присвоювання значень кінематичних та динамічних параметрів точки КСА на кулісі початковій точці А наступної групи Асура, яка може бути приєднана до механізму.

Компонент Button1: TButton ініціює виконання процедури procedure Button1Click(Sender: TObject) закриття форми TPMForm2 = class(TForm) та виходу з програми.

Активація модуля unit kr0 процедурою procedure Button2Click(Sender: TObject)

```
{
memo2.Lines.Add('Доданий кривошип');
PMForm2.Hide;
Formkr0.Show;
```

}

викликає форму TFormKr0 = class(TForm), яка представлена на рис.2.5.

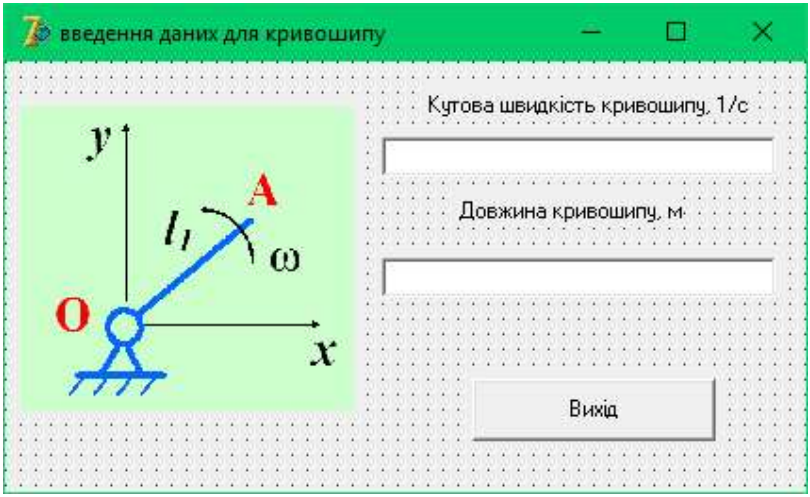


Рис.2.5

Форма TFormKr0 = class(TForm) включає в себе доступ до розрахунку кінематичних та динамічних розрахунків ведучої ланки механізму. На формі TFormKr0 = class(TForm) (рис.2.6) розташовані наступні компоненти ButtonKR01: TButton, Edit1KR0: TEdit, Edit2KR0: TEdit, Label1: TLabel, Label2: TLabel, Image1: TImage.

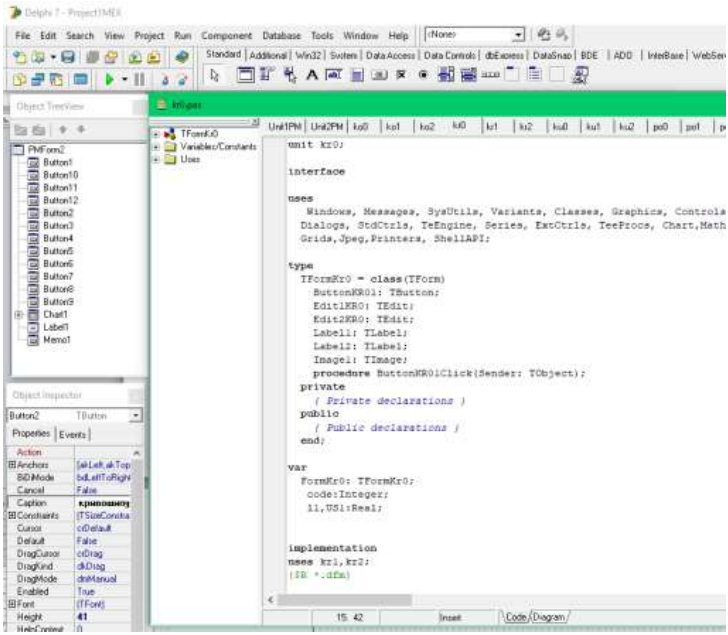


Рис.2.6

Компонент ButtonKR01: TButton ініціює виконання процедури procedure ButtonKR01Click(Sender: TObject) переходу з форми TFormKr0 = class(TForm) до модуля unit kr1 з формою TFormKr1 = class(TForm) для кінематичного розрахунку ведучої ланки механізму. Компонент Edit1KR0: TEdit представляє одно строковий компонент для введення кутової швидкості кривошипу. Компонент Edit2KR0: TEdit представляє одно строковий компонент для введення довжини кривошипу.

На рич.2.7 представлена форма TFormKr1 = class(TForm) для кінематичного розрахунку ведучої ланки механізму. На формі TFormKr1 = class(TForm) (рис.2.8) представлені наступні компоненти btnkrpo: TButton, Chart1: Tchart, Series1: TlineSeries, Series2: TlineSeries, Series3: TlineSeries, Series4: TlineSeries, Button1ks: Tbutton, Button2ks: Tbutton, Chart2: Tchart, Chart4: Tchart, Series8: TlineSeries, Series10: TlineSeries, Series11: TlineSeries, Series12: TlineSeries, Series13: TlineSeries, Series14: TlineSeries,

Series15: TlineSeries, Series16: TlineSeries, Series17: TlineSeries, StringGrid1: TstringGrid, Button1: Tbutton, Button2: Tbutton, Chart3: Tchart.

Компонент Button1ks: Tbutton на формі TFormKr1 = class(TForm) призначений для ініціалізації процедури procedure Button1ksClick(Sender: TObject)

```
{
 begin
        StringGrid2.cells[0,0]:='U1';
        StringGrid2.cells[1,0]:='x5A';
        StringGrid2.cells[2,0]:='y5A';
        StringGrid2.cells[3,0]:='x50A';
        StringGrid2.cells[4,0]:='y50A';
        StringGrid2.cells[5,0]:='v5Ax';
        StringGrid2.cells[6,0]:='v5Ay';
        StringGrid2.cells[7,0]:='v50Ax';
        StringGrid2.cells[8,0]:='v50Ay';
        StringGrid2.cells[9,0]:='w5Ax';
        StringGrid2.cells[10,0]:='w5Ay';
        StringGrid2.cells[11,0]:='w50Ax';
        StringGrid2.cells[12,0]:='w50Ay';
          end;
        for i:=0 to 360 do
        begin
        x5A:=l1*cos(U1);
        y5A:=l1*sin(U1);
        x50A:=0.5*l1*cos(U1);
        y50A:=0.5* l1*sin(U1);
        v5Ax:= -US1*l1*sin(U1);
        v5Ay:=US1*l1*cos(U1);
```

```
v50Ax:= -0.5*US1*l1*sin(U1);
v50Ay:=0.5*US1*l1*cos(U1);
w5Ax:=-sqr(US1)*l1*cos(U1);
w5Ay:=-sqr(US1)*l1*sin(U1);
w50Ay:=-0.5*sqr(US1)*l1*sin(U1);
mU1[i]:=U1;
mx5A[i]:=x5A;
my5A[i]:=y5A;
mx50A[i]:=x50A;
my50A[i]:=y50A;
mv5Ax[i]:=v5Ax;
mv5Ay[i]:=v5Ay;
mv50Ax[i]:=v50Ax;
mv50Ay[i]:=v50Ay;
mw5Ax[i]:=w5Ax;
mw5Ay[i]:=w5Ay;
mw50Ax[i]:=w50Ax;
mw50Ay[i]:=w50Ay;
end;
    for n:=0 to 360 do
        begin
  StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
  StringGrid2.cells[1,n+1]:=format('%9.4f',[mx5A[n]]);
  StringGrid2.cells[2,n+1]:=format('%9.4f',[my5A[n]]);
  StringGrid2.cells[3,n+1]:=format('%9.4f',[mx50A[n]]);
  StringGrid2.cells[4,n+1]:=format('%9.4f',[my50A[n]]);
  StringGrid2.cells[5,n+1]:=format('%9.4f',[mv5Ax[n]]);
  StringGrid2.cells[6,n+1]:=format('%9.4f',[mv5Ay[n]]);
  StringGrid2.cells[7,n+1]:=format('%9.4f',[mv50Ax[n]]);
```

```
        StringGrid2.cells[8,n+1]:=format('%9.4f',[mv50Ay[n]]);
        StringGrid2.cells[9,n+1]:=format('%9.4f',[mw5Ax[n]]);
        StringGrid2.cells[10,n+1]:=format('%9.4f',[mw5Ay[n]]);
        StringGrid2.cells[11,n+1]:=format('%9.4f',[mw50Ax[n]]);
        StringGrid2.cells[12,n+1]:=format('%9.4f',[mw50Ay[n]]);
}
```

розрахунку координат, проекцій на осі x та y векторів швидкості та прискорення точки кінця та центру мас ведучої ланки механізму. Формування таблиці відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

Компонент Button2ks: Tbutton ініціює виконання процедури

```
procedure Button2ksClick(Sender: TObject)
{
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mx5A[j],",clRed);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,my5A[j],",clGreen);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mx50A[j],",clYellow);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,my50A[j],",clBlue);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,US1,",clWhite);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mv5Ax[j],",clRed);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mv5Ay[j],",clGreen);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mv50Ax[j],",clYellow);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mv50Ay[j],",clBlue);
Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mw5Ax[j],",clRed);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mw5Ay[j],",clGreen);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mw50Ax[j],",clYellow);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mw50Ay[j],",clBlue);
}
```

для побудови графіків на компонентах Chart1: Tchart, Chart2: Tchart, Chart3: Tchart та Chart4: Tchart. Компоненти Chart1: Tchart, Chart2: Tchart,

Chart3: Tchart та Chart4: Tchart призначені для виводу графічних зображень координат, проекцій на осі x та y векторів швидкості та прискорення точки кінця та центру мас ведучої ланки механізму. На компоненті Chart1: Tchart відображаються переміщення точок. На компоненті Chart2: Tchart відображається кутова швидкість кривошипу. На компоненті Chart3: Tchart відображаються проекції на осі x та y векторів швидкості точки кінця та центру мас ведучої ланки механізму. На компоненті Chart4: Tchart відображаються проекції на осі x та y векторів прискорення точки кінця та центру мас ведучої ланки механізму.

Компонент Button2ks: Tbutton викликає процедуру procedure Button2Click(Sender: TObject) для друку результатів розрахунку даних переміщення точок, кутової швидкості кривошипу, проекцій на осі x та y векторів швидкості точки кінця та центру мас ведучої ланки механізму, проекцій на осі x та y векторів прискорення точки кінця та центру мас ведучої ланки механізму з таблиці, з використанням компоненту StringGrid, в таблиці Excel.

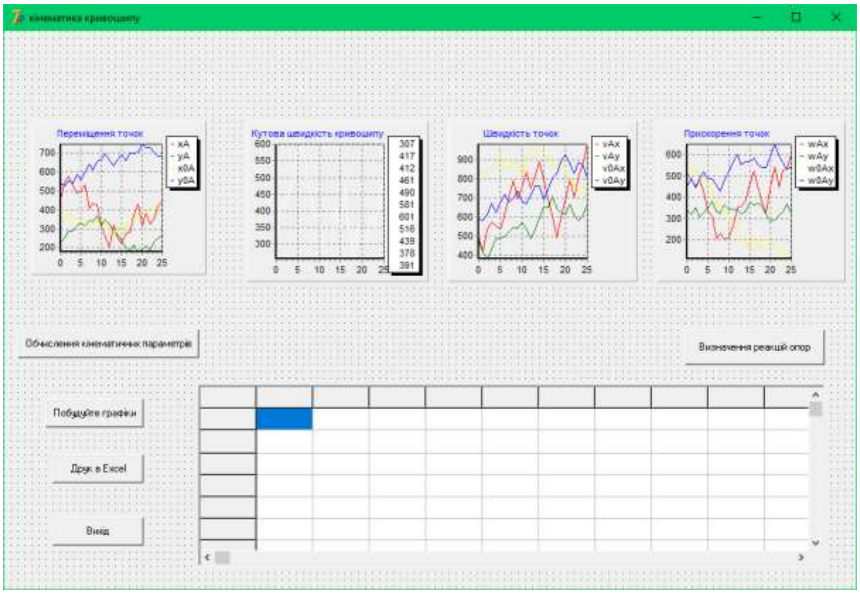

Кінематика кривошипу
Перемещення точок
Кутова швидкість кривошипу
Швидкість точок
Прискорення точок
Обчислення кінематичних параметрів
Визначення реакцій опор
Побудуйте графіки
Друк в Excel
Вихід

Рис.2.7

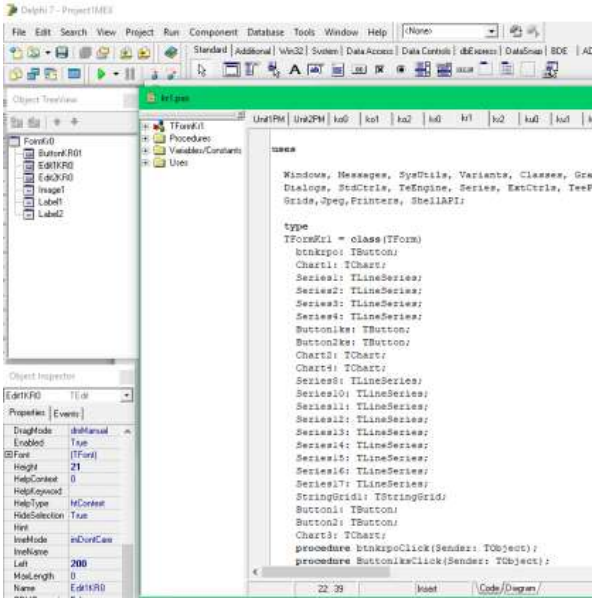


Рис.2.8

Компонент btnkrpo: Tbutton ініціює процедуру procedure btnkrpoClick(Sender: TObject) згортання форми TFormKr1 = class(TForm) та активації форми TPMForm2 = class(TForm).

Компонент Button1: Tbutton ініціює процедуру procedure Button1Click(Sender: TObject) згортання форми TFormKr1 = class(TForm) та активації форми TFormKr2 = class(TForm) для проведення кінетостатичного дослідження для визначення реакцій в шарнірах з використанням принципу Даламбера (рис.2.9).

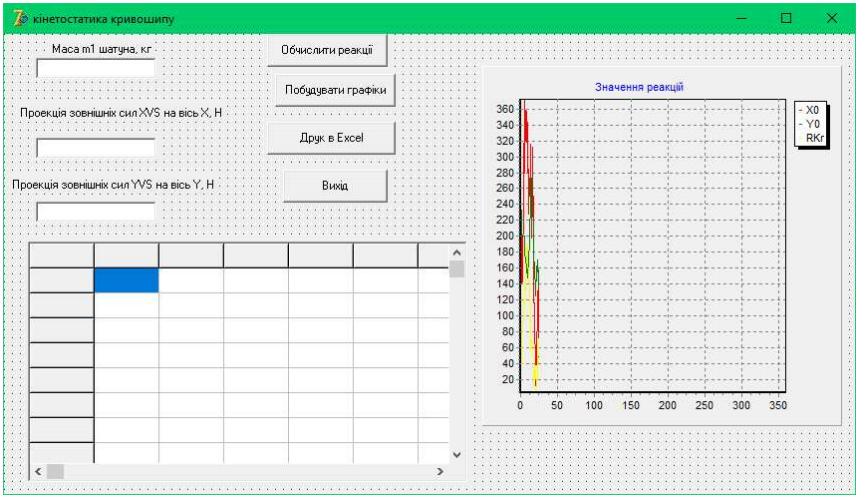


Рис.2.9

На формі TFormKr2 = class(TForm) (рис.2.10) розташовані наступні компоненти TFormKr2 = class(TForm), Edit1: TEdit, Label1: TLabel, Button1: Tbutton, Button2: Tbutton, Button3: Tbutton, StringGrid1: TstringGrid, Chart1: Tchart, Series1: TlineSeries, Series2: TlineSeries, Series4: TlineSeries, Button4: Tbutton, Label2: TLabel, Edit2: TEdit, Label4: TLabel, Edit3: TEdit. Компонент Edit1: TEdit представляє одно строковий компонент для введення маси шатуна (в кг). Компонент Edit2: TEdit представляє одно строковий компонент для введення проекції головного вектору зовнішніх сил на вісь x (в Н). Компонент Edit3: TEdit представляє одно строковий

компонент для введення проекції головного вектору зовнішніх сил на вісь y (в Н).

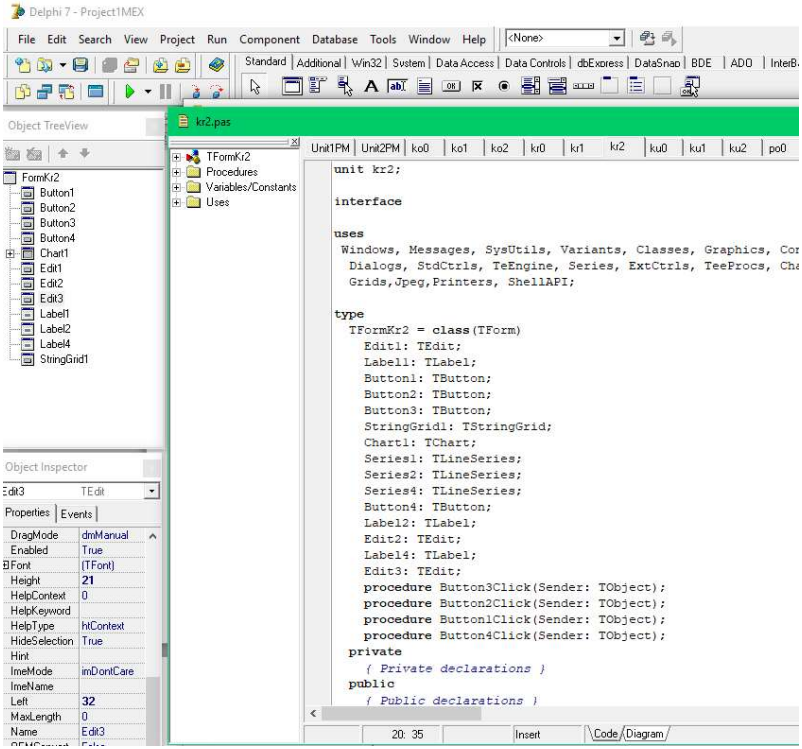


Рис.2.10

Компонент Button1: Tbutton на формі TFormKr2 = class(TForm) призначений для ініціалізації процедури procedure Button1Click(Sender: TObject)

```
{
    StringGrid2.cells[0,0]:='U1';
    StringGrid2.cells[1,0]:='XK0';
    StringGrid2.cells[2,0]:='YK0';
    StringGrid2.cells[3,0]:='RKr';
```

```
            end;
        for i:=0 to 360 do
        begin
        U1:=i*Pi/180;
        w550Ax:=mw50Ax[i];
        w550Ay:=mw50Ay[i];
        FKx:=m1*w550Ax;
        FKy:=m1*w550Ay;
        XK0:=-FKx+XVS;
        YK0:=-FKy+YVS;
        RKr:=sqrt(sqr(XK0)+sqr(YK0));
        mU1[i]:=U1;
        mXK0[i]:=XK0;
        mYK0[i]:=YK0;
        mRKr[i]:=RKr;
       end;
           for n:=0 to 360 do
               begin
          StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
          StringGrid2.cells[1,n+1]:=format('%12.4f',[mXK0[n]]);
          StringGrid2.cells[2,n+1]:=format('%12.4f',[mYK0[n]]);
          StringGrid2.cells[3,n+1]:=format('%12.4f',[mRKr[n]]);
}
```

розрахунку реакцій в шарнірах ведучої ланки механізму. Формування таблиці значень реакцій в шарнірах відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

Компонент Button2: Tbutton ініціює виконання процедури procedure Button2Click(Sender: TObject) для побудови графіків зміни реакцій в шарнірі ведучої ланки механізму на компоненті Chart1: Tchart. На

компоненті Chart1: Tchart відображаються проекції головного вектору шарніра ведучої ланки механізму в проекції на координатні осі x та y та модуль головного вектору реакції в шарнірі в залежності від кута обертання ведучої ланки навколо опори.

Компонент Button4: Tbutton викликає процедуру procedure Button4Click(Sender: TObject) для друку результатів розрахунку проекції головного вектору реакції шарніра ведучої ланки механізму в проекції на координатні осі x та y та модуль головного вектору реакції в шарнірі з таблиці, з використанням компоненту StringGrid, в таблиці Excel.

Компонент Button3: Tbutton ініціює процедуру procedure Button3Click(Sender: TObject) згортання форми TFormKr2 = class(TForm) та активації форми TFormKr1 = class(TForm).

Компонент Button4: Tbutton на формі TPMForm2 = class(TForm) ініціює виконання процедури procedure TPMForm2.Button4Click(Sender: TObject) переходу до модуля unit ko0 розрахунку механізму для шатунної коромислової групи Асура рис.2.11).

На формі TFormko0 = class(TForm) (рис.2.12) розташовані наступні компоненти TFormko0 = class(TForm), Button1: Tbutton, Image1: TImage, Image2: TImage, Label1: TLabel, Label2: TLabel, Label3: TLabel, Edit1: TEdit, Edit2: TEdit, Label4: TLabel, Edit3: TEdit, Label5: TLabel, Edit4: TEdit, Label6: TLabel, Edit5: TEdit, Label7: TLabel, Label8: TLabel, Label9: TLabel, Edit6: TEdit, Label10: TLabel, Edit7: TEdit, Label11: TLabel, Edit8: TEdit, Label12: TLabel, Edit9: TEdit, Label13: TLabel.

Компоненти Image1: TImage та Image2: TImage призначені для розміщення схем шатунної коромислової групи для 2 збирань. Перша схема збирання відповідає випадку, коли точка шарніра B розташовується вище точки шарніру C. На схемі представлені основні лінійні та кутові геометричні параметри. Друга схема збирання відповідає випадку, коли точка шарніра B розташовується нижче точки C шарніра. На схемі

представлені основні лінійні та кутові геометричні параметри для другої схеми.

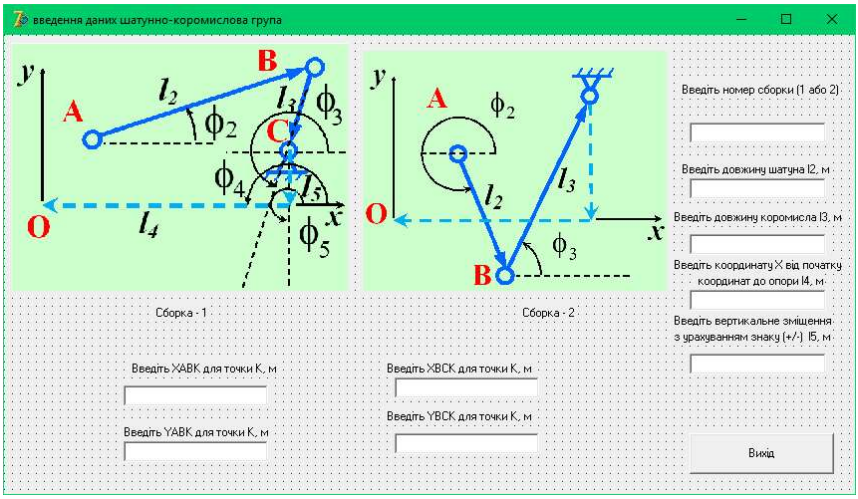


Рис.2.11

Компонент Edit1: TEdit представляє одно строковий компонент для завдання номеру збирання. Компонент Edit2: TEdit представляє одно строковий компонент для завдання довжини шатуна (в метрах). Компонент Edit3: TEdit представляє одно строковий компонент для завдання довжини коромисла (в метрах). Компонент Edit4: TEdit представляє одно строковий компонент для завдання координати x від початку координат до опори C. Початок координат розташовується в точці O шарніра ведучої ланки механізму. Компонент Edit5: TEdit представляє одно строковий компонент для завдання вертикального зміщення точки C шарніра по вертикалі. Знак «+» обирається у випадку коли напрям вектору l_5 та його проекція на вісь y буде позитивною. Знак «-»обирається у випадку коли напрям вектору l5 та його проекція на вісь y буде від'ємними.

Компонент Edit8: TEdit представляє одно строковий компонент для завдання відстані від точки A шатуна до точки K розташування центра його мас вздовж осі AB. Компонент Edit9: TEdit представляє одно

строковий компонент для завдання довжини перпендикуляру від точки К (розташування центра мас шатуна АВ) до лінії АВ шатуна.

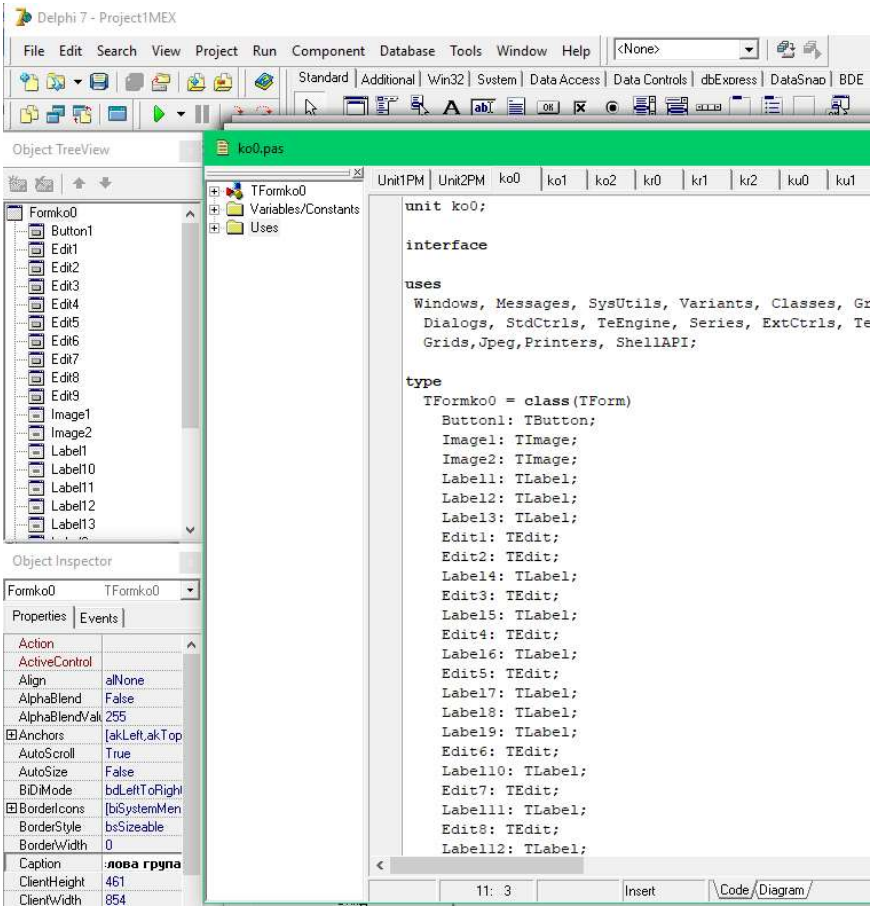


Рис.2.12

Компонент Edit6: TEdit представляє одно строковий компонент для завдання відстані від точки В коромисла до точки К розташування центра його мас вздовж осі ВС (в метрах). Компонент Edit7: TEdit представляє одно строковий компонент для завдання довжини перпендикуляру від точки К (розташування центра мас коромисла ВС) до лінії ВС коромисла.

Компонент Button1: Tbutton ініціює виконання процедури procedure Button1Click(Sender: TObject) переходу з форми TFormko0 = class(TForm) до модуля unit ko1 з формою TFormko1 = class(TForm) для кінематичного розрахунку шатунної коромислової групи механізму.

На рис.2.13 представлена форма TFormko1 = class(TForm) для кінематичного розрахунку шатунної коромислової групи механізму. На формі TFormko1 = class(TForm) (рис.2.14) представлені наступні компоненти TFormko1 = class(TForm), btnkrpo: Tbutton, Chart1: Tchart, Series3: TlineSeries, Series4: TlineSeries, Series5: TlineSeries, Button1ks: Tbutton, Button2ks: Tbutton, Series6: TlineSeries, Chart2: Tchart, Chart3: Tchart, Chart4: Tchart, Series8: TlineSeries, Series7: TlineSeries, Series9: TlineSeries, StringGrid1: TstringGrid, Series22: TlineSeries, Series23: TlineSeries, Series24: TlineSeries, Series25: TlineSeries, Series26: TlineSeries, Series12: TlineSeries, Series13: TlineSeries, Series27: TlineSeries, Series28: TlineSeries, Series29: TlineSeries, Series30: TlineSeries, Series16: TlineSeries, Series17: TlineSeries, Series18: TlineSeries, Series19: TlineSeries, Series20: TlineSeries, Series21: TlineSeries, Button1: Tbutton, Button2: Tbutton, Series1: TlineSeries, Series2: TlineSeries, Series10: TlineSeries, Series11: TlineSeries, Series14: TlineSeries, Series15: TlineSeries, Series31: TlineSeries, Series32: TlineSeries, Series33: TlineSeries, Series34: TlineSeries, Series35: TlineSeries, Series36: TlineSeries, Chart5: Tchart, Series37: TPointSeries, Series38: TPointSeries, Series39: TPointSeries, Button3: Tbutton.

Компонент Button1ks: Tbutton на формі TFormko1 = class(TForm) призначений для ініціалізації процедури procedure Button1ksClick(Sender: TObject)

```
{
    begin
   StringGrid2.cells[0,0]:='U1';
   StringGrid2.cells[1,0]:='U2';
```

```
StringGrid2.cells[2,0]:='U3';
StringGrid2.cells[3,0]:='xB';
StringGrid2.cells[4,0]:='yB';
StringGrid2.cells[5,0]:='xAB';
StringGrid2.cells[6,0]:='yAB';
StringGrid2.cells[7,0]:='xCB';
StringGrid2.cells[8,0]:='yCB';
StringGrid2.cells[9,0]:='US2';
StringGrid2.cells[10,0]:='US3';
StringGrid2.cells[11,0]:='UU2';
StringGrid2.cells[12,0]:='UU3';
StringGrid2.cells[13,0]:='vBx';
StringGrid2.cells[14,0]:='vBy';
StringGrid2.cells[15,0]:='vABx';
StringGrid2.cells[16,0]:='vABy';
StringGrid2.cells[17,0]:='vCBx';
StringGrid2.cells[18,0]:='vCBy';
StringGrid2.cells[19,0]:='wBx';
StringGrid2.cells[20,0]:='wBy';
StringGrid2.cells[21,0]:='wABx';
StringGrid2.cells[22,0]:='wABy';
StringGrid2.cells[23,0]:='wCBx';
StringGrid2.cells[24,0]:='wCBy';
StringGrid2.cells[25,0]:='XAK';
StringGrid2.cells[26,0]:='YAK';
StringGrid2.cells[27,0]:='VXAK';
StringGrid2.cells[28,0]:='VYAK';
StringGrid2.cells[29,0]:='WXAK';
StringGrid2.cells[30,0]:='WYAK';
```

```
        StringGrid2.cells[31,0]:='XBK';
        StringGrid2.cells[32,0]:='YBK';
        StringGrid2.cells[33,0]:='VXBK';
        StringGrid2.cells[34,0]:='VYBK';
        StringGrid2.cells[35,0]:='WXBK';
        StringGrid2.cells[36,0]:='WYBK';
                end;
         for i:=0 to 360 do
begin
          U1:=i*Pi/180;
         if l4-mxA[i]<>0 then
            begin
         A1:=arcTan((myA[i]-l5)/(l4-mxA[i]));
            end
                 else
                          begin
          if myA[i]>0     then
            begin
          A1:=Pi/2;
            end
                    else
                      begin
          if myA[i]<0 then
          begin
          A1:=3*Pi/2;
          end
                   else
                 begin
                A1:=0;
```

```
        end;
              end;
                   end;
 if A1<>0 then
     begin
  AC:=(myA[i]-l5)/sin(A1);
     end
         else
     begin
     AC:=l4-mxA[i];
     end;
p:=(l2+l3+AC)/2;
r:=sqrt((p-l2)*(p-l3)*(p-AC)/p);
A2:=2*arcTan(r/(p-l2));
 if nz<2  then
   begin
    U3:=2*Pi-(A1+A2);
   end
        else
     begin
   U3:=A2-A1;
     end;
xB:=l4-l3*cos(U3);
yB:=l5-l3*sin(U3);
 if xB>mxA[i]   then
   begin
  if nz<2 then
     begin
```

```
  U2:=arcTan(sqrt(sqr(l5-l3*sin(U3)-myA[i]))/sqrt(sqr(l4-l3*cos(U3)-
mxA[i])));
            end
                else
            begin
  U2:=2*Pi-arcTan(sqrt(sqr(l5-l3*sin(U3)-myA[i]))/sqrt(sqr(l4-l3*cos(U3)-
mxA[i])));
            end;
           end
                  else
                      begin
                     if xB<mxA[i] then
                        begin
  U2:=Pi+arcTan(sqrt(sqr(l5-l3*sin(U3)-myA[i]))/sqrt(sqr(l4-l3*cos(U3)-
mxA[i])));
                        end
                             else
                        begin
                      if nz<2 then
                        begin
                       U2:=Pi/2;
                        end
                            else
                        begin
                        U2:=3*Pi/2;
                        end;
                         end;
                       end;
       xAB:=mxA[i]+(l2/2)*cos(U2); yAB:=myA[i]+(l2/2)*sin(U2);
```

```
xCB:=l4-(l3/2)*cos(U3); yCB:=-(l3/2)*sin(U3)+l5;
US3:=(-mvAx[i]*cos(U2)-mvAy[i]*sin(U2))/(l3*(cos(U3)*sin(U2)-
cos(U2)*sin(U3)));
US2:= (mvAx[i]/(l2*sin(U2)))-((US3*l3*sin(U3))/(l2*sin(U2)));
UU2:=                    (-l2*US2*US2*cos(U2)*cos(U3)-
l3*US3*US3*cos(U3)*sin(U3)+mwAx[i]*cos(U3)+mwAy[i]*sin(U3)-
l2*US2*US2*sin(U2)*sin(U3))/(l2*(sin(U2)*cos(U3)-cos(U2)*sin(u3)));
UU3:=(mwAx[i]-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2)-
l3*sqr(US3)*cos(U3))/(l3*sin(U3));
vBx:=l3*US3*sin(U3); vBy:=-l3*US3*cos(U3);
vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
vCBx:=(l3/2)*US3*sin(U3); vCBy:=-(l3/2)*US3*cos(U3);
wBx:=l3*UU3*sin(U3)+l3*sqr(US3)*cos(U3);
wBy:=-l3*UU3*cos(U3)+l3*sqr(US3)*sin(U3);
wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
wCBx:=(l3/2)*UU3*sin(U3)+(l3/2)*sqr(US3)*cos(U3);
wCBy:=-(l3/2)*UU3*cos(U3)+(l3/2)*sqr(US3)*sin(U3);
 SA:=-XABK*sin(U2)-YABK*cos(U2);
 SB:=XABK*cos(U2)-YABK*sin(U2);
 XAK:=mxA[i]+SB;
 YAK:=myA[i]-SA;
 VXAK:=mvAx[i]+US2*SA;
 VYAK:=mvAy[i]+US2*SB;
 WXAK:=mwAx[i]+UU2*SA-US2*US2*SB;
 WYAK:=mwAy[i]+UU2*SB+US2*US2*SA;
CK:=-XBCK*sin(U3)-YBCK*cos(U3);
CK1:=XBCK*cos(U3)-YBCK*sin(U3);
```

```
        XBK:=xB+CK1;
        YBK:=yB-CK;
        VXBK:=vBx+CK*US3;
        VYBK:=vBy+CK1*US3;
        WXBK:=wBx+UU3*CK-US3*US3*CK1;
        WYBK:=wBy+UU3*CK1+US3*US3*CK;
mU1[i]:=U1;
         mU2[i]:=U2;
         mU3[i]:=U3;
         mxB[i]:=xB;
         myB[i]:=yB;
         mxAB[i]:=xAB;
         myAB[i]:=yAB;
         mxCB[i]:=xCB;
         myCB[i]:=yCB;
         mUS2[i]:=US2;
         mUS3[i]:=US3;
         mUU2[i]:=UU2;
         mUU3[i]:=UU3;
         mvBx[i]:=vBx;
         mvBy[i]:=vBy;
         mvABx[i]:=vABx;
         mvABy[i]:=vABy;
         mvCBx[i]:=vCBx;
         mvCBy[i]:=vCBy;
         mwBx[i]:=wBx;
         mwBy[i]:=wBy;
         mwABx[i]:=wABx;
         mwABy[i]:=wCBx;
```

```
        mwCBx[i]:=wCBy;
        mwCBy[i]:=wCBy;
        mXAK[i]:=XAK;
        mYAK[i]:=YAK;
        mVXAK[i]:=VXAK;
        mVYAK[i]:=VYAK;
        mWXAK[i]:=WXAK;
        mWYAK[i]:=WYAK;
        mXBK[i]:=XBK;
        mYBK[i]:=YBK;
        mVXBK[i]:=VXBK;
        mVYBK[i]:=VYBK;
        mWXBK[i]:=WXBK;
        mWYBK[i]:=WYBK;
end;
            for n:=0 to 360 do
              begin
          StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
          StringGrid2.cells[1,n+1]:=format('%9.2f',[mU2[n]]);
          StringGrid2.cells[2,n+1]:=format('%9.4f',[mU3[n]]);
          StringGrid2.cells[3,n+1]:=format('%9.4f',[mxB[n]]);
          StringGrid2.cells[4,n+1]:= format('%9.4f',[myB[n]]);
          StringGrid2.cells[5,n+1]:= format('%9.4f',[mxAB[n]]);
          StringGrid2.cells[6,n+1]:=format('%9.4f',[myAB[n]]);
          StringGrid2.cells[7,n+1]:=format('%9.4f',[mxCB[n]]);
          StringGrid2.cells[8,n+1]:=format('%9.4f',[myCB[n]]);
          StringGrid2.cells[9,n+1]:=format('%9.4f',[mUS2[n]]);
          StringGrid2.cells[10,n+1]:=format('%9.4f',[mUS3[n]]);
          StringGrid2.cells[11,n+1]:= format('%9.4f',[mUU2[n]]);
```

```
StringGrid2.cells[12,n+1]:=format('%9.4f',[mUU3[n]]);
StringGrid2.cells[13,n+1]:=format('%9.4f',[mvBx[n]]);
StringGrid2.cells[14,n+1]:=format('%9.4f',[mvBy[n]]);
StringGrid2.cells[15,n+1]:=format('%9.4f',[mvABx[n]]);
StringGrid2.cells[16,n+1]:=format('%9.4f',[mvABy[n]]);
StringGrid2.cells[17,n+1]:= format('%9.4f',[mvCBx[n]]);
StringGrid2.cells[18,n+1]:= format('%9.4f',[mvCBy[n]]);
StringGrid2.cells[19,n+1]:=format('%9.4f',[mwBx[n]]);
StringGrid2.cells[20,n+1]:=format('%9.4f',[mwBy[n]]);
StringGrid2.cells[21,n+1]:=format('%9.4f',[mwABx[n]]);
StringGrid2.cells[22,n+1]:=format('%9.4f',[mwABy[n]]);
StringGrid2.cells[23,n+1]:=format('%9.4f',[mwCBx[n]]);
StringGrid2.cells[24,n+1]:= format('%9.4f',[mwCBy[n]]);
StringGrid2.cells[25,n+1]:=format('%9.4f',[mXAK[n]]);
StringGrid2.cells[26,n+1]:=format('%9.4f',[mYAK[n]]);
StringGrid2.cells[27,n+1]:=format('%9.4f',[mVXAK[n]]);
StringGrid2.cells[28,n+1]:=format('%9.4f',[mVYAK[n]]);
StringGrid2.cells[29,n+1]:=format('%9.4f',[mWXAK[n]]);
StringGrid2.cells[30,n+1]:= format('%9.4f',[mWYAK[n]]);
StringGrid2.cells[31,n+1]:=format('%9.4f',[mXBK[n]]);
StringGrid2.cells[32,n+1]:=format('%9.4f',[mYBK[n]]);
StringGrid2.cells[33,n+1]:=format('%9.4f',[mVXBK[n]]);
StringGrid2.cells[34,n+1]:=format('%9.4f',[mVYBK[n]]);
StringGrid2.cells[35,n+1]:=format('%9.4f',[mWXBK[n]]);
StringGrid2.cells[36,n+1]:= format('%9.4f',[mWYBK[n]]);
        end;
}
```

розрахунку координат, проекцій на осі x та y векторів швидкості та прискорення точок шатунної коромислової групи механізму. Формування

таблиці відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

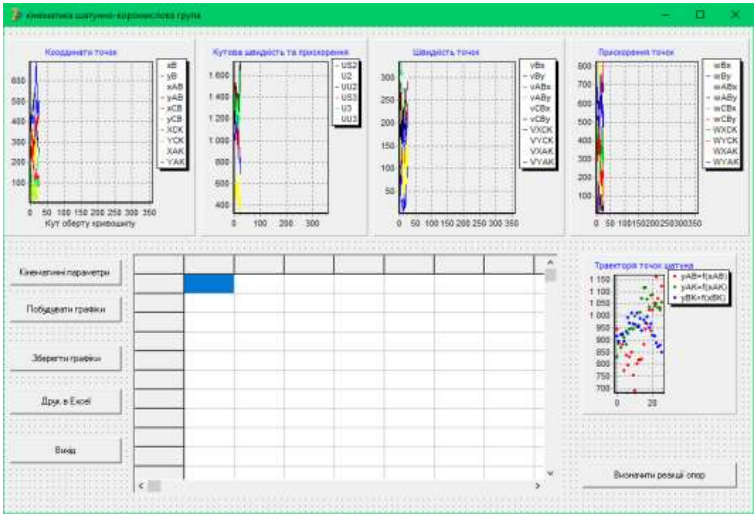


Рис.2.13

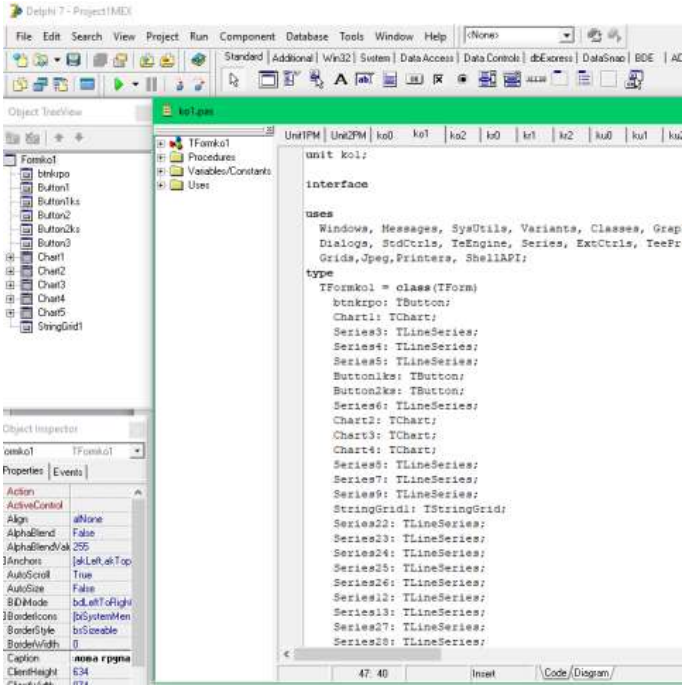


Рис.2.14

Компонент Button2ks: Tbutton ініціює виконання процедури

```
procedure Button2ksClick(Sender: TObject)
{
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart2.SeriesList[4].Clear;
Chart2.SeriesList[5].Clear;
Chart2.SeriesList[6].Clear;
Chart2.SeriesList[7].Clear;
Chart2.SeriesList[8].Clear;
```

```
Chart2.SeriesList[9].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart2.SeriesList[4].Clear;
Chart2.SeriesList[5].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
Chart3.SeriesList[4].Clear;
Chart3.SeriesList[5].Clear;
Chart3.SeriesList[6].Clear;
Chart3.SeriesList[7].Clear;
Chart3.SeriesList[8].Clear;
Chart3.SeriesList[9].Clear;
Chart4.SeriesList[0].Clear;
Chart4.SeriesList[1].Clear;
Chart4.SeriesList[2].Clear;
Chart4.SeriesList[3].Clear;
Chart4.SeriesList[4].Clear;
Chart4.SeriesList[5].Clear;
Chart4.SeriesList[6].Clear;
Chart4.SeriesList[7].Clear;
Chart4.SeriesList[8].Clear;
Chart4.SeriesList[9].Clear;
```

```
Chart5.SeriesList[0].Clear;
Chart5.SeriesList[1].Clear;
Chart5.SeriesList[2].Clear;
for j:=0 to 360 do
 begin
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mxB[j],",clYellow);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,myB[j],",clBlue);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mxAB[j],",clWhite);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,myAB[j],",clBlack);
Chart2.SeriesList[4].AddXY(mU1[j]*180/Pi,mxCB[j],",clRed);
Chart2.SeriesList[5].AddXY(mU1[j]*180/Pi,myCB[j],",clGreen);
Chart2.SeriesList[6].AddXY(mU1[j]*180/Pi,mXBK[j],",clRed);
Chart2.SeriesList[7].AddXY(mU1[j]*180/Pi,mYBK[j],",clGreen);
Chart2.SeriesList[8].AddXY(mU1[j]*180/Pi,mXAK[j],",clYellow);
Chart2.SeriesList[9].AddXY(mU1[j]*180/Pi,mYAK[j],",clBlue);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mUS2[j],",clBlue);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mU2[j],",clWhite);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mUU2[j],",clBlack);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,mUS3[j],",clRed);
Chart2.SeriesList[4].AddXY(mU1[j]*180/Pi,mU3[j],",clGreen);
Chart2.SeriesList[5].AddXY(mU1[j]*180/Pi,mUU3[j],",clYellow);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mvBx[j],",clYellow);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mvBy[j],",clBlue);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mvABx[j],",clRed);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mvABy[j],",clGreen);
Chart3.SeriesList[4].AddXY(mU1[j]*180/Pi,mvCBx[j],",clWhite);
Chart3.SeriesList[5].AddXY(mU1[j]*180/Pi,mvCBy[j],",clBlack);
Chart3.SeriesList[6].AddXY(mU1[j]*180/Pi,mVXBK[j],",clWhite);
Chart3.SeriesList[7].AddXY(mU1[j]*180/Pi,mVYBK[j],",clBlack);
```

```
Chart3.SeriesList[8].AddXY(mU1[j]*180/Pi,mVXAK[j],",clYellow);
Chart3.SeriesList[9].AddXY(mU1[j]*180/Pi,mVYAK[j],",clBlue);
Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mwBx[j],",clYellow);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mwBy[j],",clBlue);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mwABx[j],",clWhite);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mwABy[j],",clBlack);
Chart4.SeriesList[4].AddXY(mU1[j]*180/Pi,mwCBx[j],",clGreen);
Chart4.SeriesList[5].AddXY(mU1[j]*180/Pi,mwCBy[j],",clRed);
Chart4.SeriesList[6].AddXY(mU1[j]*180/Pi,mWXBK[j],",clGreen);
Chart4.SeriesList[7].AddXY(mU1[j]*180/Pi,mWYBK[j],",clRed);
Chart4.SeriesList[8].AddXY(mU1[j]*180/Pi,mWXAK[j],",clYellow);
Chart4.SeriesList[9].AddXY(mU1[j]*180/Pi,mWYAK[j],",clBlue);
Chart5.SeriesList[0].AddXY(mxAB[j],myAB[j],",clRed);
Chart5.SeriesList[1].AddXY(mXAK[j],mYAK[j],",clGreen);
Chart5.SeriesList[2].AddXY(mXBK[j],mYBK[j],",clBlue);
}
```

для побудови графіків на компонентах Chart1: Tchart, Chart2: Tchart, Chart3: Tchart , Chart4: Tchart та Chart5: Tchart шатунної коромислової групи механізму. Компоненти Chart1: Tchart, Chart2: Tchart, Chart3: Tchart , Chart4: Tchart та Chart5: Tchart призначені для виводу графічних зображень координат, проекцій на осі x та y векторів швидкості та прискорення точок та центрів мас шатунної коромислової групи механізму. На компоненті Chart1: Tchart відображаються переміщення точок шатунної коромислової групи механізму. На компоненті Chart2: Tchart відображається кутова швидкість та прискорення ланок шатунної коромислової групи механізму. На компоненті Chart3: Tchart відображаються проекції на осі x та y векторів швидкості точок та центрів мас ланок шатунної коромислової групи механізму. На компоненті Chart4: Tchart відображаються проекції на осі x та y векторів прискорення точок та центрів мас ланок шатунної

коромислової групи механізму. На компоненті Chart5: Tchart відображаються траєкторії точок шатуна.

Компонент Button2: Tbutton викликає процедуру procedure Button2Click(Sender: TObject) для друку результатів розрахунку даних переміщення точок, кутової швидкості та прискорення ланок шатунної коромислової групи, проекцій на осі x та y векторів швидкості точок та центру мас ланок шатунної коромислової групи, проекцій на осі x та y векторів прискорення точок та центрів мас ланок шатунної коромислової групи з таблиці, з використанням компоненту StringGrid, в таблиці Excel.

Компонент Button1: Tbutton ініціює процедуру procedure Button1ksClick(Sender: TObject) згортання форми TFormko1 = class(TForm) та активації форми TFormko2 = class(TForm) для проведення кінетостатичного дослідження для визначення реакцій в шарнірах шатунної коромислової групи з використанням принципу Даламбера (рис.2.15).

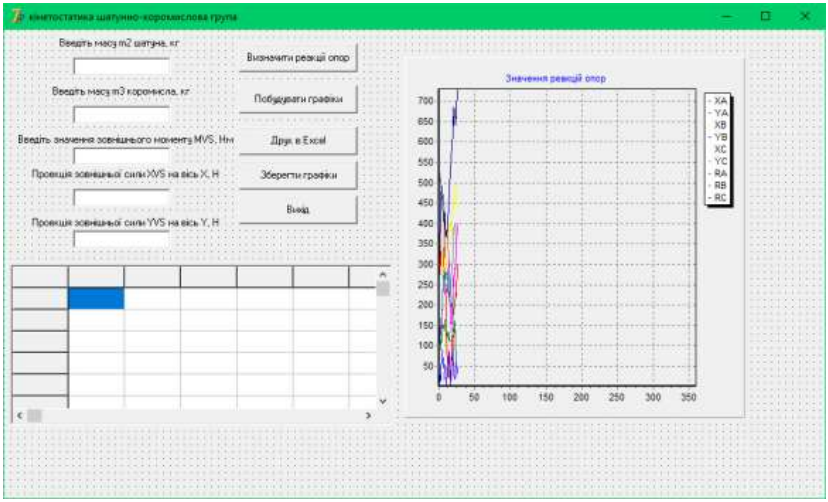


Рис.2.15

На формі TFormko2 = class(TForm) (рис.2.16) розташовані наступні компоненти TFormko2 = class(TForm), Edit1: TEdit, Label1: TLabel, Edit2:

TEdit, Label2: TLabel, Button1: Tbutton, Button2: Tbutton, Button3: Tbutton, StringGrid1: TstringGrid, Chart1: Tchart, Series1: TlineSeries, Series2: TlineSeries, Series3: TlineSeries, Series4: TlineSeries, Series5: TlineSeries, Series6: TlineSeries, Series7: TlineSeries, Series8: TlineSeries, Series9: TlineSeries, Button4: Tbutton, Edit3: TEdit, Label3: TLabel, Label4: TLabel, Edit4: TEdit, Label5: TLabel, Edit5: TEdit, Button5: TButton

Компонент Edit1: TEdit представляє одно строковий компонент для введення маси шатуна (в кг). Компонент Edit2: TEdit представляє одно строковий компонент для введення маси коромисла (в кг). Компонент Edit3: TEdit представляє одно строковий компонент для введення зовнішнього моменту (Нм). Компонент Edit4: TEdit представляє одно строковий компонент для введення проекції головного вектору зовнішніх сил на вісь x (в Н). Компонент Edit5: TEdit представляє одно строковий компонент для введення проекції головного вектору зовнішніх сил на вісь y (в Н).

Компонент Button1: Tbutton на формі TFormko2 = class(TForm) призначений для ініціалізації процедури procedure Button1Click(Sender: TObject)

```
{
        Val(Edit2.text,m2,code);
        Val(Edit2.text,m3,code);
        Val(Edit3.text,MVS,code);
        Val(Edit4.text,XVS,code);
        Val(Edit5.text,YVS,code);
          begin
        StringGrid2.cells[0,0]:='U1';
        StringGrid2.cells[1,0]:='XA';
        StringGrid2.cells[2,0]:='YA';
        StringGrid2.cells[3,0]:='XB';
```

```
        StringGrid2.cells[4,0]:='YB';
        StringGrid2.cells[5,0]:='XC';
        StringGrid2.cells[6,0]:='YC';
        StringGrid2.cells[7,0]:='RA';
        StringGrid2.cells[8,0]:='RB';
        StringGrid2.cells[9,0]:='RC';
              end;
        for i:=0 to 360 do
        begin
        I2:=m2*l2/12; I3:=m3*l3/12;
        wABx:=mwABx[i]; wABy:=mwABy[i];
        wCBx:=mwCBx[i]; wCBy:=mwCBy[i];
        FI2x:=m2*wABx;
        FI2y:=m2*wABy;
        FI3x:=m3*wCBx;
        FI3y:=m3*wCBy;
        l:=sqrt(sqr(l4-mxA[i])+sqr(myA[i]-l5));
        G:=arcTan(((myA[i]-l5)/l)/(sqrt(1-sqr((myA[i]-l5)/l))));
YA1:=(MVS+I2*mUU2[i]+I3*mUU3[i]+FI2y*(l3*cos(mU3[i])+(l2/2)*cos(mU
2[i]))+FI2x*(myA[i]-
l5+(l2/2)*sin(mU2[i]))+YVS*(l3/2)*cos(mU3[i])+FI3y*(l3/2)*cos(mU3[i])+X
VS*(l3/2)*sin(mU3[i])+FI3x*(l3/2)*sin(mU3[i]))/l;
        YC1:=-YVS*cos(G)-XVS*sin(G)-YA1-FI2y*cos(G)-FI2x*sin(G)-
FI3y*cos(G)-FI3x*sin(G);
        h:=l2*sin(mU2[i]+G); h1:=l-l2*cos(mU2[i]+G);
        XC1:=(I3*mUU3[i]+MVS-YC1*h1-FI3y*(l3/2)*cos(mU3[i])-
FI3x*(l3/2)*sin(mU3[i]))/h;
        YB1:=-YVS*cos(G)-XVS*sin(G)-YC1-FI3y*cos(G)-FI3x*sin(G);
        XB1:=FI3y*sin(G)-FI3x*cos(G)-XC1+XVS*cos(G)-YVS*sin(G);
```

```
XA1:=XB1-FI2x*cos(G)-FI2y*sin(G);
RA:=sqrt(sqr(XA1)+sqr(YA1));
RB:=sqrt(sqr(XB1)+sqr(YB1));
RC:=sqrt(sqr(XC1)+sqr(YC1));
mXA1[i]:=XA1;
mYA1[i]:=YA1;
mXB1[i]:=XB1;
mYB1[i]:=YB1;
mXC1[i]:=XC1;
mYC1[i]:=YC1;
mRA[i]:=RA;
mRB[i]:=RB;
mRC[i]:=RC;
end;
    for n:=0 to 360 do
        begin
  StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
  StringGrid2.cells[1,n+1]:=format('%12.4f',[mXA1[n]]);
  StringGrid2.cells[2,n+1]:=format('%12.4f',[mYA1[n]]);
  StringGrid2.cells[3,n+1]:=format('%12.4f',[mXB1[n]]);
  StringGrid2.cells[4,n+1]:=format('%12.4f',[mYB1[n]]);
  StringGrid2.cells[5,n+1]:=format('%12.4f',[mXC1[n]]);
  StringGrid2.cells[6,n+1]:=format('%12.4f',[mYC1[n]]);
  StringGrid2.cells[7,n+1]:=format('%12.4f',[mRA[n]]);
  StringGrid2.cells[8,n+1]:=format('%12.4f',[mRB[n]]);
  StringGrid2.cells[9,n+1]:=format('%12.4f',[mRC[n]]);
        end;
}
```

розрахунку реакцій в шарнірах шатунної коромислової групи механізму. Формування таблиці значень реакцій в шарнірах відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

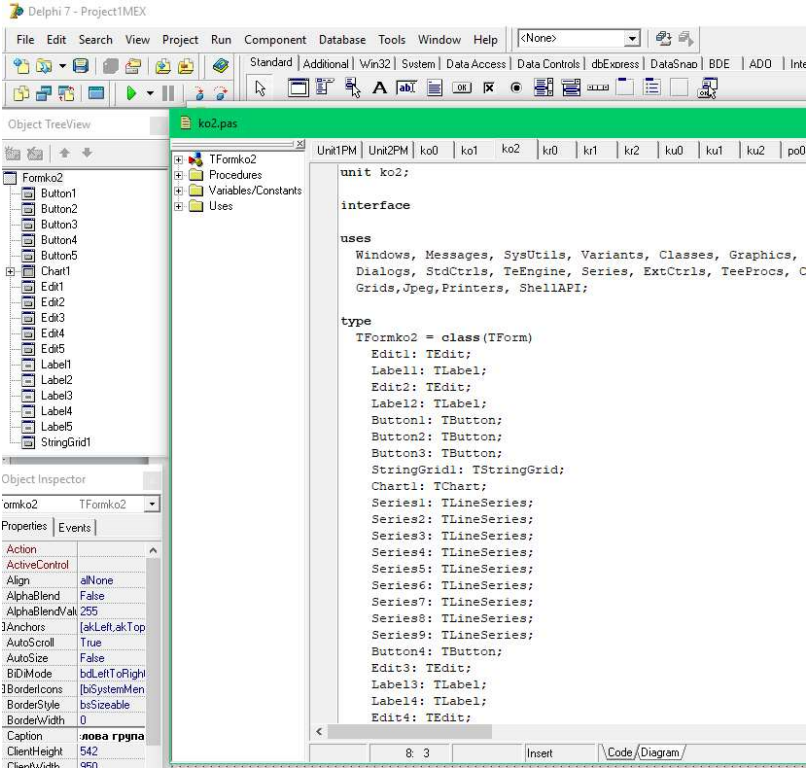


Рис.2.16

Компонент Button2: Tbutton ініціює виконання процедури procedure Button2Click(Sender: TObject) для побудови графіків зміни реакцій в шарнірах шатунної коромислової групи механізму на компоненті Chart1: Tchart. На компоненті Chart1: Tchart відображаються проекцій головних векторів шарнірів A, B, C шатунної коромислової групи механізму в проекції на координатні осі x та y та модулі головних векторів реакцій в

шарнірах A, B, C в залежності від кута обертання ведучої ланки навколо опори.

Компонент Button4: Tbutton викликає процедуру procedure Button4Click(Sender: TObject) для друку результатів розрахунку проекцій головних векторів шарнірів A, B, C шатунної коромислової групи механізму в проекції на координатні осі x та y та модулі головних векторів реакцій в шарнірах A, B, C в залежності від кута обертання ведучої ланки навколо опори з таблиці, з використанням компоненту StringGrid, в таблиці Excel.

Компонент Button3: Tbutton ініціює процедуру procedure Button3Click(Sender: TObject) згортання форми TFormko2 = class(TForm) та активації форми TFormko1 = class(TForm).

Компонент Button5: Tbutton на формі TPMForm2 = class(TForm) ініціює виконання процедури procedure Button5Click(Sender: TObject) переходу до модуля unit ku0 розрахунку механізму для кулісної групи Асура рис.2.17).

На формі TFormku0 = class(TForm) (рис.2.18) розташовані наступні компоненти TFormku0 = class(TForm), Button1: Tbutton, Image1: TImage, Image2: TImage, Label1: TLabel, Label2: TLabel, Edit1: Tedit, Label3: TLabel, Edit2: Tedit, Label4: TLabel, Label7: TLabel, Edit3: Tedit, Label8: TLabel, Edit4: Tedit, Label9: TLabel, Edit5: Tedit, Label12: TLabel, Label13: TLabel, Label14: TLabel, Edit6: Tedit, Label10: TLabel, Label5: TLabel, Edit7: Tedit.

Компоненти Image1: Timage та Image2: Timage призначені для розміщення схем кулісної групи для 2 збирань. Перша схема збирання відповідає випадку, коли кулісний камінь з шарніром A розташовується на куліси CD, яка закріплена в точці C. На схемі представлені основні лінійні та кутові геометричні параметри. Друга схема збирання відповідає випадку, коли кулісний камінь закріплений в точці C з можливістю обертання навколо неї. Куліса AD проходить скрізь нього. На схемі

представлені основні лінійні та кутові геометричні параметри для другої схеми.

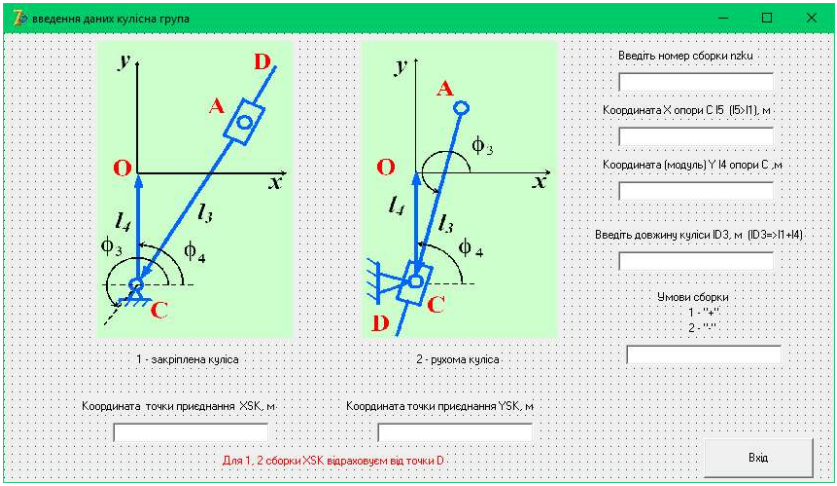


Рис.2.17

Компонент Edit3: Tedit представляє одно строковий компонент для завдання номеру збирання. Компонент Edit2: Tedit представляє одно строковий компонент для завдання координати x закріплення точки C (в метрах). Компонент Edit7: Tedit представляє одно строковий компонент для завдання координати y (в метрах). Компонент Edit1: Tedit представляє одно строковий компонент для завдання довжини куліси CD (в метрах). Початок координат розташовується в точці O шарніра ведучої ланки механізму. Компонент Edit6: Tedit представляє одно строковий компонент для завдання умов зібрання. Знак «+» обирається у випадку першого зібрання. Знак «-»обирається у випадку другого зібрання.

Компонент Edit4: Tedit представляє одно строковий компонент для завдання відстані від точки D куліси до точки K розташування центра її мас вздовж осі куліси CD. Компонент Edit5: Tedit представляє одно строковий компонент для завдання довжини перпендикуляру від точки K (розташування центра мас куліси CD) до лінії CD куліси.

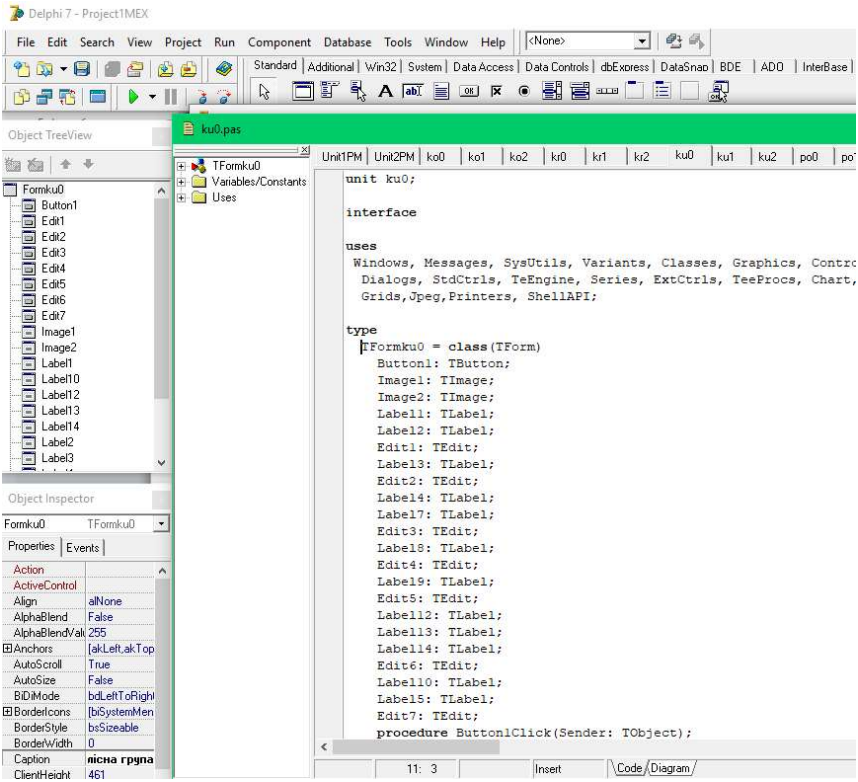


Рис.2.18

Компонент Button1: Tbutton ініціює виконання процедури procedure Button1Click(Sender: TObject) переходу з форми TFormku0 = class(TForm) до модуля unit ku1 з формою TFormku1 = class(TForm) для кінематичного розрахунку кулісної групи механізму.

На рис.2.19 представлена форма TFormku1 = class(TForm) для кінематичного розрахунку кулісної групи механізму. На формі TFormku1 = class(TForm) (рис.2.20) представлені наступні компоненти TFormku1 = class(TForm), btnkrpo: Tbutton, Chart1: Tchart, Button1ks: Tbutton, Button2ks: Tbutton, Chart2: Tchart, Chart4: Tchart, Series8: TlineSeries, Series7: TlineSeries, StringGrid1: TstringGrid, Series3: TlineSeries, Series4: TlineSeries,

Series14: TlineSeries, Series15: TlineSeries, Button1: Tbutton, Button2: Tbutton, Chart3: Tchart, Series1: TlineSeries, Series2: TlineSeries, Series5: TlineSeries, Series6: TlineSeries, Series9: TlineSeries, Series10: TlineSeries, Series11: TlineSeries, Series12: TlineSeries, Series13: TlineSeries, Series16: TlineSeries, Series17: TlineSeries, Series18: TlineSeries, Series19: TlineSeries, Series20: TlineSeries, Series21: TlineSeries, Button3: Tbutton.

Компонент Button1ks: Tbutton на формі TFormko1 = class(TForm) призначений для ініціалізації процедури procedure Button1ksClick(Sender: TObject)

```
{
    begin
    StringGrid2.cells[0,0]:='U1';
    StringGrid2.cells[1,0]:='xlD3';
    StringGrid2.cells[2,0]:='ylD3';
    StringGrid2.cells[3,0]:='U3';
    StringGrid2.cells[4,0]:='US3';
    StringGrid2.cells[5,0]:='UU3';
    StringGrid2.cells[6,0]:='vlD3x';
    StringGrid2.cells[7,0]:='vlD3y';
    StringGrid2.cells[8,0]:='wlD3x';
    StringGrid2.cells[9,0]:='wlD3y';
    StringGrid2.cells[10,0]:='XKCA';
    StringGrid2.cells[11,0]:='YKCA';
    StringGrid2.cells[12,0]:='VXKCA';
    StringGrid2.cells[13,0]:='VYKCA';
    StringGrid2.cells[14,0]:='WXKCA';
    StringGrid2.cells[15,0]:='WYKCA';
    StringGrid2.cells[16,0]:='xlSD3';
    StringGrid2.cells[17,0]:='ylSD3';
```

```
StringGrid2.cells[18,0]:='vlSD3x';
StringGrid2.cells[19,0]:='vlSD3y';
StringGrid2.cells[20,0]:='wlSD3x';
StringGrid2.cells[21,0]:='wlSD3y';
 end;
 for i:=0 to 360 do
 begin
 U1:=i*Pi/180;
 if ZPOKU<2 then
 begin
 xC:=l5;
 yC:=l4;
 U3:=arcTan((yC-myA[i])/(xC-mxA[i]));
 end
        else
 begin
  xC:=l5;
  yC:=-l4;
  U3:=2*Pi-arcTan(sqrt(sqr((yC-myA[i])/(xC-mxA[i]))));
 end;
 if nzku<2 then
 begin
 xlSD3:=xC-(lD3/2)*cos(U3);
 ylSD3:=yC-(lD3/2)*sin(U3);
 xlD3:=xC-(lD3)*cos(U3);
 ylD3:=yC-(lD3)*sin(U3);
 l3:= (xC-mxA[i])/cos(U3);
 US3:=(mvAx[i]-mvAy[i]*(cos(U3)/sin(U3)))/(l3*cos(U3)*
(cos(U3)/sin(U3))+l3*sin(U3));
```

```
        l33:=(-mvAy[i]-l3*US3*cos(U3))/sin(U3);
      UU3:=(-mwAy[i]+(sin(U3)/cos(U3))*mwAx[i]-
(sin(U3)/cos(U3))*l33*US3*sin(U3)-(sin(U3)/cos(U3))*l33*US3*sin(U3)-
(sin(U3)/cos(U3))*l3*sqr(US3)*cos(U3)-l33*US3*cos(U3)-
l33*US3*cos(U3)+l3*sqr(US3)*sin(U3))/((sin(U3)/cos(U3))*l3*sin(U3)+l3*co
s(U3));
        vlD3x:=(lD3)*US3*sin(U3);
        vlD3y:=-(lD3)*US3*cos(U3);
        vlSD3x:=(lD3/2)*US3*sin(U3);
        vlSD3y:=-(lD3/2)*US3*cos(U3);
        wlD3x:=(lD3)*UU3*sin(U3)+(lD3)*sqr(US3)*cos(U3);
        wlD3y:=-(lD3)*UU3*cos(U3)+(lD3)*sqr(US3)*sin(U3);
        wlSD3x:=(lD3/2)*UU3*sin(U3)+(lD3/2)*sqr(US3)*cos(U3);
        wlSD3y:=-(lD3/2)*UU3*cos(U3)+(lD3/2)*sqr(US3)*sin(U3);
        SA:=-XSK*sin(U3)-YSK*cos(U3);
        SB:=XSK*cos(U3)-YSK*sin(U3);
        XKCA:=xlD3+SB;
        YKCA:=ylD3-SA;
        VXKCA:=vlD3x+US3*SA;
        VYKCA:=vlD3y+US3*SB;
        WXKCA:=wlD3x+UU3*SA-US3*US3*SB;
        WYKCA:=wlD3y+UU3*SB+US3*US3*SA;
        end
              else
        begin
        xlSD3:=mxA[i]+(lD3/2)*cos(U3);
        ylSD3:=myA[i]+(lD3/2)*sin(U3);
        xlD3:=mxA[i]+(lD3)*cos(U3);
```

```
        ylD3:=myA[i]+(lD3)*sin(U3);
        l3:= (xC-mxA[i])/cos(U3);
        US3:=(mvAx[i]-mvAy[i]*(cos(U3)/sin(U3)))/(l3*cos(U3)*
(cos(U3)/sin(U3))+l3*sin(U3));
        l33:=(-mvAy[i]-l3*US3*cos(U3))/sin(U3);
        UU3:=(-mwAy[i]+(sin(U3)/cos(U3))*mwAx[i]-
(sin(U3)/cos(U3))*l33*US3*sin(U3)-(sin(U3)/cos(U3))*l33*US3*sin(U3)-
(sin(U3)/cos(U3))*l3*sqr(US3)*cos(U3)-l33*US3*cos(U3)-
l33*US3*cos(U3)+l3*sqr(US3)*sin(U3))/((sin(U3)/cos(U3))*l3*sin(U3)+l3*co
s(U3));
        vlD3x:=(lD3)*US3*sin(U3);
        vlD3y:=-(lD3)*US3*cos(U3);
        vlSD3x:=(lD3/2)*US3*sin(U3);
        vlSD3y:=-(lD3/2)*US3*cos(U3);
        wlD3x:=(lD3)*UU3*sin(U3)+(lD3)*sqr(US3)*cos(U3);
        wlD3y:=-(lD3)*UU3*cos(U3)+(lD3)*sqr(US3)*sin(U3);
        wlSD3x:=(lD3/2)*UU3*sin(U3)+(lD3/2)*sqr(US3)*cos(U3);
        wlSD3y:=-(lD3/2)*UU3*cos(U3)+(lD3/2)*sqr(US3)*sin(U3);
        SA:=-XSK*sin(U3)-YSK*cos(U3);
        SB:=XSK*cos(U3)-YSK*sin(U3);
        XKCA:=xlD3+SB;
        YKCA:=ylD3-SA;
        VXKCA:=vlD3x+US3*SA;
        VYKCA:=vlD3y+US3*SB;
        WXKCA:=wlD3x+UU3*SA-US3*US3*SB;
        WYKCA:=wlD3y+UU3*SB+US3*US3*SA;
        end;
        mU1[i]:=U1;
        mU3[i]:=U3;
```

```
mxlD3[i]:=xlD3;
mylD3[i]:=ylD3;
mUS3[i]:=US3;
mUU3[i]:=UU3;
mvlD3x[i]:=vlD3x;
mvlD3y[i]:=vlD3y;
mwlD3x[i]:=wlD3x;
mwlD3y[i]:=wlD3y;
mXKCA[i]:=XKCA;
mYKCA[i]:=YKCA;
mVXKCA[i]:=VXKCA;
mVYKCA[i]:=VYKCA;
mWXKCA[i]:=WXKCA;
mWYKCA[i]:=WYKCA;
mxlSD3[i]:=xlSD3;
mylSD3[i]:=ylSD3;
mvlSD3x[i]:=vlSD3x;
mvlSD3y[i]:=vlSD3y;
mwlSD3x[i]:=wlSD3x;
mwlSD3y[i]:=wlSD3y;
end;
    for n:=0 to 360 do
        begin
  StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
  StringGrid2.cells[1,n+1]:=format('%9.4f',[mxlD3[n]]);
  StringGrid2.cells[2,n+1]:=format('%9.4f',[mylD3[n]]);
  StringGrid2.cells[3,n+1]:=format('%9.4f',[mU3[n]]);
  StringGrid2.cells[4,n+1]:=format('%9.4f',[mUS3[n]]);
  StringGrid2.cells[5,n+1]:= format('%9.4f',[mUU3[n]]);
```

```
StringGrid2.cells[6,n+1]:=format('%9.4f',[mvlD3x[n]]);
StringGrid2.cells[7,n+1]:=format('%9.4f',[mvlD3y[n]]);
StringGrid2.cells[8,n+1]:=format('%9.4f',[mwlD3x[n]]);
StringGrid2.cells[9,n+1]:=format('%9.4f',[mwlD3y[n]]);
StringGrid2.cells[10,n+1]:=format('%9.4f',[mXKCA[n]]);
StringGrid2.cells[11,n+1]:= format('%9.4f',[mYKCA[n]]);
StringGrid2.cells[12,n+1]:=format('%9.4f',[mVXKCA[n]]);
StringGrid2.cells[13,n+1]:=format('%9.4f',[mVYKCA[n]]);
StringGrid2.cells[14,n+1]:=format('%9.4f',[mWXKCA[n]]);
StringGrid2.cells[15,n+1]:=format('%9.4f',[mWYKCA[n]]);
StringGrid2.cells[16,n+1]:=format('%9.4f',[mxlSD3[n]]);
StringGrid2.cells[17,n+1]:= format('%9.4f',[mylSD3[n]]);
StringGrid2.cells[18,n+1]:=format('%9.4f',[mvlSD3x[n]]);
StringGrid2.cells[19,n+1]:=format('%9.4f',[mvlSD3y[n]]);
StringGrid2.cells[20,n+1]:=format('%9.4f',[mwlSD3x[n]]);
StringGrid2.cells[21,n+1]:=format('%9.4f',[mwlSD3y[n]]);
                end;
}
```

розрахунку координат, проекцій на осі x та y векторів швидкості та прискорення точок кулісної групи механізму. Формування таблиці відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

Компонент Button2ks: Tbutton ініціює виконання процедури procedure Button2ksClick(Sender: TObject) для побудови графіків на компонентах Chart1: Tchart, Chart2: Tchart, Chart3: Tchart , Chart4: Tchart кулісної групи механізму. Компоненти Chart1: Tchart, Chart2: Tchart, Chart3: Tchart , Chart4: Tchart призначені для виводу графічних зображень координат, проекцій на осі x та y векторів швидкості та прискорення точок та центрів мас кулісної групи механізму. На компоненті Chart1: Tchart

відображаються переміщення точок кулісної групи механізму. На компоненті Chart2: Tchart відображається кут обертання та кутова швидкість ланок кулісної групи механізму. На компоненті Chart3: Tchart відображаються проекції на осі x та y векторів швидкості точок та центрів мас ланок кулісної групи механізму.

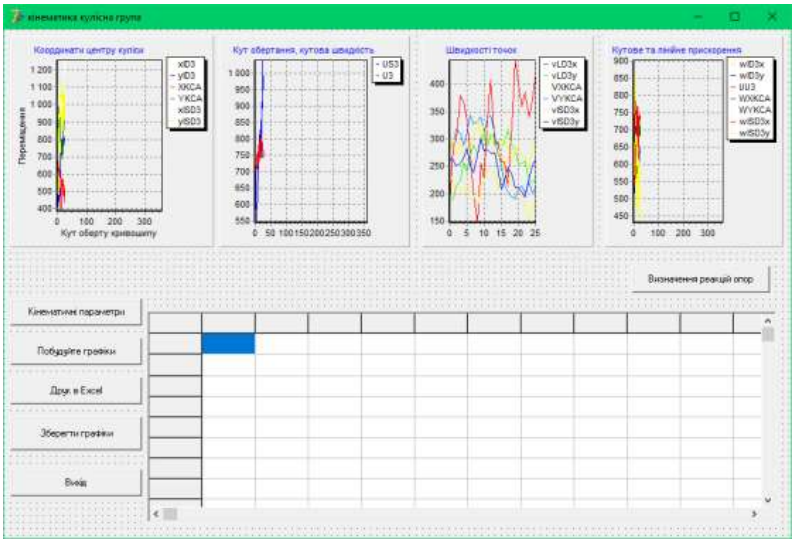


Рис.2.19

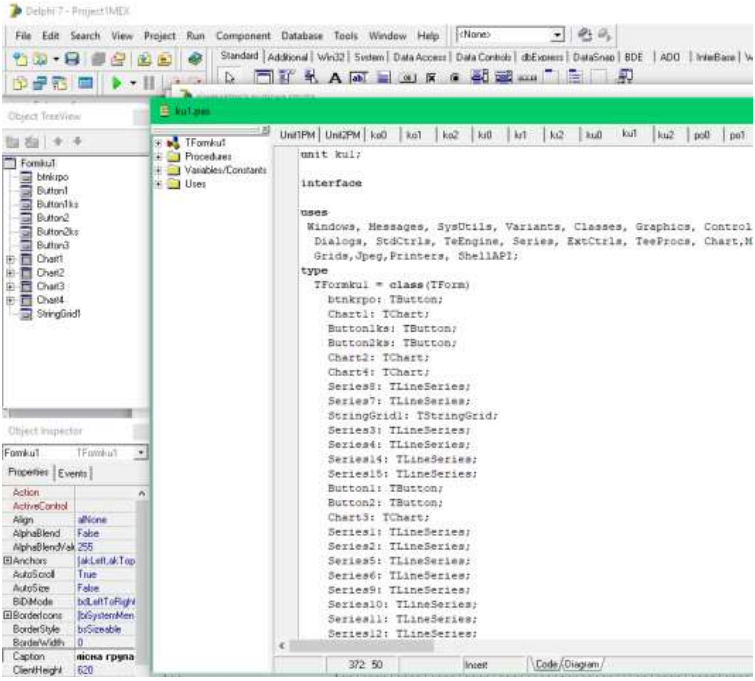


Рис.2.20

На компоненті Chart4: Tchart відображаються проекції на осі x та y векторів прискорення точок та центрів мас ланок та кутового прискорення кулісної групи механізму.

Компонент Button2: Tbutton викликає процедуру procedure Button2Click(Sender: TObject) для друку результатів розрахунку даних переміщення точок, кутової швидкості та прискорення ланок кулісної групи, проекцій на осі x та y векторів швидкості точок та центру мас ланок кулісної групи, проекцій на осі x та y векторів прискорення точок та центрів мас ланок кулісної групи з таблиці, з використанням компоненту StringGrid, в таблиці Excel.

Компонент Button1: Tbutton ініціює процедуру procedure Button1ksClick(Sender: TObject) згортання форми TFormku1 = class(TForm) та активації форми TFormku2 = class(TForm) для проведення

кінетостатичного дослідження для визначення реакцій в шарнірах кулісної групи з використанням принципу Даламбера (рис.2.15).

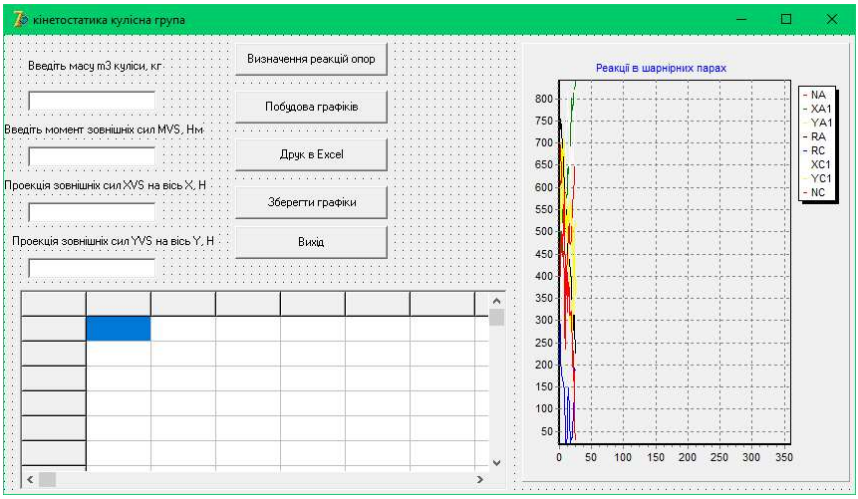


Рис.2.21

На формі TFormku2 = class(TForm) (рис.2.22) розташовані наступні компоненти TFormku2 = class(TForm), Edit1: Tedit, Label1: TLabel, Button1: Tbutton, Button2: Tbutton, Button3: Tbutton, StringGrid1: TstringGrid, Chart1: Tchart, Series1: TlineSeries, Series2: TlineSeries, Series3: TlineSeries, Series4: TlineSeries, Series5: TlineSeries, Button4: Tbutton, Label2: TLabel, Label3: TLabel, Label4: TLabel, Edit2: Tedit, Edit3: Tedit, Edit4: Tedit, Series6: TlineSeries, Series7: TlineSeries, Series8: TlineSeries, Button5: Tbutton.

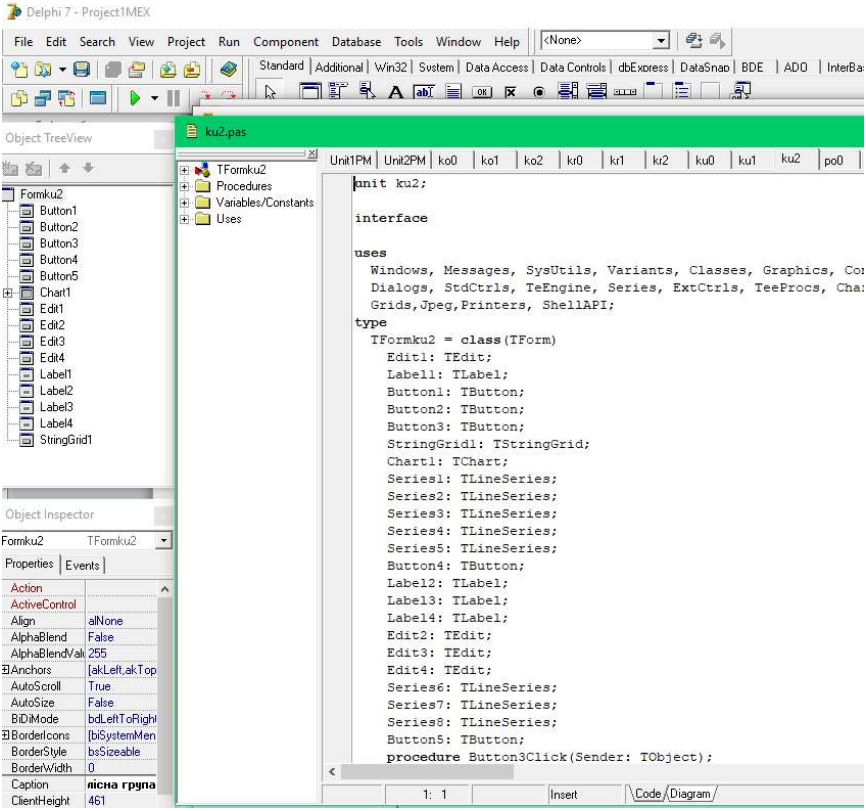


Рис.2.22

Компонент Edit1: TEdit представляє одно строковий компонент для введення маси куліси (в кг). Компонент Edit2: TEdit представляє одно строковий компонент для введення момент зовнішніх сил (в Нм). Компонент Edit3: TEdit представляє одно строковий компонент для введення проекції головного вектору зовнішніх сил на вісь x (в Н). Компонент Edit4: TEdit представляє одно строковий компонент для введення проекції головного вектору зовнішніх сил на вісь y (в Н).

Компонент Button1: Tbutton на формі TFormku2 = class(TForm) призначений для ініціалізації процедури procedure Button1Click(Sender: TObject)

```
{
Val(Edit2.text,m3,code);
   Val(Edit2.text,MVS,code);
   Val(Edit3.text,XVS,code);
   Val(Edit4.text,YVS,code);
         begin
       StringGrid2.cells[0,0]:='U1';
       StringGrid2.cells[1,0]:='NA';
       StringGrid2.cells[2,0]:='NC';
       StringGrid2.cells[3,0]:='XA1';
       StringGrid2.cells[4,0]:='YA1';
       StringGrid2.cells[5,0]:='XC1';
       StringGrid2.cells[6,0]:='YC1';
       StringGrid2.cells[7,0]:='RA';
       StringGrid2.cells[8,0]:='RC';
        end;
        for i:=0 to 360 do
    begin
        I3:=m3*lD3/12;
        wlD3x:=mwlSD3x[i]; wlD3y:=mwlSD3y[i];
        FI3x:=m3*wlSD3x;
        FI3y:=m3*wlSD3y;
            if nzku<2 then
                    begin
        NA:=(cos(Pi/2-mU3[i])/(yC-myA[i]))*(-I3*mUU3[i]-
MVS+FI3y*(lD3/2)*sin((Pi/2)-mU3[i])-
FI3x*(lD3/2)*cos((Pi/2)+YVS*(lD3)*sin((Pi/2)-mU3[i])-
XVS*(lD3)*cos((Pi/2)-mU3[i])));
        XC1:=-NA*cos((Pi/2)-mU3[i])-FI3x-XVS;
```

```
YC1:=-NA*sin((Pi/2)-mU3[i])-FI3y-YVS;
XA1:=NA*cos((Pi/2)-mU3[i]);
YA1:=NA*sin((Pi/2)-mU3[i]);
RA:=sqrt(sqr(XA1)+sqr(YA1));
RC:=sqrt(sqr(XC1)+sqr(YC1));
NC:=0;
                end
    else
begin
NC:=                                                    (sin(Pi/2-mU3[i])/(xC-
mxA[i]))*(I3*mUU3[i]+MVS+FI3y*(lD3/2)*sin((Pi/2)-mU3[i])-
FI3x*(lD3/2)*cos((Pi/2)-mU3[i])+YVS*(lD3)*sin((Pi/2)-mU3[i])-
XVS*(lD3)*cos((Pi/2)-mU3[i]));
XA1:=-NC*cos(Pi/2-mU3[i])-FI3x-XVS;
YA1:=NC*sin(Pi/2-mU3[i])-FI3y-YVS;
XC1:=NC*cos(Pi/2-mU3[i]);
YC1:=NC*sin(Pi/2-mU3[i]);
RA:=sqrt(sqr(XA1)+sqr(YA1));
RC:=sqrt(sqr(XC1)+sqr(YC1));
NA:=0;
end;
mNA[i]:=NA;
mNC[i]:=NC;
mXC1[i]:=XC1;
mYC1[i]:=YC1;
mXA1[i]:=XA1;
mYA1[i]:=YA1;
mRA[i]:=RA;
mRC[i]:=RC;
```

```
  end;
        for n:=0 to 360 do
            begin
        StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
        StringGrid2.cells[1,n+1]:=format('%12.4f',[mNA[n]]);
        StringGrid2.cells[2,n+1]:=format('%12.4f',[mNC[n]]);
        StringGrid2.cells[3,n+1]:=format('%12.4f',[mXA1[n]]);
        StringGrid2.cells[4,n+1]:=format('%12.4f',[mYA1[n]]);
        StringGrid2.cells[5,n+1]:=format('%12.4f',[mXC1[n]]);
        StringGrid2.cells[6,n+1]:=format('%12.4f',[mYC1[n]]);
        StringGrid2.cells[7,n+1]:=format('%12.4f',[mRA[n]]);
        StringGrid2.cells[8,n+1]:=format('%12.4f',[mRC[n]]);
}
```

розрахунку реакцій в шарнірах кулісної групи механізму. Формування таблиці значень реакцій в шарнірах відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

Компонент Button2: Tbutton ініціює виконання процедури procedure Button2Click(Sender: TObject) для побудови графіків зміни реакцій в шарнірах кулісної групи механізму на компоненті Chart1: Tchart. На компоненті Chart1: Tchart відображаються проекцій головних векторів шарнірів А, С кулісної групи механізму в проекції на координатні осі х та у та модулі головних векторів реакцій в шарнірах А, С в залежності від кута обертання ведучої ланки навколо опори.

Компонент Button4: Tbutton викликає процедуру procedure Button4Click(Sender: TObject) для друку результатів розрахунку проекцій головних векторів шарнірів А, С кулісної групи механізму в проекції на координатні осі х та у та модулі головних векторів реакцій в шарнірах А, С в залежності від кута обертання ведучої ланки навколо опори з таблиці, з використанням компоненту StringGrid, в таблиці Excel.

Компонент Button3: Tbutton ініціює процедуру procedure Button3Click(Sender: TObject) згортання форми TFormku2 = class(TForm) та активації форми TFormku1 = class(TForm).

Компонент Button3: Tbutton на формі TPMForm2 = class(TForm) ініціює виконання процедури procedure Button3Click(Sender: TObject) переходу до модуля unit po0 розрахунку механізму для шатунної повзункової групи Асура (рис.2.23).

На TFormPo0 = class(TForm) (рис.2.24) розташовані наступні компоненти TFormPo0 = class(TForm), Button1: Tbutton, Edit1: Tedit, Edit2: Tedit, Edit3: Tedit, Edit4: Tedit, Label1: TLabel, Label2: TLabel, Label3: TLabel, Label4: TLabel, Edit5: Tedit, Label5: TLabel, Edit6: Tedit, Label6: TLabel, Edit7: Tedit, Label7: TLabel, Image1: Timage, Label8: TLabel, Label12: TLabel, Image2: Timage, Image3: Timage, Label9: TLabel, Label13: TLabel, Label14: TLabel, Edit8: Tedit, Label15: TLabel, Label10: TLabel, Label11: TLabel, Label16: TLabel.

Компоненти Image2: Timage, Image3: Timage та Image1: Timage призначені для розміщення схем шатунної повзункової групи для 3 варіантів розташування повзуна відносно координатної системи Oxy. Перша схема розташування відповідає випадку, коли повзун рухається по горизонтальній напрямній. На схемі представлені основні лінійні та кутові геометричні параметри. Друга схема розташування відповідає випадку, коли повзун рухається по вертикальній напрямній. На схемі представлені основні лінійні та кутові геометричні параметри для другої схеми.

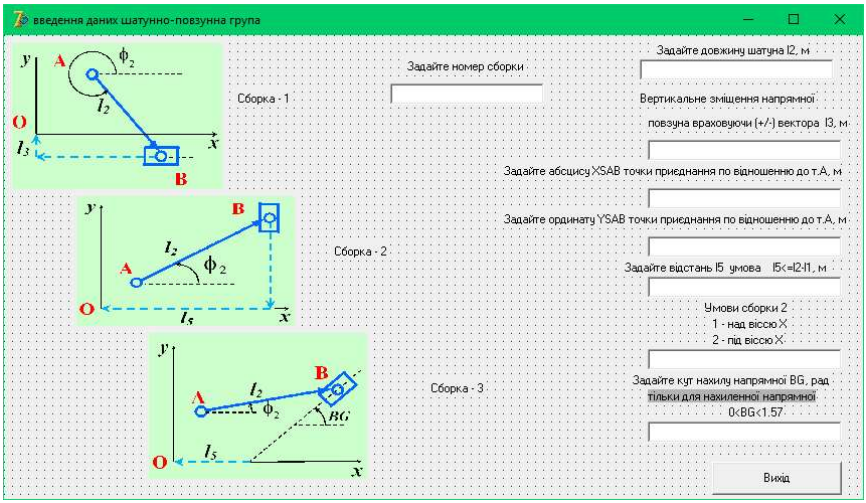


Рис.2.23

Третя схема розташування відповідає випадку, коли повзун рухається по нахиленій площині. На схемі представлені основні лінійні та кутові геометричні параметри для другої схеми.

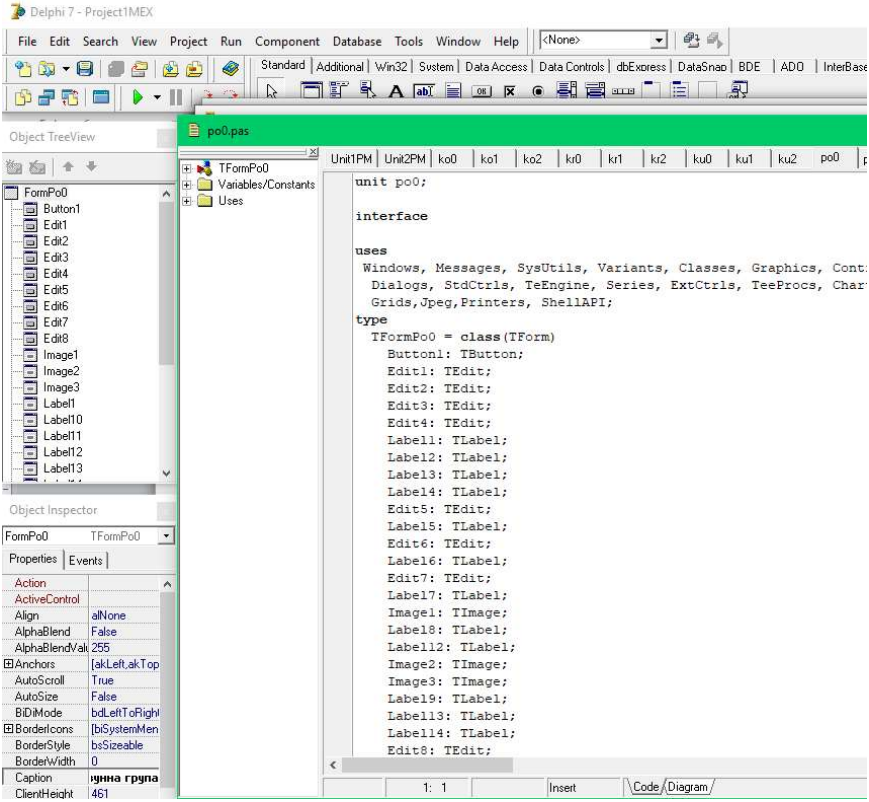


Рис.2.24

Компонент Edit8: Tedit представляє одно строковий компонент для завдання номеру розташування. Компонент Edit1: Tedit представляє одно строковий компонент для завдання довжини шатуна. Компонент Edit2: Tedit представляє одно строковий компонент для завдання вертикального зміщення горизонтальної направляючої повзуна (м). Компонент Edit3: Tedit представляє одно строковий компонент для завдання відстані від точки А шатуна до точки К розташування центра його мас вздовж осі шатуна АВ. Початок координат розташовується в точці О шарніра ведучої ланки механізму (м). Компонент Edit4: Tedit представляє одно строковий

компонент для завдання довжини перпендикуляру від точки К (розташування центра мас шатуна АВ) до лінії АВ шатуна (м).

Компонент Edit5: Tedit представляє одно строковий компонент для завдання відстані від вертикальної напрямної руху повзуна (перпендикуляр) до вертикальної осі у координатної системи Оху (м). Компонент Edit6: Tedit представляє одно строковий компонент для завдання умов зборки: «+» повзун розташований на віссю Ох; «-» повзун розташований під віссю Ох. Компонент Edit7: Tedit представляє одно строковий компонент для завдання кута нахилу напрямної (рад.)

Компонент Button1: Tbutton ініціює виконання процедури procedure Button1Click(Sender: TObject) переходу з форми TFormPo0 = class(TForm) до модуля unit po1 з формою TFormPo1 = class(TForm) для кінематичного розрахунку шатунної повзункової групи механізму.

На рис.2.25 представлена форма TFormPo1 = class(TForm) для кінематичного розрахунку шатунної повзункової групи механізму. На формі TFormPo1 = class(TForm) (рис.2.26) представлені наступні компоненти TFormPo1 = class(TForm), btnkrpo: Tbutton, Chart1: Tchart, Series3: TlineSeries, Series4: TlineSeries, Series5: TlineSeries, Button1ks: Tbutton, Button2ks: Tbutton, Series6: TlineSeries, Chart2: Tchart, Chart4: Tchart, Series8: TlineSeries, Series7: TlineSeries, Series9: TlineSeries, Series14: TlineSeries, Series15: TlineSeries, Series16: TlineSeries, Series17: TlineSeries, Series18: TlineSeries, Series19: TlineSeries, StringGrid1: TstringGrid, Button1: Tbutton, Button2: Tbutton, Chart3: Tchart, Series22: TlineSeries, Series23: TlineSeries, Series24: TlineSeries, Series25: TlineSeries, Series26: TlineSeries, Series27: TlineSeries, Chart5: Tchart, Series1: TPointSeries, Series2: TPointSeries, Series10: TlineSeries, Series11: TlineSeries, Button3: Tbutton.

Компонент Button1ks: Tbutton на формі TFormko1 = class(TForm) призначений для ініціалізації процедури procedure Button1ksClick(Sender: TObject)

```
 {
begin
        StringGrid2.cells[0,0]:='U1';
        StringGrid2.cells[1,0]:='U2';
        StringGrid2.cells[2,0]:='xB';
        StringGrid2.cells[3,0]:='yB';
        StringGrid2.cells[4,0]:='xAB';
        StringGrid2.cells[5,0]:='yAB';
        StringGrid2.cells[6,0]:='US2';
        StringGrid2.cells[7,0]:='UU2';
        StringGrid2.cells[8,0]:='vBx';
        StringGrid2.cells[9,0]:='vBy';
        StringGrid2.cells[10,0]:='wBx';
        StringGrid2.cells[11,0]:='wBy';
        StringGrid2.cells[12,0]:='vABx';
        StringGrid2.cells[13,0]:='vABy';
        StringGrid2.cells[14,0]:='wABx';
        StringGrid2.cells[15,0]:='wABy';
        StringGrid2.cells[16,0]:='xKAB';
        StringGrid2.cells[17,0]:='yKAB';
        StringGrid2.cells[18,0]:='vXKAB';
        StringGrid2.cells[19,0]:='vYKAB';
        StringGrid2.cells[20,0]:='wXKAB';
        StringGrid2.cells[21,0]:='wYKAB';
          end;
        for i:=0 to 360 do
```

```
        begin
U1:=i*Pi/180;
        if nzpo<2 then
        begin
        if ZPO<2 then
        begin
        yB:=l3;
        end
                else
        begin
        yB:=-l3;
        end;
        U2:=arcSin(-(yB+myA[i])/l2);
        xB:=mxA[i]+l2*cos(U2);
        US2:=-mvAy[i]/(l2*cos(U2));
        UU2:=(-mwAy[i]+l2*sqr(US2)*sin(U2))/(l2*cos(U2));
        vBx:=mvAx[i]-l2*US2*sin(U2);
        vBy:=0;
        wBx:=-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2)+mwAx[i];
        wBy:=0;
        xAB:=mxA[i]+(l2/2)*cos(U2);
        yAB:=myA[i]+(l2/2)*sin(U2);
        SA:=-XSAB*sin(U2)-YSAB*cos(U2);
        SB:=XSAB*cos(U2)-YSAB*sin(U2);
        XKAB:=mxA[i]+SB;
        YKAB:=myA[i]-SA;
        vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
        vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
```

```
VXKAB:=mvAx[i]+US2*SA;
VYKAB:=mvAy[i]+US2*SB;
wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
WXKAB:=mwAx[i]+UU2*SA-US2*US2*SB;
WYKAB:=mwAy[i]+UU2*SB+US2*US2*SA;
end
else
begin
if nzpo<3 then
begin
if ZPO<2 then
begin
U2:=arcCos((l5-mxA[i])/l2);
end
        else
begin
U2:=2*Pi-arcCos((l5-mxA[i])/l2);
end;
yB:=myA[i]+l2*sin(U2);
xB:=l5;
US2:=mvAx[i]/(l2*sin(U2));
UU2:=(mwAx[i]-l2*sqr(US2)*cos(U2))/(l2*sin(U2));
vBx:=0;
vBy:=mvAy[i]+l2*US2*cos(U2);
wBx:=0;
wBy:=mwAy[i]+l2*UU2*cos(U2)-l2*sqr(US2)*sin(U2);
xAB:=mxA[i]+(l2/2)*cos(U2);
yAB:=myA[i]+(l2/2)*sin(U2);
```

```
SA:=-XSAB*sin(U2)-YSAB*cos(U2);
SB:=XSAB*cos(U2)-YSAB*sin(U2);
XKAB:=mxA[i]+SB;
YKAB:=myA[i]-SA;
vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
VXKAB:=mvAx[i]+US2*SA;
VYKAB:=mvAy[i]+US2*SB;
wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
WXKAB:=mwAx[i]+UU2*SA-US2*US2*SB;
WYKAB:=mwAy[i]+UU2*SB+US2*US2*SA;
end
else
begin
vv1:=-Tan(BG)*l2;
vv2:=l2;
vv3:=-l5*Tan(BG)-myA[i]+Tan(BG)*mxA[i];
if ZPO<2 then
begin
U2:=arcSin(vv3/sqrt(sqr(vv1)+sqr(vv2)))-
arcSin(vv1/sqrt(sqr(vv1)+sqr(vv2)));
end
        else
begin
U2:=Pi+2*BG-
arcSin(vv3/sqrt(sqr(vv1)+sqr(vv2)))+arcSin(vv1/sqrt(sqr(vv1)+sqr(vv2)));
end;
xB:=mxA[i]+l2*cos(U2);
```

```
        yB:=myA[i]+l2*sin(U2);
        US2:=(mvAx[i]*Tan(BG)-mvAy[i])/(l2*sin(U2)*Tan(BG)-l2*cos(U2));
        UU2:=(l2*sqr(US2)*sin(U2)-mwAy[i]+mwAx[i]*Tan(BG)-
l2*sqr(US2)*cos(U2)*Tan(BG))/(l2*sin(U2)*Tan(BG)+l2*cos(U2));
        vBx:=mvAx[i]-l2*US2*sin(U2);
        vBy:=mvAy[i]+l2*US2*cos(U2);
        wBx:=mwAx[i]-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2);
        wBy:=mwAy[i]+l2*UU2*cos(U2)-l2*sqr(US2)*sin(U2);
        xAB:=mxA[i]+(l2/2)*cos(U2);
        yAB:=myA[i]+(l2/2)*sin(U2);
        SA:=-XSAB*sin(U2)-YSAB*cos(U2);
        SB:=XSAB*cos(U2)-YSAB*sin(U2);
        XKAB:=mxA[i]+SB;
        YKAB:=myA[i]-SA;
        vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
        vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
        VXKAB:=mvAx[i]+US2*SA;
        VYKAB:=mvAy[i]+US2*SB;
        wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
        wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
        WXKAB:=mwAx[i]+UU2*SA-US2*US2*SB;
        WYKAB:=mwAy[i]+UU2*SB+US2*US2*SA;
        end;
        end;
        mU1[i]:=U1;
        mU2[i]:=U2;
        mxB[i]:=xB;
        myB[i]:=yB;
        mxAB[i]:=xAB;
```

```
myAB[i]:=yAB;
mUS2[i]:=US2;
mUU2[i]:=UU2;
mvBx[i]:=vBx;
mvBy[i]:=vBy;
mwBx[i]:=wBx;
mwBy[i]:=wBy;
mvABx[i]:=vABx;
mvABy[i]:=vABy;
mwABx[i]:=wABx;
mwABy[i]:=wABy;
mXKAB[i]:=XKAB;
mYKAB[i]:=YKAB;
mVXKAB[i]:=VXKAB;
mVYKAB[i]:=VYKAB;
mWXKAB[i]:=WXKAB;
mWYKAB[i]:=WYKAB;
end;
    for n:=0 to 360 do
        begin
  StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
  StringGrid2.cells[1,n+1]:=format('%9.4f',[mU2[n]]);
  StringGrid2.cells[2,n+1]:=format('%9.4f',[mxB[n]]);
  StringGrid2.cells[3,n+1]:=format('%9.4f',[myB[n]]);
  StringGrid2.cells[4,n+1]:= format('%9.4f',[mxAB[n]]);
  StringGrid2.cells[5,n+1]:= format('%9.4f',[myAB[n]]);
  StringGrid2.cells[6,n+1]:=format('%9.4f',[mUS2[n]]);
  StringGrid2.cells[7,n+1]:= format('%9.4f',[mUU2[n]]);
  StringGrid2.cells[8,n+1]:=format('%9.4f',[mvBx[n]]);
```

```
        StringGrid2.cells[9,n+1]:=format('%9.4f',[mvBy[n]]);
        StringGrid2.cells[10,n+1]:=format('%9.4f',[mwBx[n]]);
        StringGrid2.cells[11,n+1]:=format('%9.4f',[mwBy[n]]);
        StringGrid2.cells[12,n+1]:=format('%9.4f',[mvABx[n]]);
        StringGrid2.cells[13,n+1]:=format('%9.4f',[mvABy[n]]);
        StringGrid2.cells[14,n+1]:=format('%9.4f',[mwABx[n]]);
        StringGrid2.cells[15,n+1]:=format('%9.4f',[mwABy[n]]);
        StringGrid2.cells[16,n+1]:=format('%9.4f',[mXKAB[n]]);
        StringGrid2.cells[17,n+1]:=format('%9.4f',[mYKAB[n]]);
        StringGrid2.cells[18,n+1]:=format('%9.4f',[mVXKAB[n]]);
        StringGrid2.cells[19,n+1]:=format('%9.4f',[mVYKAB[n]]);
        StringGrid2.cells[20,n+1]:=format('%9.4f',[mWXKAB[n]]);
        StringGrid2.cells[21,n+1]:=format('%9.4f',[mWYKAB[n]]);
}
```

розрахунку координат, проекцій на осі x та y векторів швидкості та прискорення точок шатунної повзункової групи механізму. Формування таблиці відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

Компонент Button2ks: TButton ініціює виконання процедури

```
procedure Button2ksClick(Sender: TObject)
{
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart2.SeriesList[4].Clear;
Chart2.SeriesList[5].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
```

```
Chart2.SeriesList[2].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
Chart4.SeriesList[0].Clear;
Chart4.SeriesList[1].Clear;
Chart4.SeriesList[2].Clear;
Chart4.SeriesList[3].Clear;
Chart4.SeriesList[4].Clear;
Chart4.SeriesList[5].Clear;
Chart4.SeriesList[6].Clear;
Chart4.SeriesList[7].Clear;
Chart5.SeriesList[0].Clear;
Chart5.SeriesList[1].Clear;
for j:=0 to 360 do
 begin
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mxB[j],",clRed);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,myB[j],",clBlue);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mxAB[j],",clYellow);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,myAB[j],",clGreen);
Chart2.SeriesList[4].AddXY(mU1[j]*180/Pi,mxKAB[j],",clWhite);
Chart2.SeriesList[5].AddXY(mU1[j]*180/Pi,myKAB[j],",clBlack);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mUS2[j],",clBlue);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mU2[j],",clWhite);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mUU2[j],",clRed);


Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mVXKAB[j],",clYellow);
```

```
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mVYKAB[j],'',clBlue);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mwBx[j],'',clRed);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mwBy[j],'',clGreen);
Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mvABx[j],'',clRed);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mvABy[j],'',clGreen);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mwABx[j],'',clYellow);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mwABy[j],'',clBlue);
Chart4.SeriesList[4].AddXY(mU1[j]*180/Pi,mvBx[j],'',clWhite);
Chart4.SeriesList[5].AddXY(mU1[j]*180/Pi,mvBy[j],'',clBlack);
Chart4.SeriesList[6].AddXY(mU1[j]*180/Pi,mwXKAB[j],'',clGreen);
Chart4.SeriesList[7].AddXY(mU1[j]*180/Pi,mwYKAB[j],'',clYellow);
Chart5.SeriesList[0].AddXY(mxAB[j],myAB[j],'',clRed);
Chart5.SeriesList[1].AddXY(mxKAB[j],myKAB[j],'',clGreen);
}
```

для побудови графіків на компонентах Chart1: Tchart, Chart2: Tchart, Chart3: Tchart , Chart4: Tchart та Chart5: Tchart шатунної повзункової групи механізму. Компоненти Chart1: Tchart, Chart2: Tchart, Chart3: Tchart , Chart4: Tchart та Chart5: Tchart призначені для виводу графічних зображень координат, проекцій на осі x та y векторів швидкості та прискорення точок та центрів мас шатунної повзункової групи механізму. На компоненті Chart1: Tchart відображаються переміщення точок шатунної повзункової групи механізму. На компоненті Chart2: Tchart відображається кутова швидкість та прискорення ланок шатунної повзункової групи механізму. На компоненті Chart3: Tchart відображаються проекції на осі x та y векторів швидкості та прискорення точок шатунної повзункової групи механізму.

На компоненті Chart4: Tchart відображаються проекції на осі x та y векторів швидкості та прискорення точки A шатунної повзункової групи

механізму. На компоненті Chart5: Tchart відображаються траєкторії руху центру мас та симетрії шатуна шатунної повзункової групи механізму.

Компонент Button2: Tbutton викликає процедуру procedure Button2Click(Sender: TObject) для друку результатів розрахунку даних переміщення точок, кутової швидкості та прискорення ланок шатунної повзункової групи з використанням компоненту StringGrid, в таблиці Excel.

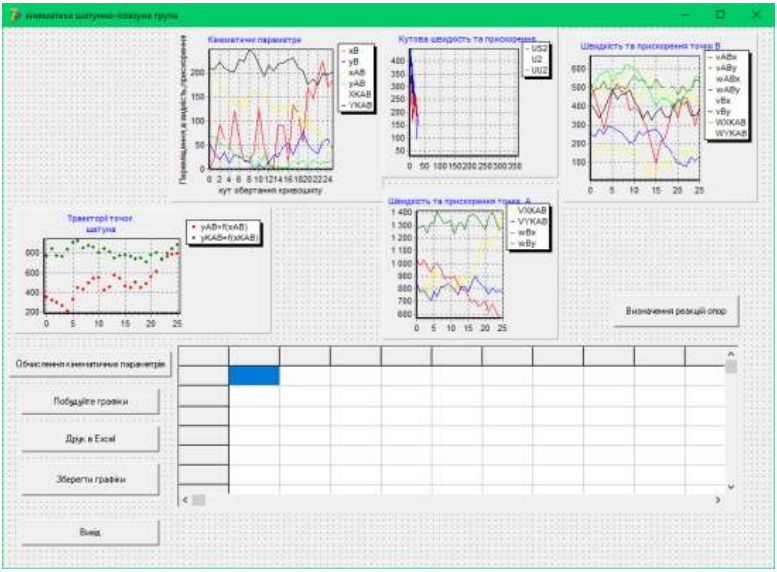


Рис.2.25

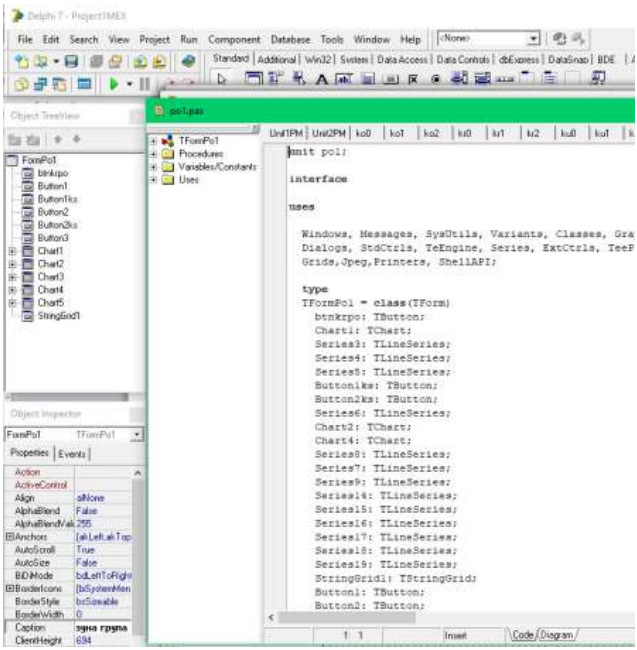


Рис.2.26

Компонент Button1: Tbutton ініціює процедуру procedure Button1ksClick(Sender: TObject) згортання форми TFormPo1 = class(TForm) та активації форми TFormPo2 = class(TForm) для проведення кінетостатичного дослідження для визначення реакцій в шарнірах шатунної повзункової групи з використанням принципу Даламбера (рис.2.27).

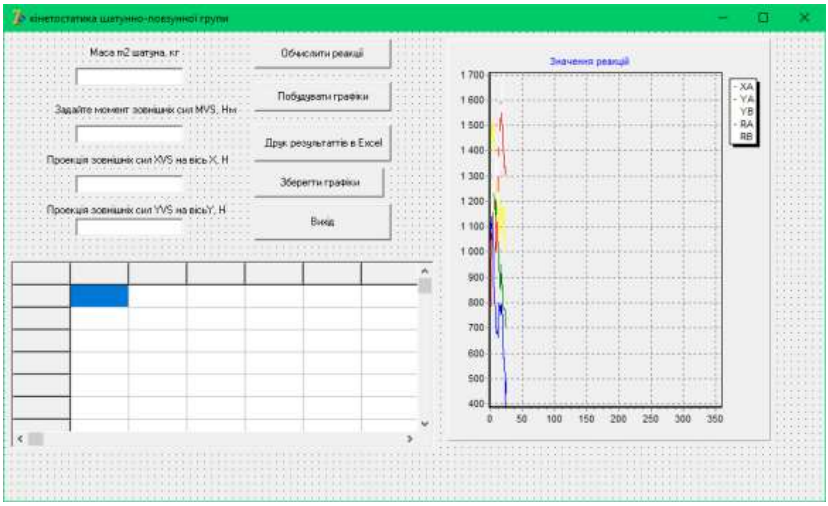


Рис.2.27

На формі TFormPo2 = class(TForm) (рис.2.28) розташовані наступні компоненти TFormPo2 = class(TForm), Edit1: Tedit, Label1: TLabel, Button1: Tbutton, Button2: Tbutton, Button3: Tbutton, StringGrid1: TstringGrid, Chart1: Tchart, Series1: TlineSeries, Series2: TlineSeries, Series3: TlineSeries, Series4: TlineSeries, Series5: TlineSeries, Button4: Tbutton, Label2: TLabel, Edit2: Tedit, Edit3: Tedit, Label3: TLabel, Label4: TLabel, Edit4: Tedit, Button5: Tbutton.

Компонент Edit1: TEdit представляє одно строковий компонент для введення маси шатуна (в кг). Компонент Edit2: TEdit представляє одно строковий компонент для введення момент зовнішніх сил (в Нм). Компонент Edit3: TEdit представляє одно строковий компонент для введення проекції головного вектору зовнішніх сил на вісь x (в Н). Компонент Edit4: TEdit представляє одно строковий компонент для введення проекції головного вектору зовнішніх сил на вісь y (в Н).

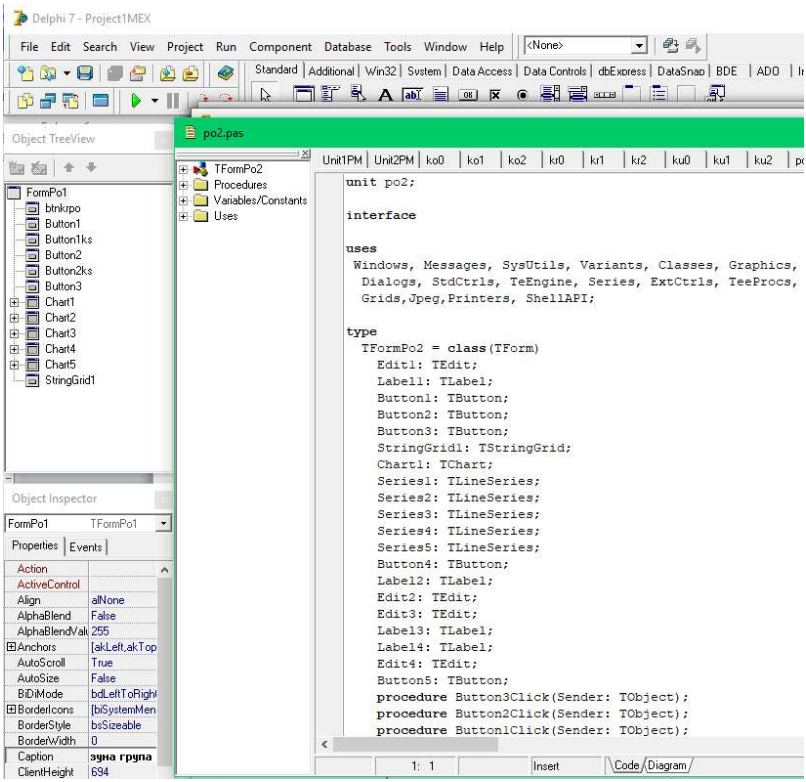


Рис.2.28

Компонент Button1: Tbutton на формі TFormPo2 = class(TForm) призначений для ініціалізації процедури procedure Button1Click(Sender: TObject)

```
{
Val(Edit2.text,m2,code);
  Val(Edit2.text,MVS,code);
  Val(Edit3.text,XVS,code);
  Val(Edit4.text,YVS,code);
        begin
      StringGrid2.cells[0,0]:='U1';
```

```
        StringGrid2.cells[1,0]:='XA';
        StringGrid2.cells[2,0]:='YA';
        StringGrid2.cells[3,0]:='YB';
        StringGrid2.cells[4,0]:='RA';
        StringGrid2.cells[5,0]:='RB';
              end;
        for i:=0 to 360 do
        begin
        I2:=m2*l2/12;
        wABx:=mwABx[i]; wABy:=mwABy[i];
        FI2x:=m2*wABx;
        FI2y:=m2*wABy;
YB1:=(MVS+FI2x*sin(mU2[i])*(l2/2)+I2*mUU2[i]+XVS*l2*sin(mU2[i])+YV
S*l2*cos(mU2[i])+FI2y*cos(mU2[i])*(l2/2))/(-l2*cos(mU2[i]));
        XA1:=-FI2x+XVS;
        YA1:=-FI2y-YB1+YVS;
        RA:=sqrt(sqr(XA1)+sqr(YA1));
        RB:=sqrt(sqr(YB1));
        mXA1[i]:=XA1;
        mYA1[i]:=YA1;
        mYB1[i]:=YB1;
        mRA[i]:=RA;
        mRB[i]:=RB;
        end;
            for n:=0 to 360 do
                begin
          StringGrid2.cells[0,n+1]:=format('%3.0d',[n]);
          StringGrid2.cells[1,n+1]:=format('%12.4f',[mXA1[n]]);
          StringGrid2.cells[2,n+1]:=format('%12.4f',[mYA1[n]]);
```

```
        StringGrid2.cells[3,n+1]:=format('%12.4f',[mYB1[n]]);
        StringGrid2.cells[4,n+1]:=format('%12.4f',[mRA[n]]);
        StringGrid2.cells[5,n+1]:=format('%12.4f',[mRB[n]]);
}
```

розрахунку реакцій в шарнірах шатунної повзункової групи механізму. Формування таблиці значень реакцій в шарнірах відбувається з використанням компоненту StringGrid для відображення різних даних в табличній формі.

Компонент Button2: Tbutton ініціює виконання процедури procedure

```
Button2Click(Sender: TObject)
{
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart2.SeriesList[4].Clear;
for j:=0 to 360 do
 begin
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mXA1[j],'',clRed);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mYA1[j],'',clGreen);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mYB1[j],'',clYellow);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,mRA[j],'',clBlue);
Chart2.SeriesList[4].AddXY(mU1[j]*180/Pi,mRB[j],'',clWhite);
}
```

для побудови графіків зміни реакцій в шарнірах шатунної повзункової групи механізму на компоненті Chart1: Tchart. На компоненті Chart1: Tchart відображаються проекцій головних векторів шарнірів А, В шатунної повзункової групи механізму в проекції на координатні осі x та y та модулі

головних векторів реакцій в шарнірах A, B залежно від кута обертання ведучої ланки навколо опори.

Компонент Button4: Tbutton викликає процедуру procedure Button4Click(Sender: TObject) для друку результатів розрахунку проекцій головних векторів шарнірів A, B шатунної повзункової групи механізму в проекції на координатні осі x та y та модулі головних векторів реакцій в шарнірах A, B в залежності від кута обертання ведучої ланки навколо опори з таблиці, з використанням компоненту StringGrid, в таблиці Excel.

Компонент Button3: Tbutton ініціює процедуру procedure Button3Click(Sender: TObject) згортання форми TFormPo2 = class(TForm) та активації форми TFormPo1 = class(TForm).

ЛІСТИНГ ПРОГРАМ ДЛЯ СТРУКТУРНОГО СИНТЕЗУ СИСТЕМИ ПОДАЧІ НИТКИ

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include <string.h>
#include "Thread.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BPathSearchClick(TObject *Sender)
{
  PathPoint *pPath, *p;
  //Формування початкових даних
  if (NP)
    MasObst = new obstacle[NP];
  for (int i = 0; i < NP; i++)
    {
```

```
   MasObst[i] = pObst -> obst;
   MasObst[i].Tv.MinSumAngle = 6.28*NP;
   MasObst[i].Tv.ptrAD = NULL;
   MasObst[i].Tn.MinSumAngle = 6.28*NP;
   MasObst[i].Tn.ptrAD = NULL;
   pObst = pObst -> next;
  }
Record = 6.28*(NP+1);
Vt.MinSumAngle = 6.28*NP;
Vt.ptrAD = NULL;
formGV();
//Визначення координат вершин знайденого шляху
pPath = new PathPoint;
pPath -> x = Xt;
pPath -> y = Vt.y;
p = pPath;
while (pRec -> nomper != -1)
  {
   p -> next = new PathPoint;
   p = p -> next;
   p -> x = MasObst[pRec -> nomper].x;
   p -> y = MasObst[pRec -> nomper].Tv.y;
   if (pRec -> ver)
     p -> y = MasObst[pRec -> nomper].Tn.y;
   pRec = pRec -> pPAD;
  }
p -> next = new PathPoint;
p = p -> next;
p -> x = Xs;
```

```
  p -> y = Vs.y;
  p -> next = NULL;
  //Друкування шляху
  Canvas -> Pen -> Color = clRed;
  Canvas -> Pen -> Width = 3;
  p = pPath;
  Canvas -> MoveTo(p -> x, p -> y);
  while (p)
    {
     Canvas -> LineTo(p->x, p->y);
     p = p -> next;
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N1Click(TObject *Sender)
{
  reg = 1;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N2Click(TObject *Sender)
{
  reg = 2;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N3Click(TObject *Sender)
{
```

```
 reg = 3;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Image1MouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
  int Xn;
  switch (reg)
   {
    case 1: if ( Button == mbLeft)
              {
               Xs = X; Vs.y = Y;
               Canvas -> Pen -> Color = clLime;
               Canvas -> Brush -> Color = clLime;
               Canvas -> Ellipse(X - 5, Y + 5, X + 5, Y - 5);
               reg = 4;
              }
            break;
    case 2: if ( Button == mbLeft)
              {
               Xt = X; Vt.y = Y;
               Canvas -> Pen -> Color = clBlue;
               Canvas -> Brush -> Color = clBlue;
               Canvas -> Ellipse(X - 5, Y + 5, X + 5, Y - 5);
               reg = 4;
              }
            break;
    case 3: if ( Button == mbLeft && !draw)
```

```
            {
              Canvas -> Pen -> Color = clOlive;
              Canvas -> Pen -> Width = 5;
              draw = true;
              reg = 4;
            }
    }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Image1MouseMove(TObject *Sender, TShiftState
Shift,
      int X, int Y)
{
  static int Xn;
  static bool r;
  ListObstacle *prev, *pT;
  if (!draw) r = false;
  if (draw)
    {
     if (!r)
       {
         Canvas -> MoveTo(X, Y);
         Xn = X;
         NP++;
         //Визначення верхньої точки перешкоди
         p = new ListObstacle;
         (p -> obst).x = X;
         (p -> obst).Tn.y = Y;
```

```
        //Розташування перешкоди в упорядкованому списку
        pT = pObst;
        prev = NULL;
        while (pT && ((pT->obst).x < X-0.0001 ||
          fabs((pT->obst).x - X)< 0.0001 && Y > (pT->obst).Tn.y))
          {
           prev = pT;
           pT = pT->next;
          }
        p->next = pT;
        if (prev)
          prev->next = p;
        else
          pObst = p;
        r = true;
       }
     Canvas -> LineTo(Xn, Y);
   }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Image1MouseUp(TObject *Sender, TMouseButton
Button,
     TShiftState Shift, int X, int Y)
{
  if (draw)
    {
     draw = false;
     Canvas -> Pen -> Color = clRed;
```

```
    Canvas -> Pen -> Width = 3;
    //Визначення нижньої точки перешкоди
   (p -> obst).Tv.y = Y;
  }
}
//---------------------------------------------------------------------------

/---------------------------------------------------------------------------

#ifndef ThreadH
#define ThreadH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <math.h>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
#include <algorithm>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:        // IDE-managed Components
      TButton *BPathSearch;
      TImage *Image1;
      TPopupMenu *PopupMenu1;
```

```
        TMenuItem *N1;
        TMenuItem *N2;
        TMenuItem *N3;
        void __fastcall BPathSearchClick(TObject *Sender);
        void __fastcall N1Click(TObject *Sender);
        void __fastcall N2Click(TObject *Sender);
        void __fastcall N3Click(TObject *Sender);
        void __fastcall Image1MouseDown(TObject *Sender,
          TMouseButton Button, TShiftState Shift, int X, int Y);
        void __fastcall Image1MouseMove(TObject *Sender, TShiftState Shift,
          int X, int Y);
        void __fastcall Image1MouseUp(TObject *Sender, TMouseButton Button,
          TShiftState Shift, int X, int Y);
private:        // User declarations
public:                 // User declarations
        __fastcall TForm1(TComponent* Owner);
};
struct ListApproachDir
{
  int nomper;
  bool ver;
  float ApprAngle, SumAngle;
  ListApproachDir *pPAD, *next;
};
struct point
{
  float y, MinSumAngle;
  ListApproachDir* ptrAD;
};
```

```
struct PathPoint
{
  float x, y;
  PathPoint *next;
};
struct obstacle
{
  point Tv, Tn;
  float x;
};
struct ListObstacle
{
   obstacle obst;
   ListObstacle *next;
};
struct ListBlockDir
{
  float hd, ld;
  ListBlockDir* next;
};
int Xn, NP = 0; // Кількість перешкод
float Xs, Xt; // Координати джерела та цілі
obstacle* MasObst;
ListObstacle  *pObst = NULL, *p;
point Vs, Vt; // Вершини - джерело та ціль
short reg;
bool draw;
//char b[100];
  point *pV;
```

```
 ListApproachDir *pAD, *pAVD, *prevAD, *CurpAD, *pNAD, *pRec;
 float x, y, dv, da, RoundAngle, CurSumAngle, Record;
float DefAngle(float x, float y, int kper, bool kv)
/*
 Визначення тангенсу напрямку на видиму вершину
 x,y - координати вихідної точки;
 kper, kv - параметри видимої вершини
*/
{
 float Xv;
  //Координати старої видимої вершини
 float Yv;
 if (kper == -1)
   {
    Xv = Xt;
    Yv = Vt.y;
   }
 else
   {
    Xv = MasObst[kper].x;
    if(kv)
      Yv = MasObst[kper].Tn.y;
    else
      Yv = MasObst[kper].Tv.y;
   }
 return (Yv - y)/(Xv - x);
}
//---------------------------------------------------------------------
void insertListApproachDir( int kper, bool kv, float x, float y,
```

```
                    point* p, bool kp, int nomp)
{
 /*
   kper -  видима перешкода
   kv -    тип видимої точки (верхня чи нижня)
   x, y -  координати вихідної точки
   p - покажчик на вихідну точку
   kp - тип вихідної точки (верхня чи нижня)
   nomp -  вихідна перешкода
 */
 float dv, CurSumAngle;
 point* pV;
 ListApproachDir *pNAD, *pPAD, *prev, *CurpAD;
 if (kper == -1)
   pV = &Vt;
 else
   {
     pV = &(MasObst[kper].Tv);
     if(kv) pV = &(MasObst[kper].Tn);
   }
 //Визначення кута переходу у наступну вершину
 dv = DefAngle(x, y, kper, kv);
 if (!(p->ptrAD))
   {
     // Вихідна точка - джерело
     pNAD = new ListApproachDir;
     pNAD->ver = false;
     pNAD->nomper = -1;
     pNAD->ApprAngle = dv;
```

```
   pNAD->SumAngle = 0;
   pNAD->next = NULL;
   pV->ptrAD = pNAD;
   pV -> MinSumAngle = 0;
   return;
  }
else
  {
   pPAD = p->ptrAD;
   //Визначення сумарного кута досягнення наступної вершини
   CurSumAngle = 6.28 * NP;
   while (pPAD && ((!kp && pPAD->ApprAngle > dv + 0.00001) ||
        (kp && pPAD->ApprAngle < dv - 0.00001)))
     {
      RoundAngle = (1 - 2*kp)*(atan(pPAD->ApprAngle) - atan(dv));
      if (pPAD->SumAngle + RoundAngle < CurSumAngle)
        {
         CurSumAngle = pPAD->SumAngle + RoundAngle;
         CurpAD = pPAD;
        }
      pPAD = pPAD -> next;
     }
   if (CurSumAngle < 6.28*NP - 0.0001) //Перехід можливий
     {
      pNAD = new ListApproachDir;
      pNAD->ver = kp;
      pNAD->nomper = nomp;
      pNAD->ApprAngle = dv;
      pNAD->SumAngle = CurSumAngle;
```

```
      pNAD->pPAD = CurpAD;
      if (CurSumAngle < pV -> MinSumAngle)
        { //Вершину досягнуто з меншим сумарним кутом
         pV -> MinSumAngle = CurSumAngle;
  /*              ShowMessage("Досягнуто "+IntToStr(pVD -> nomper)+", "+
                IntToStr((int)(pVD                                    ->
ver))+"MinSumAngle="+FloatToStr(CurSumAngle));
  */
          if (pV == &Vt) //Цільова вершина
            {
             Record = CurSumAngle;
             pRec = pNAD;
            }
        }
      //Додавання входу у видиму вершину
      pPAD = pV -> ptrAD;
      prev = NULL;
      //Пошук місця додавання входу
      while (pPAD && ((!kv && pPAD->ApprAngle > dv + 0.00001) ||
          (kv && pPAD->ApprAngle < dv - 0.00001)))
        {
         prev = pPAD;
         pPAD = pPAD -> next;
        }
      if (prev)  //Додавання у середину або у кінець списку
        {
         pNAD -> next = prev -> next;
         prev -> next = pNAD;
        }
```

```
        else    //Додавання у початок списку
          {
           pNAD -> next = pV -> ptrAD;
           pV -> ptrAD = pNAD;
          }
       }
    }
}
//---------------------------------------------------------------------------
//Звільнення памяті
void FreeMem(ListBlockDir *p)
{
  if (p -> next)
    FreeMem(p -> next);
  delete p;
}

//---------------------------------------------------------------------------
void formVDv(float x, float y, int nomp, point*v, bool kp)
//Визначення видимих вершин з поданої *v
{
  if (v -> MinSumAngle > Record - 0.00001) return;
  ListBlockDir *LBD = NULL, *p, *q, *previous;
  float vd, nd;
  int k = (nomp > 0)?nomp:0;
  //Аналіз перешкод
  for (int i = k; i < NP; i++)
    if ( MasObst[i].x > x + 0.001) //Перешкоду знайдено
      {
```

```
vd =  (MasObst[i].Tv.y - y)/(MasObst[i].x - x);
nd =  (MasObst[i].Tn.y - y)/(MasObst[i].x - x);
if (!LBD) //Перша перешкода
  {
   insertListApproachDir(i, false, x, y, v, kp, nomp); //Додавання видимих
вершин
   insertListApproachDir(i, true, x, y, v, kp, nomp);
   LBD = new ListBlockDir;
   LBD -> next = NULL;
   LBD -> hd =  vd;
   LBD -> ld =  nd;
  }
else  //Неперша перешкода
  {
   q = LBD;
   previous = NULL;
   while (q)
    {
     if ( vd >= q -> hd) //Верхня вершина вище блокованого сектора
       {
        insertListApproachDir(i, false, x, y, v, kp, nomp);
        if (nd >= q -> hd) //Уся перешкода вище
          {
           insertListApproachDir(i, true, x, y, v, kp, nomp);
          //Створення нового блокованого сектора
           p = new ListBlockDir;
           p->hd = vd;
           p->ld = nd;
           p->next = q;
```

```
      if(previous)
        previous->next = p;
      else
       LBD = p;
      break;
     }
   else
  //Розширення поточного блокованого сектора угору
     q -> hd = vd;
   if (nd > q -> ld) //Нижня вершина невидима
     break;
  }
if (vd <= q -> ld) //Верхня вершина нижче поточного сектора
  {
   previous = q;
   q = q -> next;
   continue;
  }
/*Об'єднання блокованих напрямків і перевірка видимості
нижньої вершини*/
p = q;
while (p)
  {
   if (p -> hd <= nd) //Нижня вершина видима
     {
      insertListApproachDir(i, true, x, y, v, kp, nomp);
      q -> ld = nd;
      break;
     }
```

```
    else
     if (p -> ld < nd) //Нижня вершина невидима
      {
       if (p != q) //Вилучення блокованого сектора
        {
         q -> ld = p -> ld;
         q -> next = p -> next;
         delete p;
        }
       break;
      }
     else
      {
       //Перехід до наступного блокованого сектора
       if (q != p)
        {
         q -> next = p;
         p = p -> next;
         delete (q -> next);
        }
       else
        p = p -> next;
      }
  }//Кінець циклу while p
if (!p)
  {
   q -> ld = nd;
   q -> next = NULL;
   insertListApproachDir(i, true, x, y, v, kp, nomp);
```

```
            }
            break;
          }//Кінець циклу while q
        if (!q) //Створення нового блокованого сектора
          {
            insertListApproachDir(i, false, x, y, v, kp, nomp);
            insertListApproachDir(i, true, x, y, v, kp, nomp);
            p = new ListBlockDir;
            p->hd = vd;
            p->ld = nd;
            p->next = NULL;
            previous->next = p;
          }
      }// else Неперша перешкода
   }//if Перешкоду знайдено; Кінець циклу for
if ( !LBD) //Перешкод між вершиною та ціллю нема
  insertListApproachDir(-1, false, x, y, v, kp, nomp);
else
 //Перевірка видимості цілі
  {
   q = LBD;
   vd = (Vt.y - y)/(Xt - x);
   while (q)
     {
      if (vd >= q -> hd)
        {
          insertListApproachDir(-1, false, x, y, v, kp, nomp);
          break;
        }
```

```
        else
          if (vd > q -> ld)
            break;
        q = q -> next;
       }
     if (!q) insertListApproachDir(-1, false, x, y, v, kp, nomp);
    }
  // Звільнення динамічної пам'яті
  if (LBD) FreeMem(LBD);
}
//---------------------------------------------------------------------------
void formGV()
// Формування графу видимості
{
  formVDv(Xs, Vs.y, -1, &Vs, false);
  for (int j = 0; j < NP; j++)
    {
      formVDv(MasObst[j].x, MasObst[j].Tv.y, j, &MasObst[j].Tv, false);
      formVDv(MasObst[j].x, MasObst[j].Tn.y, j, &MasObst[j].Tn, true);
    }
}
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif

struct ListApproachDir
{
  int nomper;
```

```
 bool ver;
 float ApprAngle, SumAngle;
 ListApproachDir *pPAD, *next;
};
struct point
{
 float y, MinSumAngle;
 ListApproachDir* ptrAD;
};
struct PathPoint
{
 float x, y;
 PathPoint *next;
};
struct obstacle
{
 point Tv, Tn;
 float x;
};
struct ListObstacle
{
  obstacle obst;
  ListObstacle *next;
};
struct ListBlockDir
{
 float hd, ld;
 ListBlockDir* next;
};
```

```
int Xn, NP = 0; // Кількість перешкод
float Xs, Xt; // Координати джерела та цілі
obstacle* MasObst;
ListObstacle  *pObst = NULL, *p;
point Vs, Vt; // Вершини - джерело та ціль
short reg;
bool draw;
//char b[100];
  point *pV;
  ListApproachDir *pAD, *pAVD, *prevAD, *CurpAD, *pNAD, *pRec;
  float x, y, dv, da, RoundAngle, CurSumAngle, Record;
float DefAngle(float x, float y, int kper, bool kv)
/*
  Визначення тангенсу напрямку на видиму вершину
  x,y - координати вихідної точки;
  kper, kv - параметри видимої вершини
*/
{
  float Xv;
    //Координати старої видимої вершини
  float Yv;
  if (kper == -1)
    {
      Xv = Xt;
      Yv = Vt.y;
    }
  else
    {
      Xv = MasObst[kper].x;
```

```
    if(kv)
      Yv = MasObst[kper].Tn.y;
    else
      Yv = MasObst[kper].Tv.y;
   }
 return (Yv - y)/(Xv - x);
}
//---------------------------------------------------------------------
void insertListApproachDir( int kper, bool kv, float x, float y,
                            point* p, bool kp, int nomp)
{
 /*
   kper -  видима перешкода
   kv -    тип видимої точки (верхня чи нижня)
   x, y -  координати вихідної точки
   p - покажчик на вихідну точку
   kp - тип вихідної точки (верхня чи нижня)
   nomp -  вихідна перешкода
 */
 float dv, CurSumAngle;
 point* pV;
 ListApproachDir *pNAD, *pPAD, *prev, *CurpAD;
 if (kper == -1)
   pV = &Vt;
 else
   {
    pV = &(MasObst[kper].Tv);
    if(kv) pV = &(MasObst[kper].Tn);
   }
```

```
//Визначення кута переходу у наступну вершину
dv = DefAngle(x, y, kper, kv);
if (!(p->ptrAD))
  {
   // Вихідна точка - джерело
   pNAD = new ListApproachDir;
   pNAD->ver = false;
   pNAD->nomper = -1;
   pNAD->ApprAngle = dv;
   pNAD->SumAngle = 0;
   pNAD->next = NULL;
   pV->ptrAD = pNAD;
   pV -> MinSumAngle = 0;
   return;
  }
else
  {
   pPAD = p->ptrAD;
   //Визначення сумарного кута досягнення наступної вершини
   CurSumAngle = 6.28 * NP;
   while (pPAD && ((!kp && pPAD->ApprAngle > dv + 0.00001) ||
        (kp && pPAD->ApprAngle < dv - 0.00001)))
     {
      RoundAngle = (1 - 2*kp)*(atan(pPAD->ApprAngle) - atan(dv));
      if (pPAD->SumAngle + RoundAngle < CurSumAngle)
        {
         CurSumAngle = pPAD->SumAngle + RoundAngle;
         CurpAD = pPAD;
        }
```

```
      pPAD = pPAD -> next;
     }
    if (CurSumAngle < 6.28*NP - 0.0001) //Перехід можливий
     {
      pNAD = new ListApproachDir;
      pNAD->ver = kp;
      pNAD->nomper = nomp;
      pNAD->ApprAngle = dv;
      pNAD->SumAngle = CurSumAngle;
      pNAD->pPAD = CurpAD;
      if (CurSumAngle < pV -> MinSumAngle)
       { //Вершину досягнуто з меншим сумарним кутом
        pV -> MinSumAngle = CurSumAngle;
  /*              ShowMessage("Досягнуто "+IntToStr(pVD -> nomper)+", "+
                IntToStr((int)(pVD                                    ->
ver))+"MinSumAngle="+FloatToStr(CurSumAngle));
  */
        if (pV == &Vt) //Цільова вершина
         {
          Record = CurSumAngle;
          pRec = pNAD;
         }
       }
      //Додавання входу у видиму вершину
      pPAD = pV -> ptrAD;
      prev = NULL;
      //Пошук місця додавання входу
      while (pPAD && ((!kv && pPAD->ApprAngle > dv + 0.00001) ||
          (kv && pPAD->ApprAngle < dv - 0.00001)))
```

```
        {
          prev = pPAD;
          pPAD = pPAD -> next;
        }
       if (prev)  //Додавання у середину або у кінець списку
        {
          pNAD -> next = prev -> next;
          prev -> next = pNAD;
        }
       else    //Додавання у початок списку
        {
          pNAD -> next = pV -> ptrAD;
          pV -> ptrAD = pNAD;
        }
     }
   }
}
//---------------------------------------------------------------------------
//Звільнення памяті
void FreeMem(ListBlockDir *p)
{
 if (p -> next)
   FreeMem(p -> next);
 delete p;
}

//---------------------------------------------------------------------------
void formVDv(float x, float y, int nomp, point*v, bool kp)
//Визначення видимих вершин з поданої *v
```

```
{
 if (v -> MinSumAngle > Record - 0.00001) return;
 ListBlockDir *LBD = NULL, *p, *q, *previous;
 float vd, nd;
 int k = (nomp > 0)?nomp:0;
 //Аналіз перешкод
 for (int i = k; i < NP; i++)
  if ( MasObst[i].x > x + 0.001) //Перешкоду знайдено
    {
     vd =  (MasObst[i].Tv.y - y)/(MasObst[i].x - x);
     nd =  (MasObst[i].Tn.y - y)/(MasObst[i].x - x);
     if (!LBD) //Перша перешкода
       {
         insertListApproachDir(i, false, x, y, v, kp, nomp); //Додавання видимих
вершин
         insertListApproachDir(i, true, x, y, v, kp, nomp);
         LBD = new ListBlockDir;
         LBD -> next = NULL;
         LBD -> hd =  vd;
         LBD -> ld =  nd;
       }
     else  //Неперша перешкода
       {
         q = LBD;
         previous = NULL;
         while (q)
           {
             if ( vd >= q -> hd) //Верхня вершина вище блокованого сектора
               {
```

```
    insertListApproachDir(i, false, x, y, v, kp, nomp);
    if (nd >= q -> hd) //Уся перешкода вище
      {
       insertListApproachDir(i, true, x, y, v, kp, nomp);
      //Створення нового блокованого сектора
       p = new ListBlockDir;
       p->hd = vd;
       p->ld = nd;
       p->next = q;
       if(previous)
          previous->next = p;
       else
         LBD = p;
       break;
      }
    else
  //Розширення поточного блокованого сектора угору
      q -> hd = vd;
    if (nd > q -> ld) //Нижня вершина невидима
      break;
  }
if (vd <= q -> ld) //Верхня вершина нижче поточного сектора
  {
    previous = q;
    q = q -> next;
    continue;
  }
/*Об'єднання блокованих напрямків і перевірка видимості
нижньої вершини*/
```

```
p = q;
while (p)
  {
   if (p -> hd <= nd) //Нижня вершина видима
     {
      insertListApproachDir(i, true, x, y, v, kp, nomp);
      q -> ld = nd;
      break;
     }
   else
     if (p -> ld < nd) //Нижня вершина невидима
       {
        if (p != q) //Вилучення блокованого сектора
          {
           q -> ld = p -> ld;
           q -> next = p -> next;
           delete p;
          }
        break;
       }
     else
       {
        //Перехід до наступного блокованого сектора
        if (q != p)
          {
           q -> next = p;
           p = p -> next;
           delete (q -> next);
          }
```

```
                else
                  p = p -> next;
                }
            }//Кінець циклу while p
          if (!p)
            {
              q -> ld = nd;
              q -> next = NULL;
              insertListApproachDir(i, true, x, y, v, kp, nomp);
            }
          break;
        }//Кінець циклу while q
      if (!q) //Створення нового блокованого сектора
        {
          insertListApproachDir(i, false, x, y, v, kp, nomp);
          insertListApproachDir(i, true, x, y, v, kp, nomp);
          p = new ListBlockDir;
          p->hd = vd;
          p->ld = nd;
          p->next = NULL;
          previous->next = p;
        }
    }// else Неперша перешкода
  }//if Перешкоду знайдено; Кінець циклу for
if ( !LBD) //Перешкод між вершиною та ціллю нема
  insertListApproachDir(-1, false, x, y, v, kp, nomp);
else
 //Перевірка видимості цілі
  {
```

```
    q = LBD;
    vd = (Vt.y - y)/(Xt - x);
    while (q)
      {
       if (vd >= q -> hd)
         {
          insertListApproachDir(-1, false, x, y, v, kp, nomp);
          break;
         }
       else
         if (vd > q -> ld)
          break;
       q = q -> next;
      }
    if (!q) insertListApproachDir(-1, false, x, y, v, kp, nomp);
   }
  // Звільнення динамічної пам'яті
  if (LBD) FreeMem(LBD);
}
//---------------------------------------------------------------------------
void formGV()
// Формування графу видимості
{
  formVDv(Xs, Vs.y, -1, &Vs, false);
  for (int j = 0; j < NP; j++)
    {
     formVDv(MasObst[j].x, MasObst[j].Tv.y, j, &MasObst[j].Tv, false);
     formVDv(MasObst[j].x, MasObst[j].Tn.y, j, &MasObst[j].Tn, true);
    }
```

```
}
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
public:                 // User declarations
        __fastcall TForm1(TComponent* Owner);
};
struct ListApproachDir
{
  int nomper;
  bool ver;
  float ApprAngle, SumAngle;
  ListApproachDir *pPAD, *next;
};
struct point
{
  float y, MinSumAngle;
  ListApproachDir* ptrAD;
};
struct PathPoint
{
  float x, y;
  PathPoint *next;
};
struct obstacle
{
  point Tv, Tn;
  float x;
```

```
};
struct ListObstacle
{
  obstacle obst;
  ListObstacle *next;
};
struct ListBlockDir
{
 float hd, ld;
 ListBlockDir* next;
};
int Xn, NP = 0; // Кількість перешкод
float Xs, Xt; // Координати джерела та цілі
obstacle* MasObst;
ListObstacle  *pObst = NULL, *p;
point Vs, Vt; // Вершини - джерело та ціль
short reg;
bool draw;
//char b[100];
 point *pV;
 ListApproachDir *pAD, *pAVD, *prevAD, *CurpAD, *pNAD, *pRec;
 float x, y, dv, da, RoundAngle, CurSumAngle, Record;
float DefAngle(float x, float y, int kper, bool kv)
/*
 Визначення тангенсу напрямку на видиму вершину
 x,y - координати вихідної точки;
 kper, kv - параметри видимої вершини
*/
{
```

```
 float Xv;
  //Координати старої видимої вершини
 float Yv;
 if (kper == -1)
  {
   Xv = Xt;
   Yv = Vt.y;
  }
 else
  {
   Xv = MasObst[kper].x;
   if(kv)
     Yv = MasObst[kper].Tn.y;
   else
     Yv = MasObst[kper].Tv.y;
  }
 return (Yv - y)/(Xv - x);
}
//---------------------------------------------------------------------
void insertListApproachDir( int kper, bool kv, float x, float y,
                        point* p, bool kp, int nomp)
{
 /*
   kper -  видима перешкода
   kv -    тип видимої точки (верхня чи нижня)
   x, y -  координати вихідної точки
   p - покажчик на вихідну точку
   kp - тип вихідної точки (верхня чи нижня)
   nomp -  вихідна перешкода
```

```
*/
float dv, CurSumAngle;
point* pV;
ListApproachDir *pNAD, *pPAD, *prev, *CurpAD;
if (kper == -1)
  pV = &Vt;
else
  {
    pV = &(MasObst[kper].Tv);
    if(kv) pV = &(MasObst[kper].Tn);
  }
//Визначення кута переходу у наступну вершину
dv = DefAngle(x, y, kper, kv);
if (!(p->ptrAD))
  {
    // Вихідна точка - джерело
    pNAD = new ListApproachDir;
    pNAD->ver = false;
    pNAD->nomper = -1;
    pNAD->ApprAngle = dv;
    pNAD->SumAngle = 0;
    pNAD->next = NULL;
    pV->ptrAD = pNAD;
    pV -> MinSumAngle = 0;
    return;
  }
else
  {
    pPAD = p->ptrAD;
```

```
    //Визначення сумарного кута досягнення наступної вершини
    CurSumAngle = 6.28 * NP;
    while (pPAD && ((!kp && pPAD->ApprAngle > dv + 0.00001) ||
        (kp && pPAD->ApprAngle < dv - 0.00001)))
     {
      RoundAngle = (1 - 2*kp)*(atan(pPAD->ApprAngle) - atan(dv));
      if (pPAD->SumAngle + RoundAngle < CurSumAngle)
        {
         CurSumAngle = pPAD->SumAngle + RoundAngle;
         CurpAD = pPAD;
        }
      pPAD = pPAD -> next;
     }
    if (CurSumAngle < 6.28*NP - 0.0001) //Перехід можливий
     {
      pNAD = new ListApproachDir;
      pNAD->ver = kp;
      pNAD->nomper = nomp;
      pNAD->ApprAngle = dv;
      pNAD->SumAngle = CurSumAngle;
      pNAD->pPAD = CurpAD;
      if (CurSumAngle < pV -> MinSumAngle)
        { //Вершину досягнуто з меншим сумарним кутом
         pV -> MinSumAngle = CurSumAngle;
   /*              ShowMessage("Досягнуто "+IntToStr(pVD -> nomper)+", "+
                  IntToStr((int)(pVD                                    ->
ver))+"MinSumAngle="+FloatToStr(CurSumAngle));
   */
         if (pV == &Vt) //Цільова вершина
```

```
            {
              Record = CurSumAngle;
              pRec = pNAD;
            }
          }
        //Додавання входу у видиму вершину
        pPAD = pV -> ptrAD;
        prev = NULL;
        //Пошук місця додавання входу
        while (pPAD && ((!kv && pPAD->ApprAngle > dv + 0.00001) ||
            (kv && pPAD->ApprAngle < dv - 0.00001)))
          {
            prev = pPAD;
            pPAD = pPAD -> next;
          }
        if (prev)  //Додавання у середину або у кінець списку
          {
            pNAD -> next = prev -> next;
            prev -> next = pNAD;
          }
        else    //Додавання у початок списку
          {
            pNAD -> next = pV -> ptrAD;
            pV -> ptrAD = pNAD;
          }
      }
    }
}
//---------------------------------------------------------------------------
```

```
//Звільнення памяті
void FreeMem(ListBlockDir *p)
{
 if (p -> next)
   FreeMem(p -> next);
 delete p;
}

//---------------------------------------------------------------------------
void formVDv(float x, float y, int nomp, point*v, bool kp)
//Визначення видимих вершин з поданої *v
{
 if (v -> MinSumAngle > Record - 0.00001) return;
 ListBlockDir *LBD = NULL, *p, *q, *previous;
 float vd, nd;
 int k = (nomp > 0)?nomp:0;
 //Аналіз перешкод
 for (int i = k; i < NP; i++)
   if ( MasObst[i].x > x + 0.001) //Перешкоду знайдено
     {
      vd =  (MasObst[i].Tv.y - y)/(MasObst[i].x - x);
      nd =  (MasObst[i].Tn.y - y)/(MasObst[i].x - x);
      if (!LBD) //Перша перешкода
        {
          insertListApproachDir(i, false, x, y, v, kp, nomp); //Додавання видимих
вершин
          insertListApproachDir(i, true, x, y, v, kp, nomp);
          LBD = new ListBlockDir;
          LBD -> next = NULL;
```

```
    LBD -> hd =  vd;
    LBD -> ld =  nd;
   }
 else  //Неперша перешкода
   {
    q = LBD;
    previous = NULL;
    while (q)
      {
       if ( vd >= q -> hd) //Верхня вершина вище блокованого сектора
         {
          insertListApproachDir(i, false, x, y, v, kp, nomp);
          if (nd >= q -> hd) //Уся перешкода вище
            {
             insertListApproachDir(i, true, x, y, v, kp, nomp);
             //Створення нового блокованого сектора
             p = new ListBlockDir;
             p->hd = vd;
             p->ld = nd;
             p->next = q;
             if(previous)
                previous->next = p;
             else
               LBD = p;
             break;
            }
          else
         //Розширення поточного блокованого сектора угору
            q -> hd = vd;
```

```
    if (nd > q -> ld) //Нижня вершина невидима
      break;
   }
if (vd <= q -> ld) //Верхня вершина нижче поточного сектора
   {
    previous = q;
    q = q -> next;
    continue;
   }
/*Об'єднання блокованих напрямків і перевірка видимості
нижньої вершини*/
p = q;
while (p)
   {
    if (p -> hd <= nd) //Нижня вершина видима
      {
       insertListApproachDir(i, true, x, y, v, kp, nomp);
       q -> ld = nd;
       break;
      }
    else
      if (p -> ld < nd) //Нижня вершина невидима
        {
         if (p != q) //Вилучення блокованого сектора
           {
            q -> ld = p -> ld;
            q -> next = p -> next;
            delete p;
           }
```

```
            break;
          }
        else
          {
           //Перехід до наступного блокованого сектора
           if (q != p)
             {
              q -> next = p;
              p = p -> next;
              delete (q -> next);
             }
           else
             p = p -> next;
          }
      }//Кінець циклу while p
    if (!p)
      {
       q -> ld = nd;
       q -> next = NULL;
       insertListApproachDir(i, true, x, y, v, kp, nomp);
      }
    break;
  }//Кінець циклу while q
 if (!q) //Створення нового блокованого сектора
  {
   insertListApproachDir(i, false, x, y, v, kp, nomp);
   insertListApproachDir(i, true, x, y, v, kp, nomp);
   p = new ListBlockDir;
   p->hd = vd;
```

```
            p->ld = nd;
            p->next = NULL;
            previous->next = p;
           }
       }// else Неперша перешкода
    }//if Перешкоду знайдено; Кінець циклу for
if ( !LBD) //Перешкод між вершиною та ціллю нема
  insertListApproachDir(-1, false, x, y, v, kp, nomp);
else
 //Перевірка видимості цілі
  {
    q = LBD;
    vd = (Vt.y - y)/(Xt - x);
    while (q)
      {
       if (vd >= q -> hd)
         {
          insertListApproachDir(-1, false, x, y, v, kp, nomp);
          break;
         }
       else
         if (vd > q -> ld)
          break;
       q = q -> next;
      }
    if (!q) insertListApproachDir(-1, false, x, y, v, kp, nomp);
  }
// Звільнення динамічної пам'яті
if (LBD) FreeMem(LBD);
```

```
}
//---------------------------------------------------------------------------
void formGV()
// Формування графу видимості
{
  formVDv(Xs, Vs.y, -1, &Vs, false);
  for (int j = 0; j < NP; j++)
    {
      formVDv(MasObst[j].x, MasObst[j].Tv.y, j, &MasObst[j].Tv, false);
      formVDv(MasObst[j].x, MasObst[j].Tn.y, j, &MasObst[j].Tn, true);
    }
}
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
struct ListApproachDir
{
  int nomper;
  bool ver;
  float ApprAngle, SumAngle;
  ListApproachDir *pPAD, *next;
};
struct point
{
  float y, MinSumAngle;
  ListApproachDir* ptrAD;
};
struct PathPoint
```

```
{
 float x, y;
 PathPoint *next;
};
struct obstacle
{
 point Tv, Tn;
 float x;
};
struct ListObstacle
{
  obstacle obst;
  ListObstacle *next;
};
struct ListBlockDir
{
 float hd, ld;
 ListBlockDir* next;
};
int Xn, NP = 0; // Кількість перешкод
float Xs, Xt; // Координати джерела та цілі
obstacle* MasObst;
ListObstacle  *pObst = NULL, *p;
point Vs, Vt; // Вершини - джерело та ціль
short reg;
bool draw;
//char b[100];
 point *pV;
 ListApproachDir *pAD, *pAVD, *prevAD, *CurpAD, *pNAD, *pRec;
```

```
 float x, y, dv, da, RoundAngle, CurSumAngle, Record;
float DefAngle(float x, float y, int kper, bool kv)
/*
 Визначення тангенсу напрямку на видиму вершину
 x,y - координати вихідної точки;
 kper, kv - параметри видимої вершини
*/
{
 float Xv;
  //Координати старої видимої вершини
 float Yv;
 if (kper == -1)
  {
   Xv = Xt;
   Yv = Vt.y;
  }
 else
  {
   Xv = MasObst[kper].x;
   if(kv)
     Yv = MasObst[kper].Tn.y;
   else
     Yv = MasObst[kper].Tv.y;
  }
 return (Yv - y)/(Xv - x);
}
//---------------------------------------------------------------------
void insertListApproachDir( int kper, bool kv, float x, float y,
                            point* p, bool kp, int nomp)
```

```
{
 /*
  kper -  видима перешкода
  kv -    тип видимої точки (верхня чи нижня)
  x, y -  координати вихідної точки
  p - покажчик на вихідну точку
  kp - тип вихідної точки (верхня чи нижня)
  nomp -  вихідна перешкода
 */
 float dv, CurSumAngle;
 point* pV;
 ListApproachDir *pNAD, *pPAD, *prev, *CurpAD;
 if (kper == -1)
  pV = &Vt;
 else
  {
   pV = &(MasObst[kper].Tv);
   if(kv) pV = &(MasObst[kper].Tn);
  }
 //Визначення кута переходу у наступну вершину
 dv = DefAngle(x, y, kper, kv);
 if (!(p->ptrAD))
  {
   // Вихідна точка - джерело
   pNAD = new ListApproachDir;
   pNAD->ver = false;
   pNAD->nomper = -1;
   pNAD->ApprAngle = dv;
   pNAD->SumAngle = 0;
```

```
    pNAD->next = NULL;
    pV->ptrAD = pNAD;
    pV -> MinSumAngle = 0;
    return;
   }
 else
   {
    pPAD = p->ptrAD;
    //Визначення сумарного кута досягнення наступної вершини
    CurSumAngle = 6.28 * NP;
    while (pPAD && ((!kp && pPAD->ApprAngle > dv + 0.00001) ||
        (kp && pPAD->ApprAngle < dv - 0.00001)))
      {
       RoundAngle = (1 - 2*kp)*(atan(pPAD->ApprAngle) - atan(dv));
       if (pPAD->SumAngle + RoundAngle < CurSumAngle)
         {
          CurSumAngle = pPAD->SumAngle + RoundAngle;
          CurpAD = pPAD;
         }
       pPAD = pPAD -> next;
      }
    if (CurSumAngle < 6.28*NP - 0.0001) //Перехід можливий
      {
       pNAD = new ListApproachDir;
       pNAD->ver = kp;
       pNAD->nomper = nomp;
       pNAD->ApprAngle = dv;
       pNAD->SumAngle = CurSumAngle;
       pNAD->pPAD = CurpAD;
```

```
        if (CurSumAngle < pV -> MinSumAngle)
          { //Вершину досягнуто з меншим сумарним кутом
           pV -> MinSumAngle = CurSumAngle;
    /*            ShowMessage("Досягнуто "+IntToStr(pVD -> nomper)+", "+
                  IntToStr((int)(pVD                                    ->
ver))+"MinSumAngle="+FloatToStr(CurSumAngle));
    */
           if (pV == &Vt) //Цільова вершина
             {
              Record = CurSumAngle;
              pRec = pNAD;
             }
          }
        //Додавання входу у видиму вершину
        pPAD = pV -> ptrAD;
        prev = NULL;
        //Пошук місця додавання входу
        while (pPAD && ((!kv && pPAD->ApprAngle > dv + 0.00001) ||
            (kv && pPAD->ApprAngle < dv - 0.00001)))
          {
           prev = pPAD;
           pPAD = pPAD -> next;
          }
        if (prev)  //Додавання у середину або у кінець списку
          {
           pNAD -> next = prev -> next;
           prev -> next = pNAD;
          }
        else    //Додавання у початок списку
```

```
          {
            pNAD -> next = pV -> ptrAD;
            pV -> ptrAD = pNAD;
          }
        }
    }
}
//---------------------------------------------------------------------------
//Звільнення памяті
void FreeMem(ListBlockDir *p)
{
 if (p -> next)
   FreeMem(p -> next);
 delete p;
}

//---------------------------------------------------------------------------
void formVDv(float x, float y, int nomp, point*v, bool kp)
//Визначення видимих вершин з поданої *v
{
 if (v -> MinSumAngle > Record - 0.00001) return;
 ListBlockDir *LBD = NULL, *p, *q, *previous;
 float vd, nd;
 int k = (nomp > 0)?nomp:0;
 //Аналіз перешкод
 for (int i = k; i < NP; i++)
   if ( MasObst[i].x > x + 0.001) //Перешкоду знайдено
     {
       vd =  (MasObst[i].Tv.y - y)/(MasObst[i].x - x);
```

```
nd =  (MasObst[i].Tn.y - y)/(MasObst[i].x - x);
if (!LBD) //Перша перешкода
  {
   insertListApproachDir(i, false, x, y, v, kp, nomp); //Додавання видимих
вершин
   insertListApproachDir(i, true, x, y, v, kp, nomp);
   LBD = new ListBlockDir;
   LBD -> next = NULL;
   LBD -> hd =  vd;
   LBD -> ld =  nd;
  }
else  //Неперша перешкода
  {
   q = LBD;
   previous = NULL;
   while (q)
    {
     if ( vd >= q -> hd) //Верхня вершина вище блокованого сектора
       {
        insertListApproachDir(i, false, x, y, v, kp, nomp);
        if (nd >= q -> hd) //Уся перешкода вище
          {
           insertListApproachDir(i, true, x, y, v, kp, nomp);
           //Створення нового блокованого сектора
           p = new ListBlockDir;
           p->hd = vd;
           p->ld = nd;
           p->next = q;
           if(previous)
```

```
          previous->next = p;
        else
          LBD = p;
        break;
      }
    else
  //Розширення поточного блокованого сектора угору
      q -> hd = vd;
    if (nd > q -> ld) //Нижня вершина невидима
      break;
  }
if (vd <= q -> ld) //Верхня вершина нижче поточного сектора
  {
    previous = q;
    q = q -> next;
    continue;
  }
/*Об'єднання блокованих напрямків і перевірка видимості
нижньої вершини*/
p = q;
while (p)
  {
    if (p -> hd <= nd) //Нижня вершина видима
      {
        insertListApproachDir(i, true, x, y, v, kp, nomp);
        q -> ld = nd;
        break;
      }
    else
```

```
      if (p -> ld < nd) //Нижня вершина невидима
        {
         if (p != q) //Вилучення блокованого сектора
           {
            q -> ld = p -> ld;
            q -> next = p -> next;
            delete p;
           }
         break;
        }
      else
        {
         //Перехід до наступного блокованого сектора
         if (q != p)
           {
            q -> next = p;
            p = p -> next;
            delete (q -> next);
           }
         else
           p = p -> next;
        }
    }//Кінець циклу while p
  if (!p)
    {
     q -> ld = nd;
     q -> next = NULL;
     insertListApproachDir(i, true, x, y, v, kp, nomp);
    }
```

```
            break;
          }//Кінець циклу while q
        if (!q) //Створення нового блокованого сектора
          {
            insertListApproachDir(i, false, x, y, v, kp, nomp);
            insertListApproachDir(i, true, x, y, v, kp, nomp);
            p = new ListBlockDir;
            p->hd = vd;
            p->ld = nd;
            p->next = NULL;
            previous->next = p;
          }
      }// else Неперша перешкода
   }//if Перешкоду знайдено; Кінець циклу for
if ( !LBD) //Перешкод між вершиною та ціллю нема
  insertListApproachDir(-1, false, x, y, v, kp, nomp);
else
 //Перевірка видимості цілі
  {
    q = LBD;
    vd = (Vt.y - y)/(Xt - x);
    while (q)
      {
       if (vd >= q -> hd)
         {
           insertListApproachDir(-1, false, x, y, v, kp, nomp);
           break;
         }
       else
```

```
        if (vd > q -> ld)
          break;
        q = q -> next;
       }
     if (!q) insertListApproachDir(-1, false, x, y, v, kp, nomp);
    }
  // Звільнення динамічної пам'яті
  if (LBD) FreeMem(LBD);
}
//---------------------------------------------------------------------------
void formGV()
// Формування графу видимості
{
  formVDv(Xs, Vs.y, -1, &Vs, false);
  for (int j = 0; j < NP; j++)
    {
     formVDv(MasObst[j].x, MasObst[j].Tv.y, j, &MasObst[j].Tv, false);
     formVDv(MasObst[j].x, MasObst[j].Tn.y, j, &MasObst[j].Tn, true);
    }
}
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif

//---------------------------------------------------------------------------

#ifndef ThreadH
#define ThreadH
```

```
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <math.h>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
#include <algorithm>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:        // IDE-managed Components
        TButton *BPathSearch;
        TImage *Image1;
        TPopupMenu *PopupMenu1;
        TMenuItem *N1;
        TMenuItem *N2;
        TMenuItem *N3;
        void __fastcall BPathSearchClick(TObject *Sender);
        void __fastcall N1Click(TObject *Sender);
        void __fastcall N2Click(TObject *Sender);
        void __fastcall N3Click(TObject *Sender);
        void __fastcall Image1MouseDown(TObject *Sender,
          TMouseButton Button, TShiftState Shift, int X, int Y);
        void __fastcall Image1MouseMove(TObject *Sender, TShiftState Shift,
          int X, int Y);
```

```
    void __fastcall Image1MouseUp(TObject *Sender, TMouseButton Button,
      TShiftState Shift, int X, int Y);
private:       // User declarations
public:              // User declarations
    __fastcall TForm1(TComponent* Owner);
};
struct ListApproachDir
{
 int nomper;
 bool ver;
 float ApprAngle, SumAngle;
 ListApproachDir *pPAD, *next;
};
struct point
{
 float y, MinSumAngle;
 ListApproachDir* ptrAD;
};
struct PathPoint
{
 float x, y;
 PathPoint *next;
};
struct obstacle
{
 point Tv, Tn;
 float x;
};
struct ListObstacle
```

```
{
  obstacle obst;
  ListObstacle *next;
};
struct ListBlockDir
{
 float hd, ld;
 ListBlockDir* next;
};
int Xn, NP = 0; // Кількість перешкод
float Xs, Xt; // Координати джерела та цілі
obstacle* MasObst;
ListObstacle  *pObst = NULL, *p;
point Vs, Vt; // Вершини - джерело та ціль
short reg;
bool draw;
//char b[100];
 point *pV;
 ListApproachDir *pAD, *pAVD, *prevAD, *CurpAD, *pNAD, *pRec;
 float x, y, dv, da, RoundAngle, CurSumAngle, Record;
float DefAngle(float x, float y, int kper, bool kv)
/*
 Визначення тангенсу напрямку на видиму вершину
 x,y - координати вихідної точки;
 kper, kv - параметри видимої вершини
*/
{
 float Xv;
   //Координати старої видимої вершини
```

```
 float Yv;
 if (kper == -1)
   {
    Xv = Xt;
    Yv = Vt.y;
   }
 else
   {
    Xv = MasObst[kper].x;
    if(kv)
      Yv = MasObst[kper].Tn.y;
    else
      Yv = MasObst[kper].Tv.y;
   }
 return (Yv - y)/(Xv - x);
}
//---------------------------------------------------------------------
void insertListApproachDir( int kper, bool kv, float x, float y,
                        point* p, bool kp, int nomp)
{
 /*
   kper -  видима перешкода
   kv -    тип видимої точки (верхня чи нижня)
   x, y -  координати вихідної точки
   p - покажчик на вихідну точку
   kp - тип вихідної точки (верхня чи нижня)
   nomp -  вихідна перешкода
 */
 float dv, CurSumAngle;
```

```
point* pV;
ListApproachDir *pNAD, *pPAD, *prev, *CurpAD;
if (kper == -1)
  pV = &Vt;
else
  {
    pV = &(MasObst[kper].Tv);
    if(kv) pV = &(MasObst[kper].Tn);
  }
//Визначення кута переходу у наступну вершину
dv = DefAngle(x, y, kper, kv);
if (!(p->ptrAD))
  {
    // Вихідна точка - джерело
    pNAD = new ListApproachDir;
    pNAD->ver = false;
    pNAD->nomper = -1;
    pNAD->ApprAngle = dv;
    pNAD->SumAngle = 0;
    pNAD->next = NULL;
    pV->ptrAD = pNAD;
    pV -> MinSumAngle = 0;
    return;
  }
else
  {
    pPAD = p->ptrAD;
    //Визначення сумарного кута досягнення наступної вершини
    CurSumAngle = 6.28 * NP;
```

```
    while (pPAD && ((!kp && pPAD->ApprAngle > dv + 0.00001) ||
        (kp && pPAD->ApprAngle < dv - 0.00001)))
      {
       RoundAngle = (1 - 2*kp)*(atan(pPAD->ApprAngle) - atan(dv));
       if (pPAD->SumAngle + RoundAngle < CurSumAngle)
         {
          CurSumAngle = pPAD->SumAngle + RoundAngle;
          CurpAD = pPAD;
         }
       pPAD = pPAD -> next;
      }
    if (CurSumAngle < 6.28*NP - 0.0001) //Перехід можливий
      {
       pNAD = new ListApproachDir;
       pNAD->ver = kp;
       pNAD->nomper = nomp;
       pNAD->ApprAngle = dv;
       pNAD->SumAngle = CurSumAngle;
       pNAD->pPAD = CurpAD;
       if (CurSumAngle < pV -> MinSumAngle)
         { //Вершину досягнуто з меншим сумарним кутом
          pV -> MinSumAngle = CurSumAngle;
   /*              ShowMessage("Досягнуто "+IntToStr(pVD -> nomper)+", "+
                  IntToStr((int)(pVD                                    ->
ver))+"MinSumAngle="+FloatToStr(CurSumAngle));
   */
          if (pV == &Vt) //Цільова вершина
            {
             Record = CurSumAngle;
```

```
            pRec = pNAD;
           }
        }
      //Додавання входу у видиму вершину
      pPAD = pV -> ptrAD;
      prev = NULL;
      //Пошук місця додавання входу
      while (pPAD && ((!kv && pPAD->ApprAngle > dv + 0.00001) ||
          (kv && pPAD->ApprAngle < dv - 0.00001)))
        {
         prev = pPAD;
         pPAD = pPAD -> next;
        }
      if (prev)  //Додавання у середину або у кінець списку
        {
         pNAD -> next = prev -> next;
         prev -> next = pNAD;
        }
      else    //Додавання у початок списку
        {
         pNAD -> next = pV -> ptrAD;
         pV -> ptrAD = pNAD;
        }
     }
   }
}
//---------------------------------------------------------------------------
//Звільнення памяті
void FreeMem(ListBlockDir *p)
```

```
{
 if (p -> next)
   FreeMem(p -> next);
 delete p;
}

//---------------------------------------------------------------------------
void formVDv(float x, float y, int nomp, point*v, bool kp)
//Визначення видимих вершин з поданої *v
{
 if (v -> MinSumAngle > Record - 0.00001) return;
 ListBlockDir *LBD = NULL, *p, *q, *previous;
 float vd, nd;
 int k = (nomp > 0)?nomp:0;
 //Аналіз перешкод
 for (int i = k; i < NP; i++)
   if ( MasObst[i].x > x + 0.001) //Перешкоду знайдено
     {
      vd =  (MasObst[i].Tv.y - y)/(MasObst[i].x - x);
      nd =  (MasObst[i].Tn.y - y)/(MasObst[i].x - x);
      if (!LBD) //Перша перешкода
        {
          insertListApproachDir(i, false, x, y, v, kp, nomp); //Додавання видимих
вершин
          insertListApproachDir(i, true, x, y, v, kp, nomp);
          LBD = new ListBlockDir;
          LBD -> next = NULL;
          LBD -> hd =  vd;
          LBD -> ld =  nd;
```

```
  }
else  //Неперша перешкода
  {
   q = LBD;
   previous = NULL;
   while (q)
    {
     if ( vd >= q -> hd) //Верхня вершина вище блокованого сектора
      {
       insertListApproachDir(i, false, x, y, v, kp, nomp);
       if (nd >= q -> hd) //Уся перешкода вище
        {
         insertListApproachDir(i, true, x, y, v, kp, nomp);
         //Створення нового блокованого сектора
         p = new ListBlockDir;
         p->hd = vd;
         p->ld = nd;
         p->next = q;
         if(previous)
           previous->next = p;
         else
          LBD = p;
         break;
        }
       else
       //Розширення поточного блокованого сектора угору
         q -> hd = vd;
       if (nd > q -> ld) //Нижня вершина невидима
         break;
```

```
  }
if (vd <= q -> ld) //Верхня вершина нижче поточного сектора
  {
   previous = q;
   q = q -> next;
   continue;
  }
/*Об'єднання блокованих напрямків і перевірка видимості
нижньої вершини*/
p = q;
while (p)
  {
   if (p -> hd <= nd) //Нижня вершина видима
     {
      insertListApproachDir(i, true, x, y, v, kp, nomp);
      q -> ld = nd;
      break;
     }
   else
    if (p -> ld < nd) //Нижня вершина невидима
      {
       if (p != q) //Вилучення блокованого сектора
         {
          q -> ld = p -> ld;
          q -> next = p -> next;
          delete p;
         }
       break;
      }
```

```
        else
          {
           //Перехід до наступного блокованого сектора
           if (q != p)
             {
              q -> next = p;
              p = p -> next;
              delete (q -> next);
             }
           else
             p = p -> next;
          }
      }//Кінець циклу while p
    if (!p)
      {
       q -> ld = nd;
       q -> next = NULL;
       insertListApproachDir(i, true, x, y, v, kp, nomp);
      }
    break;
  }//Кінець циклу while q
if (!q) //Створення нового блокованого сектора
  {
   insertListApproachDir(i, false, x, y, v, kp, nomp);
   insertListApproachDir(i, true, x, y, v, kp, nomp);
   p = new ListBlockDir;
   p->hd = vd;
   p->ld = nd;
   p->next = NULL;
```

```
            previous->next = p;
          }
        }// else Неперша перешкода
    }//if Перешкоду знайдено; Кінець циклу for
 if ( !LBD) //Перешкод між вершиною та ціллю нема
   insertListApproachDir(-1, false, x, y, v, kp, nomp);
 else
  //Перевірка видимості цілі
   {
    q = LBD;
    vd = (Vt.y - y)/(Xt - x);
    while (q)
      {
       if (vd >= q -> hd)
         {
          insertListApproachDir(-1, false, x, y, v, kp, nomp);
          break;
         }
       else
         if (vd > q -> ld)
          break;
       q = q -> next;
      }
    if (!q) insertListApproachDir(-1, false, x, y, v, kp, nomp);
   }
  // Звільнення динамічної пам'яті
  if (LBD) FreeMem(LBD);
}
//---------------------------------------------------------------------------
```

```
void formGV()
// Формування графу видимості
{
  formVDv(Xs, Vs.y, -1, &Vs, false);
  for (int j = 0; j < NP; j++)
    {
      formVDv(MasObst[j].x, MasObst[j].Tv.y, j, &MasObst[j].Tv, false);
      formVDv(MasObst[j].x, MasObst[j].Tn.y, j, &MasObst[j].Tn, true);
    }
}
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

ЛІСТИНГИ ПРОГРАМ РЕАЛІЗАЦІЇ АЛГОРИТМУ ДЕЙКСТРИ ПРИ ВИЗНАЧЕННІ ФОРМИ ЗАПРАВКИ НИТКИ

```
unit MainUnit;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Math, Menus;
type
  TForm1 = class(TForm)
    Image1: TImage;
    ScrollBar1: TScrollBar;
    ScrollBar2: TScrollBar;
    Panel1: TPanel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit4: TEdit;
    Button3: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Button8: TButton;
    Panel2: TPanel;
    Edit7: TEdit;
    Edit8: TEdit;
    Label6: TLabel;
    Label7: TLabel;
    Button9: TButton;
    Button10: TButton;
    Label3: TLabel;
```

MainMenu1: TMainMenu;
N12: TMenuItem;
N13: TMenuItem;
N14: TMenuItem;
N15: TMenuItem;
N16: TMenuItem;
N17: TMenuItem;
N18: TMenuItem;
N19: TMenuItem;
N20: TMenuItem;
N21: TMenuItem;
N22: TMenuItem;
N23: TMenuItem;
N24: TMenuItem;
N26: TMenuItem;
N27: TMenuItem;
N28: TMenuItem;
N29: TMenuItem;
N30: TMenuItem;
N31: TMenuItem;
N32: TMenuItem;
Label5: TLabel;
Label8: TLabel;
N2: TMenuItem;
N3: TMenuItem;
N4: TMenuItem;
N5: TMenuItem;
N6: TMenuItem;
OpenDialog1: TOpenDialog;

```
SaveDialog1: TSaveDialog;
MainMenu2: TMainMenu;
N1: TMenuItem;
Button1: TButton;
procedure ScrollBar1Scroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
procedure ScrollBar2Scroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
procedure Button3Click(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure Button8Click(Sender: TObject);
procedure Button9Click(Sender: TObject);
procedure Button10Click(Sender: TObject);
procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Image1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure N29Click(Sender: TObject);
procedure N27Click(Sender: TObject);
procedure N28Click(Sender: TObject);
procedure N31Click(Sender: TObject);
procedure N32Click(Sender: TObject);
procedure N16Click(Sender: TObject);
procedure N17Click(Sender: TObject);
procedure N18Click(Sender: TObject);
procedure N19Click(Sender: TObject);
procedure N13Click(Sender: TObject);
```

```
  procedure N14Click(Sender: TObject);
  procedure N21Click(Sender: TObject);
  procedure N20Click(Sender: TObject);
  procedure N23Click(Sender: TObject);
  procedure N24Click(Sender: TObject);
  procedure N2Click(Sender: TObject);
  procedure N3Click(Sender: TObject);
  procedure N4Click(Sender: TObject);
  procedure N5Click(Sender: TObject);
  procedure N6Click(Sender: TObject);
  procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

type
  PMyListE = ^ElList;    //
  ElList = record
    x_c,y_c,R:real;
    elnumber:string{char};
end;

type
  PMyListM = ^MdList;    //
  MdList = record
    numrebra:integer;
    x1,y1,x2,y2:real;
```

```
    el1,el2:string;
    dest_in:char;
    naklon:real;
end;
type
  PListL = ^LList;    //список L
  LList = record
    NumEl:string;
    x1,y1,x2,y2:real;
    numrebra:integer;
    sumang:real;
    path:string;
    vn:char;
end;
type
  PListM = ^MList;    //
  MList = record
    NumEl:string;
    x1,y1,x2,y2:real;
    numrebra:integer;
    sumang:real;
    path:string;
    vn:char;
end;
type
  PListSm = ^SmList;    //
  SmList = record
   numreb:integer;
   numsmreb:array[1..50]of integer;
```

```
    kolsmreber:integer;
  end;
  type
   MyD = record          //
      NumEl:string;
      x1,y1,x2,y2:real;
      numrebra:integer;
      sumang:real;
      path:string;
      vn:char;
  end;
var
  Form1: TForm1;
  ksi,eta,x0,y0,math_x,math_y:real; //
  mx,my:integer; //

  MyListE: TList;  //
  ElRecord1: PMyListE; //
  MyListM: TList;     //
  MdRecord: PMyListM; //
  ListL: TList;      //список L
  ListM: TList;      //список M
  MRecord: PListM;   // M
  ListSm: TList;     //
  ListReb: TList;
  RebRecord: PMyListM;
  scr_pos1,scr_pos2:integer; //
  xmax,ymax:integer; //
```

```
  num_el,num_rebra:integer; //
  is_source,is_target:boolean; //
  source,target:boolean; //
  clear:boolean; //
  editing:boolean; //
  moving:boolean; //
  num:integer; //
  CopyElRecord:PMyListE;
  x_start,y_start:real;  //
  elnum:string;          //...
  finding_rebro:boolean; //
implementation
 uses Unit2;
{$R *.dfm}
procedure Koord(u,v:real);
//
begin
  if (u<320)and(v>240)then
   begin
     ksi:=x0+u*mx;
     eta:=y0+v*my;
   end;
  if (u<320)and(v<240)then
   begin
     ksi:=x0+u*mx;
     eta:=y0-v*my;
   end;
  if (u>320)and(v<240)then
   begin
```

```
    ksi:=x0-u*mx;
    eta:=y0-v*my;
   end;

  if (u>320)and(v>240)then
   begin
    ksi:=x0-u*mx;
    eta:=y0+v*my;
   end;
end;
procedure MathKoord(ks,et:real);
//
begin
  math_x:=(ks-x0)/mx;
  math_y:=(et-y0)/(-my);
end;
function FindAngle(x1_1,y1_1,x1_2,y1_2,x2_1,y2_1,x2_2,y2_2:real):real;
//)
var a,b:real;
begin
 if (x1_1=x1_2)and(y1_1=y1_2) then
 FindAngle:=0
 else
  begin
   if (x1_2=x1_1)or(x2_2=x2_1) then
    begin
     FindAngle:=0;
     exit;
    end;
```

```
  a:=arctan((y1_2-y1_1)/(x1_2-x1_1))*180/pi;
  b:=arctan((y2_2-y2_1)/(x2_2-x2_1))*180/pi;
  if (a>=0)and(b>=0)and(b<a) then
   FindAngle:=a-b
  else if (a>=0)and(b>=0)and(b>a) then
   FindAngle:=b-a
  else if (a>=0)and(b<=0) then
   FindAngle:=a+abs(b)
  else if (a<=0)and(b<=0)and(b<a) then
   FindAngle:=abs(b)-abs(a)
  else if (a<=0)and(b<=0)and(b>a) then
   FindAngle:=abs(a)-abs(b)
  else {if (a<0)and(b>0) then }
   FindAngle:=abs(a)+b;
 end;
end;
function FindNaklon(x1,y1,x2,y2:real):real;
//пошук кута нахилу дотичної
begin
 if x1=x2 then
 begin
 FindNaklon:=0;
 exit;
 end
 else
 begin
  FindNaklon:=arctan((y2-y1)/(x2-x1))*180/pi;
 end;
```

```
end;
procedure DrawAxis(sx,fx,sy,fy:integer);
//
var
  i,k:integer;
begin
  form1.Image1.Canvas.Pen.Color:=clBlack;
  k:=0;
  for i:=sx to fx do
   begin
     Koord(k,0);
     k:=k+1;
     form1.Image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
     form1.Image1.Canvas.LineTo(trunc(ksi),trunc(eta));
     form1.Image1.Canvas.Pen.Width:=1;
     form1.Image1.Canvas.LineTo(trunc(ksi),0);
     if i<>0 then
       form1.Image1.Canvas.TextOut(trunc(ksi),trunc(eta),inttostr(i));
     form1.Image1.Canvas.Pen.Width:=4;
   end;
  k:=0;
  for i:=sy to fy do
   begin
     Koord(0,k);
     k:=k+1;
     form1.Image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
     form1.Image1.Canvas.LineTo(trunc(ksi),trunc(eta));
```

```
    form1.Image1.Canvas.Pen.Width:=1;
    form1.Image1.Canvas.LineTo(640,trunc(eta));
    if i<>0 then
      form1.Image1.Canvas.TextOut(trunc(ksi),trunc(eta),inttostr(i));
    form1.Image1.Canvas.Pen.Width:=4;
   end;
  form1.Image1.Canvas.Pen.Width:=1;
  form1.Image1.Canvas.Pen.Color:=clBlack;
end;
Procedure DrawMyEllipse(xc,yc,R:real;mytype:string);
//рисование окружности
var R_p,xc_p,yc_p:integer; //
begin
  koord(xc,yc);
  xc_p:=trunc(ksi);
  yc_p:=trunc(eta);
  koord(R,0);
  R_p:=trunc(ksi);
  form1.Image1.Canvas.Pen.Color:=clRed;
  form1.Image1.Canvas.Ellipse(xc_p-R_p,yc_p+R_p,xc_p+R_p,yc_p-R_p);
    Form1.Image1.Canvas.TextOut(xc_p-2,yc_p-5,mytype);
  form1.Image1.Canvas.Pen.Color:=clBlack;
end;
Procedure ReDrawEllipse(shift_x,shift_y:integer);
//
var
  i:integer;
  xc,yc,R:real;//
  x1,x2,y1,y2:real;
```

```
 path:string; //
begin
 if yListE.Count<>0 then
 //
  begin
    for i:=0 to MyListE.Count-1 do
     begin
       ElRecord1:=MyListE.Items[i];
       xc:=ElRecord1.x_c-shift_x;
       yc:=ElRecord1.y_c-shift_y;
       R:=ElRecord1.R;
       DrawMyEllipse(xc,yc,R,ElRecord1.elnumber)
     end;
  end;
 if clear<>true then
 begin
  if MyListM.Count<>0 then
  //перерисовка рёбер
   begin
     form1.image1.Canvas.Pen.Color:=clBlue;
     for i:=0 to MyListM.Count-1 do
      begin
        MdRecord:=MyListM.Items[i];
        x1:=MdRecord.x1-shift_x;
        y1:=MdRecord.y1-shift_y;
        Koord(x1,y1);
        form1.Image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
        x2:=MdRecord.x2-shift_x;
        y2:=MdRecord.y2-shift_y;
```

```
      Koord(x2,y2);
      form1.Image1.Canvas.LineTo(trunc(ksi),trunc(eta));
    end;
   form1.image1.Canvas.Pen.Color:=clBlack;
  end;
end;
 if ListM.Count<>0 then
 //   begin
   MRecord:=ListM.Items[ListM.Count-1];
   Form1.Image1.Canvas.Pen.Width:=2;
   Form1.Image1.Canvas.Pen.Color:=clGreen;
   path:=MRecord.path;
   delete(path,length(path),1);
   while path<>'' do
    begin
     x1:=strtofloat(copy(path,1,pos('*',path)-1));
     delete(path,1,pos('*',path));
     y1:=strtofloat(copy(path,1,pos('*',path)-1));
     delete(path,1,pos('*',path));
     x2:=strtofloat(copy(path,1,pos('*',path)-1));
     delete(path,1,pos('*',path));
     if pos('*',path)=0 then
      begin
       y2:=strtofloat(path);
       delete(path,1,length(path));
      end
      else
      begin
       y2:=strtofloat(copy(path,1,pos('*',path)-1));
```

```
        delete(path,1,pos('*',path));
       end;
      Koord(x1-shift_x,y1-shift_y);
      Form1.Image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
      Koord(x2-shift_x,y2-shift_y);
      Form1.Image1.Canvas.LineTo(trunc(ksi),trunc(eta));
     end;
    Form1.Image1.Canvas.Pen.Width:=1;
    Form1.Image1.Canvas.Pen.Color:=clBlack;
   end;
end;
procedure Sort;
//
var i,j,flag:integer;
   TmpRecord1,TmpRecord2: PMyListE; //
begin
 i:=1;
 repeat
  flag:=0;
  for j:=MyListE.Count-1 downto i do
   begin
    ElRecord1:=MyListE.Items[j];
    TmpRecord1:=MyListE.Items[j-1];
    if (ElRecord1.x_c=TmpRecord1.x_c)and(ElRecord1.y_c<TmpRecord1.y_c)
then
     begin
      TmpRecord2:=TmpRecord1;
      MyListE.Items[j-1]:=MyListE.Items[j];
      MyListE.Items[j]:=TmpRecord2;
```

```
    end;
   if (ElRecord1.x_c<TmpRecord1.x_c) then
    begin
     TmpRecord2:=TmpRecord1;
     MyListE.Items[j-1]:=MyListE.Items[j];
     MyListE.Items[j]:=TmpRecord2;
     flag:=1;
    end;
  end;
 i:=i+1;
until flag=0;
//
j:=1;
for i:=0 to MyListE.Count-1 do
 begin
  ElRecord1:=MyListE.Items[i];
  if (ElRecord1.elnumber<>'s')and(ElRecord1.elnumber<>'t')then
   begin
    ElRecord1.elnumber:=inttostr(j);
    MyListE.Items[i]:=ElRecord1;
    j:=j+1;
   end;
 end;
end;
procedure BuildTangent(el1,el2:string;xc1,yc1,R1,xc2,yc2,R2:real);
//
var
   d:real; //
   beta:real; //
```

```
   sn_beta,cs_beta:real;  //
   sn_alpha,cs_alpha:real; //
   x1,y1,x2,y2:real; //
begin
 if xc1=xc2 then
 d:=abs(yc1-yc2)
 else
 d:=sqrt((yc2-yc1)*(yc2-yc1)+(xc2-xc1)*(xc2-xc1));
 if xc2=xc1 then beta:=90*pi/180 else beta:=arctan((yc2-yc1)/(xc2-xc1));
 sn_beta:=sin(beta);
 cs_beta:=cos(beta);
 //
 sn_alpha:=(R2-R1)/d;
 cs_alpha:=sqrt(1-sn_alpha*sn_alpha);
 //
 x1:=((xc1*cs_beta+yc1*sn_beta)-R1*sn_alpha)*cs_beta-((-
xc1*sn_beta+yc1*cs_beta)+R1*cs_alpha)*sn_beta;
 y1:=((xc1*cs_beta+yc1*sn_beta)-R1*sn_alpha)*sn_beta+((-
xc1*sn_beta+yc1*cs_beta)+R1*cs_alpha)*cs_beta;
 x2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*cs_beta-((-
xc2*sn_beta+yc2*cs_beta)+R2*cs_alpha)*sn_beta;
 y2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*sn_beta+((-
xc2*sn_beta+yc2*cs_beta)+R2*cs_alpha)*cs_beta;
 num_rebra:=num_rebra+1;
 New(MdRecord); //)
  MdRecord.el1:=el1;
  MdRecord.el2:=el2;
  MdRecord.numrebra:=num_rebra;
  MdRecord.x1:=x1;
```

```
 MdRecord.y1:=y1;
 MdRecord.x2:=x2;
 MdRecord.y2:=y2;
 MdRecord.naklon:=FindNaklon(x1,y1,x2,y2);
MyListM.Add(MdRecord);
//
x1:=((xc1*cs_beta+yc1*sn_beta)-R1*sn_alpha)*cs_beta-((-
xc1*sn_beta+yc1*cs_beta)-R1*cs_alpha)*sn_beta;
y1:=((xc1*cs_beta+yc1*sn_beta)-R1*sn_alpha)*sn_beta+((-
xc1*sn_beta+yc1*cs_beta)-R1*cs_alpha)*cs_beta;
x2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*cs_beta-((-
xc2*sn_beta+yc2*cs_beta)-R2*cs_alpha)*sn_beta;
y2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*sn_beta+((-
xc2*sn_beta+yc2*cs_beta)-R2*cs_alpha)*cs_beta;
num_rebra:=num_rebra+1;
New(MdRecord); //)
 MdRecord.el1:=el1;
 MdRecord.el2:=el2;
 MdRecord.numrebra:=num_rebra;
 MdRecord.x1:=x1;
 MdRecord.y1:=y1;
 MdRecord.x2:=x2;
 MdRecord.y2:=y2;
 MdRecord.naklon:=FindNaklon(x1,y1,x2,y2);
MyListM.Add(MdRecord);
//-------------------------------------------------------------------------
//
sn_alpha:=(R1+R2)/d;
cs_alpha:=sqrt(1-sn_alpha*sn_alpha);
```

```
//
 x1:=((xc1*cs_beta+yc1*sn_beta)+R1*sn_alpha)*cs_beta-((-
xc1*sn_beta+yc1*cs_beta)+R1*cs_alpha)*sn_beta;
 y1:=((xc1*cs_beta+yc1*sn_beta)+R1*sn_alpha)*sn_beta+((-
xc1*sn_beta+yc1*cs_beta)+R1*cs_alpha)*cs_beta;
 x2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*cs_beta-((-
xc2*sn_beta+yc2*cs_beta)-R2*cs_alpha)*sn_beta;
 y2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*sn_beta+((-
xc2*sn_beta+yc2*cs_beta)-R2*cs_alpha)*cs_beta;
 num_rebra:=num_rebra+1;
 New(MdRecord); //)
  MdRecord.el1:=el1;
  MdRecord.el2:=el2;
  MdRecord.numrebra:=num_rebra;
  MdRecord.x1:=x1;
  MdRecord.y1:=y1;
  MdRecord.x2:=x2;
  MdRecord.y2:=y2;
  MdRecord.naklon:=FindNaklon(x1,y1,x2,y2);
 MyListM.Add(MdRecord);
 //
 x1:=((xc1*cs_beta+yc1*sn_beta)+R1*sn_alpha)*cs_beta-((-
xc1*sn_beta+yc1*cs_beta)-R1*cs_alpha)*sn_beta;
 y1:=((xc1*cs_beta+yc1*sn_beta)+R1*sn_alpha)*sn_beta+((-
xc1*sn_beta+yc1*cs_beta)-R1*cs_alpha)*cs_beta;
 x2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*cs_beta-((-
xc2*sn_beta+yc2*cs_beta)+R2*cs_alpha)*sn_beta;
```

```
 y2:=((xc2*cs_beta+yc2*sn_beta)-R2*sn_alpha)*sn_beta+((-
xc2*sn_beta+yc2*cs_beta)+R2*cs_alpha)*cs_beta;
 num_rebra:=num_rebra+1;
 New(MdRecord); //)
  MdRecord.el1:=el1;
  MdRecord.el2:=el2;
  MdRecord.numrebra:=num_rebra;
  MdRecord.x1:=x1;
  MdRecord.y1:=y1;
  MdRecord.x2:=x2;
  MdRecord.y2:=y2;
  MdRecord.naklon:=FindNaklon(x1,y1,x2,y2);
 MyListM.Add(MdRecord);
end;
function FindCross(x1,y1,x2,y2,xc,yc,R:real):boolean;
var D,D2:real; //
    a,b,c:real;
    x,y:real;
begin
 a:=(x1-x2)*(x1-x2)+(y1-y2)*(y1-y2);
 b:=2*(y1-y2)*(x1*y2-x2*y1)-2*xc*(x1-x2)*(x1-x2)-2*yc*(x1-x2)*(y1-y2);
 c:=xc*xc*(x1-x2)*(x1-x2)+(x1*y2-x2*y1)*(x1*y2-x2*y1)-2*yc*(x1-
x2)*(x1*y2-x2*y1)+
    yc*yc*(x1-x2)*(x1-x2)-R*R*(x1-x2)*(x1-x2);
 if (x1=x2)and(y1=y2)then
  begin
   FindCross:=false;
   exit;
  end;
```

```
 D:=b*b-4*a*c;
 if D>=0 then
  begin
   x:=(-b+sqrt(D))/(2*a);
   b:=2*(x1-x2)*(x2*y1-x1*y2)-2*yc*(y1-y2)*(y1-y2)-2*xc*(x1-x2)*(y1-y2);
   c:=(x2*y1-x1*y2)*(x2*y1-x1*y2)-2*xc*(x2*y1-x1*y2)*(y1-
y2)+(xc*xc+yc*yc-R*R)*
      (y1-y2)*(y1-y2);
   D2:=b*b-4*a*c;
   if D2>0 then
   y:=(-b+sqrt(D2))/(2*a)
   else y:=0;
  end
 else
  begin
   x:=0;
   y:=0;
  end;
 if (x1=x2)and(x=x1) then
  begin
   if (y1<y2)and(y>y1)and(y<y2) then
    begin
     FindCross:=true;
     exit;
    end
   else if (y1>y2)and(y<y1)and(y>y2) then
    begin
     FindCross:=true;
```

```
    exit;
   end;
  end;
 if (D<0)or((D>-0.0000001)and(D<0.0000001)) then FindCross:=false
 else //D>0
  begin
   if (x1<x2)and(x>x1)and(x<x2) then FindCross:=true
   else if (x1>x2)and(x>x2)and(x<x1) then FindCross:=true
   else FindCross:=false;
  end;
end;
function PointtoComma(mystr:string):string;
//
var i:integer;
begin
   for i:=1 to length(mystr) do
    begin
     if copy(mystr,i,1)='.' then
      begin
       delete(mystr,i,1);
       insert(',',mystr,i);
      end;
    end;
  PointtoComma:=mystr;
  end;
procedure TForm1.ScrollBar1Scroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
//прокрутка по оси X
begin
```

```
  scr_pos1:=ScrollPos;
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
  DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
  ReDrawEllipse(scr_pos1,scr_pos2);
end;
procedure TForm1.ScrollBar2Scroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
// begin
  scr_pos2:=ScrollBar2.Max-Scrollpos;
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
  DrawAxis(scr_pos1,scr_pos1+45,Scr_Pos2,Scr_Pos2+45);
  ReDrawEllipse(scr_pos1,scr_pos2);
end;
procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
//)
//
var i:integer;
begin
  if (mx<>0)and(my<>0) then
   begin
     MathKoord(X,Y);
     math_x:=math_x+ScrollBar1.Position;
     math_y:=math_y+ScrollBar2.Max-ScrollBar2.Position;
Form1.Caption:='('+FloatToStrf(math_x,ffFixed,3,1)+';'+FloatToStrf(math_y,ff
Fixed,3,1)+')';
   end;
```

```
 //)
 if
(editing=true)and(MyListE.Count<>0)and(moving=false)and(finding_rebro<>tr
ue) then
  begin
    for i:=0 to MyListE.Count-1 do
     begin
      ElRecord1:=MyListE.Items[i];
      if (math_x>ElRecord1.x_c-
ElRecord1.R)and(math_x<ElRecord1.x_c+ElRecord1.R)and
        (math_y>ElRecord1.y_c-
ElRecord1.R)and(math_y<ElRecord1.y_c+ElRecord1.R)then
       begin
        Image1.Cursor:=crHandPoint;
        num:=i;
        break;
       end
      else Image1.Cursor:=crDefault;
     end;
  CopyElRecord:=ElRecord1;
 end;
 if (moving=true) then
 begin
  ElRecord1:=CopyElRecord;
  ElRecord1.x_c:=math_x;
  ElRecord1.y_c:=math_y;
  CopyElRecord:=ElRecord1;
  MyListE.Items[num]:=ElRecord1;
```

```
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
  DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
  ReDrawEllipse(scr_pos1,scr_pos2);
 end;

 if (fiding_rebro=true)and(MyListE.Count<>0)and(MyListM.Count<>0)then
  begin
  for i:=0 to MyListM.Count-1 do
   begin
    MdRecord:=MyListM.Items[i];
    if
(MdRecord.x1*math_y+MdRecord.x2*MdRecord.y1+math_x*MdRecord.y2-
      MdRecord.x2*math_y-MdRecord.x1*MdRecord.y2-
math_x*MdRecord.y1>-0.1)and

(MdRecord.x1*math_y+MdRecord.x2*MdRecord.y1+math_x*MdRecord.y2-
      MdRecord.x2*math_y-MdRecord.x1*MdRecord.y2-
math_x*MdRecord.y1<0.1)then
      begin
       if
((MdRecord.x1<MdRecord.x2)and(math_x>=MdRecord.x1)and(math_x<=MdR
ecord.x2))

or((MdRecord.x1>MdRecord.x2)and(math_x<=MdRecord.x1)and(math_x>=M
dRecord.x2))then
        begin
       Image1.Cursor:=crHandPoint;
       num:=MdRecord.numrebra;
       Form1.Caption:=inttostr(num);
```

```
        break;
        end;
       end
       else Image1.Cursor:=crDefault;
    end;
  end;
end;
procedure TForm1.Button3Click(Sender: TObject);
//
var
  xc,yc,R:real; //
  d:real; //
  i:integer;
  cross:boolean;   //
begin
  try
   xc:=strtofloat(edit1.Text);
  except
   ShowMessage('Помилка в значенні Xc');
   exit;
  end;
  try
   yc:=strtofloat(edit2.Text);
  except
   ShowMessage('Помилка в значенні Yc');
   exit;
  end;
  try
   R:=strtofloat(edit4.Text);
```

```
except
 ShowMessage('Помилка в значенні R');
 exit;
end;

if (xc>xmax+21)or(yc>ymax+16)or(xc-R<0)or(yc+R<0) then
 begin
  ShowMessage('Коло виходить за межі поля');
  exit;
 end;
//
cross:=false;
if MyListE.Count>0 then
 begin
  for i:=0 to MyListE.Count-1 do
   begin
    ElRecord1:=MyListE.Items[i];
    if ElRecord1.x_c=xc then d:=abs(ElRecord1.y_c-yc)
     else d:=sqrt((yc-ElRecord1.y_c)*(yc-ElRecord1.y_c)+(xc-
ElRecord1.x_c)*(xc-ElRecord1.x_c));
    if d<ElRecord1.R+R then
     begin
      cross:=true;
      break;
     end;
   end;
 end;
//
if cross<>true then
```

```
begin
 New(ElRecord1); //)
 ElRecord1.x_c:=xc;
 ElRecord1.y_c:=yc;
 ElRecord1.R:=R;
 if is_source=true then
  begin
   ElRecord1.elnumber:='s';
   source:=true;
   N16.Enabled:=false;
  end
 else if is_target=true then
  begin
   ElRecord1.elnumber:='t';
   target:=true;
   N17.Enabled:=false;
  end
 else
  begin
   num_el:=num_el+1;
   ElRecord1.elnumber:=inttostr(num_el);
  end;
 MyListE.Add(ElRecord1);
 DrawMyEllipse(xc-scr_pos1,yc,R,ElRecord1.elnumber);
 panel1.Visible:=false;
 Form1.Menu:=MainMenu1;
 is_source:=false;
 is_target:=false;
 N22.Enabled:=true;
```

```
   if (source=true)and(target=true) then  N30.Enabled:=true;
   end
  else ShowMessage('Коло перетинає інше коло');
  end;
procedure TForm1.FormShow(Sender: TObject);
//
begin
 finding_rebro:=false;
num:=0;
  xmax:=100;ymax:=100;
  Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
  num_rebra:=0;
  num_el:=0;
  mx:=30;
  my:=30;
  x0:=0;
  y0:=480;
  DrawAxis(0,100,0,100);
  MyListE := TList.Create;
  MyListM := TList.Create;
  ListSm := TList.Create;
  ListL := TList.Create;
  ListM := TList.Create;
  scr_pos1:=0;scr_pos2:=0;
  is_source:=false;is_target:=false;
  source:=false;target:=false;
  editing:=false;
end;
```

```
procedure TForm1.Button8Click(Sender: TObject);
//
begin
  panel1.Visible:=false;
  Form1.Menu:=MainMenu1;
end;
procedure TForm1.Button9Click(Sender: TObject);
//
var
    i:integer;
    tmp_x,tmp_y:integer;
begin
 try
  tmp_x:=strtoint(edit7.Text);
 except
  ShowMessage('Помилка в значенні Xmax');
  exit;
 end;
 try
  tmp_y:=strtoint(edit8.Text);
 except
  ShowMessage('Помилка в значенні Ymax');
  exit;
 end;
 if tmp_x<21 then
  begin
   ShowMessage('Xmax не должно быть менее 21');
   exit;
```

```
 end;

if tmp_y<16 then
 begin
  ShowMessage('Ymax 16');
  exit;
 end;
if target=true then
 begin
  for i:=0 to MyListE.Count-1 do
   begin
    ElRecord1:=MyListE.Items[i];
    if ElRecord1.elnumber='t' then
     begin
      koord(ElRecord1.y_c+ElRecord1.R,ElRecord1.y_c+ElRecord1.R);
      if (ksi>tmp_x)or(eta>tmp_y) then
       begin
        ShowMessage();
        exit;
       end;
     end;
   end;
 end;
xmax:=tmp_x-21;
ymax:=tmp_y-16;
ScrollBar1.Max:=xmax;
ScrollBar2.Max:=ymax;
ScrollBar2.Position:=ScrollBar2.Max;
scr_pos1:=0;scr_pos2:=0;
```

```
 panel2.Visible:=false;
 Form1.Menu:=MainMenu1;
end;
procedure TForm1.Button10Click(Sender: TObject);
//
begin
 panel2.Visible:=false;
 Form1.Menu:=MainMenu1;
end;
procedure TForm1.Image1MouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var i:integer;
    TmpRecord:PMyListM; //
    found:boolean;      //
begin
 if (editing=true)and(image1.Cursor=crHandPoint)then
  begin
   ElRecord1:=MyListE.Items[num];
   x_start:=ElRecord1.x_c;
   y_start:=ElRecord1.y_c;
   elnum:=ElRecord1.elnumber;
   moving:=true;
  end;
 //
 if (finding_rebro=true)and(Image1.Cursor=crHandPoint) then
  begin
   New(RebRecord);
    RebRecord:=MyListM.Items[num-1];
```

```
  //)
   found:=false;
   if ListReb.Count<>0 then
    for i:=0 to ListReb.Count-1 do
     begin
      TmpRecord:=ListReb.Items[i];
      if TmpRecord.numrebra=RebRecord.numrebra then
       begin
        found:=true;
        break;
       end;
     end;
  if found=false then ListReb.Add(RebRecord);
  Form1.Image1.Canvas.Pen.Width:=2;
  Form1.Image1.Canvas.Pen.Color:=clMaroon;
   Koord(RebRecord.x1-scr_pos1,RebRecord.y1-scr_pos2);
   Form1.Image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
   Koord(RebRecord.x2-scr_pos1,RebRecord.y2-scr_pos2);
   Form1.Image1.Canvas.LineTo(trunc(ksi),trunc(eta));
  Form1.Image1.Canvas.Pen.Width:=1;
  Form1.Image1.Canvas.Pen.Color:=clBlack;
 end;
end;
procedure TForm1.Image1MouseUp(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
var i:integer;
   d:real;
   cross:boolean;
```

```
begin
 cross:=false;
  if (editing=true)and(moving=true) then
  begin
   moving:=false;
     for i:=0 to MyListE.Count-1 do
      begin
       ElRecord1:=MyListE.Items[i];
       if CopyElRecord.elnumber<>ElRecord1.elnumber then
        begin
         if CopyElRecord.x_c=ElRecord1.x_c then d:=abs(CopyElRecord.y_c-
ElRecord1.y_c)
          else d:=sqrt((ElRecord1.y_c-CopyElRecord.y_c)*(ElRecord1.y_c-
CopyElRecord.y_c)+(ElRecord1.x_c-CopyElRecord.x_c)*(ElRecord1.x_c-
CopyElRecord.x_c));
         if d<CopyElRecord.R+ElRecord1.R then
          begin
           cross:=true;
           break;
          end;
        end;
      end;
  if cross=true then
   begin
    for i:=0 to MyListE.Count-1 do
     begin
      ElRecord1:=MyListE.Items[i];
      if ElRecord1.elnumber=elnum then
```

```
      begin
       ElRecord1.x_c:=x_start;
       ElRecord1.y_c:=y_start;
       MyListE.Items[i]:=ElRecord1;

       Iage1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
       DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
       ReDrawEllipse(scr_pos1,scr_pos2);
       break;
      end;
    end;
  end;

  nd;

end;

proedure TForm1.N29Click(Sender: TObject);
//очистка экрана (с перерисовкой коорд. сетки и очищением списков)
var
  i:integer;
begin
  clear:=false;
  ScrollBar1.Position:=0;
  ScrollBar2.Position:=100;
  xmax:=100;ymax:=100;
  scr_pos1:=0;
  scr_pos2:=0;
  mx:=30;my:=30;
```

```
label8.Caption:='';
label8.Visible:=false;
label5.Visible:=false;
source:=false;target:=false;
Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
DrawAxis(0,100,0,100);
num_rebra:=0;
num_el:=0;
finding_rebro:=false;
if MyListE.Count<>0 then
  begin
    for i:=0 to MyListE.Count-1 do MyListE.Delete(0);
  end;
if MyListM.Count<>0 then
  begin
    for i:=0 to MyListM.Count-1 do MyListM.Delete(0);
  end;
if ListL.Count<>0 then
  begin
    for i:=0 to ListL.Count-1 do ListL.Delete(0);
  end;
if ListM.Count<>0 then
  begin
    for i:=0 to ListM.Count-1 do ListM.Delete(0);
end;
if ListSm.Count<>0 then
  begin
    for i:=0 to ListSm.Count-1 do ListSm.Delete(0);
  end;
```

```
  N30.Enabled:=false;
  N31.Enabled:=false;
  N32.Enabled:=false;
  N20.Enabled:=false;
  N22.Enabled:=false;
  N23.Enabled:=false;
  N24.Enabled:=false;
  N16.Enabled:=true;
  N17.Enabled:=true;
  N15.Enabled:=true;
  N13.Enabled:=true;
  N27.Enabled:=true;
  N28.Enabled:=true;
  N2.Enabled:=false;
  N3.Enabled:=false;
  N4.Enabled:=false;
  N21.Enabled:=true;
end;
procedure TForm1.N27Click(Sender: TObject);
//
begin
 mx:=mx*2;my:=my*2;
 Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
 DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
 ReDrawEllipse(scr_pos1,scr_pos2);
 if mx>=240 then
  begin
```

```
  N27.Enabled:=false;
  exit;
 end;
N28.Enabled:=true;
end;
procedure TForm1.N28Click(Sender: TObject);
//
begin
 mx:=trunc(mx/2);my:=trunc(my/2);
 Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
 DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
 ReDrawEllipse(scr_pos1,scr_pos2);
 if mx<=15 then
  begin
   N28.Enabled:=false;
   exit;
  end;
 N27.Enabled:=true;
end;
procedure TForm1.N31Click(Sender: TObject);
//
var f:textfile;  //файл
   i,j:integer;
   found:boolean; //найдена ли заданая запись в списке
   ElRecord2,ElRecord3: PMyListE;
   cross:boolean;
   tmp_el:string;
   tmp_kr:real;
begin
```

```
 Sort;
 Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
 DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
 ReDrawEllipse(scr_pos1,scr_pos2);
//
 found:=false;
 for i:=0 to MyListE.Count-1 do
  begin
   ElRecord1:=MyListE.Items[i];
   if ElRecord1.elnumber='s' then found:=true;
  end;
 if found=false then
  begin
   ShowMessage();
   exit;
  end;
//
 found:=false;
 for i:=0 to MyListE.Count-1 do
  begin
   ElRecord1:=MyListE.Items[i];
   if ElRecord1.elnumber='t' then found:=true;
  end;
 if found=false then
  begin
   ShowMessage();
   exit;
  end;
```

```
//
 for i:=0 to MyListE.Count-2 do
  begin
   ElRecord1:=MyListE.Items[i];
    j:=i+1;
    while j<>MyListE.Count do
    begin
     ElRecord2:=MyListE.Items[j];
     j:=j+1;
     if (ElRecord1.elnumber<>ElRecord2.elnumber) then
BuildTangent(ElRecord1.elnumber,ElRecord2.elnumber,ElRecord1.x_c,ElReco
rd1.y_c,ElRecord1.R,ElRecord2.x_c,ElRecord2.y_c,ElRecord2.R);
    end;
  end;
//
 for i:=0 to MyListM.Count-2 do
  begin
   for j:=0 to MyListM.Count-1 do
   begin
    MdRecord:=MyListM.Items[j];
    if MdRecord.numrebra=i+1 then break;
   end;
   cross:=false;
   for j:=0 to MyListE.Count-2 do
    begin
     ElRecord1:=MyListE.Items[j];
     if ElRecord1.elnumber=MdRecord.el1 then break;
    end;
```

```
for j:=0 to MyListE.Count-1 do
 begin
  ElRecord2:=MyListE.Items[j];
  if ElRecord2.elnumber=MdRecord.el2 then break;
 end;
for j:=1 to MyListE.Count-2 do
 begin
  ElRecord3:=MyListE.Items[j];
  if (ElRecord1.elnumber<>ElRecord3.elnumber)then
   begin
    cross:=FindCross(MdRecord.x1,MdRecord.y1,MdRecord.x2,MdRecord.y2,
                ElRecord3.x_c,ElRecord3.y_c,ElRecord3.R);
    if cross=true then break;
   end;
 end;
//
for j:=0 to MyListM.Count-1 do
 begin
  MdRecord:=MyListM.Items[j];
  if MdRecord.numrebra=i+1 then break;
 end;
if (MdRecord.x1>MdRecord.x2)and(MdRecord.el2<>'t') then
 begin
  tmp_el:=mdrecord.el1;
  MdRecord.el1:=MdRecord.el2;
  MdRecord.el2:=tmp_el;
  tmp_kr:=MdRecord.x1;
  MdRecord.x1:=MdRecord.x2;
```

```
    MdRecord.x2:=tmp_kr;
    tmp_kr:=MdRecord.y1;
    MdRecord.y1:=MdRecord.y2;
    MdRecord.y2:=tmp_kr;
    MyListM.Items[j]:=MdRecord;
   end;
   if cross=true then
    begin
     for j:=0 to MyListM.Count-1 do
      begin
       MdRecord:=MyListM.Items[j];
       if MdRecord.numrebra=i+1 then break;
      end;
     MyListM.Delete(j);
    end;
 end;
 for i:=0 to MyListM.Count-1 do
 begin
  MdRecord:=MyListM.Items[i];
  MdRecord.numrebra:=i+1;
 end;
 assignfile(f,'model1.txt');
 rewrite(f);
 for i:=0 to MyListM.Count-1 do
  begin
   MdRecord:=MyListM.Items[i];
   writeln(f,MdRecord.numrebra,' ',MdRecord.el1,' ',MdRecord.el2,'
',floattostrf(MdRecord.x1,ffFixed,5,4),' ',
```

```
            floattostrf(MdRecord.y1,ffFixed,5,4),'
',floattostrf(MdRecord.x2,ffFixed,5,4),' ',floattostrf(MdRecord.y2,ffFixed,5,4));
   end;
 closefile(f);
 //
 for i:=0 to MyListM.Count-1 do
  begin
   MdRecord:=MyListM.Items[i];
   koord(MdRecord.x1-scr_pos1,MdRecord.y1-scr_pos2);
   image1.Canvas.Pen.Color:=clBlue;
   image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
   koord(MdRecord.x2-scr_pos1,MdRecord.y2-scr_pos2);
   image1.Canvas.LineTo(trunc(ksi),trunc(eta));
   image1.Canvas.Pen.Color:=clBlack;
   //ShowMessage('OK');
  end;
 N32.Enabled:=true;
 N15.Enabled:=false;
 N21.Enabled:=false;
 N23.Enabled:=false;
 N31.Enabled:=false;
 N2.Enabled:=true;
end;
procedure TForm1.N32Click(Sender: TObject);
var
    D: MyD; // D
    ang:real;  //
    i,j,k:integer; //
    min:real; //
```

```
   found:boolean; //
   mywhere:string; //
   path:string; //
   ind:integer; //
   x1,y1,x2,y2:real;
   LRecord: PListL;   // L
   SmRecord: PListSm; //
   MdRecordTmp: PMyListM;  //
   f:textfile;
  // tmp:integer;
   label label1;
begin
 //tmp:=1;
//assignfile(f,'F:\UNIVER\Diploma\1\MyProject\debug.doc');
//     rewrite(f);
 N31.Enabled:=false;
 N32.Enabled:=false;
 //
 for i:=0 to MyListM.Count-1 do
  begin
   MdRecord:=MyListM.Items[i];
   for j:=0 to MyListE.Count-1 do
    begin
     ElRecord1:=MyListE.Items[j];
     if ElRecord1.elnumber=MdRecord.el2 then break;
    end;
   if MdRecord.y2>ElRecord1.y_c then MdRecord.dest_in:='v'
   else MdRecord.dest_in:='n'
```

```
 end;
//
 New(SmRecord);
 SmRecord.numreb:=0;
 SmRecord.kolsmreber:=0;
 j:=0;
 for i:=0 to MyListM.Count-1 do
  begin
   MdRecord:=MyListM.Items[i];
   if MdRecord.el1='s' then
    begin
     j:=j+1;
     SmRecord.numsmreb[j]:=MdRecord.numrebra;
     SmRecord.kolsmreber:=SmRecord.kolsmreber+1;
    end;
  end;
 ListSm.Add(SmRecord);

 for k:=0 to MyListM.Count-1 do
  begin
   MdRecord:=MyListM.Items[k];
   New(SmRecord);
   SmRecord.kolsmreber:=0;
   SmRecord.numreb:=MdRecord.numrebra;
   for i:=0 to MyListE.Count-1 do
    begin
     ElRecord1:=MyListE.Items[i];
     if ElRecord1.elnumber=MdRecord.el2 then break;
    end;
```

```
  if MdRecord.y2>ElRecord1.y_c then
   begin
    j:=1;
    for i:=0 to MyListM.Count-1 do
     begin
      MdRecordTmp:=MyListM.Items[i];
      if MdRecordTmp.el1=MdRecord.el2 then
      if
(MdRecordTmp.naklon<MdRecord.naklon)and(MdRecordTmp.y1>ElRecord1.
y_c) then
       begin
        SmRecord.numsmreb[j]:=MdRecordTmp.numrebra;
        SmRecord.kolsmreber:=SmRecord.kolsmreber+1;
        j:=j+1;
       end;
     end;
     ListSm.Add(SmRecord);
   end;

   if MdRecord.y2<=ElRecord1.y_c then
   begin
    j:=1;
    for i:=0 to MyListM.Count-1 do
     begin
      MdRecordTmp:=MyListM.Items[i];
      if MdRecordTmp.el1=MdRecord.el2 then
```

```
      if
(MdRecordTmp.naklon>MdRecord.naklon)and(MdRecordTmp.y1<ElRecord1.
y_c) then
       begin
        SmRecord.numsmreb[j]:=MdRecordTmp.numrebra;
        SmRecord.kolsmreber:=SmRecord.kolsmreber+1;
        j:=j+1;
       end;
     end;
     ListSm.Add(SmRecord);
   end;

 end;
// D
MdRecord:=MyListM.Items[0];
D.NumEl:='s';
D.x1:=0;
D.y1:=0;
D.x2:=0;
D.y2:=0;
D.numrebra:=0;
D.sumang:=0;
D.vn:='s';
D.path:='';
// D в L
New(LRecord);
 LRecord.NumEl:=D.NumEl;
 LRecord.x1:=D.x1;
 LRecord.y1:=D.y1;
```

```
 LRecord.x2:=D.x2;
 LRecord.y2:=D.y2;
 LRecord.numrebra:=D.numrebra;
 LRecord.sumang:=D.sumang;
 LRecord.vn:=D.vn;
 LRecord.path:='';
ListL.Add(LRecord);
N20.Enabled:=true;
label1:
if ListL.Count=0 then
 begin
  ShowMessage();
  exit;
 end;
//
ind:=0;
LRecord:=ListL.Items[0];
min:=LRecord.sumang;
for i:=1 to ListL.Count-1 do
 begin
  LRecord:=ListL.Items[i];
  if min>LRecord.sumang then
   begin
    min:=LRecord.sumang;
    ind:=i;
   end;
 end;
LRecord:=ListL.Items[ind];
```

```
// M
New(MRecord);
 MRecord.NumEl:=LRecord.NumEl;
 MRecord.numrebra:=LRecord.numrebra;
 MRecord.x1:=LRecord.x1;
 MRecord.y1:=LRecord.y1;
 MRecord.x2:=LRecord.x2;
 MRecord.y2:=LRecord.y2;
 MRecord.sumang:=Lrecord.sumang;
 MRecord.path:=LRecord.path;
 MRecord.vn:=LRecord.vn;
ListM.Add(MRecord);
ListL.Delete(ind);
if MRecord.NumEl='t' then  //
 begin
  MRecord:=ListM.Items[ListM.Count-1];
  label5.Visible:=true;
  label8.Visible:=true;
  label8.Caption:=floattostrf(MRecord.sumang,ffFixed,3,5);
  //
  Form1.Image1.Canvas.Pen.Width:=2;
  Form1.Image1.Canvas.Pen.Color:=clGreen;
  path:=MRecord.path;
  delete(path,length(path),1);
  while path<>'' do
  begin
   x1:=strtofloat(copy(path,1,pos('*',path)-1));
   delete(path,1,pos('*',path));
```

```
  y1:=strtofloat(copy(path,1,pos('*',path)-1));
  delete(path,1,pos('*',path));
  x2:=strtofloat(copy(path,1,pos('*',path)-1));
  delete(path,1,pos('*',path));
  if pos('*',path)=0 then
  begin
   y2:=strtofloat(path);
   delete(path,1,length(path));
  end
  else
  begin
   y2:=strtofloat(copy(path,1,pos('*',path)-1));
   delete(path,1,pos('*',path));
  end;
  Koord(x1-scr_pos1,y1-scr_pos2);
  Form1.Image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
  Koord(x2-scr_pos1,y2-scr_pos2);
  Form1.Image1.Canvas.LineTo(trunc(ksi),trunc(eta));
 end;
 Form1.Image1.Canvas.Pen.Width:=1;
 Form1.Image1.Canvas.Pen.Color:=clBlack;
 exit;
end;
//
for i:=0 to ListSm.Count-1 do
 begin
  SmRecord:=ListSm.Items[i];
  if SmRecord.numreb=MRecord.numrebra then break;
 end;
```

```
i:=1;
//
while i<=SmRecord.kolsmreber do
 begin
  //
  for j:=0 to MyListM.Count-1 do
   begin
    MdRecord:=MyListM.Items[j];
    if MdRecord.numrebra=SmRecord.numsmreb[i] then break;
   end;
ang:=FindAngle(MRecord.x1,MRecord.y1,MRecord.x2,MRecord.y2,MdRecord
.x1,MdRecord.y1,
          MdRecord.x2,MdRecord.y2);
  //writeln(f,'x1_1=' + floattostrf(mRecord.x1,ffFixed,5,2) + ';'+'y1_1=' +
floattostrf(mRecord.y1,ffFixed,5,2) + ';'+'x1_2=' +
floattostrf(mRecord.x2,ffFixed,5,2) + ';'+'y1_2=' +
floattostrf(mRecord.y2,ffFixed,5,2) + ';'+'x2_1=' +
floattostrf(mdRecord.x1,ffFixed,5,2) + ';'+'y2_1=' +
floattostrf(mdRecord.y1,ffFixed,5,2) + ';'+'x2_2=' +
floattostrf(mdRecord.x2,ffFixed,5,2) + ';'+'y2_2=' +
floattostrf(mdRecord.y2,ffFixed,5,2));
  //  writeln(f, floattostrf(ang,ffFixed,5,2));
    //tmp:=tmp+1;
    //if tmp=23 then ShowMessage('24');
    //closefile(f);
  // D'
  D.NumEl:=MdRecord.el2;
  D.x1:=MdRecord.x1;
```

```
  D.y1:=MdRecord.y1;
  D.x2:=MdRecord.x2;
  D.y2:=MdRecord.y2;
  D.numrebra:=MdRecord.numrebra;
  D.sumang:=MRecord.sumang+ang;
  D.vn:=MdRecord.dest_in;

D.path:=MRecord.path+floattostr(MdRecord.x1)+'*'+floattostr(MdRecord.y1)+'
*'
        +floattostr(MdRecord.x2)+'*'+floattostr(MdRecord.y2)+'*';
  found:=false;mywhere:='';
  // D
  for j:=0 to ListL.Count-1 do
   begin
    LRecord:=ListL.Items[j];
    if
(LRecord.NumEl=D.NumEl)and(LRecord.vn=D.vn)and(LRecord.sumang>D.su
mang) then
     begin
      found:=true;mywhere:=mywhere+'L';ind:=j;break;
     end
   end;
   // D
   for j:=0 to ListM.Count-1 do
    begin
     MRecord:=ListM.Items[j];
     if
(MRecord.NumEl=D.NumEl)and(MRecord.vn=D.vn)and(MRecord.sumang>D.
sumang) then
```

```
    begin
     found:=true;mywhere:=mywhere+'M';ind:=j;break;
    end
  end;

//
if found=false then
 begin
  //помещаем D в L
  New(LRecord);
   LRecord.NumEl:=D.NumEl;
   LRecord.x1:=D.x1;
   LRecord.y1:=D.y1;
   LRecord.x2:=D.x2;
   LRecord.y2:=D.y2;
   LRecord.numrebra:=D.numrebra;
   LRecord.sumang:=D.sumang;
   LRecord.vn:=D.vn;
   LRecord.path:=D.path;
  ListL.Add(LRecord);
  i:=i+1;
  continue;
 end
//
else begin
      if mywhere='LM' then
      begin
       if MRecord.sumang>D.sumang then ListM.Delete(ind);
       if LRecord.sumang>=D.sumang then
```

```
  begin
  // L
   if LRecord.sumang>D.sumang then ListL.Delete(ind);
   New(LRecord);
    LRecord.NumEl:=D.NumEl;
    LRecord.x1:=D.x1;
    LRecord.y1:=D.y1;
    LRecord.x2:=D.x2;
    LRecord.y2:=D.y2;
    LRecord.numrebra:=D.numrebra;
    LRecord.sumang:=D.sumang;
    LRecord.vn:=D.vn;
    LRecord.path:=D.path;
   ListL.Add(LRecord);
   i:=i+1;
   continue;
  end  else begin i:=i+1; continue;end;
 end;
if mywhere='L' then
 begin
  if LRecord.sumang>=D.sumang then
   begin
   // L
    if LRecord.sumang>D.sumang then ListL.Delete(ind);
    New(LRecord);
     LRecord.NumEl:=D.NumEl;
     LRecord.x1:=D.x1;
     LRecord.y1:=D.y1;
     LRecord.x2:=D.x2;
```

```
     LRecord.y2:=D.y2;
     LRecord.numrebra:=D.numrebra;
     LRecord.sumang:=D.sumang;
     LRecord.vn:=D.vn;
     LRecord.path:=D.path;
    ListL.Add(LRecord);
    i:=i+1;
    continue;
   end else begin i:=i+1; continue;end;
 end;
if mywhere='M' then
 begin
  if MRecord.sumang>=D.sumang then
   begin
   // M
    if MRecord.sumang>D.sumang then ListM.Delete(ind);
    New(lRecord);
     lRecord.NumEl:=D.NumEl;
     lRecord.x1:=D.x1;
     lRecord.y1:=D.y1;
     lRecord.x2:=D.x2;
     lRecord.y2:=D.y2;
     lRecord.numrebra:=D.numrebra;
     lRecord.sumang:=D.sumang;
     lRecord.vn:=D.vn;
     lRecord.path:=D.path;
    Listl.Add(MRecord);
    i:=i+1;
    continue;
```

```
          end else begin i:=i+1;continue;end;
         end;
       end;
  end;
 goto label1;
  //  closefile(f);
end;
procedure TForm1.N16Click(Sender: TObject);
//
begin
 panel1.Visible:=true;
 Form1.Menu:=MainMenu2;
 is_source:=true;
 edit1.SetFocus;
 if target=true then N31.Enabled:=true;
end;
procedure TForm1.N17Click(Sender: TObject);
//
begin
 panel1.Visible:=true;
 Form1.Menu:=MainMenu2;
 is_target:=true;
 edit1.SetFocus;
 if source=true then N31.Enabled:=true;
end;
procedure TForm1.N18Click(Sender: TObject);
//
begin
  panel1.Visible:=true;
```

```
 Form1.Menu:=MainMenu2;
 edit1.SetFocus;
end;
procedure TForm1.N19Click(Sender: TObject);
begin
 if MyListE.Count<>0 then
  begin
   ElRecord1:=MyListE.Items[MyListE.Count-1];
   if ElRecord1.elnumber='s' then
    begin
     source:=false;
     N16.Enabled:=true;
     N31.Enabled:=false;
     N32.Enabled:=false;
    end
   else if ElRecord1.elnumber='t' then
    begin
     target:=false;
     N17.Enabled:=true;
     N31.Enabled:=false;
     N32.Enabled:=false;
    end;
   MyListE.Delete(MyListE.Count-1);
   num_el:=num_el-1;
   Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
   DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
   ReDrawEllipse(scr_pos1,scr_pos2);

  end;
```

```
end;
procedure TForm1.N13Click(Sender: TObject);
//
var f:textfile; //
    str:string;
    xc,yc,R:real;
    num:string; //
    i:integer;//
    ch:string; //
begin
 if opendialog1.Execute and fileexists(opendialog1.FileName) then
  begin
   assignfile(f,opendialog1.FileName);
   try
    reset(f);
   except
    showmessage(+ opendialog1.FileName);
    exit;
   end;
   while not eof(f) do
    begin
     readln(f,str);
     str:=PointtoComma(str);
     //
     num:='';
     xc:=0;
     yc:=0;
     R:=0;
     i:=1;
```

```
repeat
 ch:=copy(str,i,1);
 if (num<>'')and(xc<>0)and(yc<>0)and(R=0) then
  begin
   R:=strtofloat(str);
   break;
  end else
 if ch=' ' then
  begin
   if num='' then num:=copy(str,1,i-1) else
   if xc=0 then xc:=strtofloat(copy(str,1,i-1)) else
   if yc=0 then yc:=strtofloat(copy(str,1,i-1)) else
   R:=strtofloat(copy(str,1,i-1));
   delete(str,1,i);
   i:=1;
  end else i:=i+1;
until str='';

if nm='s' then
 begin
  if source=true then
   begin
    ShowMessage('Источник уже задан');
    exit;
   end;
  DrawMyEllipse(xc-scr_pos1,yc,R,'s');
  N16.Enabled:=false;
  source:=true;
```

```
     end
    else if num='t' then
     begin
      if target=true then
       begin
        ShowMessage('Цель уже задана');
        N31.Enabled:=true;
        exit;
       end;
      DrawMyEllipse(xc-scr_pos1,yc,R,'t');
      N17.Enabled:=false;
      target:=true;
     end
    else DrawMyEllipse(xc-scr_pos1,yc,R,num);
    New(ElRecord1); //добавление новой окружности (новой записи) в
список
    if (num='s')or(num='t') then ElRecord1.elnumber:=num
    else
     begin
      num_el:=num_el+1;
      ElRecord1.elnumber:=inttostr(num_el);
     end;
     ElRecord1.x_c:=xc;
     ElRecord1.y_c:=yc;
     ElRecord1.R:=R;
    MyListE.Add(ElRecord1);
    end;
  closefile(f);
  end;
```

```
  if MyListE.Count<>0 then
   begin
    N30.Enabled:=true;
    N31.Enabled:=true;
    N13.Enabled:=false;
    N22.Enabled:=true;
    N23.Enabled:=true;
  end;
end;
procedure TForm1.N14Click(Sender: TObject);
//сохранение в файл
var f:textfile;
   i,j:integer;
   str:string;
begin
 if savedialog1.Execute then
  begin
   assignfile(f,savedialog1.FileName);
   rewrite(f);
   if MyListE.Count<>0 then
    begin
     for i:=0 to MyListE.Count-1 do
      begin
       ElRecord1:=MyListE.Items[i];
       str:=ElRecord1.elnumber+' '+floattostr(ElRecord1.x_c)+'
'+floattostr(ElRecord1.y_c)+
            ' '+floattostr(ElRecord1.R);
       //
       for j:=1 to length(str) do
```

```
      begin
       if copy(str,j,1)=',' then
        begin
         delete(str,j,1);
         insert('.',str,j);
        end;
       end;
      writeln(f,str);
     end;
   end;
  closefile(f);
 end;
end;
procedure TForm1.N21Click(Sender: TObject);
//вызов окна для задание размеров поля
begin
 Panel2.Visible:=true;
 edit7.SetFocus;
 Form1.Menu:=MainMenu2;
end;
procedure TForm1.N20Click(Sender: TObject);
var
  i:integer;
  xc,yc,R:real;//
  x1,x2,y1,y2:real;
  path:string; //
begin
 N2.Enabled:=false;
 Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
```

```
DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
 if MyListE.Count<>0 then
 //
  begin
    for i:=0 to MyListE.Count-1 do
     begin
       ElRecord1:=MyListE.Items[i];
       xc:=ElRecord1.x_c-scr_pos1;
       yc:=ElRecord1.y_c-scr_pos2;
       R:=ElRecord1.R;
       DrawMyEllipse(xc,yc,R,ElRecord1.elnumber)
     end;
  end;
  if ListM.Count<>0 then
  //
   begin
    MRecord:=ListM.Items[ListM.Count-1];
    Form1.Image1.Canvas.Pen.Width:=2;
    Form1.Image1.Canvas.Pen.Color:=clGreen;
    path:=MRecord.path;
    delete(path,length(path),1);
    while path<>'' do
     begin
      x1:=strtofloat(copy(path,1,pos('*',path)-1));
      delete(path,1,pos('*',path));
      y1:=strtofloat(copy(path,1,pos('*',path)-1));
      delete(path,1,pos('*',path));
      x2:=strtofloat(copy(path,1,pos('*',path)-1));
      delete(path,1,pos('*',path));
```

```
      if pos('*',path)=0 then
       begin
        y2:=strtofloat(path);
        delete(path,1,length(path));
       end
       else
       begin
        y2:=strtofloat(copy(path,1,pos('*',path)-1));
        delete(path,1,pos('*',path));
       end;
      Koord(x1-scr_pos1,y1-scr_pos2);
      Form1.Image1.Canvas.MoveTo(trunc(ksi),trunc(eta));
      Koord(x2-scr_pos1,y2-scr_pos2);
      Form1.Image1.Canvas.LineTo(trunc(ksi),trunc(eta));
     end;
    Form1.Image1.Canvas.Pen.Width:=1;
    Form1.Image1.Canvas.Pen.Color:=clBlack;
   end;
  clear:=true;
end;
procedure TForm1.N23Click(Sender: TObject);
//
begin
 editing:=true;
 label3.Visible:=true;
 N31.Visible:=false;
 N32.Visible:=false;
 N23.Enabled:=false;
```

```
 N24.Enabled:=true;
 N12.Enabled:=false;
 N15.Enabled:=false;
 N30.Enabled:=false;
 N20.Enabled:=false;
 N21.Enabled:=false;
end;
procedure TForm1.N24Click(Sender: TObject);
//
begin
 editing:=false;
 label3.Visible:=false;
 N31.Visible:=true;
 N32.Visible:=true;
 N23.Enabled:=true;
 N24.Enabled:=false;
 N12.Enabled:=true;
 N15.Enabled:=true;
 N30.Enabled:=true;
 N20.Enabled:=true;
 N21.Enabled:=true;
 Sort;
 Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
 DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
 ReDrawEllipse(scr_pos1,scr_pos2);
end;
procedure TForm1.N2Click(Sender: TObject);
//
begin
```

```
 ListReb:=TList.Create;
 if MyListM.Count>0 then finding_rebro:=true;
 N3.Enabled:=true;
end;
procedure TForm1.N3Click(Sender: TObject);
//
var i,j:integer;
    sum_angle:real;
    TmpRecord: PMyListM;
    SmRecord: PListSm; //
    found:boolean;
    ind:integer;
 begin

 if (finding_rebro=true)and (ListReb.Count<>0)then
  begin
   sum_angle:=0;
   //
   RebRecord:=ListReb.Items[ListReb.Count-1];
   if RebRecord.el2<>'t' then
    begin
     ShowMessage('Последнее ребро не входит в цель');
     exit;
    end;
   ind:=0;
   for i:=0 to ListReb.Count-2 do
    begin
     found:=false;
     RebRecord:=ListReb.Items[i];
```

```
    TmpRecord:=ListReb.Items[i+1];
    for j:=0 to ListSm.Count-1 do
     begin
      SmRecord:=ListSm.Items[j];
      if SmRecord.numreb=RebRecord.numrebra then
       begin
        found:=true;
        ind:=j;
        break;
       end;
     end;
    if found=true then
     begin
      SmRecord:=ListSm.Items[ind];
      for j:=0 to SmRecord.kolsmreber do
       begin
        if TmpRecord.numrebra=SmRecord.numsmreb[j] then break;
       end;
     end;
   end;
  for i:=0 to ListReb.Count-2 do
   begin
    RebRecord:=ListReb.Items[i];
    TmpRecord:=ListReb.Items[i+1];
um_angle:=sum_angle+FindAngle(RebRecord.x1,RebRecord.y1,RebRecord.x2,
RebRecord.y2,
 TmpRecord.x1,TmpRecord.y1,TmpRecord.x2,TmpRecord.y2);
   end;
   Label5.Visible:=true;
```

```
   Label8.Visible:=true;
   Label8.Caption:=floattostr(sum_angle);
   n4.Enabled:=true;
  end;
   if ListReb.Count<>0 then
    begin
     for i:=0 to ListReb.Count-1 do ListReb.Delete(0);
    end;
end;
procedure TForm1.N4Click(Sender: TObject);
//
var i:integer;
begin
 Image1.Canvas.Rectangle(0,0,Image1.Width,Image1.Height);
 DrawAxis(scr_pos1,scr_pos1+45,scr_pos2,scr_pos2+45);
 ReDrawEllipse(scr_pos1,scr_pos2);
 Label5.Visible:=false;
 Label8.Visible:=false;
 if ListReb.Count<>0 then
  begin
   for i:=0 to ListReb.Count-1 do ListReb.Delete(0);
  end;
end;
procedure TForm1.N5Click(Sender: TObject);
//
var
  i:integer;
begin
```

```
 if MyListE.Count<>0 then
  begin
    for i:=0 to MyListE.Count-1 do MyListE.Delete(0);
    MyListE.Free;
  end;
 if MyListM.Count<>0 then
  begin
    for i:=0 to MyListM.Count-1 do MyListM.Delete(0);
    MyListM.Free;
  end;
 if ListL.Count<>0 then
   begin
     for i:=0 to ListL.Count-1 do ListL.Delete(0);
     ListL.Free;
   end;
 if ListM.Count<>0 then
   begin
     for i:=0 to ListM.Count-1 do ListM.Delete(0);
     ListM.Free;
   end;
 if ListSm.Count<>0 then
   begin
     for i:=0 to ListSm.Count-1 do ListSm.Delete(0);
     ListSm.Free;
   end;
  Close;
end;
procedure TForm1.N6Click(Sender: TObject);
//
```

```
begin
 if MessageDlg('Вывести на печать',mtConfirmation, [mbYes, mbNo], 0) =
mrYes then
  begin
   Form1.Color:=clWhite;
   Form1.Print;
   Form1.Color:=clBtnFace;
  end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
Form2.Close;
end;
unit Unit2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;
type
  TForm2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Image1: TImage;
    Label11: TLabel;
    Label6: TLabel;
```

```
    Label7: TLabel;
    Label8: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
implementation
 uses MainUnit;
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Hide;
Form1.Show;
end;
```

ЛІСТИНГИ ПРОГРАМ ДЛЯ ВИЗНАЧЕННЯ НАПРУЖЕНОСТІ ТЕХНОЛОГІЧНОГО ПРОЦЕСУ

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, jpeg, ExtCtrls, StdCtrls;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Label3: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    Image5: TImage;
    Image6: TImage;
    Label2: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
```

```
    Label15: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
uses Unit2;
procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Hide;
Form2.Show;
end;
unit Unit2;
interface
uses
   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj
, Forms,
  Dialogs, StdCtrls,Menus,TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm2 = class(TForm)
    MainMenu1: TMainMenu;
    Button1: TButton;
    N2: TMenuItem;
```

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N6: TMenuItem;

N7: TMenuItem;

N8: TMenuItem;

N12: TMenuItem;

N14: TMenuItem;

N16: TMenuItem;

N18: TMenuItem;

N19: TMenuItem;

N20: TMenuItem;

N21: TMenuItem;

N22: TMenuItem;

N23: TMenuItem;

N24: TMenuItem;

N25: TMenuItem;

N26: TMenuItem;

N27: TMenuItem;

N28: TMenuItem;

N29: TMenuItem;

N32: TMenuItem;

N33: TMenuItem;

N34: TMenuItem;

N35: TMenuItem;

N36: TMenuItem;

N37: TMenuItem;

N38: TMenuItem;

N39: TMenuItem;

N40: TMenuItem;

N1: TMenuItem;

Memo1: TMemo;

N30: TMenuItem;

N31: TMenuItem;

N41: TMenuItem;

N42: TMenuItem;

N43: TMenuItem;

N44: TMenuItem;

Memo2: TMemo;

Label1: TLabel;

Chart1: TChart;

Series1: TLineSeries;

Chart2: TChart;

Series2: TBarSeries;

Series3: TBarSeries;

Chart3: TChart;

Series4: THorizBarSeries;

Series5: THorizBarSeries;

N9: TMenuItem;

N10: TMenuItem;

N45: TMenuItem;

N46: TMenuItem;

N47: TMenuItem;

N48: TMenuItem;

N49: TMenuItem;

Label2: TLabel;

Label3: TLabel;

Chart4: TChart;

```
Series6: TFastLineSeries;
StringGrid1: TStringGrid;
N11: TMenuItem;
N13: TMenuItem;
Label4: TLabel;
N15: TMenuItem;
N17: TMenuItem;
procedure Button1Click(Sender: TObject);
procedure N33Click(Sender: TObject);
procedure N34Click(Sender: TObject);
procedure N35Click(Sender: TObject);
procedure N1Click(Sender: TObject);
procedure N28Click(Sender: TObject);
procedure N29Click(Sender: TObject);
procedure N36Click(Sender: TObject);
procedure N37Click(Sender: TObject);
procedure N38Click(Sender: TObject);
procedure N39Click(Sender: TObject);
procedure N40Click(Sender: TObject);
procedure N30Click(Sender: TObject);
procedure N18Click(Sender: TObject);
procedure N27Click(Sender: TObject);
procedure N41Click(Sender: TObject);
procedure N9Click(Sender: TObject);
procedure N10Click(Sender: TObject);
procedure N47Click(Sender: TObject);
procedure N26Click(Sender: TObject);
procedure N19Click(Sender: TObject);
procedure N20Click(Sender: TObject);
```

```
    procedure N21Click(Sender: TObject);
    procedure N22Click(Sender: TObject);
    procedure N23Click(Sender: TObject);
    procedure N24Click(Sender: TObject);
    procedure N25Click(Sender: TObject);
    procedure N45Click(Sender: TObject);
    procedure N46Click(Sender: TObject);
    procedure N43Click(Sender: TObject);
    procedure N44Click(Sender: TObject);
    procedure N48Click(Sender: TObject);
    procedure N49Click(Sender: TObject);
    procedure N12Click(Sender: TObject);
    procedure N16Click(Sender: TObject);
    procedure N14Click(Sender: TObject);
    procedure N31Click(Sender: TObject);
    procedure N13Click(Sender: TObject);
    procedure N11Click(Sender: TObject);
    procedure N15Click(Sender: TObject);
    procedure N17Click(Sender: TObject);
      private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
  TCEVL,TPOET,TKAMN,TLEN, TLA, TVI, TV,TBV,TSHL,
TKA,DN,BGN,E1N,BSLN,AT,BT:Real;
  DN1,BGN1,E1N1,BSLN1,AT1,BT1:String;
```

```
 RN:Real;
 BER,FP,R1:Real;
 x,F,c,kt,d0,d,V11,SS0,SS1,GG0,GG1,USS0,USS1,UGG0,UGG1:Real;
 a,b,e,h:Real;
SKUTKOM,URNAT,KTP:Real;
s,DN2:Real;
 ALSH,RMSH,RBSH,NDSH:Real;
  TRUXU,UNPA,KOPA,KTPA:Real;
 P0,t,vrux1,dlinz1:Real;
 PZDI0,PZDI1,PZDI2,PZDI3,PZDI4,PZDI5,PZDI6:Real;
 PZNA0,PZNA1,PZNA2,PZNA3,PZNA4,PZNA5,PZNA6:Real;
 jj1,kivuz1:Integer;
 i,code:Integer;
 k:Integer;
 s11,s22,t022,t011,TRUXU2,kkkkk1,kkkk1,kk,kkk,kkk1:String;
 w:Integer;
KTPA1,KTP1,BER2,BER1,P02,P011,P022,FP1,R,FP2,R2,kivuz,vrux,dlinz,kiv
uz2,vrux2,dlinz2:String;
 s1,DN22,vvv,otnat,nat1,nat,P01,t1: array [0..360] of Real;
 OBOZ1,OBOZ2,OBOZ3,OBOZ4:array [1..100] of String;
implementation
Uses
Unit1,Synt,UErrors,Unit3,Unit4,Unit5,Unit6,Unit7,Unit8,Unit9,Unit10,Unit11,
Unit12,Unit13,Unit14,Unit15,Unit16,Unit17,Unit18,Unit19,Unit20,Unit21,Unit
22,Unit23,Unit24,Unit25,Unit26;
{$R *.dfm}
 procedure V(var P0:Real;t:Real);
        begin
```

```
        SetData('P0',P0);
        SetData('t',t);
        Calculate(P0);
      end;
procedure V1(var
F:Real;X,P0,BSLN,RN,E1N,BGN,FP,AT,BT,R1,d0,kt,V11:Real);
      begin
        SetData('P0',P0);
        SetData('BSLN',BSLN);
        SetData('RN',RN);
        SetData('E1N',E1N);
        SetData('BGN',BGN);
        SetData('FP',FP);
        SetData('AT',AT);
        SetData('BT',BT);
        SetData('R1',R1);
        SetData('d0',d0);
        SetData('kt',kt);
        SetData('V11',V11);
        SetData('X',X);
        F:=-X+P0*(1+((R1+RN)/(R1+RN*(1-d0)))*(exp(kt*V11)-
1))+(BGN/(2*sqr(R1+RN*(1-d0))))-(BGN/(2*sqr(R1+RN*(1-
d0))))*(1+((R1+RN)/(R1+RN*(1-d0)))*(exp(kt*V11)-1));
      end;
 procedure V2(var
F:Real;X,P0,BSLN,RN,E1N,BGN,FP,BER,AT,BT,R1,d0,kt,V11:Real);
      begin
        SetData('P0',P0);
        SetData('BSLN',BSLN);
```

```
        SetData('RN',RN);
        SetData('E1N',E1N);
        SetData('BGN',BGN);
        SetData('FP',FP);
        SetData('BER',BER);
        SetData('AT',AT);
        SetData('BT',BT);
        SetData('R1',R1);
        SetData('d0',d0);
        SetData('kt',kt);
        SetData('V11',V11);
        SetData('X',X);
        F:=-X+P0*(1+((R1+RN)/(R1+RN*(1-d0)))*(exp(kt*V11)-
1))+(BGN/(2*sqr(R1+RN*(1-d0))))-(BGN/(2*sqr(R1+RN*(1-
d0))))*(1+((R1+RN)/(R1+RN*(1-d0)))*(exp(kt*V11)-1));
      end;
procedure V3(var DN2:Real;s:Real);
      begin
        SetData('DN2',DN2);
        SetData('s',s);
        Calculate(DN2);
      end;
procedure TForm2.Button1Click(Sender: TObject);
begin
Form1.Close;
end;
procedure TForm2.N33Click(Sender: TObject);
begin
```

```
 kivuz:=InputBox('Введіть кількість вузлів',' kivuz=','');
 Val(kivuz,kivuz1,code);
end;

procedre TForm2.N34Click(Sender: TObject);
begin
 vrux:=InputBox('Введіть швидкість руху, мм/с',' vrux=','');
 Val(vrux,vrux1,code);
end;

rocedure TForm2.N35Click(Sender: TObject);
begin
  dlinz:=InputBox('Довжина лінії заправки, мм',' dlinz=','');
  Val(dlinz,dlinz1,code);

end;

procdure TForm2.N1Click(Sender: TObject);
begin

  kvuz2:=format('%3.0d',[kivuz1]);
  vrux2:=format('%9.5f',[vrux1]);
  dlinz2:=format('%9.5f',[dlinz1]);
  Memo1.Lines.Add('Кількість вузлів kivuz='+kivuz2);
  Memo1.Lines.Add('Швидкість руху vrux='+vrux2);
  Memo1.Lines.Add('Довжина лінії запрапвки dlinz='+dlinz2);



end;
```

```
procedure TForm2.N41Click(Sender: TObject);
begin
      jj1:=1;
      P011:=InputBox('Введіть початковий натяг, cH',' P0=','');
      Val(P011,P0,code);

      P022:=format('%9.5f',[P0]);
      Memo1.Lines.Add('Початковий натяг P0='+P022);
      nat[jj1]:=P0;
      nat1[jj1]:=P0;
end;
procedure TForm2.N28Click(Sender: TObject);
begin
Form2.Hide;
Form3.Show;
end;
procedure TForm2.N29Click(Sender: TObject);
begin
Form2.Hide;
Form4.Show;
end;
procedure TForm2.N36Click(Sender: TObject);
begin
Form2.Hide;
Form5.Show;
end;
procedure TForm2.N37Click(Sender: TObject);
begin
```

```
Form2.Hide;
Form6.Show;
pocedure TForm2.N38Click(Sender: TObject);
bgin
Frm2.Hide;
Form7.Show;
end;
rocedure TForm2.N39Click(Sender: TObject);
begin
Form2.Hide;
Form8.Show;
procedure TForm2.N40Click(Sender: TObject);
begin
Form2.Hide;
Form9.Show;
end;
procedure TForm2.N30Click(Sender: TObject);
begin
Form2.Hide;
Form10.Show;
end;
procedure TForm2.N18Click(Sender: TObject);
begin
      jj1:=1;
     t:=0;
       for i:=1 to 100 do
  begin
     t:=t+((dlinz1/vrux1)/100);
                    //
```

```
  if (FErrors <> nil) then FErrors.Close;
  //
  if not CreatePZ(Memo2.Text)
  then
  begin
  //
  Application.CreateForm(TFErrors, FErrors);
  FErrors.LBErrors.Items.Assign(ErrorList);
  FErrors.Show;
  exit;
  end;
  V(P0,t);
  P01[i]:=P0;
  t1[i]:=t;
 end;
                    for i:=1 to 100 do
                  begin
                Chart1.SeriesList[0].AddXY(t1[i],P01[i],'',clRed);
                  end;
    t011:=InputBox('Введіть значення часу t, c',' t=','');
    Val(t011,t,code);
        V(P0,t);
    t022:=format('%9.5f',[P0]);
    Memo1.Lines.Add('Початковий натяг P0='+t022);
    nat[jj1]:=P0;
    nat1[jj1]:=P0;
end;
procedure TForm2.N27Click(Sender: TObject);
begin
```

```
  P0:=nat[jj1];
  jj1:=jj1+1;
  nat1[jj1]:=P0;
  vvv[jj1]:=2;
  OBOZ1[jj1]:='напрямна';
  OBOZ2[jj1]:='без рад.ох.';
  OBOZ3[jj1]:='Yellow';
  OBOZ4[jj1]:='NRO';
  FP1:=InputBox('Введіть розрахунковий кут охоплення, рад',' FP=','');
  Val(FP1,FP,code);
  R:=InputBox('Введіть радіус напрямної, мм',' R=','');
  Val(R,R1,code);

              a:=P0;
      b:=300;
      e:=0.01;
      h:=0.01;
       c:=h; k:=0; x:=a;
    RN:=DN/2;
    kt:=AT/exp(BT*ln(P0/R1));
d0:=P0*(R1+RN)/(RN*P0+E1N*BSLN*sqr(R1+RN));
   if d0>1 then d0:=1;
d:=d0*exp(kt*FP);
   if d>1 then d:=1;
SS0:=1-d0*(2*RN/R1);
SS1:=1-d*(2*RN/R1);
GG0:=1-(BGN/(2*P0*sqr(R1+RN)));
GG1:=1-(BGN/(2*X*sqr(R1+RN)));
USS0:=(Pi/2)-arctan(SS0/sqrt(1-SS0*SS0));
```

```
    USS1:=(Pi/2)-arctan(SS1/sqrt(1-SS1*SS1));
    UGG0:=(Pi/2)-arctan(GG0/sqrt(1-GG0*GG0));
    UGG1:=(Pi/2)-arctan(GG1/sqrt(1-GG1*GG1));
    V11:=FP+USS0+USS1-UGG0-UGG1;
            V1(F,X,P0,BSLN,RN,E1N,BGN,FP,AT,BT,R1,d0,kt,V11);
            w:= Trunc(F/abs(F));
          repeat
           x:=x+c;
           if x-c>=b then Break;
            V1(F,X,P0,BSLN,RN,E1N,BGN,FP,AT,BT,R1,d0,kt,V11);
           if F*w/c>0 then Continue;
           c:=-c/4;
           if abs(c)>(e/4) then Continue;
           k:=k+1;
          kk:=format('%5.2f',[R1]);
          kkk:=format('%17.8f',[x]);
          kkkkk1:=format('%17.8f',[P0]);
          kkkk1:=format('%3.0d',[jj1-1]);
          memo1.Lines.Add('Без радіального охоплення (Yellow) - '+kkkk1);
          memo1.Lines.Add('P('+kk+')='+kkk);
          memo1.Lines.Add('P0('+kk+')='+kkkkk1);
          nat[jj1]:=X;
           c:=h;
           w:=-w;
          until False;
         otnat[jj1]:= nat[jj1]/nat1[jj1];
end;
procedure TForm2.N9Click(Sender: TObject);
begin
```

```
   kivuz1:=kivuz1+1;
         Series2.Clear;
         Series3.Clear;
         for jj1:=1 to kivuz1 do

        begin
         Series2.Add(nat1[jj1],'',clBlue);
         Series3.Add(nat[jj1],'',clRed);
         end;

end;
procedre TForm2.N10Click(Sender: TObject);
begin
          for jj1:=1 to kivuz1 do
         begin
         Series4.Add(otnat[jj1],'',clGreen);
          if  vvv[jj1]=1then
                begin
                    Series5.Add(vvv[jj1],'',clRed);
                    end
                else
                    begin
                     if  vvv[jj1]=2 then
                     begin
                     Series5.Add(vvv[jj1]/2,'',clYellow);
                     end
                    else
                      if  vvv[jj1]=3 then
                     begin
```

```
  Series5.Add(vvv[jj1]/3,",clBlue);
  end
else
   if  vvv[jj1]=4 then
  begin
  Series5.Add(vvv[jj1]/4,",clWhite);
  end
else
   if  vvv[jj1]=5 then
  begin
  Series5.Add(vvv[jj1]/5,",clAqua);
  end
else
  if  vvv[jj1]=6 then
  begin
  Series5.Add(vvv[jj1]/6,",clRed);
  end
else
  if  vvv[jj1]=7 then
  begin
  Series5.Add(vvv[jj1]/7,",clYellow);
  end
else
  if  vvv[jj1]=8 then
  begin
  Series5.Add(vvv[jj1]/8,",clBlue);
  end
else
  if  vvv[jj1]=9 then
```

```
                    begin
                    Series5.Add(vvv[jj1]/9,'',clWhite);
                    end
                  else
                    if  vvv[jj1]=10 then
                    begin
                    Series5.Add(vvv[jj1]/10,'',clAqua);
                    end
                  else
                    begin
                    Series5.Add(vvv[jj1]/11,'',clRed);
                    end;
                end;
        end;
end;
procedure TForm2.N47Click(Sender: TObject);
begin
Form2.Hide;
Form11.Show;
end;
procedure TForm2.N26Click(Sender: TObject);
begin
      P0:=nat[jj1];
      jj1:=jj1+1;
      nat1[jj1]:=P0;
      vvv[jj1]:=1;
      OBOZ1[jj1]:='напрямна';
      OBOZ2[jj1]:='з рад.ох.';
      OBOZ3[jj1]:='Red';
```

```
  OBOZ4[jj1]:='ERO';
  FP1:=InputBox('Введіть розрахунковий кут охоплення, рад',' FP=','');
  Val(FP1,FP,code);
  R:=InputBox('Введіть радіус напрямної, мм',' R=','');
  Val(R,R1,code);
  BER1:=InputBox('Введіть радіальний кут охоплення, рад',' BER=','');
  Val(BER1,BER,code);
       a:=P0;
       b:=1000;
       e:=0.01;
       h:=0.01;
        c:=h; k:=0; x:=a;
    RN:=DN/2;
    kt:= (4*sin(BER/2)/(BER+sin(BER)))*AT/exp(BT*ln(P0/R1));
d0:=P0*(R1+RN)/(RN*P0+E1N*BSLN*sqr(R1+RN));
   if d0>1 then d0:=1;
d:=d0*exp(kt*FP);
   if d>1 then d:=1;
SS0:=1-d0*(2*RN/R1);
SS1:=1-d*(2*RN/R1);
GG0:=1-(BGN/(2*P0*sqr(R1+RN)));
GG1:=1-(BGN/(2*X*sqr(R1+RN)));
USS0:=1.57-arctan(SS0/sqrt(1-SS0*SS0));
USS1:=1.57-arctan(SS1/sqrt(1-SS1*SS1));
UGG0:=1.57-arctan(GG0/sqrt(1-GG0*GG0));
UGG1:=1.57-arctan(GG1/sqrt(1-GG1*GG1));
V11:=FP+USS0+USS1-UGG0-UGG1;
          V2(F,X,P0,BSLN,RN,E1N,BGN,FP,BER,AT,BT,R1,d0,kt,V11);
```

```
            w:= Trunc(F/abs(F));
          repeat
           x:=x+c;
           if x-c>=b then Break;
            V2(F,X,P0,BSLN,RN,E1N,BGN,FP,BER,AT,BT,R1,d0,kt,V11);

           if F*w/c>0 then Continue;
           c:=-c/4;
           if abs(c)>(e/4) then Continue;
           k:=k+1;
          kk:=format('%5.2f',[R1]);
          kkk:=format('%17.8f',[x]);
          kkkkk1:=format('%17.8f',[P0]);
          kkkk1:=format('%3.0d',[jj1-1]);
          memo1.Lines.Add('З радіальним охопленням (RED) - '+kkkk1);
          memo1.Lines.Add('P('+kk+')='+kkk);
          memo1.Lines.Add('P0('+kk+')='+kkkkk1);
          nat[jj1]:=X;
           c:=h;
           w:=-w;
         until False;
        otnat[jj1]:= nat[jj1]/nat1[jj1];
end;
procedure TForm2.N19Click(Sender: TObject);
begin
  Form2.Hide;
  Form12.Show;

end;
```

```
procedure TForm2.N20Click(Sender: TObject);
begin
      P0:=nat[jj1];
      jj1:=jj1+1;
      nat1[jj1]:=P0;
      vvv[jj1]:=4;
      OBOZ1[jj1]:='натягувач';
      OBOZ2[jj1]:='гребінчат.';
      OBOZ3[jj1]:='White';
      OBOZ4[jj1]:='NPA';
      FP1:=InputBox('Введіть результуючий кут охоплення гребінчатого
натягувача, рад',' FPP=','');
      Val(FP1,FP,code);
      R:=InputBox('Введіть радіус стріжней гребінчатого натягувача, мм','
r=','');
      Val(R,R1,code);
          a:=P0;
          b:=1000;
          e:=0.01;
          h:=0.01;
           c:=h; k:=0; x:=a;


        RN:=DN/2;
        kt:=AT/exp(BT*ln(P0/R1));
    d0:=P0*(R1+RN)/(RN*P0+E1N*BSLN*sqr(R1+RN));
       if d0>1 then d0:=1;
    d:=d0*exp(kt*FP);
```

```
   if d>1 then d:=1;
SS0:=1-d0*(2*RN/R1);
SS1:=1-d*(2*RN/R1);
GG0:=1-(BGN/(2*P0*sqr(R1+RN)));
GG1:=1-(BGN/(2*X*sqr(R1+RN)));
USS0:=1.57-arctan(SS0/sqrt(1-SS0*SS0));
USS1:=1.57-arctan(SS1/sqrt(1-SS1*SS1));
UGG0:=1.57-arctan(GG0/sqrt(1-GG0*GG0));
UGG1:=1.57-arctan(GG1/sqrt(1-GG1*GG1));
V11:=FP+USS0+USS1-UGG0-UGG1;
          V1(F,X,P0,BSLN,RN,E1N,BGN,FP,AT,BT,R1,d0,kt,V11);
          w:= Trunc(F/abs(F));
        repeat
         x:=x+c;
         if x-c>=b then Break;
          V1(F,X,P0,BSLN,RN,E1N,BGN,FP,AT,BT,R1,d0,kt,V11);
         if F*w/c>0 then Continue;
         c:=-c/4;
         if abs(c)>(e/4) then Continue;
         k:=k+1;
        kk:=format('%5.2f',[R1]);
        kkk:=format('%17.8f',[x]);
        kkkkk1:=format('%17.8f',[P0]);
        kkkk1:=format('%3.0d',[jj1-1]);
        memo1.Lines.Add('Натягувач гребінчатий (White) - '+kkkk1);
        memo1.Lines.Add('P('+kk+')='+kkk);
        memo1.Lines.Add('P0('+kk+')='+kkkkk1);
```

```
            nat[jj1]:=X;
             c:=h;
             w:=-w;
            until False;
           otnat[jj1]:= nat[jj1]/nat1[jj1];
end;
procedure TForm2.N21Click(Sender: TObject);
begin
  Form2.Hide;
  Form13.Show;
end;
procedure TForm2.N22Click(Sender: TObject);
begin
 Form2.Hide;
 Form14.Show;
end;
procedure TForm2.N23Click(Sender: TObject);
begin
Form2.Hide;
Form15.Show;
end;
procedure TForm2.N24Click(Sender: TObject);
begin
Form2.Hide;
Form16.Show;
end;
procedure TForm2.N25Click(Sender: TObject);
begin
```

```
Form2.Hide;
Form17.Show;
end;
procedure TForm2.N45Click(Sender: TObject);
begin
Form2.Hide;
Form18.Show;
end;
procedure TForm2.N46Click(Sender: TObject);
begin
Form2.Hide;
Form19.Show;
end;
procedure TForm2.N43Click(Sender: TObject);
begin
Form2.Hide;
Form20.Show;
end;
procedure TForm2.N44Click(Sender: TObject);
begin
Form2.Hide;
Form21.Show;
end;
procedure TForm2.N48Click(Sender: TObject);
    var
    Prn:TextFile;
    k:Integer;
begin
```

```
      AssignPrn(Prn);
       Rewrite(Prn);
       Printer.Canvas.Font:=memo1.Font;
       for k:=0 to memo1.Lines.Count-1 do
       WriteLn(Prn,memo1.Lines[k]);
       CloseFile(Prn);
end;
procedure TForm2.N49Click(Sender: TObject);
begin
  TRUXU:= dlinz1/vrux1;
  TRUXU2:=format('%9.1f',[TRUXU]);
  Memo1.Lines.Add('Час t руху нитки, с  ='+TRUXU2);
end;
procedure TForm2.N12Click(Sender: TObject);
begin
Form2.Hide;
Form22.Show;
end;
procedure TForm2.N16Click(Sender: TObject);
begin
 Form2.Hide;
 Form23.Show;
end;
procedure TForm2.N14Click(Sender: TObject);
begin
      s:=0;
        for i:=1 to 100 do
   begin
      s:=s+(dlinz1)/100;
```

```
                    //
   if (FErrors <> nil) then FErrors.Close;
   //
   if not CreatePZ(Memo2.Text)
   then
   begin
   //
   Application.CreateForm(TFErrors, FErrors);
   FErrors.LBErrors.Items.Assign(ErrorList);
   FErrors.Show;
   exit;
   end;
   V3(DN2,s);
    DN22[i]:=DN2;
   s1[i]:=s;
  end;
                      for i:=1 to 100 do
                    begin
                  Form2.Chart4.SeriesList[0].AddXY(s1[i],DN22[i],'',clGreen);
                    end;
      s11:=InputBox('Значення дугової координати, s',' s=','');
      Val(s11,s,code);
      s22:=format('%9.5f',[DN2]);
      Form2.Memo1.Lines.Add('Диаметр перетину DN2='+s22);
end;
procedure TForm2.N31Click(Sender: TObject);
begin
Form2.Hide;
Form24.Show;
```

```
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
var
  FStream: TFileStream;
  I, J: Integer;
begin
  Result := False;
```

```
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
 end;
procedure TForm2.N13Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
procedure TForm2.N11Click(Sender: TObject);
begin
                    begin
        StringGrid1.cells[0,0]:='NU';
        StringGrid1.cells[1,0]:='KU';
        StringGrid1.cells[2,0]:='DN';
        StringGrid1.cells[3,0]:='V';
```

```
StringGrid1.cells[4,0]:='L';
StringGrid1.cells[5,0]:='T';
StringGrid1.cells[6,0]:='P0';
StringGrid1.cells[7,0]:='P';
StringGrid1.cells[8,0]:='P/P0';
          end;
//    StringGrid1.cells[0,1]:=format('%9.4f',[mt[n]]);
   StringGrid1.cells[1,1]:=format('%3.0d',[kivuz1-1]);
   StringGrid1.cells[2,1]:=format('%9.4f',[DN]);
   StringGrid1.cells[3,1]:=format('%9.4f',[vrux1]);
   StringGrid1.cells[4,1]:=format('%9.4f',[dlinz1]);
   StringGrid1.cells[5,1]:=format('%9.4f',[TRUXU]);
  // StringGrid1.cells[6,1]:=format('%9.4f',[]);
  // StringGrid1.cells[7,1]:=format('%9.4f',[mmpux11[n]]);
  // StringGrid1.cells[8,1]:=format('%9.4f',[mmpux11[n]]);
     for jj1:=1 to kivuz1-1 do
        begin
   StringGrid1.cells[0,jj1+1]:=format('%3.0d',[jj1]);
//   StringGrid1.cells[1,n+1]:=format('%9.4f',[momg11[n]]);
//   StringGrid1.cells[2,n+1]:=format('%9.4f',[mfi11[n]]);
//   StringGrid1.cells[3,n+1]:=format('%9.4f',[meps11[n]]);
//   StringGrid1.cells[4,n+1]:=format('%9.4f',[momg21[n]]);
//   StringGrid1.cells[5,n+1]:=format('%9.4f',[mfi21[n]]);
   StringGrid1.cells[6,jj1+1]:=format('%9.4f',[nat1[jj1+1]]);
   StringGrid1.cells[7,jj1+1]:=format('%9.4f',[nat[jj1+1]]);
   StringGrid1.cells[8,jj1+1]:=format('%9.4f',[otnat[jj1+1]]);
          end;
     for jj1:=1 to kivuz1-1 do
        begin
```

```
            StringGrid1.cells[1,jj1+1]:=OBOZ1[jj1+1];
            StringGrid1.cells[2,jj1+1]:=OBOZ2[jj1+1];
            StringGrid1.cells[3,jj1+1]:=OBOZ3[jj1+1];
            StringGrid1.cells[4,jj1+1]:=OBOZ4[jj1+1];
                end;
end;
procedure TForm2.N15Click(Sender: TObject);
begin
Form2.Hide;
Form25.Show;
end;
procedure TForm2.N17Click(Sender: TObject);
begin
  Form2.Hide;
  Form26.Show;
end;
unit Unit3;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj
, Forms,
  Dialogs, StdCtrls,Menus,TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm3 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
```

```
    Label2: TLabel;
    Memo1: TMemo;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
      private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form3: TForm3;
 { TBV,DN,BGN,E1N,BSLN,AT,BT:Real;
  code:Integer;
  DN1,BGN1,E1N1,BSLN1,AT1,BT1:String; }
implementation
 Uses Unit2;
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
  Val(Edit1.text,TBV,code);
    DN:=0.0395*sqrt(TBV);
    BGN:= TBV*0.0022/27.6;
    E1N:=712.2;
    BSLN:=0.012;
    AT:=0.1656;
    BT:=0.590;
    KTP:=0.29;
    KTPA:=0.32;
```

```
   DN1:=format('%9.5f',[DN]);
   Memo1.Lines.Add('Діаметр нитки,мм D=2r='+DN1);
   BGN1:=format('%9.5f',[BGN]);
   Memo1.Lines.Add('Коефіцієнт жорсткості на згин,сН/мм2 B='+BGN1);
   E1N1:=format('%9.5f',[E1N]);
   Memo1.Lines.Add('Модуль пружності нитки при стисненні,сН/мм
E1='+E1N1);
   BSLN1:=format('%9.5f',[BSLN]);
   Memo1.Lines.Add('Ширина зони контакту,мм b='+BSLN1);
   AT1:=format('%9.5f',[AT]);
   Memo1.Lines.Add('Постійна для коефіцієнту тертя a='+AT1);
   BT1:=format('%9.5f',[BT]);
   Memo1.Lines.Add('Постійна для коефіцієнту тертя b='+BT1);
   KTP1:=format('%9.5f',[KTP]);
   Memo1.Lines.Add('Коефіцієнт тертя по поверхні KTP='+KTP1);
   KTPA1:=format('%9.5f',[KTPA]);
   Memo1.Lines.Add('Коефіцієнт тертя по поверхні KTP='+KTPA1);
   Form2.Memo1.Lines.Add(Form3.Memo1.Text);
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
  Form3.Hide;
  Form2.Show;
end;
interface
uses
   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj
, Forms,
```

```
  Dialogs, StdCtrls,Menus,TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm4 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Button1: TButton;
    Button2: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form4: TForm4;
 { TV,DN,BGN,E1N,BSLN,AT,BT:Real;
  code:Integer;
  DN1,BGN1,E1N1,BSLN1,AT1,BT1:String;  }
  implementation
Uses Unit2;
{$R *.dfm}
procedure TForm4.Button1Click(Sender: TObject);
begin
    Val(Edit1.text,TV,code);
    DN:=0.0427*sqrt(TV);
```

```
  BGN:= TV*0.0023/29.9;
  E1N:=599.0;
  BSLN:=0.015;
  AT:=0.1330;
  BT:=0.0910;
  KTP:=0.21;
  KTPA:=0.25;
  N1:=format('%9.5f',[DN]);
  Memo1.Lines.Add('Діаметр нитки,мм D=2r='+DN1);
  BN1:=format('%9.5f',[BGN]);
  Memo1.Lines.Add('Коефіцієнт жорсткості на згин,сН/мм2 B='+BGN1);
  1N1:=format('%9.5f',[E1N]);
  Memo1.Lines.Add('Модуль пружності нитки при стисненні,сН/мм
E1='+E1N1);
  BSL1:=format('%9.5f',[BSLN]);
  Memo1.Lines.Add('Ширина зони контакту,мм b='+BSLN1);
  AT:=format('%9.5f',[AT]);
  Memo1.Lines.Add('Постійна для коефіцієнту тертя a='+AT1);
  BT1:=format('%9.5f',[BT]);
  Memo1.Lines.Add('Постійна для коефіцієнту тертя b='+BT1);
  KTP1:=format('%9.5f',[KTP]);
  Memo1.Lines.Add('Коефіцієнт тертя по поверхні KTP='+KTP1);
  KTPA1:=format('%9.5f',[KTPA]);
  Memo1.Lines.Add('Коефіцієнт тертя по поверхні KTP='+KTPA1);
  Form2.Memo1.Lines.Add(Form4.Memo1.Text);
end;
procedure TForm4.Button2Click(Sender: TObject);
begin
  Form4.Hide;
```

```
  Form2.Show;
end;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj
, Forms,
 Dialogs, StdCtrls,Menus,TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
 Grids,Jpeg,Printers, ShellAPI;
type
 TForm5 = class(TForm)
   Label1: TLabel;
   Edit1: TEdit;
   Button1: TButton;
   Memo1: TMemo;
   Button2: TButton;
   Label2: TLabel;
   procedure Button2Click(Sender: TObject);
   procedure Button1Click(Sender: TObject);
 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
 Form5: TForm5;
{ TSHL,DN,BGN,E1N,BSLN,AT,BT:Real;
 code:Integer;
 DN1,BGN1,E1N1,BSLN1,AT1,BT1:String;  }
```

```
implementation
Uses Unit2;
{$R *.dfm}
procedure TForm5.Button2Click(Sender: TObject);
begin
Form5.Hide;
Form2.Show;
end;
procedure TForm5.Button1Click(Sender: TObject);
begin
   Val(Edit1.text,TSHL,code);
   DN:=0.0411*sqrt(TSHL);
   BGN:= TSHL*0.0012/29.9;
   E1N:=623.4;
   BSLN:=0.012;
   AT:=0.1455;
   BT:=0.550;
   KTP:=0.24;
   KTPA:=0.23;
   DN1:=format('%9.5f',[DN]);
   Memo1.Lines.Add('Діаметр нитки,мм D=2r='+DN1);
   BGN1:=format('%9.5f',[BGN]);
   Memo1.Lines.Add('Коефіцієнт жорсткості на згин,сН/мм2 B='+BGN1);
   E1N1:=format('%9.5f',[E1N]);
   Memo1.Lines.Add('Модуль пружності нитки при стисненні,сН/мм
E1='+E1N1);
```

```
    BSLN1:=format('%9.5f',[BSLN]);
    Memo1.Lines.Add('Ширина зони контакту,мм b='+BSLN1);
    AT1:=format('%9.5f',[AT]);
    Memo1.Lines.Add('Постійна для коефіцієнту тертя a='+AT1);
    BT1:=format('%9.5f',[BT]);
    Memo1.Lines.Add('Постійна для коефіцієнту тертя b='+BT1);
    KTP1:=format('%9.5f',[KTP]);
    Memo1.Lines.Add('Коефіцієнт тертя по поверхні KTP='+KTP1);
    KTPA1:=format('%9.5f',[KTPA]);
    Memo1.Lines.Add('Коефіцієнт тертя по поверхні KTP='+KTPA1);
    Form2.Memo1.Lines.Add(Form5.Memo1.Text);
end;
interface
uses classes;
type
 TData = record
   Name: string;
   Data:real;
 end;
var
 NConst: integer = 100;
 ErrorList: TStringList;
 PZ: array of integer;
 DataList: array of TData;
const
 MConst = 2;
procedure SyntItem(S:string; First:boolean=false; Pos:Integer=1);
```

```
function CreatePZ(S:string):boolean;
function Calculate(var R:real):boolean;
function SetData(Name:string; Data:real):boolean;
function GetData(Name:string; var Data:real):boolean;
implementation
uses Sysutils, Math, Dialogs,Unit2;
type
 TType = (None, Number, Divider, Ident, Func, Part, All);
 TSynt = record
    mode: TType;
    Number:real;
    Ident:string;
    Error:boolean;
    Pos1,Pos2:integer;
 end;
const
    SetNum: set of char=['0'..'9', ','];
    SetDiv: set of char=[';', '(', ')', '=', '+', '-', '/', '*', '^',
                         '{', ' ', #13];
    SetChar: set of char=['a'..'z','A'..'Z','_'];
    NFunc = 10;
    Functions: array[1..NFunc] of string =
      ('exp','sin','cos','sqrt','abs','ln','tg','arctan','arccos','sqr');
var
 SItem: TSynt;
 TrStack: array of char;
 ConstList: array of real;
```

```
 Position: Integer;
procedure SyntItem(S:string;First:boolean=false;Pos:Integer=1);
var i:integer;
begin
 if (S = '') then begin
  SItem.mode := All;
  exit;
 end;
 if(First) then Position := Pos;
 repeat
  if (S[Position] = '{')
   then begin
    repeat
     Inc(Position)
    until (Position >= Length(S)) or (S[Position] = '}');
    Inc(Position);
   end;
  if(Position <= Length(S)) then
   while ((S[Position] = ' ')or
         (S[Position] = #13)or
         (S[Position] = #10)or
         (S[Position] = #0))
   do Inc(Position);
 until (S[Position] <> '{');
 SItem.Error:=false;
 SItem.Pos1:=Position;
 if(Position > Length(S)) then begin
  SItem.mode := All;
  exit;
```

```
end;
SItem.Ident := S[Position];
if (S[Position] in SetChar)
   then SItem.mode := Ident
else if (S[Position] in SetNum)
   then SItem.mode := Number
else if (S[Position] in SetDiv)
   then begin
    if (S[Position] <> ';')
     then SItem.mode := Divider
     else SItem.mode := Part;
    Inc(Position);
    exit;
   end
else begin
 SItem.mode := None;
 Inc(Position);
 exit;
end;
repeat
 Inc(Position);
 if (SItem.mode = Number)and
   ((S[Position] = '-')or(S[Position] = '+'))and
   (UpCase(S[Position-1])='E')
  then SItem.Ident := SItem.Ident + S[Position]
 else if ((Position > Length(S))or(S[Position] in SetDiv))
  then begin
   if(SItem.mode = Number)
     then try
```

```
        SItem.Number := StrToFloat(SItem.Ident)
       except
        on EConvertError do SItem.Error := true;
       end;
     for i:=1 to NFunc do
      if (LowerCase(SItem.Ident) = Functions[i])
        then begin
         SItem.mode:=Func;
         SItem.Number:=i;
         break;
        end;
      SItem.Pos2:=Position-1;
      exit;
    end
    else SItem.Ident := SItem.Ident + S[Position];
 until false;
end;
procedure ClearPZ;
begin
 ErrorList.Clear;
 SetLength(ConstList,0);
 SetLength(DataList,MConst);
 SetLength(PZ,0);
end;
function CreatePZ(S:string):boolean;
var
    lend:boolean;
    i:integer;
    Assign:boolean;
```

```
   Adress: integer;
   OldMode: TType;
   OldS: char;
procedure code;
begin
 SetLength(PZ,High(PZ)+2);
 case TrStack[High(TrStack)] of
  '+': PZ[High(PZ)] := -1;
  '-': PZ[High(PZ)] := -2;
  '*': PZ[High(PZ)] := -3;
  '/': PZ[High(PZ)] := -4;
  '^': PZ[High(PZ)] := -5;
  'M': PZ[High(PZ)] := -6;
 end;
end;
procedure proc1;
begin
 SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc2;
begin
 code;
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc3;
begin
 code;
 SetLength(TrStack,High(TrStack));
```

```
 lend:=false;
end;
procedure proc4;
begin
 SetLength(TrStack,High(TrStack));
end;
procedure proc5;
begin
 SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := Chr(127+Round(SItem.Number));
end;
procedure proc6;
begin
 SetLength(PZ,High(PZ)+2);
 PZ[High(PZ)] := -Ord(TrStack[High(TrStack)])+27;
 SetLength(TrStack,High(TrStack));
end;
begin
 ClearPZ;
 SetLength(TrStack,1);
 TrStack[0] := '0';
 OldMode := None;
 OldS := ' ';
 Assign := true;
 Adress := 0;
 SyntItem(S,true);
 if (SItem.mode = All)
  then begin
   ErrorList.Add();
```

```
 Result := false;
 exit;
end;
repeat
 if ((OldMode = Func)and(SItem.Ident[1] <> '('))
  then ErrorList.Add('+IntToStr(SItem.Pos1));
 case SItem.mode of
  Number: begin
          if((OldMode <> Divider)and(OldMode <> None)and
            (OldMode <> Part))
           then ErrorList.Add('');
          if (SItem.Error)
           then ErrorList.Add('+IntToStr(SItem.Pos1)+
                              ' - '+IntToStr(SItem.Pos2))
           else begin
            SetLength(ConstList,High(ConstList)+2);
            ConstList[High(ConstList)] := SItem.Number;
            SetLength(PZ,High(PZ)+2);
            PZ[High(PZ)] := High(ConstList);
           end;
          Assign:=false;
         end;
  Ident:  begin
          if((OldMode <> Divider)and(OldMode <> None)and
            (OldMode <> Part))
           then ErrorList.Add('+IntToStr(SItem.Pos1)+' ');
          for i:=0 to High(DataList) do
           begin
            if (UpperCase(SItem.Ident) = DataList[i].Name)
```

```
          then begin
           SetLength(PZ,High(PZ)+2);
           PZ[High(PZ)] := NConst+i;
           break;
          end;
         if(i = High(DataList)) then
          begin
           SetLength(DataList,High(DataList)+2);
           DataList[High(DataList)].Name:=UpperCase(SItem.Ident);
           DataList[High(DataList)].Data:=0;
           SetLength(PZ,High(PZ)+2);
           PZ[High(PZ)] := NConst+High(DataList);
          end;
        end;
       end;
All,Part:begin
       repeat
        lend:=true;
        case TrStack[High(TrStack)] of
         '0': begin
             if (Adress <> 0) then begin
               SetLength(PZ,High(PZ)+3);
               PZ[High(PZ)-1] := -7;
               PZ[High(PZ)] := Adress;
               Adress := 0;
              end;
             break;
             end;
         '(': ErrorList.Add();
```

```
          else proc3;
         end;
        until lend;
        if (ErrorList.Count = 0)
         then Result:=true
         else Result:=false;
        if (SItem.mode = All) then exit
        else begin
         Assign := true;
         SItem.mode := None;
        end
       end;
Divider:begin
       if((OldMode = Divider)and
         ((SItem.Ident[1]<>'=')and
         (SItem.Ident[1]<>'(')and
         (SItem.Ident[1] <> ')'))and
         ((OldS <> '(')and
         (OldS <> ')')and
         (OldS <> '=')))
        then begin
         ErrorList.Add();
         break;
        end;
       repeat
        lend:=true;
        case SItem.Ident[1] of
         '=': if Assign and (OldMode = Ident)
              then begin
```

```
        Adress := PZ[High(PZ)];
        SetLength(PZ,High(PZ));
        SItem.mode := None;
       end
       else ErrorList.Add(' '+IntToStr(SItem.Pos1)+
                          ':);
  '(': if(OldMode = Ident) or (OldMode = Number)
       then ErrorList.Add(' '+IntToStr(SItem.Pos1))
       else  proc1;
  '+','-','M': begin
       if((OldMode = None)or(OldS = '('))
        //
        then if (SItem.Ident[1] = '+') //
         then break
        //                 else SItem.Ident[1] := 'M';
       case TrStack[High(TrStack)] of
        '0','(': proc1;
        '+','-','M': proc2;
        '*','/','^': proc3;
       end;
       end;
  '*','/':
       if OldS = '('
       then ErrorList.Add('+IntToStr(SItem.Pos1))
       else
       case TrStack[High(TrStack)] of
        '0','(','+','-','M': proc1;
        '*','/': proc2;
        '^': proc3;
```

```
            end;
        '^':
            if OldS = '('
            then ErrorList.Add('+IntToStr(SItem.Pos1))
            else
            case TrStack[High(TrStack)] of
             '0','(','+','-','*','/','M': proc1;
             '^': proc2;
            end;
        ')':
            case TrStack[High(TrStack)] of
             '0': ErrorList.Add();
             '(': begin
                   proc4;
                   if (Ord(TrStack[High(TrStack)]) > 127)
                    then proc6;
                  end;
             '+','-','*','/','^','M': proc3;
            end;
       end;
      until lend;
      Assign:=false;
      end;
Func:   begin
      repeat
       lend:=true;
       proc5
      until lend;
      Assign:=false;
```

```
        end;
  None:   ErrorList.Add('+IntToStr(SItem.Pos1));
 end;
 OldMode := SItem.mode;
 OldS := SItem.Ident[1];
 SyntItem(S);
until false;
if(ErrorList.Count = 0)
then Result := true
else Result := false;
end;
function Calculate(var R:real):boolean;
var Stack: array of real;
   i:integer;
begin
 for i:=0 to High(PZ) do begin
  if (i > 0) then
   if (PZ[i-1] = -7) and (i < High(PZ)) then Continue;
  if PZ[i] < -100
   then begin
    try
     case -PZ[i]-100 of
      1: Stack[High(Stack)]:=Exp(Stack[High(Stack)]);
      2: Stack[High(Stack)]:=Sin(Stack[High(Stack)]);
      3: Stack[High(Stack)]:=Cos(Stack[High(Stack)]);
      4: Stack[High(Stack)]:=Sqrt(Stack[High(Stack)]);
      5: Stack[High(Stack)]:=Abs(Stack[High(Stack)]);
      6: Stack[High(Stack)]:=Ln(Stack[High(Stack)]);
      7: Stack[High(Stack)]:=Tan(Stack[High(Stack)]);
```

```
  8: Stack[High(Stack)]:=ArcTan(Stack[High(Stack)]);
  9: Stack[High(Stack)]:=ArcCos(Stack[High(Stack)]);
  10: Stack[High(Stack)]:=Sqr(Stack[High(Stack)]);
 end
 except
  Result := false;
  exit;
 end;
 if(FloatToStr(Stack[High(Stack)]) = 'NAN') or
  (FloatToStr(Stack[High(Stack)]) = 'INF') or
  (FloatToStr(Stack[High(Stack)]) = '-INF')
 then begin
  Result := false;
  exit;
 end
end
else if PZ[i] < 0
then begin
try
 case -PZ[i] of
  1: Stack[High(Stack)-1]:=
    Stack[High(Stack)-1]+Stack[High(Stack)];
  2: Stack[High(Stack)-1]:=
    Stack[High(Stack)-1]-Stack[High(Stack)];
  3: Stack[High(Stack)-1]:=
    Stack[High(Stack)-1]*Stack[High(Stack)];
  4: Stack[High(Stack)-1]:=
      Stack[High(Stack)-1]/Stack[High(Stack)];
  5: Stack[High(Stack)-1]:=
```

```
        Power(Stack[High(Stack)-1],Stack[High(Stack)]);
     6: Stack[High(Stack)]:= -Stack[High(Stack)];
     7: DataList[PZ[i+1]-NConst].Data := Stack[High(Stack)];
    end;
   except
    Result := false;
    exit;
   end;
    if (PZ[i] <> -6)
      then SetLength(Stack,High(Stack));
   end
   else begin
    SetLength(Stack,High(Stack)+2);
    if (PZ[i] < NConst)
     then Stack[High(Stack)]:=ConstList[PZ[i]]
     else Stack[High(Stack)]:=DataList[PZ[i]-NConst].Data;
   end;
 end;
 Result := true;
 R :=Stack[High(Stack)];
end;

function SetData(Name:string; Data:real):boolean;
var i:integer;
begin
 for i:=MConst to High(DataList) do
  if (UpperCase(Name) = DataList[i].Name)
   then  begin
    DataList[i].Data := Data;
```

```
   Result:=true;
   exit;
  end;
 Result := false;
end;
function GetData(Name:string; var Data:real):boolean;
var i:integer;
begin
 for i:=0 to High(DataList) do
  if (UpperCase(Name) = DataList[i].Name)
   then  begin
    Data := DataList[i].Data;
    Result:=true;
    exit;
   end;
 Result := false;
end;
initialization
 SetLength(DataList,MConst);
 DataList[0].Name:='PI';
 DataList[0].Data:=Pi;
 DataList[1].Name:='E';
 DataList[1].Data:=2.71828183;
 ErrorList := TStringList.Create;
finalization
 ErrorList.Free;
end.
```

ЛІСТИНГ ПРОГРАМ ДЛЯ КІНЕМАТИЧНОГО ТА КІНЕТОСТАТИЧНОГО АНАЛІЗУ МЕХАНІЗМІВ

```
unit Unit1PM;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TPMForm1 = class(TForm)
    PMLabel1: TLabel;
    PMLabel2: TLabel;
    PMLabel3: TLabel;
    PMLabel4: TLabel;
    PMLabel5: TLabel;
    PMImage1: TImage;
    PMButton1: TButton;
    Label1: TLabel;
    procedure PMButton1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  PMForm1: TPMForm1;
implementation
uses Unit2PM;
```

```
{$R *.dfm}
procedure TPMForm1.PMButton1Click(Sender: TObject);
begin
PMForm1.Hide;
PMForm2.Show;
end;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TPMForm2 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Memo1: TMemo;
    Label1: TLabel;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Chart1: TChart;
    Button7: TButton;
    Series1: TPointSeries;
    Button8: TButton;
    Button9: TButton;
    Button10: TButton;
    Button11: TButton;
```

```
    Button12: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure Button11Click(Sender: TObject);
    procedure Button12Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  PMForm2: TPMForm2;
  i,j:Integer;
   mxA,myA, mvAx,mvAy,mwAx,mwAy:array [0..361] of Real;
implementation
uses Unit1PM,kr0,kr1,po0,po1,ko0,ko1,ku0,ku1;
{$R *.dfm}
procedure TPMForm2.Button1Click(Sender: TObject);
begin
PMForm1.Close;
end;
```

```
procedure TPMForm2.Button2Click(Sender: TObject);
begin
memo1.Lines.Add('Доданий кривошип');
PMForm2.Hide;
Formkr0.Show;
end;
procedure TPMForm2.Button3Click(Sender: TObject);
begin
memo1.Lines.Add('Додана шатунно-повзунна група Асура');
PMForm2.Hide;
Formpo0.Show;
end;
procedure TPMForm2.Button4Click(Sender: TObject);
begin
memo1.Lines.Add('Додана шатунно-коромислова група Асура');
PMForm2.Hide;
Formko0.Show;
end;
procedure TPMForm2.Button5Click(Sender: TObject);
begin
memo1.Lines.Add('Додана кулісна група Асура');
PMForm2.Hide;
Formku0.Show;
end;
procedure TPMForm2.Button6Click(Sender: TObject);
begin
memo1.Lines.Add('Перейменовано кінематичні параметри від 5А до А');
for i:=0 to 360 do mxA[i]:=mx5A[i];
```

```
for i:=0 to 360 do myA[i]:=my5A[i];
for i:=0 to 360 do mvAx[i]:=mv5Ax[i];
for i:=0 to 360 do mvAy[i]:=mv5Ay[i];
for i:=0 to 360 do mwAx[i]:=mw5Ax[i];
for i:=0 to 360 do mwAy[i]:=mw5Ay[i];
end;
procedure TPMForm2.Button7Click(Sender: TObject);
begin
 Chart1.SeriesList[0].Clear;
 for i:=0 to 360 do
 begin
 Chart1.SeriesList[0].AddXY(mxA[i],myA[i],'',clRed);
 end;
end;
procedure TPMForm2.Button8Click(Sender: TObject);
begin
memo1.Lines.Add('Перейменовано кінематичні параметри від KAB до A');
for i:=0 to 360 do mxA[i]:=mXKAB[i];
for i:=0 to 360 do myA[i]:=mYKAB[i];
for i:=0 to 360 do mvAx[i]:=mVXKAB[i];
for i:=0 to 360 do mvAy[i]:=mVYKAB[i];
for i:=0 to 360 do mwAx[i]:=mWXKAB[i];
for i:=0 to 360 do mwAy[i]:=mWYKAB[i];
end;
procedure TPMForm2.Button9Click(Sender: TObject);
begin
memo1.Lines.Add('Перейменовано кінематичні параметри від AK до A');
for i:=0 to 360 do mxA[i]:=mXAK[i];
for i:=0 to 360 do myA[i]:=mYAK[i];
```

```
for i:=0 to 360 do mvAx[i]:=mVXAK[i];
for i:=0 to 360 do mvAy[i]:=mVYAK[i];
for i:=0 to 360 do mwAx[i]:=mWXAK[i];
for i:=0 to 360 do mwAy[i]:=mWYAK[i];

end;
procedure TPMForm2.Button10Click(Sender: TObject);
begin
memo1.Lines.Add('Перейменовано кінематичні параметри від ВК до А');
for i:=0 to 360 do mxA[i]:=mXBK[i];
for i:=0 to 360 do myA[i]:=mYBK[i];
for i:=0 to 360 do mvAx[i]:=mVXBK[i];
for i:=0 to 360 do mvAy[i]:=mVYBK[i];
for i:=0 to 360 do mwAx[i]:=mWXBK[i];
for i:=0 to 360 do mwAy[i]:=mWYBK[i];
end;
procedure TPMForm2.Button11Click(Sender: TObject);
begin
memo1.Lines.Add('Перейменовано кінематичні параметри від КСА до А');
for i:=0 to 360 do mxA[i]:=mXKCA[i];
for i:=0 to 360 do myA[i]:=mYKCA[i];
for i:=0 to 360 do mvAx[i]:=mVXKCA[i];
for i:=0 to 360 do mvAy[i]:=mVYKCA[i];
for i:=0 to 360 do mwAx[i]:=mWXKCA[i];
for i:=0 to 360 do mwAy[i]:=mWYKCA[i];
end;
procedure TPMForm2.Button12Click(Sender: TObject);
interface
```

```
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormko0 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    Image2: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Label4: TLabel;
    Edit3: TEdit;
    Label5: TLabel;
    Edit4: TEdit;
    Label6: TLabel;
    Edit5: TEdit;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Edit6: TEdit;
    Label10: TLabel;
    Edit7: TEdit;
    Label11: TLabel;
    Edit8: TEdit;
```

```
    Label12: TLabel;
    Edit9: TEdit;
    Label13: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Formko0: TFormko0;
   code:Integer;
   nz:Real;
   XABK,YABK,XBCK,YBCK,l2,l3,l4,l5:Real;
  implementation
uses ko2,ko1;
{$R *.dfm}
procedure TFormko0.Button1Click(Sender: TObject);
begin
Val(Edit1.text,nz,code);
Val(Edit2.text,l2,code);
Val(Edit3.text,l3,code);
Val(Edit4.text,l4,code);
Val(Edit5.text,l5,code);
Val(Edit6.text,XBCK,code);
Val(Edit7.text,YBCK,code);
Val(Edit8.text,XABK,code);
Val(Edit9.text,YABK,code);
```

```
Formko0.Hide;
Formko1.show;
end;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormko1 = class(TForm)
    btnkrpo: TButton;
    Chart1: TChart;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    Button1ks: TButton;
    Button2ks: TButton;
    Series6: TLineSeries;
    Chart2: TChart;
    Chart3: TChart;
    Chart4: TChart;
    Series8: TLineSeries;
    Series7: TLineSeries;
    Series9: TLineSeries;
    StringGrid1: TStringGrid;
    Series22: TLineSeries;
    Series23: TLineSeries;
    Series24: TLineSeries;
```

Series25: TLineSeries;
Series26: TLineSeries;
Series12: TLineSeries;
Series13: TLineSeries;
Series27: TLineSeries;
Series28: TLineSeries;
Series29: TLineSeries;
Series30: TLineSeries;
Series16: TLineSeries;
Series17: TLineSeries;
Series18: TLineSeries;
Series19: TLineSeries;
Series20: TLineSeries;
Series21: TLineSeries;
Button1: TButton;
Button2: TButton;
Series1: TLineSeries;
Series2: TLineSeries;
Series10: TLineSeries;
Series11: TLineSeries;
Series14: TLineSeries;
Series15: TLineSeries;
Series31: TLineSeries;
Series32: TLineSeries;
Series33: TLineSeries;
Series34: TLineSeries;
Series35: TLineSeries;
Series36: TLineSeries;
Chart5: TChart;

```
    Series37: TPointSeries;
    Series38: TPointSeries;
    Series39: TPointSeries;
    Button3: TButton;
    procedure btnkrpoClick(Sender: TObject);
    procedure Button1ksClick(Sender: TObject);
    procedure Button2ksClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Formko1: TFormko1;
  i,j,n:Integer;
  U1,A1,AC,p,r,A2,U3,U2,xB,yB,xAB,yAB,xCB,yCB:Real;
  SA,SB,XAK,YAK,VXAK,VYAK,WXAK,WYAK:Real;
  CK1,CK,XBK,YBK,VXBK,VYBK,WXBK,WYBK:Real;
  US2,UU2,US3,UU3:Real;
  vBx,vBy,wBx,wBy,vCBx,vCBy:Real;
  vABx,vABy,wABx,wABy,wCBx,wCBy:Real;
  mU1,mU2,mU3,mxB,myB,mxAB,myAB,mxCB,myCB:array [0..361] of Real;
  mUS2,mUU2,mUS3,mUU3:array [0..361] of Real;
  mvBx,mvBy,mwBx,mwBy,mvABx,mvABy,mwABx,mwABy:array [0..361]
of Real;
  mvCBx,mvCBy,mwCBx,mwCBy:array [0..361] of Real;
```

```
 mXBK,mYBK,mVXBK,mVYBK,mWXBK,mWYBK:array [0..361] of Real;
 mXAK,mYAK,mVXAK,mVYAK,mWXAK,mWYAK:array [0..361] of Real;
implementation
 uses Unit2PM,ko0,ko2;
{$R *.dfm}
procedure TFormko1.btnkrpoClick(Sender: TObject);
begin
formko1.Hide;
PMform2.Show;
end;
procedure TFormko1.Button1ksClick(Sender: TObject);
begin
        begin
       StringGrid1.cells[0,0]:='U1';
       StringGrid1.cells[1,0]:='U2';
       StringGrid1.cells[2,0]:='U3';
       StringGrid1.cells[3,0]:='xB';
       StringGrid1.cells[4,0]:='yB';
       StringGrid1.cells[5,0]:='xAB';
       StringGrid1.cells[6,0]:='yAB';
       StringGrid1.cells[7,0]:='xCB';
       StringGrid1.cells[8,0]:='yCB';
       StringGrid1.cells[9,0]:='US2';
       StringGrid1.cells[10,0]:='US3';
       StringGrid1.cells[11,0]:='UU2';
       StringGrid1.cells[12,0]:='UU3';
       StringGrid1.cells[13,0]:='vBx';
       StringGrid1.cells[14,0]:='vBy';
       StringGrid1.cells[15,0]:='vABx';
```

```
        StringGrid1.cells[16,0]:='vABy';
        StringGrid1.cells[17,0]:='vCBx';
        StringGrid1.cells[18,0]:='vCBy';
        StringGrid1.cells[19,0]:='wBx';
        StringGrid1.cells[20,0]:='wBy';
        StringGrid1.cells[21,0]:='wABx';
        StringGrid1.cells[22,0]:='wABy';
        StringGrid1.cells[23,0]:='wCBx';
        StringGrid1.cells[24,0]:='wCBy';
        StringGrid1.cells[25,0]:='XAK';
        StringGrid1.cells[26,0]:='YAK';
        StringGrid1.cells[27,0]:='VXAK';
        StringGrid1.cells[28,0]:='VYAK';
        StringGrid1.cells[29,0]:='WXAK';
        StringGrid1.cells[30,0]:='WYAK';
        StringGrid1.cells[31,0]:='XBK';
        StringGrid1.cells[32,0]:='YBK';
        StringGrid1.cells[33,0]:='VXBK';
        StringGrid1.cells[34,0]:='VYBK';
        StringGrid1.cells[35,0]:='WXBK';
        StringGrid1.cells[36,0]:='WYBK';
                end;
       for i:=0 to 360 do
begin
        U1:=i*Pi/180;
       if l4-mxA[i]<>0 then
          begin
       A1:=arcTan((myA[i]-l5)/(l4-mxA[i]));
          end
```

```
          else
                    begin
 if myA[i]>0     then
    begin
 A1:=Pi/2;
    end
              else
                begin
  if myA[i]<0 then
  begin
  A1:=3*Pi/2;
  end
             else
        begin
       A1:=0;
        end;
                end;
                    end;
 if A1<>0 then
     begin
  AC:=(myA[i]-l5)/sin(A1);
     end
         else
     begin
     AC:=l4-mxA[i];
     end;
p:=(l2+l3+AC)/2;
r:=sqrt((p-l2)*(p-l3)*(p-AC)/p);
A2:=2*arcTan(r/(p-l2));
```

```
        if nz<2  then
          begin
           U3:=2*Pi-(A1+A2);
          end
               else
           begin
          U3:=A2-A1;
           end;
        xB:=l4-l3*cos(U3);
        yB:=l5-l3*sin(U3);
         if xB>mxA[i]   then
          begin
         if nz<2 then
           begin
  U2:=arcTan(sqrt(sqr(l5-l3*sin(U3)-myA[i]))/sqrt(sqr(l4-l3*cos(U3)-
mxA[i])));
           end
               else
           begin
  U2:=2*Pi-arcTan(sqrt(sqr(l5-l3*sin(U3)-myA[i]))/sqrt(sqr(l4-l3*cos(U3)-
mxA[i])));
           end;
          end
                  else
                     begin
                    if xB<mxA[i] then
                       begin
  U2:=Pi+arcTan(sqrt(sqr(l5-l3*sin(U3)-myA[i]))/sqrt(sqr(l4-l3*cos(U3)-
mxA[i])));
```

```
                    end
                        else
                    begin
                  if nz<2 then
                    begin
                   U2:=Pi/2;
                    end
                       else
                    begin
                    U2:=3*Pi/2;
                    end;
                     end;
                   end;
      xAB:=mxA[i]+(l2/2)*cos(U2); yAB:=myA[i]+(l2/2)*sin(U2);
      xCB:=l4-(l3/2)*cos(U3); yCB:=-(l3/2)*sin(U3)+l5;
      US3:=(-mvAx[i]*cos(U2)-mvAy[i]*sin(U2))/(l3*(cos(U3)*sin(U2)-
cos(U2)*sin(U3)));
      US2:= (mvAx[i]/(l2*sin(U2)))-((US3*l3*sin(U3))/(l2*sin(U2)));
      UU2:= (-l2*US2*US2*cos(U2)*cos(U3)-
l3*US3*US3*cos(U3)*sin(U3)+mwAx[i]*cos(U3)+mwAy[i]*sin(U3)-
l2*US2*US2*sin(U2)*sin(U3))/(l2*(sin(U2)*cos(U3)-cos(U2)*sin(u3)));
      UU3:=(mwAx[i]-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2)-
l3*sqr(US3)*cos(U3))/(l3*sin(U3));
      vBx:=l3*US3*sin(U3); vBy:=-l3*US3*cos(U3);
      vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
      vCBx:=(l3/2)*US3*sin(U3); vCBy:=-(l3/2)*US3*cos(U3);
      wBx:=l3*UU3*sin(U3)+l3*sqr(US3)*cos(U3);
      wBy:=-l3*UU3*cos(U3)+l3*sqr(US3)*sin(U3);
```

```
wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
wCBx:=(l3/2)*UU3*sin(U3)+(l3/2)*sqr(US3)*cos(U3);
wCBy:=-(l3/2)*UU3*cos(U3)+(l3/2)*sqr(US3)*sin(U3);
 SA:=-XABK*sin(U2)-YABK*cos(U2);
 SB:=XABK*cos(U2)-YABK*sin(U2);
 XAK:=mxA[i]+SB;
 YAK:=myA[i]-SA;
 VXAK:=mvAx[i]+US2*SA;
 VYAK:=mvAy[i]+US2*SB;
 WXAK:=mwAx[i]+UU2*SA-US2*US2*SB;
 WYAK:=mwAy[i]+UU2*SB+US2*US2*SA;
CK:=-XBCK*sin(U3)-YBCK*cos(U3);
CK1:=XBCK*cos(U3)-YBCK*sin(U3);
XBK:=xB+CK1;
YBK:=yB-CK;
VXBK:=vBx+CK*US3;
VYBK:=vBy+CK1*US3;
WXBK:=wBx+UU3*CK-US3*US3*CK1;
WYBK:=wBy+UU3*CK1+US3*US3*CK;
 mU1[i]:=U1;
 mU2[i]:=U2;
 mU3[i]:=U3;
 mxB[i]:=xB;
 myB[i]:=yB;
 mxAB[i]:=xAB;
 myAB[i]:=yAB;
 mxCB[i]:=xCB;
 myCB[i]:=yCB;
```

```
mUS2[i]:=US2;
mUS3[i]:=US3;
mUU2[i]:=UU2;
mUU3[i]:=UU3;
mvBx[i]:=vBx;
mvBy[i]:=vBy;
mvABx[i]:=vABx;
mvABy[i]:=vABy;
mvCBx[i]:=vCBx;
mvCBy[i]:=vCBy;
mwBx[i]:=wBx;
mwBy[i]:=wBy;
mwABx[i]:=wABx;
mwABy[i]:=wCBx;
mwCBx[i]:=wCBy;
mwCBy[i]:=wCBy;
mXAK[i]:=XAK;
mYAK[i]:=YAK;
mVXAK[i]:=VXAK;
mVYAK[i]:=VYAK;
mWXAK[i]:=WXAK;
mWYAK[i]:=WYAK;
mXBK[i]:=XBK;
mYBK[i]:=YBK;
mVXBK[i]:=VXBK;
mVYBK[i]:=VYBK;
mWXBK[i]:=WXBK;
mWYBK[i]:=WYBK;
```

```
end;
        for n:=0 to 360 do
          begin
      StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
      StringGrid1.cells[1,n+1]:=format('%9.2f',[mU2[n]]);
      StringGrid1.cells[2,n+1]:=format('%9.4f',[mU3[n]]);
      StringGrid1.cells[3,n+1]:=format('%9.4f',[mxB[n]]);
      StringGrid1.cells[4,n+1]:= format('%9.4f',[myB[n]]);
      StringGrid1.cells[5,n+1]:= format('%9.4f',[mxAB[n]]);
      StringGrid1.cells[6,n+1]:=format('%9.4f',[myAB[n]]);
      StringGrid1.cells[7,n+1]:=format('%9.4f',[mxCB[n]]);
      StringGrid1.cells[8,n+1]:=format('%9.4f',[myCB[n]]);
      StringGrid1.cells[9,n+1]:=format('%9.4f',[mUS2[n]]);
      StringGrid1.cells[10,n+1]:=format('%9.4f',[mUS3[n]]);
      StringGrid1.cells[11,n+1]:= format('%9.4f',[mUU2[n]]);
      StringGrid1.cells[12,n+1]:=format('%9.4f',[mUU3[n]]);
      StringGrid1.cells[13,n+1]:=format('%9.4f',[mvBx[n]]);
      StringGrid1.cells[14,n+1]:=format('%9.4f',[mvBy[n]]);
      StringGrid1.cells[15,n+1]:=format('%9.4f',[mvABx[n]]);
      StringGrid1.cells[16,n+1]:=format('%9.4f',[mvABy[n]]);
      StringGrid1.cells[17,n+1]:= format('%9.4f',[mvCBx[n]]);
      StringGrid1.cells[18,n+1]:= format('%9.4f',[mvCBy[n]]);
      StringGrid1.cells[19,n+1]:=format('%9.4f',[mwBx[n]]);
      StringGrid1.cells[20,n+1]:=format('%9.4f',[mwBy[n]]);
      StringGrid1.cells[21,n+1]:=format('%9.4f',[mwABx[n]]);
      StringGrid1.cells[22,n+1]:=format('%9.4f',[mwABy[n]]);
      StringGrid1.cells[23,n+1]:=format('%9.4f',[mwCBx[n]]);
      StringGrid1.cells[24,n+1]:= format('%9.4f',[mwCBy[n]]);
```

```
            StringGrid1.cells[25,n+1]:=format('%9.4f',[mXAK[n]]);
            StringGrid1.cells[26,n+1]:=format('%9.4f',[mYAK[n]]);
            StringGrid1.cells[27,n+1]:=format('%9.4f',[mVXAK[n]]);
            StringGrid1.cells[28,n+1]:=format('%9.4f',[mVYAK[n]]);
            StringGrid1.cells[29,n+1]:=format('%9.4f',[mWXAK[n]]);
            StringGrid1.cells[30,n+1]:= format('%9.4f',[mWYAK[n]]);
            StringGrid1.cells[31,n+1]:=format('%9.4f',[mXBK[n]]);
            StringGrid1.cells[32,n+1]:=format('%9.4f',[mYBK[n]]);
            StringGrid1.cells[33,n+1]:=format('%9.4f',[mVXBK[n]]);
            StringGrid1.cells[34,n+1]:=format('%9.4f',[mVYBK[n]]);
            StringGrid1.cells[35,n+1]:=format('%9.4f',[mWXBK[n]]);
            StringGrid1.cells[36,n+1]:= format('%9.4f',[mWYBK[n]]);
                        end;
end;
procedure TFormko1.Button2ksClick(Sender: TObject);
begin
 Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart1.SeriesList[4].Clear;
Chart1.SeriesList[5].Clear;
Chart1.SeriesList[6].Clear;
Chart1.SeriesList[7].Clear;
Chart1.SeriesList[8].Clear;
Chart1.SeriesList[9].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
```

```
Chart2.SeriesList[3].Clear;
Chart2.SeriesList[4].Clear;
Chart2.SeriesList[5].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
Chart3.SeriesList[4].Clear;
Chart3.SeriesList[5].Clear;
Chart3.SeriesList[6].Clear;
Chart3.SeriesList[7].Clear;
Chart3.SeriesList[8].Clear;
Chart3.SeriesList[9].Clear;
Chart4.SeriesList[0].Clear;
Chart4.SeriesList[1].Clear;
Chart4.SeriesList[2].Clear;
Chart4.SeriesList[3].Clear;
Chart4.SeriesList[4].Clear;
Chart4.SeriesList[5].Clear;
Chart4.SeriesList[6].Clear;
Chart4.SeriesList[7].Clear;
Chart4.SeriesList[8].Clear;
Chart4.SeriesList[9].Clear;
Chart5.SeriesList[0].Clear;
Chart5.SeriesList[1].Clear;
Chart5.SeriesList[2].Clear;
for j:=0 to 360 do
 begin
```

```
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mxB[j],",clYellow);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,myB[j],",clBlue);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mxAB[j],",clWhite);
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,myAB[j],",clBlack);
Chart1.SeriesList[4].AddXY(mU1[j]*180/Pi,mxCB[j],",clRed);
Chart1.SeriesList[5].AddXY(mU1[j]*180/Pi,myCB[j],",clGreen);
Chart1.SeriesList[6].AddXY(mU1[j]*180/Pi,mXBK[j],",clRed);
Chart1.SeriesList[7].AddXY(mU1[j]*180/Pi,mYBK[j],",clGreen);
Chart1.SeriesList[8].AddXY(mU1[j]*180/Pi,mXAK[j],",clYellow);
Chart1.SeriesList[9].AddXY(mU1[j]*180/Pi,mYAK[j],",clBlue);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mUS2[j],",clBlue);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mU2[j],",clWhite);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mUU2[j],",clBlack);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,mUS3[j],",clRed);
Chart2.SeriesList[4].AddXY(mU1[j]*180/Pi,mU3[j],",clGreen);
Chart2.SeriesList[5].AddXY(mU1[j]*180/Pi,mUU3[j],",clYellow);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mvBx[j],",clYellow);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mvBy[j],",clBlue);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mvABx[j],",clRed);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mvABy[j],",clGreen);
Chart3.SeriesList[4].AddXY(mU1[j]*180/Pi,mvCBx[j],",clWhite);
Chart3.SeriesList[5].AddXY(mU1[j]*180/Pi,mvCBy[j],",clBlack);
Chart3.SeriesList[6].AddXY(mU1[j]*180/Pi,mVXBK[j],",clWhite);
Chart3.SeriesList[7].AddXY(mU1[j]*180/Pi,mVYBK[j],",clBlack);
Chart3.SeriesList[8].AddXY(mU1[j]*180/Pi,mVXAK[j],",clYellow);
Chart3.SeriesList[9].AddXY(mU1[j]*180/Pi,mVYAK[j],",clBlue);
```

```
Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mwBx[j],",clYellow);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mwBy[j],",clBlue);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mwABx[j],",clWhite);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mwABy[j],",clBlack);
Chart4.SeriesList[4].AddXY(mU1[j]*180/Pi,mwCBx[j],",clGreen);
Chart4.SeriesList[5].AddXY(mU1[j]*180/Pi,mwCBy[j],",clRed);
Chart4.SeriesList[6].AddXY(mU1[j]*180/Pi,mWXBK[j],",clGreen);
Chart4.SeriesList[7].AddXY(mU1[j]*180/Pi,mWYBK[j],",clRed);
Chart4.SeriesList[8].AddXY(mU1[j]*180/Pi,mWXAK[j],",clYellow);
Chart4.SeriesList[9].AddXY(mU1[j]*180/Pi,mWYAK[j],",clBlue);
Chart5.SeriesList[0].AddXY(mxAB[j],myAB[j],",clRed);
Chart5.SeriesList[1].AddXY(mXAK[j],mYAK[j],",clGreen);
Chart5.SeriesList[2].AddXY(mXBK[j],mYBK[j],",clBlue);
end;
end;
procedure TFormko1.Button1Click(Sender: TObject);
begin
Formko1.Hide;
Formko2.Show;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
```

```
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
```

```
  end;
 end;
procedure TFormko1.Button2Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then

ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormko2 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Edit2: TEdit;
    Label2: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    StringGrid1: TStringGrid;
    Chart1: TChart;
    Series1: TLineSeries;
```

```
    Series2: TLineSeries;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    Series6: TLineSeries;
    Series7: TLineSeries;
    Series8: TLineSeries;
    Series9: TLineSeries;
    Button4: TButton;
    Edit3: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Edit4: TEdit;
    Label5: TLabel;
    Edit5: TEdit;
    Button5: TButton;
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Formko2: TFormko2;
  i,j,n:Integer;
```

```
XVS,YVS,MVS,m2,m3,I2,I3,FI2x,FI2y,FI3x,FI3y,l,G,h,h1,XB1,YB1,XA1,YA
1,XC1,YC1,RA,RB,RC:Real;
  mXA1,mYA1,mXB1,mYB1,mXC1,mYC1,mRA,mRB,mRC:array [0..361] of
Real;
 implementation
uses Unit2PM,ko1,ko0;
{$R *.dfm}
procedure TFormko2.Button3Click(Sender: TObject);
begin
Formko2.Hide;
Formko1.Show;
end;
procedure TFormko2.Button2Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart1.SeriesList[4].Clear;
Chart1.SeriesList[5].Clear;
Chart1.SeriesList[6].Clear;
Chart1.SeriesList[7].Clear;
Chart1.SeriesList[8].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mXA1[j],",clRed);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,mYA1[j],",clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mXB1[j],",clYellow);
```

```
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,mYB1[j],'',clBlue);
Chart1.SeriesList[4].AddXY(mU1[j]*180/Pi,mXC1[j],'',clWhite);
Chart1.SeriesList[5].AddXY(mU1[j]*180/Pi,mYC1[j],'',clBlack);
Chart1.SeriesList[6].AddXY(mU1[j]*180/Pi,mRA[j],'',clWhite);
Chart1.SeriesList[7].AddXY(mU1[j]*180/Pi,mRB[j],'',clBlack);
Chart1.SeriesList[8].AddXY(mU1[j]*180/Pi,mRC[j],'',clBlue);

end;
procedure TFormko2.Button1Click(Sender: TObject);
begin
        Val(Edit1.text,m2,code);
        Val(Edit2.text,m3,code);
        Val(Edit3.text,MVS,code);
        Val(Edit4.text,XVS,code);
        Val(Edit5.text,YVS,code);
          begin
        StringGrid1.cells[0,0]:='U1';
        StringGrid1.cells[1,0]:='XA';
        StringGrid1.cells[2,0]:='YA';
        StringGrid1.cells[3,0]:='XB';
        StringGrid1.cells[4,0]:='YB';
        StringGrid1.cells[5,0]:='XC';
        StringGrid1.cells[6,0]:='YC';
        StringGrid1.cells[7,0]:='RA';
        StringGrid1.cells[8,0]:='RB';
        StringGrid1.cells[9,0]:='RC';
              end;
```

```
for i:=0 to 360 do
begin
I2:=m2*l2/12; I3:=m3*l3/12;
wABx:=mwABx[i]; wABy:=mwABy[i];
wCBx:=mwCBx[i]; wCBy:=mwCBy[i];
FI2x:=m2*wABx;
FI2y:=m2*wABy;
FI3x:=m3*wCBx;
FI3y:=m3*wCBy;
l:=sqrt(sqr(l4-mxA[i])+sqr(myA[i]-l5));
G:=arcTan(((myA[i]-l5)/l)/(sqrt(1-sqr((myA[i]-l5)/l))));
YA1:=(MVS+I2*mUU2[i]+I3*mUU3[i]+FI2y*(l3*cos(mU3[i])+(l2/2)*cos(mU
2[i]))+FI2x*(myA[i]-
l5+(l2/2)*sin(mU2[i]))+YVS*(l3/2)*cos(mU3[i])+FI3y*(l3/2)*cos(mU3[i])+X
VS*(l3/2)*sin(mU3[i])+FI3x*(l3/2)*sin(mU3[i]))/l;
YC1:=-YVS*cos(G)-XVS*sin(G)-YA1-FI2y*cos(G)-FI2x*sin(G)-
FI3y*cos(G)-FI3x*sin(G);
h:=l2*sin(mU2[i]+G); h1:=l-l2*cos(mU2[i]+G);
XC1:=(I3*mUU3[i]+MVS-YC1*h1-FI3y*(l3/2)*cos(mU3[i])-
FI3x*(l3/2)*sin(mU3[i]))/h;
YB1:=-YVS*cos(G)-XVS*sin(G)-YC1-FI3y*cos(G)-FI3x*sin(G);
XB1:=FI3y*sin(G)-FI3x*cos(G)-XC1+XVS*cos(G)-YVS*sin(G);
XA1:=XB1-FI2x*cos(G)-FI2y*sin(G);
RA:=sqrt(sqr(XA1)+sqr(YA1));
RB:=sqrt(sqr(XB1)+sqr(YB1));
RC:=sqrt(sqr(XC1)+sqr(YC1));
mXA1[i]:=XA1;
mYA1[i]:=YA1;
mXB1[i]:=XB1;
```

```
        mYB1[i]:=YB1;
        mXC1[i]:=XC1;
        mYC1[i]:=YC1;
        mRA[i]:=RA;
        mRB[i]:=RB;
        mRC[i]:=RC;
        end;
            for n:=0 to 360 do
                begin
          StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
          StringGrid1.cells[1,n+1]:=format('%11.4f',[mXA1[n]]);
          StringGrid1.cells[2,n+1]:=format('%11.4f',[mYA1[n]]);
          StringGrid1.cells[3,n+1]:=format('%11.4f',[mXB1[n]]);
          StringGrid1.cells[4,n+1]:=format('%11.4f',[mYB1[n]]);
          StringGrid1.cells[5,n+1]:=format('%11.4f',[mXC1[n]]);
          StringGrid1.cells[6,n+1]:=format('%11.4f',[mYC1[n]]);
          StringGrid1.cells[7,n+1]:=format('%11.4f',[mRA[n]]);
          StringGrid1.cells[8,n+1]:=format('%11.4f',[mRB[n]]);
          StringGrid1.cells[9,n+1]:=format('%11.4f',[mRC[n]]);
                end;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
```

```
 begin
   L := Length(AValue);
   CXlsLabel[1] := 8 + L;
   CXlsLabel[2] := ARow;
   CXlsLabel[3] := ACol;
   CXlsLabel[5] := L;
   XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
   XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
   {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
   CXlsEof: array[0..1] of Word = ($0A, 00);
 var
   FStream: TFileStream;
   I, J: Integer;
 begin
   Result := False;
   FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
   try
     CXlsBof[4] := 0;
     FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
     for i := 0 to AGrid.ColCount - 1 do
       for j := 0 to AGrid.RowCount - 1 do
         XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
     FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
     Result := True;
   finally
```

```
    FStream.Free;
  end;
 end;
procedure TFormko2.Button4Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
interface
uses
   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj
, Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormKr0 = class(TForm)
    ButtonKR01: TButton;
    Edit1KR0: TEdit;
    Edit2KR0: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    procedure ButtonKR01Click(Sender: TObject);
  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;
var
  FormKr0: TFormKr0;
   code:Integer;
   l1,US1:Real;
implementation
uses kr1,kr2;
{$R *.dfm}
procedure TFormKr0.ButtonKR01Click(Sender: TObject);
begin
        Val(Edit1KR0.text,US1,code);
        Val(Edit2KR0.text,l1,code);
formKr0.Hide;
formKr1.Show;
end;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
  type
  TFormKr1 = class(TForm)
    btnkrpo: TButton;
    Chart1: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series3: TLineSeries;
```

```
  Series4: TLineSeries;
  Button1ks: TButton;
  Button2ks: TButton;
  Chart2: TChart;
  Chart4: TChart;
  Series8: TLineSeries;
  Series10: TLineSeries;
  Series11: TLineSeries;
  Series12: TLineSeries;
  Series13: TLineSeries;
  Series14: TLineSeries;
  Series15: TLineSeries;
  Series16: TLineSeries;
  Series17: TLineSeries;
  StringGrid1: TStringGrid;
  Button1: TButton;
  Button2: TButton;
  Chart3: TChart;
  procedure btnkrpoClick(Sender: TObject);
  procedure Button1ksClick(Sender: TObject);
  procedure Button2ksClick(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

```
var
  FormKr1: TFormKr1;
  code,i,j,n:Integer;
  U1,x5A,y5A,x50A,y50A:Real;
  v5Ax,v5Ay,v50Ax,v50Ay:Real;
  w5Ax,w5Ay,w50Ax,w50Ay:Real;
  mU1,mx5A,my5A,mx50A,my50A,mv5Ax,mv5Ay,mv50Ax,mv50Ay:array
[0..361] of Real;
   mw5Ax,mw5Ay,mw50Ax,mw50Ay:array [0..361] of Real;
implementation
 uses Unit2PM,kr2,kr0;
{$R *.dfm}
procedure TFormKr1.btnkrpoClick(Sender: TObject);
begin
formKr1.Hide;
PMform2.Show;
end;
procedure TFormKr1.Button1ksClick(Sender: TObject);
begin
          begin
        StringGrid1.cells[0,0]:='U1';
        StringGrid1.cells[1,0]:='x5A';
        StringGrid1.cells[2,0]:='y5A';
        StringGrid1.cells[3,0]:='x50A';
        StringGrid1.cells[4,0]:='y50A';
        StringGrid1.cells[5,0]:='v5Ax';
        StringGrid1.cells[6,0]:='v5Ay';
        StringGrid1.cells[7,0]:='v50Ax';
        StringGrid1.cells[8,0]:='v50Ay';
```

```
StringGrid1.cells[9,0]:='w5Ax';
StringGrid1.cells[10,0]:='w5Ay';
StringGrid1.cells[11,0]:='w50Ax';
StringGrid1.cells[12,0]:='w50Ay';
  end;
 for i:=0 to 360 do
 begin
 U1:=i*Pi/180;
 x5A:=l1*cos(U1);
 y5A:=l1*sin(U1);
 x50A:=0.5*l1*cos(U1);
 y50A:=0.5* l1*sin(U1);
 v5Ax:= -US1*l1*sin(U1);
 v5Ay:=US1*l1*cos(U1);
 v50Ax:= -0.5*US1*l1*sin(U1);
 v50Ay:=0.5*US1*l1*cos(U1);
 w5Ax:=-sqr(US1)*l1*cos(U1);
 w5Ay:=-sqr(US1)*l1*sin(U1);
 w50Ax:=-0.5*sqr(US1)*l1*cos(U1);
 w50Ay:=-0.5*sqr(US1)*l1*sin(U1);
 mU1[i]:=U1;
 mx5A[i]:=x5A;
 my5A[i]:=y5A;
 mx50A[i]:=x50A;
 my50A[i]:=y50A;
 mv5Ax[i]:=v5Ax;
 mv5Ay[i]:=v5Ay;
 mv50Ax[i]:=v50Ax;
 mv50Ay[i]:=v50Ay;
```

```
        mw5Ax[i]:=w5Ax;
        mw5Ay[i]:=w5Ay;
        mw50Ax[i]:=w50Ax;
        mw50Ay[i]:=w50Ay;
         end;
            for n:=0 to 360 do
                begin
          StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
          StringGrid1.cells[1,n+1]:=format('%9.4f',[mx5A[n]]);
          StringGrid1.cells[2,n+1]:=format('%9.4f',[my5A[n]]);
          StringGrid1.cells[3,n+1]:=format('%9.4f',[mx50A[n]]);
          StringGrid1.cells[4,n+1]:=format('%9.4f',[my50A[n]]);
          StringGrid1.cells[5,n+1]:=format('%9.4f',[mv5Ax[n]]);
          StringGrid1.cells[6,n+1]:=format('%9.4f',[mv5Ay[n]]);
          StringGrid1.cells[7,n+1]:=format('%9.4f',[mv50Ax[n]]);
          StringGrid1.cells[8,n+1]:=format('%9.4f',[mv50Ay[n]]);
          StringGrid1.cells[9,n+1]:=format('%9.4f',[mw5Ax[n]]);
          StringGrid1.cells[10,n+1]:=format('%9.4f',[mw5Ay[n]]);
          StringGrid1.cells[11,n+1]:=format('%9.4f',[mw50Ax[n]]);
          StringGrid1.cells[12,n+1]:=format('%9.4f',[mw50Ay[n]]);
              end;
end;
procedure TFormKr1.Button2ksClick(Sender: TObject);
begin
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mx5A[j],'',clRed);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,my5A[j],'',clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mx50A[j],'',clYellow);
```

```
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,my50A[j],",clBlue);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,US1,",clWhite);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mv5Ax[j],",clRed);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mv5Ay[j],",clGreen);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mv50Ax[j],",clYellow);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mv50Ay[j],",clBlue);
Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mw5Ax[j],",clRed);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mw5Ay[j],",clGreen);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mw50Ax[j],",clYellow);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mw50Ay[j],",clBlue);
end;
end;
procedure TFormKr1.Button1Click(Sender: TObject);
begin
FormKr1.Hide;
FormKr2.Show;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
```

```
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
   CXlsBof[4] := 0;
   FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
   for i := 0 to AGrid.ColCount - 1 do
    for j := 0 to AGrid.RowCount - 1 do
     XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
   FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
   Result := True;
  finally
   FStream.Free;
  end;
 end;
```

```
procedure TFormKr1.Button2Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormKr2 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    StringGrid1: TStringGrid;
    Chart1: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series4: TLineSeries;
    Button4: TButton;
    Label2: TLabel;
    Edit2: TEdit;
```

```
    Label4: TLabel;
    Edit3: TEdit;
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FormKr2: TFormKr2;
  i,j,n:Integer;
   w550Ax,w550Ay, m1,XVS,YVS,FKx,FKy,XK0,YK0,RKr:Real;
  mXK0,mYK0,mRKr:array [0..361] of Real;
 implementation
 uses kr0,kr1;
{$R *.dfm}
procedure TFormKr2.Button3Click(Sender: TObject);
begin
formKr1.Show;
formKr2.Hide;
end;
procedure TFormKr2.Button2Click(Sender: TObject);
begin
 for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mXK0[j],'',clRed);
```

```
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,mYK0[j],'',clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mRKr[j],'',clYellow);


end;
procedure TFormKr2.Button1Click(Sender: TObject);
begin
  Val(Edit1.text,m1,code);
  Val(Edit2.text,XVS,code);
  Val(Edit3.text,YVS,code);
        begin
       StringGrid1.cells[0,0]:='U1';
       StringGrid1.cells[1,0]:='XK0';
       StringGrid1.cells[2,0]:='YK0';
       StringGrid1.cells[3,0]:='RKr';
            end;
       for i:=0 to 360 do
       begin
       U1:=i*Pi/180;
       w550Ax:=mw50Ax[i];
       w550Ay:=mw50Ay[i];
       FKx:=m1*w550Ax;
       FKy:=m1*w550Ay;
       XK0:=-FKx+XVS;
       YK0:=-FKy+YVS;
       RKr:=sqrt(sqr(XK0)+sqr(YK0));
       mU1[i]:=U1;
       mXK0[i]:=XK0;
       mYK0[i]:=YK0;
       mRKr[i]:=RKr;
```

```
        end;
            for n:=0 to 360 do
                begin
            StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
            StringGrid1.cells[1,n+1]:=format('%11.4f',[mXK0[n]]);
            StringGrid1.cells[2,n+1]:=format('%11.4f',[mYK0[n]]);
            StringGrid1.cells[3,n+1]:=format('%11.4f',[mRKr[n]]);
                end;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
```

```
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
   CXlsBof[4] := 0;
   FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
   for i := 0 to AGrid.ColCount - 1 do
    for j := 0 to AGrid.RowCount - 1 do
     XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
   FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
   Result := True;
  finally
   FStream.Free;
  end;
 end;
procedure TFormKr2.Button4Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
   {ShowMessage('StringGrid saved!')};
end;
```

```
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormku0 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    Image2: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Label3: TLabel;
    Edit2: TEdit;
    Label4: TLabel;
    Label7: TLabel;
    Edit3: TEdit;
    Label8: TLabel;
    Edit4: TEdit;
    Label9: TLabel;
    Edit5: TEdit;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Edit6: TEdit;
    Label10: TLabel;
    Label5: TLabel;
```

```
    Edit7: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Formku0: TFormku0;
  code:Integer;
  l5,ZPOKU,XSK,YSK,nzku,l4,lD3:Real;
implementation
uses ku2,ku1;
{$R *.dfm}
procedure TFormku0.Button1Click(Sender: TObject);
begin
   Val(Edit1.text,lD3,code);
   Val(Edit2.text,l5,code);
   Val(Edit3.text,nzku,code);
   Val(Edit4.text,XSK,code);
   Val(Edit5.text,YSK,code);
   Val(Edit6.text,ZPOKU,code);
   Val(Edit7.text,l4,code);
Formku0.Hide;
Formku1.Show;
end;
interface
uses
```

```
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormku1 = class(TForm)
    btnkrpo: TButton;
    Chart1: TChart;
    Button1ks: TButton;
    Button2ks: TButton;
    Chart2: TChart;
    Chart4: TChart;
    Series8: TLineSeries;
    Series7: TLineSeries;
    StringGrid1: TStringGrid;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series14: TLineSeries;
    Series15: TLineSeries;
    Button1: TButton;
    Button2: TButton;
    Chart3: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series5: TLineSeries;
    Series6: TLineSeries;
    Series9: TLineSeries;
    Series10: TLineSeries;
    Series11: TLineSeries;
```

```
    Series12: TLineSeries;
    Series13: TLineSeries;
    Series16: TLineSeries;
    Series17: TLineSeries;
    Series18: TLineSeries;
    Series19: TLineSeries;
    Series20: TLineSeries;
    Series21: TLineSeries;
    Button3: TButton;
    procedure btnkrpoClick(Sender: TObject);
    procedure Button1ksClick(Sender: TObject);
    procedure Button2ksClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Formku1: TFormku1;
  code,i,j,n:Integer;
  SA,SB,l33,l3,xC,yC,U3:Real;
  U1,US3,UU3:Real;
  xlSD3,ylSD3,vlSD3x,vlSD3y,xlD3,ylD3,vlD3x,vlD3y:Real;
  wlSD3x,wlSD3y,wlD3x,wlD3y:Real;
  XKCA,YKCA,VXKCA,VYKCA,WXKCA,WYKCA:Real;
  mU1,mU3,mxlD3,mylD3:array [0..361] of Real;
```

```
 mUS3,mUU3,mvlD3x,mvlD3y,mwlD3x,mwlD3y:array [0..361] of Real;
 mXKCA,mYKCA,mVXKCA,mVYKCA,mWXKCA,mWYKCA:array
[0..361] of Real;
 mxlSD3,mylSD3,mvlSD3x,mvlSD3y,mwlSD3x,mwlSD3y:array [0..361] of
Real;
implementation
 uses Unit2PM,ku0,ku2;
{$R *.dfm}
procedure TFormku1.btnkrpoClick(Sender: TObject);
begin
formku1.Hide;
PMform2.Show;
end;
procedure TFormku1.Button1ksClick(Sender: TObject);
begin
        begin
       StringGrid1.cells[0,0]:='U1';
       StringGrid1.cells[1,0]:='xlD3';
       StringGrid1.cells[2,0]:='ylD3';
       StringGrid1.cells[3,0]:='U3';
       StringGrid1.cells[4,0]:='US3';
       StringGrid1.cells[5,0]:='UU3';
       StringGrid1.cells[6,0]:='vlD3x';
       StringGrid1.cells[7,0]:='vlD3y';
       StringGrid1.cells[8,0]:='wlD3x';
       StringGrid1.cells[9,0]:='wlD3y';
       StringGrid1.cells[10,0]:='XKCA';
       StringGrid1.cells[11,0]:='YKCA';
       StringGrid1.cells[12,0]:='VXKCA';
```

```
StringGrid1.cells[13,0]:='VYKCA';
StringGrid1.cells[14,0]:='WXKCA';
StringGrid1.cells[15,0]:='WYKCA';
StringGrid1.cells[16,0]:='xlSD3';
StringGrid1.cells[17,0]:='ylSD3';
StringGrid1.cells[18,0]:='vlSD3x';
StringGrid1.cells[19,0]:='vlSD3y';
StringGrid1.cells[20,0]:='wlSD3x';
StringGrid1.cells[21,0]:='wlSD3y';
 end;
 for i:=0 to 360 do
 begin
 U1:=i*Pi/180;
 if ZPOKU<2 then
 begin
 xC:=l5;
 yC:=l4;
 U3:=arcTan((yC-myA[i])/(xC-mxA[i]));
 end
          else
 begin
  xC:=l5;
  yC:=-l4;
  U3:=2*Pi-arcTan(sqrt(sqr((yC-myA[i])/(xC-mxA[i]))));
 end;
 if nzku<2 then
 begin
 xlSD3:=xC-(lD3/2)*cos(U3);
 ylSD3:=yC-(lD3/2)*sin(U3);
```

```
xlD3:=xC-(lD3)*cos(U3);
ylD3:=yC-(lD3)*sin(U3);
l3:= (xC-mxA[i])/cos(U3);
US3:=(mvAx[i]-mvAy[i]*(cos(U3)/sin(U3)))/(l3*cos(U3)*
(cos(U3)/sin(U3))+l3*sin(U3));
l33:=(-mvAy[i]-l3*US3*cos(U3))/sin(U3);
UU3:=(-mwAy[i]+(sin(U3)/cos(U3))*mwAx[i]-
(sin(U3)/cos(U3))*l33*US3*sin(U3)-(sin(U3)/cos(U3))*l33*US3*sin(U3)-
(sin(U3)/cos(U3))*l3*sqr(US3)*cos(U3)-l33*US3*cos(U3)-
l33*US3*cos(U3)+l3*sqr(US3)*sin(U3))/((sin(U3)/cos(U3))*l3*sin(U3)+l3*co
s(U3));
vlD3x:=(lD3)*US3*sin(U3);
vlD3y:=-(lD3)*US3*cos(U3);
vlSD3x:=(lD3/2)*US3*sin(U3);
vlSD3y:=-(lD3/2)*US3*cos(U3);
wlD3x:=(lD3)*UU3*sin(U3)+(lD3)*sqr(US3)*cos(U3);
wlD3y:=-(lD3)*UU3*cos(U3)+(lD3)*sqr(US3)*sin(U3);
wlSD3x:=(lD3/2)*UU3*sin(U3)+(lD3/2)*sqr(US3)*cos(U3);
wlSD3y:=-(lD3/2)*UU3*cos(U3)+(lD3/2)*sqr(US3)*sin(U3);
SA:=-XSK*sin(U3)-YSK*cos(U3);
SB:=XSK*cos(U3)-YSK*sin(U3);
XKCA:=xlD3+SB;
YKCA:=ylD3-SA;
VXKCA:=vlD3x+US3*SA;
VYKCA:=vlD3y+US3*SB;
WXKCA:=wlD3x+UU3*SA-US3*US3*SB;
WYKCA:=wlD3y+UU3*SB+US3*US3*SA;
end
```

```
            else
        begin
        xlSD3:=mxA[i]+(lD3/2)*cos(U3);
        ylSD3:=myA[i]+(lD3/2)*sin(U3);
        xlD3:=mxA[i]+(lD3)*cos(U3);
        ylD3:=myA[i]+(lD3)*sin(U3);
        l3:= (xC-mxA[i])/cos(U3);
        US3:=(mvAx[i]-mvAy[i]*(cos(U3)/sin(U3)))/(l3*cos(U3)*
(cos(U3)/sin(U3))+l3*sin(U3));
        l33:=(-mvAy[i]-l3*US3*cos(U3))/sin(U3);
        UU3:=(-mwAy[i]+(sin(U3)/cos(U3))*mwAx[i]-
(sin(U3)/cos(U3))*l33*US3*sin(U3)-(sin(U3)/cos(U3))*l33*US3*sin(U3)-
(sin(U3)/cos(U3))*l3*sqr(US3)*cos(U3)-l33*US3*cos(U3)-
l33*US3*cos(U3)+l3*sqr(US3)*sin(U3))/((sin(U3)/cos(U3))*l3*sin(U3)+l3*co
s(U3));
        vlD3x:=(lD3)*US3*sin(U3);
        vlD3y:=-(lD3)*US3*cos(U3);
        vlSD3x:=(lD3/2)*US3*sin(U3);
        vlSD3y:=-(lD3/2)*US3*cos(U3);
        wlD3x:=(lD3)*UU3*sin(U3)+(lD3)*sqr(US3)*cos(U3);
        wlD3y:=-(lD3)*UU3*cos(U3)+(lD3)*sqr(US3)*sin(U3);
        wlSD3x:=(lD3/2)*UU3*sin(U3)+(lD3/2)*sqr(US3)*cos(U3);
        wlSD3y:=-(lD3/2)*UU3*cos(U3)+(lD3/2)*sqr(US3)*sin(U3);
        SA:=-XSK*sin(U3)-YSK*cos(U3);
        SB:=XSK*cos(U3)-YSK*sin(U3);
        XKCA:=xlD3+SB;
        YKCA:=ylD3-SA;
        VXKCA:=vlD3x+US3*SA;
        VYKCA:=vlD3y+US3*SB;
```

```
WXKCA:=wlD3x+UU3*SA-US3*US3*SB;
WYKCA:=wlD3y+UU3*SB+US3*US3*SA;
end;
mU1[i]:=U1;
mU3[i]:=U3;
mxlD3[i]:=xlD3;
mylD3[i]:=ylD3;
mUS3[i]:=US3;
mUU3[i]:=UU3;
mvlD3x[i]:=vlD3x;
mvlD3y[i]:=vlD3y;
mwlD3x[i]:=wlD3x;
mwlD3y[i]:=wlD3y;
mXKCA[i]:=XKCA;
mYKCA[i]:=YKCA;
mVXKCA[i]:=VXKCA;
mVYKCA[i]:=VYKCA;
mWXKCA[i]:=WXKCA;
mWYKCA[i]:=WYKCA;
mxlSD3[i]:=xlSD3;
mylSD3[i]:=ylSD3;
mvlSD3x[i]:=vlSD3x;
mvlSD3y[i]:=vlSD3y;
mwlSD3x[i]:=wlSD3x;
mwlSD3y[i]:=wlSD3y;
end;
    for n:=0 to 360 do
        begin
  StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
```

```
        StringGrid1.cells[1,n+1]:=format('%9.4f',[mxlD3[n]]);
        StringGrid1.cells[2,n+1]:=format('%9.4f',[mylD3[n]]);
        StringGrid1.cells[3,n+1]:=format('%9.4f',[mU3[n]]);
        StringGrid1.cells[4,n+1]:=format('%9.4f',[mUS3[n]]);
        StringGrid1.cells[5,n+1]:= format('%9.4f',[mUU3[n]]);
        StringGrid1.cells[6,n+1]:=format('%9.4f',[mvlD3x[n]]);
        StringGrid1.cells[7,n+1]:=format('%9.4f',[mvlD3y[n]]);
        StringGrid1.cells[8,n+1]:=format('%9.4f',[mwlD3x[n]]);
        StringGrid1.cells[9,n+1]:=format('%9.4f',[mwlD3y[n]]);
        StringGrid1.cells[10,n+1]:=format('%9.4f',[mXKCA[n]]);
        StringGrid1.cells[11,n+1]:= format('%9.4f',[mYKCA[n]]);
        StringGrid1.cells[12,n+1]:=format('%9.4f',[mVXKCA[n]]);
        StringGrid1.cells[13,n+1]:=format('%9.4f',[mVYKCA[n]]);
        StringGrid1.cells[14,n+1]:=format('%9.4f',[mWXKCA[n]]);
        StringGrid1.cells[15,n+1]:=format('%9.4f',[mWYKCA[n]]);
        StringGrid1.cells[16,n+1]:=format('%9.4f',[mxlSD3[n]]);
        StringGrid1.cells[17,n+1]:= format('%9.4f',[mylSD3[n]]);
        StringGrid1.cells[18,n+1]:=format('%9.4f',[mvlSD3x[n]]);
        StringGrid1.cells[19,n+1]:=format('%9.4f',[mvlSD3y[n]]);
        StringGrid1.cells[20,n+1]:=format('%9.4f',[mwlSD3x[n]]);
        StringGrid1.cells[21,n+1]:=format('%9.4f',[mwlSD3y[n]]);
                        end;
end;
procedure TFormku1.Button2ksClick(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
```

```
Chart1.SeriesList[4].Clear;
Chart1.SeriesList[5].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
Chart3.SeriesList[4].Clear;
Chart3.SeriesList[5].Clear;
Chart4.SeriesList[0].Clear;
Chart4.SeriesList[1].Clear;
Chart4.SeriesList[2].Clear;
Chart4.SeriesList[3].Clear;
Chart4.SeriesList[4].Clear;
Chart4.SeriesList[5].Clear;
Chart4.SeriesList[6].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mxlD3[j],",clYellow);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,mylD3[j],",clBlue);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mXKCA[j],",clRed);
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,mYKCA[j],",clGreen);
Chart1.SeriesList[4].AddXY(mU1[j]*180/Pi,mxlSD3[j],",clWhite);
Chart1.SeriesList[5].AddXY(mU1[j]*180/Pi,mylSD3[j],",clYellow);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mUS3[j],",clBlue);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mU3[j],",clRed);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mvlD3x[j],",clRed);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mvlD3y[j],",clGreen);
```

```
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mVXKCA[j],'',clYellow);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mVYKCA[j],'',clBlue);
Chart3.SeriesList[4].AddXY(mU1[j]*180/Pi,mvlSD3x[j],'',clYellow);
Chart3.SeriesList[5].AddXY(mU1[j]*180/Pi,mvlSD3y[j],'',clBlue);

Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mUU3[j],'',clRed);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mwlD3x[j],'',clYellow);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mwlD3y[j],'',clBlue);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mWXKCA[j],'',clGreen);
Chart4.SeriesList[4].AddXY(mU1[j]*180/Pi,mWYKCA[j],'',clWhite);
Chart4.SeriesList[5].AddXY(mU1[j]*180/Pi,mwlSD3x[j],'',clRed);
Chart4.SeriesList[6].AddXY(mU1[j]*180/Pi,mwlSD3y[j],'',clYellow);

end;
procedure TFormku1.Button1Click(Sender: TObject);
begin
Formku1.Hide;
Formku2.Show;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
```

```
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
```

```
 end;
procedure TFormku1.Button2Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
procedure TFormku1.Button3Click(Sender: TObject);
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormku2 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    StringGrid1: TStringGrid;
    Chart1: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series3: TLineSeries;
    Series4: TLineSeries;
```

```
    Series5: TLineSeries;
    Button4: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Series6: TLineSeries;
    Series7: TLineSeries;
    Series8: TLineSeries;
    Button5: TButton;
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Formku2: TFormku2;
  i,j,n:Integer;
  NC,MVS,XVS,YVS,m3,I3,FI3x,FI3y,NA,XA1,YA1,XC1,YC1,RA,RC:Real;
  mNC,mXC1,mYC1,mNA,mXA1,mYA1,mRA,mRC:array [0..361] of Real;
implementation
 uses Unit2PM,ku1,ku0;
```

```
{$R *.dfm}
procedure TFormku2.Button3Click(Sender: TObject);
begin
Formku2.Hide;
Formku1.Show;
end;
procedure TFormku2.Button2Click(Sender: TObject);
begin
 Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart1.SeriesList[4].Clear;
Chart1.SeriesList[5].Clear;
Chart1.SeriesList[6].Clear;
Chart1.SeriesList[7].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mNA[j],",clRed);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,mXA1[j],",clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mYA1[j],",clYellow);
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,mXC1[j],",clWhite);
Chart1.SeriesList[4].AddXY(mU1[j]*180/Pi,mYC1[j],",clYellow);
Chart1.SeriesList[5].AddXY(mU1[j]*180/Pi,mRA[j],",clBlack);
Chart1.SeriesList[6].AddXY(mU1[j]*180/Pi,mRC[j],",clBlue);
Chart1.SeriesList[7].AddXY(mU1[j]*180/Pi,mNC[j],",clRed);
end;
end;
```

```
procedure TFormku2.Button1Click(Sender: TObject);
begin
   Val(Edit1.text,m3,code);
   Val(Edit2.text,MVS,code);
   Val(Edit3.text,XVS,code);
   Val(Edit4.text,YVS,code);
        begin
       StringGrid1.cells[0,0]:='U1';
       StringGrid1.cells[1,0]:='NA';
       StringGrid1.cells[2,0]:='NC';
       StringGrid1.cells[3,0]:='XA1';
       StringGrid1.cells[4,0]:='YA1';
       StringGrid1.cells[5,0]:='XC1';
       StringGrid1.cells[6,0]:='YC1';
       StringGrid1.cells[7,0]:='RA';
       StringGrid1.cells[8,0]:='RC';
        end;
        for i:=0 to 360 do
    begin
       I3:=m3*lD3/12;
       wlD3x:=mwlSD3x[i]; wlD3y:=mwlSD3y[i];
       FI3x:=m3*wlSD3x;
       FI3y:=m3*wlSD3y;
            if nzku<2 then
                    begin
       NA:=(cos(Pi/2-mU3[i])/(yC-myA[i]))*(-I3*mUU3[i]-
MVS+FI3y*(lD3/2)*sin((Pi/2)-mU3[i])-
FI3x*(lD3/2)*cos((Pi/2)+YVS*(lD3)*sin((Pi/2)-mU3[i])-
XVS*(lD3)*cos((Pi/2)-mU3[i])));
```

```
        XC1:=-NA*cos((Pi/2)-mU3[i])-FI3x-XVS;
        YC1:=-NA*sin((Pi/2)-mU3[i])-FI3y-YVS;
        XA1:=NA*cos((Pi/2)-mU3[i]);
        YA1:=NA*sin((Pi/2)-mU3[i]);
        RA:=sqrt(sqr(XA1)+sqr(YA1));
        RC:=sqrt(sqr(XC1)+sqr(YC1));
        NC:=0;
                        end
              else
        begin
        NC:= (sin(Pi/2-mU3[i])/(xC-
mxA[i]))*(I3*mUU3[i]+MVS+FI3y*(lD3/2)*sin((Pi/2)-mU3[i])-
FI3x*(lD3/2)*cos((Pi/2)-mU3[i])+YVS*(lD3)*sin((Pi/2)-mU3[i])-
XVS*(lD3)*cos((Pi/2)-mU3[i]));
        XA1:=-NC*cos(Pi/2-mU3[i])-FI3x-XVS;
        YA1:=NC*sin(Pi/2-mU3[i])-FI3y-YVS;
        XC1:=NC*cos(Pi/2-mU3[i]);
        YC1:=NC*sin(Pi/2-mU3[i]);
        RA:=sqrt(sqr(XA1)+sqr(YA1));
        RC:=sqrt(sqr(XC1)+sqr(YC1));
        NA:=0;
        end;

mNA[i]:=NA;
        mNC[i]:=NC;
        mXC1[i]:=XC1;
        mYC1[i]:=YC1;
        mXA1[i]:=XA1;
        mYA1[i]:=YA1;
```

```
        mRA[i]:=RA;
        mRC[i]:=RC;
    end;
            for n:=0 to 360 do
                begin
            StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
            StringGrid1.cells[1,n+1]:=format('%11.4f',[mNA[n]]);
            StringGrid1.cells[2,n+1]:=format('%11.4f',[mNC[n]]);
            StringGrid1.cells[3,n+1]:=format('%11.4f',[mXA1[n]]);
            StringGrid1.cells[4,n+1]:=format('%11.4f',[mYA1[n]]);
            StringGrid1.cells[5,n+1]:=format('%11.4f',[mXC1[n]]);
            StringGrid1.cells[6,n+1]:=format('%11.4f',[mYC1[n]]);
            StringGrid1.cells[7,n+1]:=format('%11.4f',[mRA[n]]);
            StringGrid1.cells[8,n+1]:=format('%11.4f',[mRC[n]]);
                end;
end;
 procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
var
  L: Word;
const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
```

```
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
   CXlsBof[4] := 0;
   FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
   for i := 0 to AGrid.ColCount - 1 do
    for j := 0 to AGrid.RowCount - 1 do
     XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
   FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
   Result := True;
  finally
   FStream.Free;
  end;
 end;
procedure TFormku2.Button4Click(Sender: TObject);
begin
```

```
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
procedure TFormku2.Button5Click(Sender: TObject);
interface

uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormPo0 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit5: TEdit;
    Label5: TLabel;
    Edit6: TEdit;
    Label6: TLabel;
```

```
    Edit7: TEdit;
    Label7: TLabel;
    Image1: TImage;
    Label8: TLabel;
    Label12: TLabel;
    Image2: TImage;
    Image3: TImage;
    Label9: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Edit8: TEdit;
    Label15: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label16: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FormPo0: TFormPo0;
  code:Integer;
   nzpo,l2,l3,l5,ZPO,BG,XSAB,YSAB:Real;
implementation
  uses po1,po2;
{$R *.dfm}
```

```
procedure TFormPo0.Button1Click(Sender: TObject);
begin
  Val(Edit1.text,l2,code);
  Val(Edit2.text,l3,code);
  Val(Edit3.text,XSAB,code);
  Val(Edit4.text,YSAB,code);
  Val(Edit5.text,l5,code);
  Val(Edit6.text,ZPO,code);
  Val(Edit7.text,BG,code);
  Val(Edit8.text,nzpo,code);
formpo0.Hide;
formpo1.Show;
end;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
   type
  TFormPo1 = class(TForm)
    btnkrpo: TButton;
    Chart1: TChart;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    Button1ks: TButton;
    Button2ks: TButton;
    Series6: TLineSeries;
```

Chart2: TChart;
Chart4: TChart;
Series8: TLineSeries;
Series7: TLineSeries;
Series9: TLineSeries;
Series14: TLineSeries;
Series15: TLineSeries;
Series16: TLineSeries;
Series17: TLineSeries;
Series18: TLineSeries;
Series19: TLineSeries;
StringGrid1: TStringGrid;
Button1: TButton;
Button2: TButton;
Chart3: TChart;
Series22: TLineSeries;
Series23: TLineSeries;
Series24: TLineSeries;
Series25: TLineSeries;
Series26: TLineSeries;
Series27: TLineSeries;
Chart5: TChart;
Series1: TPointSeries;
Series2: TPointSeries;
Series10: TLineSeries;
Series11: TLineSeries;
Button3: TButton;
procedure btnkrpoClick(Sender: TObject);
procedure Button1ksClick(Sender: TObject);

```
    procedure Button2ksClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FormPo1: TFormPo1;
  code,i,j,n:Integer;
  U1,U2,xB,yB,xAB,yAB,SA,SB:Real;
  US2,UU2,vv1,vv2,vv3:Real;
  vBx,vBy,wBx,wBy:Real;
  XKAB,YKAB,VXKAB,VYKAB,WXKAB,WYKAB:Real;
  vABx,vABy,wABx,wABy:Real;
  mU1,mU2,mxB,myB,mxAB,myAB:array [0..361] of Real;
mUS2,mUU2,mvBx,mvBy,mwBx,mwBy,mvABx,mvABy,mwABx,mwABy:arr
ay [0..361] of Real;
  mXKAB,mYKAB,mVXKAB,mVYKAB,mWXKAB,mWYKAB:array
[0..361] of Real;
implementation
 uses Unit2PM,po2,po0;
{$R *.dfm}
procedure TFormPo1.btnkrpoClick(Sender: TObject);
begin
formpo1.Hide;
PMform2.Show;
```

```
end;
procedure TFormPo1.Button1ksClick(Sender: TObject);
begin
        begin
      StringGrid1.cells[0,0]:='U1';
      StringGrid1.cells[1,0]:='U2';
      StringGrid1.cells[2,0]:='xB';
      StringGrid1.cells[3,0]:='yB';
      StringGrid1.cells[4,0]:='xAB';
      StringGrid1.cells[5,0]:='yAB';
      StringGrid1.cells[6,0]:='US2';
      StringGrid1.cells[7,0]:='UU2';
      StringGrid1.cells[8,0]:='vBx';
      StringGrid1.cells[9,0]:='vBy';
      StringGrid1.cells[10,0]:='wBx';
      StringGrid1.cells[11,0]:='wBy';
      StringGrid1.cells[12,0]:='vABx';
      StringGrid1.cells[13,0]:='vABy';
      StringGrid1.cells[14,0]:='wABx';
      StringGrid1.cells[15,0]:='wABy';
      StringGrid1.cells[16,0]:='xKAB';
      StringGrid1.cells[17,0]:='yKAB';
      StringGrid1.cells[18,0]:='vXKAB';
      StringGrid1.cells[19,0]:='vYKAB';
      StringGrid1.cells[20,0]:='wXKAB';
      StringGrid1.cells[21,0]:='wYKAB';
        end;
      for i:=0 to 360 do
```

```
begin
U1:=i*Pi/180;
if nzpo<2 then
begin
if ZPO<2 then
begin
yB:=l3;
end
        else
begin
yB:=-l3;
end;
U2:=arcSin(-(yB+myA[i])/l2);
xB:=mxA[i]+l2*cos(U2);
US2:=-mvAy[i]/(l2*cos(U2));
UU2:=(-mwAy[i]+l2*sqr(US2)*sin(U2))/(l2*cos(U2));
vBx:=mvAx[i]-l2*US2*sin(U2);
vBy:=0;
wBx:=-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2)+mwAx[i];
wBy:=0;
xAB:=mxA[i]+(l2/2)*cos(U2);
yAB:=myA[i]+(l2/2)*sin(U2);
SA:=-XSAB*sin(U2)-YSAB*cos(U2);
SB:=XSAB*cos(U2)-YSAB*sin(U2);
XKAB:=mxA[i]+SB;
YKAB:=myA[i]-SA;
vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
VXKAB:=mvAx[i]+US2*SA;
```

```
VYKAB:=mvAy[i]+US2*SB;
wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
WXKAB:=mwAx[i]+UU2*SA-US2*US2*SB;
WYKAB:=mwAy[i]+UU2*SB+US2*US2*SA;
end
else
begin
if nzpo<3 then
begin
if ZPO<2 then
begin
U2:=arcCos((l5-mxA[i])/l2);
end
        else
begin
U2:=2*Pi-arcCos((l5-mxA[i])/l2);
end;
yB:=myA[i]+l2*sin(U2);
xB:=l5;
US2:=mvAx[i]/(l2*sin(U2));
UU2:=(mwAx[i]-l2*sqr(US2)*cos(U2))/(l2*sin(U2));
vBx:=0;
vBy:=mvAy[i]+l2*US2*cos(U2);
wBx:=0;
wBy:=mwAy[i]+l2*UU2*cos(U2)-l2*sqr(US2)*sin(U2);
xAB:=mxA[i]+(l2/2)*cos(U2);
yAB:=myA[i]+(l2/2)*sin(U2);
SA:=-XSAB*sin(U2)-YSAB*cos(U2);
```

```
SB:=XSAB*cos(U2)-YSAB*sin(U2);
XKAB:=mxA[i]+SB;
YKAB:=myA[i]-SA;
vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
VXKAB:=mvAx[i]+US2*SA;
VYKAB:=mvAy[i]+US2*SB;
wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
WXKAB:=mwAx[i]+UU2*SA-US2*US2*SB;
WYKAB:=mwAy[i]+UU2*SB+US2*US2*SA;
end
else
begin
vv1:=-Tan(BG)*l2;
vv2:=l2;
vv3:=-l5*Tan(BG)-myA[i]+Tan(BG)*mxA[i];
if ZPO<2 then
begin
U2:=arcSin(vv3/sqrt(sqr(vv1)+sqr(vv2)))-
arcSin(vv1/sqrt(sqr(vv1)+sqr(vv2)));
end
        else
begin
U2:=Pi+2*BG-
arcSin(vv3/sqrt(sqr(vv1)+sqr(vv2)))+arcSin(vv1/sqrt(sqr(vv1)+sqr(vv2)));
end;
xB:=mxA[i]+l2*cos(U2);
yB:=myA[i]+l2*sin(U2);
```

```
        US2:=(mvAx[i]*Tan(BG)-mvAy[i])/(l2*sin(U2)*Tan(BG)-l2*cos(U2));
        UU2:=(l2*sqr(US2)*sin(U2)-mwAy[i]+mwAx[i]*Tan(BG)-
l2*sqr(US2)*cos(U2)*Tan(BG))/(l2*sin(U2)*Tan(BG)+l2*cos(U2));
        vBx:=mvAx[i]-l2*US2*sin(U2);
        vBy:=mvAy[i]+l2*US2*cos(U2);
        wBx:=mwAx[i]-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2);
        wBy:=mwAy[i]+l2*UU2*cos(U2)-l2*sqr(US2)*sin(U2);
        xAB:=mxA[i]+(l2/2)*cos(U2);
        yAB:=myA[i]+(l2/2)*sin(U2);
        SA:=-XSAB*sin(U2)-YSAB*cos(U2);
        SB:=XSAB*cos(U2)-YSAB*sin(U2);
        XKAB:=mxA[i]+SB;
        YKAB:=myA[i]-SA;
        vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
        vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
        VXKAB:=mvAx[i]+US2*SA;
        VYKAB:=mvAy[i]+US2*SB;
        wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
        wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
        WXKAB:=mwAx[i]+UU2*SA-US2*US2*SB;
        WYKAB:=mwAy[i]+UU2*SB+US2*US2*SA;
        end;
        end;
        mU1[i]:=U1;
        mU2[i]:=U2;
        mxB[i]:=xB;
        myB[i]:=yB;
        mxAB[i]:=xAB;
        myAB[i]:=yAB;
```

```
mUS2[i]:=US2;
mUU2[i]:=UU2;
mvBx[i]:=vBx;
mvBy[i]:=vBy;
mwBx[i]:=wBx;
mwBy[i]:=wBy;
mvABx[i]:=vABx;
mvABy[i]:=vABy;
mwABx[i]:=wABx;
mwABy[i]:=wABy;
mXKAB[i]:=XKAB;
mYKAB[i]:=YKAB;
mVXKAB[i]:=VXKAB;
mVYKAB[i]:=VYKAB;
mWXKAB[i]:=WXKAB;
mWYKAB[i]:=WYKAB;
end;
    for n:=0 to 360 do
        begin
  StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
  StringGrid1.cells[1,n+1]:=format('%9.4f',[mU2[n]]);
  StringGrid1.cells[2,n+1]:=format('%9.4f',[mxB[n]]);
  StringGrid1.cells[3,n+1]:=format('%9.4f',[myB[n]]);
  StringGrid1.cells[4,n+1]:= format('%9.4f',[mxAB[n]]);
  StringGrid1.cells[5,n+1]:= format('%9.4f',[myAB[n]]);
  StringGrid1.cells[6,n+1]:=format('%9.4f',[mUS2[n]]);
  StringGrid1.cells[7,n+1]:= format('%9.4f',[mUU2[n]]);
  StringGrid1.cells[8,n+1]:=format('%9.4f',[mvBx[n]]);
  StringGrid1.cells[9,n+1]:=format('%9.4f',[mvBy[n]]);
```

```
        StringGrid1.cells[10,n+1]:=format('%9.4f',[mwBx[n]]);
        StringGrid1.cells[11,n+1]:=format('%9.4f',[mwBy[n]]);
        StringGrid1.cells[12,n+1]:=format('%9.4f',[mvABx[n]]);
        StringGrid1.cells[13,n+1]:=format('%9.4f',[mvABy[n]]);
        StringGrid1.cells[14,n+1]:=format('%9.4f',[mwABx[n]]);
        StringGrid1.cells[15,n+1]:=format('%9.4f',[mwABy[n]]);
        StringGrid1.cells[16,n+1]:=format('%9.4f',[mXKAB[n]]);
        StringGrid1.cells[17,n+1]:=format('%9.4f',[mYKAB[n]]);
        StringGrid1.cells[18,n+1]:=format('%9.4f',[mVXKAB[n]]);
        StringGrid1.cells[19,n+1]:=format('%9.4f',[mVYKAB[n]]);
        StringGrid1.cells[20,n+1]:=format('%9.4f',[mWXKAB[n]]);
        StringGrid1.cells[21,n+1]:=format('%9.4f',[mWYKAB[n]]);
            end;
end;
procedure TFormPo1.Button2ksClick(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart1.SeriesList[4].Clear;
Chart1.SeriesList[5].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
```

```
Chart4.SeriesList[0].Clear;
Chart4.SeriesList[1].Clear;
Chart4.SeriesList[2].Clear;
Chart4.SeriesList[3].Clear;
Chart4.SeriesList[4].Clear;
Chart4.SeriesList[5].Clear;
Chart4.SeriesList[6].Clear;
Chart4.SeriesList[7].Clear;
Chart5.SeriesList[0].Clear;
Chart5.SeriesList[1].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mxB[j],",clRed);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,myB[j],",clBlue);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mxAB[j],",clYellow);
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,myAB[j],",clGreen);
Chart1.SeriesList[4].AddXY(mU1[j]*180/Pi,mxKAB[j],",clWhite);
Chart1.SeriesList[5].AddXY(mU1[j]*180/Pi,myKAB[j],",clBlack);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mUS2[j],",clBlue);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mU2[j],",clWhite);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mUU2[j],",clRed);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mVXKAB[j],",clYellow);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mVYKAB[j],",clBlue);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mwBx[j],",clRed);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mwBy[j],",clGreen);
Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mvABx[j],",clRed);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mvABy[j],",clGreen);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mwABx[j],",clYellow);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mwABy[j],",clBlue);
```

```
Chart4.SeriesList[4].AddXY(mU1[j]*180/Pi,mvBx[j],'',clWhite);
Chart4.SeriesList[5].AddXY(mU1[j]*180/Pi,mvBy[j],'',clBlack);
Chart4.SeriesList[6].AddXY(mU1[j]*180/Pi,mwXKAB[j],'',clGreen);
Chart4.SeriesList[7].AddXY(mU1[j]*180/Pi,mwYKAB[j],'',clYellow);

hart5.SeriesList[0].AddXY(mxAB[j],myAB[j],'',clRed);
Chart5.SeriesList[1].AddXY(mxKAB[j],myKAB[j],'',clGreen);
end;
end;
procedure TFormPo1.Button1Click(Sender: TObject);
begin
Formpo1.Hide;
Formpo2.Show;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
```

```
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
var
  FStream: TFileStream;
  I, J: Integer;
begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
end;
procedure TFormPo1.Button2Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
```

```
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
procedure TFormPo1.Button3Click(Sender: TObject);
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TFormPo2 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    StringGrid1: TStringGrid;
    Chart1: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    Button4: TButton;
    Label2: TLabel;
    Edit2: TEdit;
```

```
    Edit3: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Edit4: TEdit;
    Button5: TButton;
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FormPo2: TFormPo2;
  i,j,n:Integer;
  XVS,YVS,MVS,m2,I2,FI2x,FI2y,YB1,XA1,YA1,RA,RB:Real;
   mXA1,mYA1,mYB1,mRA,mRB:array [0..361] of Real;
implementation
 uses Unit2PM,po1,po0;
{$R *.dfm}
procedure TFormPo2.Button3Click(Sender: TObject);
begin
formpo1.Show;
formpo2.Hide;
end;
```

```
procedure TFormPo2.Button2Click(Sender: TObject);
begin
 Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart1.SeriesList[4].Clear;
 for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mXA1[j],'',clRed);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,mYA1[j],'',clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mYB1[j],'',clYellow);
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,mRA[j],'',clBlue);
Chart1.SeriesList[4].AddXY(mU1[j]*180/Pi,mRB[j],'',clWhite);
end;
end;
procedure TFormPo2.Button1Click(Sender: TObject);
begin
  Val(Edit1.text,m2,code);
  Val(Edit2.text,MVS,code);
  Val(Edit3.text,XVS,code);
  Val(Edit4.text,YVS,code);
        begin
      StringGrid1.cells[0,0]:='U1';
      StringGrid1.cells[1,0]:='XA';
      StringGrid1.cells[2,0]:='YA';
      StringGrid1.cells[3,0]:='YB';
      StringGrid1.cells[4,0]:='RA';
      StringGrid1.cells[5,0]:='RB';
```

```
            end;
        for i:=0 to 360 do
        begin
        I2:=m2*l2/12;
        wABx:=mwABx[i]; wABy:=mwABy[i];
        FI2x:=m2*wABx;
        FI2y:=m2*wABy;
YB1:=(MVS+FI2x*sin(mU2[i])*(l2/2)+I2*mUU2[i]+XVS*l2*sin(mU2[i])+YV
S*l2*cos(mU2[i])+FI2y*cos(mU2[i])*(l2/2))/(-l2*cos(mU2[i]));
        XA1:=-FI2x+XVS;
        YA1:=-FI2y-YB1+YVS;
        RA:=sqrt(sqr(XA1)+sqr(YA1));
        RB:=sqrt(sqr(YB1));
        mXA1[i]:=XA1;
        mYA1[i]:=YA1;
        mYB1[i]:=YB1;
        mRA[i]:=RA;
        mRB[i]:=RB;
        end;
            for n:=0 to 360 do
                begin
          StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
          StringGrid1.cells[1,n+1]:=format('%11.4f',[mXA1[n]]);
          StringGrid1.cells[2,n+1]:=format('%11.4f',[mYA1[n]]);
          StringGrid1.cells[3,n+1]:=format('%11.4f',[mYB1[n]]);
          StringGrid1.cells[4,n+1]:=format('%11.4f',[mRA[n]]);
          StringGrid1.cells[5,n+1]:=format('%11.4f',[mRB[n]]);
                end;
```

```
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
```

```
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
 end;
procedure TFormPo2.Button4Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
procedure TFormPo2.Button5Click(Sender: TObject);
```

ЛІСТИНГ ПРОГРАМИ ДЛЯ КІНЕМАТИЧНОГО АНАЛІЗУ І СИНТЕЗУ МЕХАНІЗМІВ І МАШИН ЛЕГКОЇ ПРОМИСЛОВОСТІ

```
program newton (input,output);
{$i graph.p}
{$i intg }
{$i rea }
label 10,20,30,40;
type t506=array [1..30,1..6] of real;t503=array [1..30,1..3] of
real;t504=array [1..30,1..4] of real;t502=array [1..30,1..2] of real;
t50=array [1..30] of real;t8=array [1..8] of real;
t6=array [1..6] of real;t4=array [1..4] of real;t3=array [1..3] of real;
t2=array [1..2] of real;ti50=array [1..30] of integer;t=integer;
var mu:array [1..8] of integer;
infileK,oufileK:text;
fvn:array [1..36] of real;
dg:array [1..200] of byte;
mexanzname:string[60];
infilenamK:string[14];
oufilenamK:string[14];
q:t506;p:t503;g1:t50;sab,scb:t504;r2:t8;s1,s2:t4;r:t502;
a,b,c,d,e,r1:t6;a1,p1,p2,p4:t3;p3,p5,p6:t2;jg,ja,jb,jc,jd,je,prinK:ti50;
i,j,i1,i2:t;
as,Hag,ugpre,u,z,sg,x0,y0,x1,y1,x2,y2,mgr,pxa,pya,pxc,pyc,abm,cbm,px,py,
smr,uvn,pmax,pmin,sfvn,pxb,pyb,
cs:real;
error,kinem,filed,n,nd,nn,menuK,menuDel,element,Nprin,Khag,ACPU,
menuKin,m,Rerror,l,page,outwrite,MaxN,mult,x,y,ii,jj,cont,z0,color,
kursor,sp,am,bm,xm,Rdin,fileV,l1,menu5,menu3,grvn,k,cila,cia,cv,
stoik:integer;
```

```
    OK,eoj:boolean;
   ch:char;
overlay procedure gr2;
    begin
textbackground(6);window(19,17,65,24);clrscr;
repeat textbackground(7);textcolor(0);window(17,16,63,23);clrscr;writeln;
  write('     1. Вектор ЦТ звена  ',ja[nn]:2,'-',jb[nn]:2,'   : ');
  sab[nn,1]:=rea(sab[nn,1]);writeln;
  write('     2. Угол ЦТ звена    ',ja[nn]:2,'-',jb[nn]:2,'   : ');
  sab[nn,2]:=rea(sab[nn,2]);writeln;
  write('     3. Момент инерции   ',ja[nn]:2,'-',jb[nn]:2,'   : ');
  sab[nn,3]:=rea(sab[nn,3]);writeln;
  write('     4. Масса звена      ',ja[nn]:2,'-',jb[nn]:2,'   : ');
  sab[nn,4]:=rea(sab[nn,4]);writeln;
  clrscr;writeln;
  write('     1. Вектор ЦТ звена  ',jc[nn]:2,'-',jb[nn]:2,'   : ');
  scb[nn,1]:=rea(scb[nn,1]);writeln;
  write('     2. Угол ЦТ звена    ',jc[nn]:2,'-',jb[nn]:2,'   : ');
  scb[nn,2]:=rea(scb[nn,2]);writeln;
  write('     3. Момент инерции   ',jc[nn]:2,'-',jb[nn]:2,'   : ');
  scb[nn,3]:=rea(scb[nn,3]);writeln;
  write('     4. Масса звена      ',jc[nn]:2,'-',jb[nn]:2,'   : ');
  scb[nn,4]:=rea(scb[nn,4]);writeln;
textbackground(4);textcolor(7);window(18,19,40,20);clrscr;
write(' Обнаружена ');textcolor(0+16);writeln('');textcolor(7);
write('    (Да-1/Нет-0) : ');error:=intg(error);if error=1 then begin
textbackground(1);window(20,20,48,21);clrscr;
textbackground(4);window(18,19,46,20);clrscr;
writeln('');
```

```
write(' - <ENTER>');read;end;
until error=0;
end;
overlay procedure gr3;
    begin
    textbackground(6);window(13,19,71,24);clrscr;
repeat textbackground(7);textcolor(0);window(11,18,69,23);clrscr;writeln;
 write('     1. Вектор ЦТ з ',ja[nn]:2,' -',jb[nn]:2,'   : ');
 sab[nn,1]:=rea(sab[nn,1]);writeln;
 write('     ',ja[nn]:2,' -',jb[nn]:2,'   : ');
 sab[nn,2]:=rea(sab[nn,2]);writeln;
 write(',ja[nn]:2,' -',jb[nn]:2,'   : ');
 sab[nn,3]:=rea(sab[nn,3]);writeln;
 write(',ja[nn]:2,' -',jb[nn]:2,'   : ');
 sab[nn,4]:=rea(sab[nn,4]);
 clrscr;writeln;
 write(',jc[nn]:2,' -',jb[nn]:2,'   : ');
 scb[nn,1]:=rea(scb[nn,1]);writeln;
 write(',jc[nn]:2,' -',jb[nn]:2,'   : ');
 scb[nn,2]:=rea(scb[nn,2]);writeln;
 write(',jd[nn]:2,' -',je[nn]:2,'   : ');
 scb[nn,3]:=rea(scb[nn,3]);writeln;
 write(',jc[nn]:2,' и',jb[nn]:2,'   : ');
 scb[nn,4]:=rea(scb[nn,4]);
 textbackground(4);textcolor(7);window(13,20,35,21);clrscr;
write(' Обнаружена ');textcolor(0+16);writeln(');textcolor(7);
write('    (Да-1/Нет-0) : ');error:=intg(error);if error=1 then begin
textbackground(1);window(15,21,43,22);clrscr;
textbackground(4);window(13,20,41,21);clrscr;
```

```
writeln('!');
write(' - <ENTER>');read;end;
until error=0;
    end;
    overlay procedure d1;
    begin
 gotoxy(24,3);
 writeln();
 write(' |              : ');textbackground(7);
 textcolor(0);writeln(' ',nd:2,' ');textbackground(1);textcolor(7);writeln;
 if element=1 then begin
 write(' |    ',ja[nd]:2,'-',jb[nd]:2);
 write('               : ');
 writeln(p[nd,1]:7:2);
 write(' |    (в град.) : ');
 writeln(p[nd,2]:7:2);
 write(' (c-1)                : ');
 writeln(p[nd,3]:7:2);textbackground(7);textcolor(0);
 gotoxy(52,6);p[nd,1]:=rea(p[nd,1]);
 gotoxy(52,7);p[nd,2]:=rea(p[nd,2]);
 gotoxy(52,8);p[nd,3]:=rea(p[nd,3]);end else begin
 gotoxy(15,6);write('1. ',ja[nd]:2,'      : ');
 writeln(q[ja[nd],1]:7:2);
 gotoxy(15,7);write('2. ',ja[nd]:2,'      : ');
 writeln(q[ja[nd],2]:7:2);textbackground(7);textcolor(0);
 gotoxy(49,6);q[ja[nd],1]:=rea(q[ja[nd],1]);
 gotoxy(49,7);q[ja[nd],2]:=rea(q[ja[nd],2]);
 end;end;
 overlay procedure d2;
```

```
begin
gotoxy(24,3);
writeln();
write(');textbackground(7);
textcolor(0);writeln(' ',nd:2,' ');textbackground(1);textcolor(7);writeln;
if element=1 then begin
write(' |          ',ja[nd]:2,'-',jb[nd]:2);
write('          : ');
writeln(p[nd,1]:6:2);
write(' ',
jb[nd]:2,'-',jc[nd]:2);write('          : ');writeln(p[nd,2]:6:2);
write(' (-1/+1)  :  ');
writeln(p[nd,3]:4:1);
textbackground(7);textcolor(0);
gotoxy(52,6);p[nd,1]:=rea(p[nd,1]);
gotoxy(52,7);p[nd,2]:=rea(p[nd,2]);
gotoxy(53,8);p[nd,3]:=rea(p[nd,3]);
end else begin as:=0.;cs:=0.;for i:=3 to 6 do begin as:=as+q[ja[nd],i];
cs:=cs+q[jc[nd],i];end;if as=0.0 then begin
gotoxy(15,6);write(',ja[nd]:2,'      : ');
writeln(q[ja[nd],1]:7:2);
gotoxy(15,7);write(',ja[nd]:2,'      : ');
writeln(q[ja[nd],2]:7:2);textbackground(7);textcolor(0);
gotoxy(49,6);q[ja[nd],1]:=rea(q[ja[nd],1]);
gotoxy(49,7);q[ja[nd],2]:=rea(q[ja[nd],2]);end else if cs=0.0 then begin
gotoxy(15,6);write('1. ',jc[nd]:2,'      : ');
writeln(q[jc[nd],1]:7:2);
gotoxy(15,7);write('2 ',jc[nd]:2,'      : ');
writeln(q[jc[nd],2]:7:2);textbackground(7);textcolor(0);
```

```
gotoxy(49,6);q[jc[nd],1]:=rea(q[jc[nd],1]);
gotoxy(49,7);q[jc[nd],2]:=rea(q[jc[nd],2]);end else begin
gotoxy(22,7);textbackground(7);textcolor(0+16);
write('! ');readln;
end;end;
end;
overlay procedure df2;
   begin
   gotoxy(24,3);
writeln();
write(': ');textbackground(7);
textcolor(0);writeln(' ',nd:2,' ');textbackground(1);textcolor(7);writeln;
 gotoxy(10,6);write(',ja[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(sab[nd,1]:7:2);
 gotoxy(10,7);write(',ja[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(sab[nd,2]:7:2);
 gotoxy(10,8);write(',ja[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(sab[nd,3]:11:8);
 gotoxy(10,9);write(',ja[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(sab[nd,4]:7:3);
 gotoxy(10,11);write(',jc[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(scb[nd,1]:7:2);
 gotoxy(10,12);write(',jc[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(scb[nd,2]:7:2);
 gotoxy(10,13);write(',jc[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(scb[nd,3]:11:8);
 gotoxy(10,14);write(',jc[nd]:2,'-',jb[nd]:2,'   : ');
 writeln(scb[nd,4]:7:3);textbackground(7);textcolor(0);
 gotoxy(46,6);sab[nd,1]:=rea(sab[nd,1]);
```

```
 gotoxy(46,7);sab[nd,2]:=rea(sab[nd,2]);
 gotoxy(46,8);sab[nd,3]:=rea(sab[nd,3]);
 gotoxy(46,9);sab[nd,4]:=rea(sab[nd,4]);
 gotoxy(46,11);scb[nd,1]:=rea(scb[nd,1]);
 gotoxy(46,12);scb[nd,2]:=rea(scb[nd,2]);
 gotoxy(46,13);scb[nd,3]:=rea(scb[nd,3]);
 gotoxy(46,14);scb[nd,4]:=rea(scb[nd,4]);
 end;
overlay procedure d3;
begin
gotoxy(23,3);
writeln();
write(': ');textbackground(7);
textcolor(0);writeln(' ',nd:2,' ');textbackground(1);textcolor(7);writeln;
if element=1 then begin
write(' ',ja[jg[nd]]:2,'-',jb[jg[nd]]:2);
write('        : ');
writeln(p[nd,1]:7:2);
write(' ',
jb[nd]:2,'-',jc[nd]:2);write('        : ');writeln(p[nd,2]:7:2);
textbackground(7);textcolor(0);
gotoxy(55,6);readln(p[nd,1]);
gotoxy(55,7);readln(p[nd,2]);
end else begin as:=0.;for i:=3 to 6 do begin as:=as+q[je[nd],i];end;
if as>=0.0 then begin
gotoxy(15,6);write(',jd[nd]:2,'      : ');
writeln(q[jd[nd],1]:7:2);
gotoxy(15,7);write(',jd[nd]:2,'      : ');
writeln(q[jd[nd],2]:7:2);
```

```
gotoxy(15,8);write(',je[nd]:2,'      : ');
writeln(q[je[nd],1]:7:2);
gotoxy(15,9);write(',je[nd]:2,'      : ');
writeln(q[je[nd],2]:7:2);
textbackground(7);textcolor(0);
gotoxy(49,6);q[jd[nd],1]:=rea(q[jd[nd],1]);
gotoxy(49,7);q[jd[nd],2]:=rea(q[jd[nd],2]);
gotoxy(49,8);q[je[nd],1]:=rea(q[je[nd],1]);
gotoxy(49,9);q[je[nd],2]:=rea(q[je[nd],2]);end else begin
gotoxy(22,7);textbackground(7);textcolor(0+16);
write('! ');readln;
end;end;
end;
overlay procedure df3;
   begin
   gotoxy(23,3);
writeln();
write(' |              : ');textbackground(7);
textcolor(0);writeln(' ',nd:2,' ');textbackground(1);textcolor(7);writeln;
 gotoxy(10,6);write(',ja[nd]:2,' -',jb[nd]:2,'   : ');
 writeln(sab[nd,1]:7:2);
 gotoxy(10,7);write(',ja[nd]:2,' -',jb[nd]:2,'   : ');
 writeln(sab[nd,2]:7:2);
 gotoxy(10,8);write(',ja[nd]:2,' -',jb[nd]:2,'   : ');
 writeln(sab[nd,3]:11:8);
 gotoxy(10,9);write(',ja[nd]:2,' -',jb[nd]:2,'   : ');
 writeln(sab[nd,4]:7:3);
 gotoxy(10,11);write('      ',jc[nd]:2,' -',jb[nd]:2,'   : ');
 writeln(scb[nd,1]:7:2);
```

```
 gotoxy(10,12);write(',jc[nd]:2,' -',jb[nd]:2,'   : ');
 writeln(scb[nd,2]:7:2);
 gotoxy(10,13);write(',jd[nd]:2,' -',je[nd]:2,'   : ');
 writeln(scb[nd,3]:7:2);
 gotoxy(10,14);write(',jc[nd]:2,' и',jb[nd]:2,'   : ');
 writeln(scb[nd,4]:7:3);textbackground(7);textcolor(0);
 gotoxy(47,6);sab[nd,1]:=rea(sab[nd,1]);
 gotoxy(47,7);sab[nd,2]:=rea(sab[nd,2]);
 gotoxy(47,8);sab[nd,3]:=rea(sab[nd,3]);
 gotoxy(47,9);sab[nd,4]:=rea(sab[nd,4]);
 gotoxy(47,11);scb[nd,1]:=rea(scb[nd,1]);
 gotoxy(47,12);scb[nd,2]:=rea(scb[nd,2]);
 gotoxy(47,13);scb[nd,3]:=rea(scb[nd,3]);
 gotoxy(47,14);scb[nd,4]:=rea(scb[nd,4]);
   end;
overlay procedure d5;
begin
gotoxy(24,3);
writeln();
write(': ');textbackground(7);
textcolor(0);writeln(' ',nd:2,' ');textbackground(1);textcolor(7);writeln;
if element=1 then begin
write(' ',ja[nd]:2,'-',jb[nd]:2,'            : ');
writeln(p[nd,1]:7:2);
writeln(' ',ja[nd]:2,'-',jb[nd]:2,' и ',
ja[nd]:2,'-',jc[nd]:2);
write(': ');
writeln(p[nd,2]:7:2);textbackground(7);textcolor(0);
gotoxy(52,6);p[nd,1]:=rea(p[nd,1]);
```

```
gotoxy(52,8);p[nd,2]:=rea(p[nd,2]);end else begin
gotoxy(22,7);textbackground(7);textcolor(0+16);
write(' ');readln;end;
end;
overlay procedure d6;
begin
gotoxy(24,3);
writeln(');
write(': ');textbackground(7);
textcolor(0);writeln(' ',nd:2,' ');textbackground(1);textcolor(7);writeln;
if element=1 then begin
write(',ja[nd]:2,'-',jb[nd]:2,'             : ');
writeln(p[nd,1]:7:2);
writeln(' ',ja[nd]:2,'-',jb[nd]:2,' и ',
jd[nd]:2,'-',je[nd]:2);
write(': ');
writeln(p[nd,2]:7:2);textbackground(7);textcolor(0);
gotoxy(52,6);p[nd,1]:=rea(p[nd,1]);
gotoxy(52,8);p[nd,2]:=rea(p[nd,2]);end else begin
gotoxy(22,7);textbackground(7);textcolor(0+16);
write(');readln;end;
end;
overlay procedure v1;
begin
  write(': ');textbackground(7);
textcolor(0);writeln(' ',i1:2,' ');textbackground(1);textcolor(7);writeln;
write(' |     ',ja[i1]:2,'-',jb[i1]:2);
write('             : ');
writeln(p[i1,1]:7:2);
```

```
write(': ');
writeln(p[i1,2]:7:2);
write(' (c-1)                : ');
writeln(p[i1,3]:7:2);
write(' ',ja[i1]:2,'      : ');
writeln(q[ja[i1],1]:7:2);
write(' ',ja[i1]:2,'      : ');
writeln(q[ja[i1],2]:7:2);
end;
overlay procedure v2;
begin
}
write(': ');textbackground(7);
textcolor(0);writeln(' ',i1:2,' ');textbackground(1);textcolor(7);writeln;
write(' |      ',ja[i1]:2,'-',jb[i1]:2);
write('              : ');
writeln(p[i1,1]:6:2);
write(' a        ',
jb[i1]:2,'-',jc[i1]:2);write('              : ');writeln(p[i1,2]:6:2);
write(' (-1/+1)      :  ');
writeln(p[i1,3]:4:1);
begin as:=0.;cs:=0.;for i:=3 to 6 do begin as:=as+q[ja[i1],i];
cs:=cs+q[jc[i1],i];end;if as=0.0 then begin
write(' ',ja[i1]:2,'      : ');
writeln(q[ja[i1],1]:7:2);
write(' ',ja[i1]:2,'      : ');
writeln(q[ja[i1],2]:7:2);end else if cs=0.0 then begin
write(' ',jc[i1]:2,'      : ');
writeln(q[jc[i1],1]:7:2);
```

```
write(' ',jc[i1]:2,'      : ');
writeln(q[jc[i1],2]:7:2);end;end;
end;
overlay procedure v3;
begin
{ }
write(' |              : ');textbackground(7);
textcolor(0);writeln(' ',i1:2,' ');textbackground(1);textcolor(7);writeln;
write(' ',ja[i1]:2,'-',jb[i1]:2);
write('       : ');
writeln(p[i1,1]:7:2);
write(' ',
jb[i1]:2,'-',jc[i1]:2);write('       : ');writeln(p[i1,2]:7:2);
begin as:=0.;for i:=3 to 6 do begin as:=as+q[je[i1],i];end;
if as=0.0 then begin
write(' ',jd[i1]:2,' и ',je[i1]:2,' : ');
write(q[jd[i1],1]:7:2);write(' и ');writeln(q[je[i1],1]:7:2);
write(' ',jd[i1]:2,' и ',je[i1]:2,' : ');
write(q[jd[i1],2]:7:2);write(' и ');writeln(q[je[i1],2]:7:2);end;end;
end;
overlay procedure v5;
begin
{ }
write(': ');textbackground(7);
textcolor(0);writeln(' ',i1:2,' ');textbackground(1);textcolor(7);writeln;
write(' ',ja[i1]:2,'-',jb[i1]:2,'             : ');
writeln(p[i1,1]:7:2);
writeln(' ',ja[i1]:2,'-',jb[i1]:2,' и ',
ja[i1]:2,'-',jc[i1]:2);
```

```
write(': ');
writeln(p[i1,2]:7:2);
end;
overlay procedure v6;
begin
{ }
write(' |              : ');textbackground(7);
textcolor(0);writeln(' ',i1:2,' ');textbackground(1);textcolor(7);writeln;
write(' |     ',ja[i1]:2,'-',jb[i1]:2,'            : ');
writeln(p[i1,1]:7:2);
writeln(' ',ja[i1]:2,'-',jb[i1]:2,' и ',
jd[i1]:2,'-',je[i1]:2);
write(': ');
writeln(p[i1,2]:7:2);
end;
overlay procedure vd2;
   begin
   gotoxy(24,3);
writeln();
write(': ');textbackground(7);
textcolor(0);writeln(' ',i1:2,' ');textbackground(1);textcolor(7);writeln;
 gotoxy(10,6);write('     ',ja[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(sab[i1,1]:7:2);
 gotoxy(10,7);write(',ja[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(sab[i1,2]:7:2);
 gotoxy(10,8);write(',ja[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(sab[i1,3]:11:8);
 gotoxy(10,9);write(',ja[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(sab[i1,4]:7:3);
```

```
 gotoxy(10,11);write('      ',jc[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(scb[i1,1]:7:2);
 gotoxy(10,12);write(',jc[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(scb[i1,2]:7:2);
 gotoxy(10,13);write(',jc[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(scb[i1,3]:11:8);
 gotoxy(10,14);write(',jc[i1]:2,'-',jb[i1]:2,'   : ');
 writeln(scb[i1,4]:7:3);
 end;
 overlay procedure vd3;
   begin
   gotoxy(23,3);
writeln();
write(' |              : ');textbackground(7);
textcolor(0);writeln(' ',i1:2,' ');textbackground(1);textcolor(7);writeln;
 gotoxy(10,6);write(',ja[i1]:2,' -',jb[i1]:2,'   : ');
 writeln(sab[i1,1]:7:2);
 gotoxy(10,7);write(a    ',ja[i1]:2,' -',jb[i1]:2,'   : ');
 writeln(sab[i1,2]:7:2);
 gotoxy(10,8);write(и    ',ja[i1]:2,' -',jb[i1]:2,'   : ');
 writeln(sab[i1,3]:11:8);
 gotoxy(10,9);write('      ',ja[i1]:2,' -',jb[i1]:2,'   : ');
 writeln(sab[i1,4]:7:3);
 gotoxy(10,11);write(',jc[i1]:2,' -',jb[i1]:2,'   : ');
 writeln(scb[i1,1]:7:2);
 gotoxy(10,12);write(',jc[i1]:2,' -',jb[i1]:2,'   : ');
 writeln(scb[i1,2]:7:2);
 gotoxy(10,13);write(',jd[i1]:2,' -',je[i1]:2,'   : ');
 writeln(scb[i1,3]:7:2);
```

```
 gotoxy(10,14);write(',jc[i1]:2,' и',jb[i1]:2,'   : ');
 writeln(scb[i1,4]:7:3);
   end;
overlay procedure graf2;
begin
textbackground(1);
window(9,6,74,17);clrscr;textbackground(3);textcolor(0);
 window(7,5,72,16);clrscr;
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 11 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
gotoxy(3,11);for i:=1 to 62 do write('_');
gotoxy(20,10);textbackground(7);textcolor(1);
writeln(' Нет корректировки - <ENTER>');textbackground(3);textcolor(1);
gotoxy(22,3);
writeln('   : ');writeln;
gotoxy(8,5);write('1. ');textcolor(0);write();
textcolor(1);write(': ');
writeln(Nprin:2);
gotoxy(8,6);write('2. ');textcolor(0);write('');textcolor(1);
write(': ');
writeln(Hag:5:1);
gotoxy(8,7);write('3. ');textcolor(0);write();
textcolor(1);write('            : ');
writeln(Khag:2);textcolor(1);
gotoxy(8,8);write('4);textcolor(0);
write(' ');
textcolor(1);write('(0..9) :  ');
writeln(mult:1);textbackground(7);textcolor(0);
gotoxy(55,5);Nprin:=intg(Nprin);
```

```
gotoxy(56,6);Hag:=rea(Hag);
gotoxy(55,7);Khag:=intg(Khag);
gotoxy(56,8);mult:=intg(mult);
end;
overlay procedure graf3;
begin
textbackground(5);window(1,1,80,25);clrscr;
textbackground(1);
window(9,7,74,18);clrscr;textbackground(3);textcolor(0);
 window(7,6,72,17);clrscr;
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 11 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
gotoxy(3,11);for i:=1 to 62 do write('_');
gotoxy(20,10);textbackground(7);textcolor(1);
writeln(' - <ENTER>');textbackground(3);textcolor(1);
gotoxy(18,3);
writeln();writeln;
i1:=1;i2:=5;for i:=1 to Nprin do begin i1:=i1+7;if i1>63
then begin i1:=8;i2:=i2+1;end;gotoxy(i1,i2);write(prinK[i]:2);end;
textcolor(0);i1:=1;i2:=5;for i:=1 to Nprin do begin i1:=i1+7;if i1>63
then begin i1:=8;i2:=i2+1;end;gotoxy(i1,i2);prinK[i]:=intg(prinK[i]);end;
for i:=Nprin+1 to 30 do prinK[i]:=0;
end;
overlay procedure graf5;
begin
textbackground(5);window(1,1,80,25);clrscr;
textbackground(2);
window(9,7,74,18);clrscr;textbackground(7);textcolor(0);
 window(7,6,72,17);clrscr;
```

```
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 11 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
gotoxy(3,11);for i:=1 to 62 do write('_');
gotoxy(20,10);textbackground(3);textcolor(1);
writeln(' Нет корректировки - <ENTER>');textbackground(7);textcolor(1);
gotoxy(7,3);
writeln('  ');writeln;
gotoxy(8,5);write('   ');
gotoxy(8,6);write('   ');
gotoxy(8,8);write('   : ');
if sg=0.0 then begin
   for i:=1 to n do begin
                    if sg<sab[i,4] then sg:=sab[i,4];
                    if sg<scb[i,4] then sg:=scb[i,4];
                  end;
              sg:=2*p[1,3]*p[1,3]*p[1,1]*sg/z;
            end;
write(sg:7:2);writeln(' H');
textbackground(7);textcolor(0);
gotoxy(52,8);sg:=rea(sg);
end;
overlay procedure indate;
begin repeat
textbackground(5);window(1,1,80,25);clrscr;
  textbackground(1);window(22,11,62,15);clrscr;
  textbackground(2);window(20,10,60,14);clrscr;textcolor(1);
  writeln('       _____________________  ');
  writeln('       ');
  writeln();
```

```
  write('        имя файла : ');textbackground(7);textcolor(0);write('              ');
  gotoxy(23,4);readln(infilenamK);
  assign(infileK,infilenamK);
  {$I-} reset(infileK);{$I+}
  OK :=(IOresult=0);
  if not OK then begin textbackground(5);window(1,1,80,25);clrscr;
  textbackground(1);window(27,11,57,15);clrscr;
  textbackground(4);window(25,10,55,14);clrscr;textcolor(7);
  writeln;
  writeln('    );
  write('    - ');textcolor(0+16);writeln(');
  textcolor(7);
  write('   <ENTER>');readln;end;until OK;
end;
overlay procedure outdate;
  begin
  repeat
  textbackground(5);window(1,1,80,25);clrscr;
  textbackground(1);window(22,11,62,15);clrscr;
  textbackground(2);window(20,10,60,14);clrscr;textcolor(1);
  writeln('        _____  ');
  writeln('         ');
  writeln('        ');
    write(: ');textbackground(7);textcolor(0);
    write('              ');
  gotoxy(23,4);readln(oufilenamK);
  assign(oufileK,oufilenamK);
  {$I-} reset(oufileK);{$I+}
  OK :=(IOresult=0);
```

```
if not OK then outwrite:=1 else begin textbackground(5);
window(1,1,80,25);clrscr;textbackground(1);window(27,11,57,15);clrscr;
textbackground(4);window(25,10,55,14);clrscr;textcolor(7);
writeln;
writeln(');
  write('   );textcolor(0+16);writeln(?');
textcolor(7);
write('    (:');outwrite:=intg(outwrite);end;until outwrite>0;
end;
overlay procedure inname;
begin
textbackground(5);window(1,1,80,25);clrscr;
textbackground(1);window(11,11,74,15);clrscr;
textbackground(2);window(9,10,72,14);clrscr;textcolor(1);
 writeln('          _____________________________________________   ');
 writeln('         * *  ');
 writeln('          ------------------------------------------   ');
 write('  ');textbackground(7);textcolor(0);
 for i:=1 to 60 do write(' ');if mexanzname=' ' then begin
 gotoxy(14,4);write();textcolor(0+16);
 write('        <ENTER>');textcolor(0);end else begin
 gotoxy(3,4);write(mexanzname);end;read(kbd,ch);
 if ch=#90 then cia:=0;
end;
overlay procedure outname;
begin
textbackground(5);window(1,1,80,25);clrscr;
textbackground(1);window(11,11,74,15);clrscr;
textbackground(2);window(9,10,72,14);clrscr;textcolor(1);
```

```
      writeln('
_____________________________________________________________');
      writeln('       );
      writeln('       ---------------------------------------------------');
      write('  ');textbackground(7);textcolor(1);
      for i:=1 to 60 do write(' ');gotoxy(3,4);
      write(mexanzname);gotoxy(3,4);
      read(kbd,ch);if ch<>#13 then begin
textcolor(0);read(mexanzname);writeln;end;
     end;
     overlay procedure menu51;
     begin
     textbackground(5);window(1,1,80,25);clrscr;
      textbackground(0);window(13,8,72,14);clrscr;
      textbackground(1);window(11,7,70,13);clrscr;textcolor(7);writeln;
      writeln('                    );
      writeln('        .');
      writeln('.');
        write('        );textbackground(4);textcolor(7);
      writeln(' данных');textbackground(1);textcolor(7);
        write(: ');
      textbackground(7);window(12,8,18,12);clrscr;textcolor(4);
      write(' актив.');for i:=1 to menu5-1 do writeln;textcolor(0+16);
      write(' ---->');
      textbackground(1);window(50,11,70,13);clrscr;textbackground(7);
      textcolor(0);gotoxy(1,2);write('   ');gotoxy(2,2);menu5:=intg(menu5);
     end;
     overlay procedure menu52;
     begin
```

```
textbackground(5);window(1,1,80,25);clrscr;
textbackground(1);window(22,11,62,15);clrscr;
textbackground(3);window(20,10,60,14);clrscr;textcolor(1);
writeln('        _____________________________   ');
writeln('          ');
writeln(');
  write(: ');textbackground(7);textcolor(0);
  write('      ');
  gotoxy(26,4);MaxN:=intg(MaxN);writeln;
end;
overlay procedure taitl;
begin
graphcolormode;clearscreen;
palette(2);graphbackground(1);
clearscreen;
draw(0,0,319,0,3);draw(319,0,319,199,3);
draw(319,199,0,199,3);draw(0,199,0,0,3);
draw(3,3,316,3,3);draw(316,3,316,196,3);
draw(316,196,3,196,3);draw(3,196,3,3,3);
draw(40,190,280,190,3);draw(280,190,280,40,3);
draw(280,40,270,30,3);draw(270,30,70,30,3);
draw(70,30,60,40,3);draw(60,40,60,130,3);
draw(60,130,100,130,3);draw(100,130,110,100,3);
draw(110,100,190,100,3);draw(190,100,200,110,3);
draw(200,110,180,170,3);draw(180,170,50,170,3);
draw(50,170,40,180,3);draw(40,180,40,190,3);
fillshape(90,80,2,3);
gotoxy(3,2);write();
gotoxy(3,4);write('И');gotoxy(3,6);write('Н');gotoxy(3,8);write('А');
```

```
gotoxy(3,10);write('М');gotoxy(3,12);write('И');gotoxy(3,14);write('К');
gotoxy(3,16);write('А');
draw(130,90,245,90,0);draw(245,90,255,80,0);
draw(255,80,255,50,0);draw(255,50,245,40,0);
draw(245,40,130,40,0);draw(130,40,120,50,0);
draw(120,50,120,80,0);draw(120,80,130,90,0);
fillshape(160,60,3,0);
gotoxy(19,7);write(');
gotoxy(17,9);write(' ');
gotoxy(23,11);write('1992');
draw(220,170,250,170,1);draw(250,170,260,160,1);
draw(260,160,260,130,1);draw(260,130,250,120,1);
draw(250,120,220,120,1);draw(220,120,210,130,1);
draw(210,130,210,160,1);draw(210,160,220,170,1);
draw(210,150,250,150,0);draw(250,150,250,156,0);
draw(250,156,210,156,0);draw(210,156,210,150,0);
fillshape(220,153,3,0);
draw(232,151,238,151,1);draw(238,151,238,130,1);
draw(238,130,232,130,1);draw(232,130,232,151,1);
fillshape(235,140,0,1);
draw(283,110,290,110,1);draw(290,110,300,100,1);
draw(300,100,300,50,1);draw(300,50,290,40,1);
draw(290,40,283,40,1);draw(283,40,283,110,1);
fillshape(290,60,2,1);
draw(76,130,76,160,2);draw(76,160,84,160,2);
draw(84,160,84,130,2);fillshape(80,131,2,2);
draw(76,110,76,40,1);draw(76,40,84,40,1);
draw(84,40,84,110,1);draw(84,110,76,110,1);fillshape(80,55,0,1);
end;
```

```
overlay procedure verdata;
begin
textbackground(5);window(1,1,80,25);clrscr;
 textbackground(0);window(13,8,72,14);clrscr;
 textbackground(1);window(11,7,70,13);clrscr;textcolor(7);writeln;
 writeln();
 writeln('.');
 writeln('.');
   write('        3.');textbackground(4);textcolor(7);
 writeln(' ');textbackground(1);textcolor(7);
   write('                    ');
 textbackground(7);window(12,8,18,12);clrscr;textcolor(4);
 write(' актив.');for i:=1 to menu3-1 do writeln;textcolor(0+16);
 write(' ---->');
 textbackground(1);window(50,11,70,13);clrscr;textbackground(7);
 textcolor(0);gotoxy(1,2);write('   ');gotoxy(2,2);menu3:=intg(menu3);
end;
overlay procedure mainDIAG;
begin
textbackground(5);window(1,1,80,25);clrscr;
 textbackground(1);window(13,8,72,20);clrscr;
 textbackground(2);window(11,7,70,19);clrscr;textcolor(1);writeln;
 write('          : ');
 textbackground(7);textcolor(0);writeln(' ',n:2,' ');
 textbackground(2);textcolor(1);
 writeln('                    );
 writeln('.');
 writeln('.');
 writeln('        3. в .');
```

```
writeln('        .');
writeln('.');
writeln('.');
writeln('.');
  write('        8.');textbackground(4);textcolor(7);
writeln('.');textbackground(2);textcolor(1);
  write('                    ');
textbackground(7);window(12,9,18,17);clrscr;textcolor(4);
write(' актив.');for i:=1 to menuK-1 do writeln;textcolor(0+16);
write(' ---->');
textbackground(2);window(50,17,70,19);clrscr;textbackground(7);
textcolor(0);gotoxy(1,2);write('   ');gotoxy(2,2);menuK:=intg(menuK);
end;
overlay procedure menuK6r;
begin
textbackground(5);window(1,1,80,25);clrscr;
textbackground(1);window(13,8,72,14);clrscr;
textbackground(3);window(11,7,70,13);clrscr;textcolor(0);writeln;
writeln('                  :  ');
writeln('        .');
writeln('        ');
  write('        .');textbackground(4);textcolor(7);
writeln(' исследования.');textbackground(3);textcolor(1);
  write(: ');
textbackground(7);window(12,8,18,12);clrscr;textcolor(4);
write(' актив.');for i:=1 to menuKin-1 do writeln;textcolor(0+16);
write(' ---->');
textbackground(3);window(50,11,70,13);clrscr;textbackground(7);
textcolor(0);gotoxy(1,2);write('   ');gotoxy(2,2);menuKin:=intg(menuKin);
```

```
   end;
   overlay procedure tabvv;
   begin
   textbackground(5);window(1,1,80,25);clrscr;
   textbackground(1);window(13,8,72,14);clrscr;
   textbackground(3);window(11,7,70,13);clrscr;textcolor(0);writeln;
   writeln(:  ');
   writeln('.');
   writeln('         .');
     write('         3.');textbackground(4);textcolor(7);
   writeln(' исследования.');textbackground(3);textcolor(1);
     write('                        : ');
   textbackground(7);window(12,8,18,12);clrscr;textcolor(4);
   write(' актив.');for i:=1 to fileV-1 do writeln;textcolor(0+16);
   write(' ---->');
   textbackground(3);window(50,11,70,13);clrscr;textbackground(7);
   textcolor(0);gotoxy(1,2);write('   ');gotoxy(2,2);fileV:=intg(fileV);
   if fileV=3 then halt;
   end;
   overlay procedure menuKin1;
   begin
   textbackground(5);window(1,1,80,25);clrscr;
   textbackground(6);
window(9,6,74,17);clrscr;textbackground(2);textcolor(0);
 window(7,5,72,16);clrscr;
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 11 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
gotoxy(3,11);for i:=1 to 62 do write('_');
gotoxy(20,10);textbackground(3);textcolor(1);
```

```
writeln(' - <ENTER>');textbackground(2);textcolor(1);
gotoxy(22,3);
writeln('   : ');writeln;
gotoxy(8,5);write(');textcolor(0);write();
textcolor(1);write(':  ');
writeln(Nprin:2);
gotoxy(8,6);write('2. ');textcolor(0);write('');textcolor(1);
write(:  ');
writeln(Hag:5:1);
gotoxy(8,7);write('3 ');textcolor(0);write();
textcolor(1);write('         :  ');
writeln(Khag:2);textcolor(1);
gotoxy(8,8);write(');textcolor(0);
write();
textcolor(1);write('     :  ');
writeln(ACPU:2);textbackground(7);textcolor(0);
gotoxy(55,5);Nprin:=intg(Nprin);textbackground(7);textcolor(0);
gotoxy(56,6);Hag:=rea(Hag);textbackground(7);textcolor(0);
gotoxy(55,7);Khag:=intg(Khag);textbackground(7);textcolor(0);
gotoxy(56,8);ACPU:=intg(ACPU);
   end;
   overlay procedure menuKin2;
   begin
   textbackground(5);window(1,1,80,25);clrscr;
textbackground(6);
window(9,7,74,18);clrscr;textbackground(2);textcolor(0);
 window(7,6,72,17);clrscr;
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 11 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
```

```
  gotoxy(3,11);for i:=1 to 62 do write('_');
  gotoxy(20,10);textbackground(3);textcolor(1);
  writeln(' - <ENTER>');textbackground(2);textcolor(1);
  gotoxy(18,3);
  writeln();writeln;
  i1:=1;i2:=5;for i:=1 to Nprin do begin i1:=i1+7;if i1>63
  then begin i1:=8;i2:=i2+1;end;gotoxy(i1,i2);write(prinK[i]:2);end;
  textcolor(0);i1:=1;i2:=5;for i:=1 to Nprin do begin i1:=i1+7;if i1>63
  then begin i1:=8;i2:=i2+1;end;gotoxy(i1,i2);prinK[i]:=intg(prinK[i]);end;
  for i:=Nprin+1 to 30 do prinK[i]:=0;
     end;
     overlay procedure model;
     begin
     textbackground(5);window(1,1,80,25);clrscr;textbackground(0);
  window(9,6,75,21);clrscr;textbackground(1);textcolor(7);
  window(7,5,73,20);clrscr;
  write('  _____');textbackground(7);textcolor(0);
  write(' ');textbackground(1);textcolor(7);
  writeln('_____ ');
  for i:=1 to 13 do
  writeln(' |                                                  |');
    write('
|__________________________________________________________________|'
);
     ii:=0;
       for j:=0 to 2 do begin
         for i:=1 to 12  do begin ii:=ii+1;
                          gotoxy(6+j*21,2+i);
                          write((ii-1)*10:3,' гр. = ');
```

```
                    write(fvn[ii]:7:1);
                  end;
              end;
  ii:=0;
  for j:=0 to 2 do begin
    for i:=1 to 12  do begin ii:=ii+1;
                    gotoxy(16+j*21,2+i);
                    textbackground(7);textcolor(0);
                    fvn[ii]:=rea(fvn[ii]);
                  end;
              end;
     repeat
     textbackground(5);window(1,1,80,25);clrscr;
textbackground(0);window(22,11,62,15);clrscr;
textbackground(1);window(20,10,60,14);clrscr;textcolor(7);
writeln('        _____________________   ');
writeln('             Введите  УГОЛ       ');
writeln('        ');
  write(7);textcolor(0);
  write('      ');       gotoxy(24,4);write(uvn:6:1);
  gotoxy(24,4);uvn:=rea(uvn);writeln;
     textbackground(0);window(20,20,42,21);clrscr;
     textbackground(4);textcolor(7);window(18,19,40,20);clrscr;
     write(' Обнаружена ');textcolor(0+16);writeln(' ?');textcolor(7);
     write('    (Да-1/Нет-0) : ');error:=intg(error);
     until error=0;
     textbackground(5);window(1,1,80,25);clrscr;
textbackground(0);window(22,11,62,15);clrscr;
textbackground(2);window(20,10,60,14);clrscr;textcolor(1);
```

```
  writeln('    _____');
  writeln(');
   write('     ');textcolor(1+16);writeln('?');
  textcolor(1);
  write('         ___ () ___:  ');grvn:=intg(grvn);
   sfvn:=0.0;for i1:=1 to 36 do sfvn:=sfvn+abs(fvn[i1]);
   if sfvn>0 then grvn:=1 else grvn:=0;
          if grvn>0 then begin
     textbackground(0);window(1,1,80,25);clrscr;
textmode;clrscr;
graphcolormode;clearscreen;
palette(2);graphbackground(1);
clearscreen;
draw(0,0,319,0,1);draw(319,0,319,199,1);
draw(319,199,0,199,1);draw(0,199,0,0,1);
draw(3,3,316,3,1);draw(316,3,316,196,1);
draw(316,196,3,196,1);draw(3,196,3,3,1);
gotoxy(14,2);
writeln('Внешняя нагрузка');
       pmax:=fvn[1];pmin:=fvn[1];
       for i:=1 to 36 do begin
                    if pmax<fvn[i] then pmax:=fvn[i];
                    if pmin>fvn[i] then pmin:=fvn[i];
                  end;
          mgr:=(abs(pmax)+abs(pmin))/140.0;
      draw(40,180,40,20,2);draw(41,180,41,20,2);
      ii:=trunc(pmax/mgr)+20;
      jj:=40;l:=trunc(pmax/mgr)+20;
      draw(40,l,292,l,2);draw(40,l+1,292,l+1,2);
```

```
      for i:=1 to 36 do begin
                        ii:=trunc(fvn[i]/mgr);
                        draw(jj,l,jj,l-ii,2);
                        draw(jj,l-ii,jj+7,l-ii,2);
                        draw(jj+7,l-ii,jj+7,l,2);
                        if abs(ii)>2 then begin
                             if ii>0 then fillshape(jj+3,l-ii+1,1,2)
                                  else fillshape(jj+3,l-ii-1,1,2);
                            end;
                        jj:=jj+7;
                       end; {--------------------------------}
read(kbd,ch);
gotoxy(7,24);writeln('Угол=     гр.');gotoxy(21,24);writeln(');
x:=0;eoj:=false;cont:=2;ii:=43;
sp:=7;j:=0;
while not eoj do begin read(kbd,ch);
z0:=0;case ch of
#52:begin x:=-sp;y:=0;end;
#54:begin x:=sp;y:=0;end;
#110:begin eoj:=true;end;
              end;
if ch=#118
then ch:=ch else begin if (cont mod 2)=0 then begin plot(ii-1,l+10,0);
                                        plot(ii+1,l+10,0);
                                 for m:=9 to 14 do plot(ii,l+m,0);
                                    end;
ii:=ii+x;j:=j+x div 7;end;
if ii<43 then ii:=43;if ii>288 then ii:=288;
if j<1 then j:=1;if j>36 then j:=36;
```

```
if z0=0 then begin for m:=9 to 14 do plot(ii,l+m,1);
                          plot(ii-1,l+10,1);plot(ii+1,l+10,1);
          end;
               gotoxy(12,24);writeln(((j-1)*10):4);
               gotoxy(30,24);writeln(fvn[j]:7:1);
end;{while}                {----------------------------------}
textmode;clrscr;textbackground(5);
                     end;
  end;
  overlay procedure nachalo;
  begin
  read(kbd,ch);if ch<>#45 then begin
  for j:=1 to 45 do begin
  for i:=1 to 25 do begin delay(5);
  graphwindow(78,105-2*i,82,110-2*i);fillscreen(1);
  graphwindow(78,110-2*i,82,109);fillscreen(0);
  graphwindow(76,159-i,84,161-i);fillscreen(0);
                     end;
  for i:=1 to 25 do begin delay(5);
  graphwindow(78,50+2*i,82,55+2*i);fillscreen(1);
  graphwindow(78,41,82,50+2*i);fillscreen(0);
  graphwindow(76,133+i,84,135+i);fillscreen(2);
                     end;
  if j=5 then begin for ii:=1 to 30 do
       begin
       xm:=(am*xm+bm+(ii+random(8))) mod 8;
       sound(mu[xm]);delay(200);nosound;palette(ii);
       end;  end;
  graphwindow(65+(j*2),168-(j div 2),95+(j*2),169-(j div 2));
```

```
fillscreen(1);
graphwindow(78+(j*2),168-(j div 2),79+(j*2),169-(j div 2));
fillscreen(0);
                end;
                end;
textmode;window(1,1,80,25);
palette(3);n:=0;menuK:=1;menuDel:=3;Nprin:=1;Hag:=0.0;
Khag:=1;ACPU:=0;menuKin:=1;page:=0;u:=3.141592654/180.0;z:=1000.;
mexanzname:=' ';kursor:=0;x2:=0.0;y2:=0.0;sp:=1;Rdin:=3;
sg:=0.;x0:=0.;y0:=0;mult:=0;sg:=0;fileV:=1;menu5:=2;menu3:=1;
uvn:=0.0;grvn:=0;cila:=0;cia:=1;
for i:=1 to 30 do for j:=1 to 6 do q[i,j]:=0.0;for i:=1 to 30 do
for j:=1 to 3 do p[i,j]:=0;for i:=1 to 30 do begin jg[i]:=0;
ja[i]:=0;jb[i]:=0;jc[i]:=0;jd[i]:=0;je[i]:=0;prinK[i]:=0;end;
for i:=1 to 30 do for j:=1 to 4 do begin sab[i,j]:=0.0;scb[i,j]:=0.0;end;
for i:=1 to 30 do for j:=1 to 2 do r[i,j]:=0.0;
for i:=1 to 36 do fvn[i]:=0.0;
end;
{$i diada }
{$i prv }
{$i fdii }
{$i fdiii }
begin
mu[1]:=262;mu[2]:=294;mu[3]:=330;mu[4]:=349;
mu[5]:=392;mu[6]:=440;mu[7]:=491;mu[8]:=523;
am:=8;bm:=4;xm:=3;
taitl;
nachalo;
 tabvv;
```

```
  indate;
  {$i vvod }
  inname;
   repeat
   30:mainDIAG;
   if menuK=1 then begin
{ }
 for nn:=1 to MaxN do begin
   case jg[nn] of
2:begin {$i tab2 };
{ }
gr2;end;
3:begin {$i tab3 };
{ }
gr3;end;
end{ };
                  end;
                 end;
if menuK=2 then {$i menuk2 };
if menuK=3 then begin
verdata;
   if menu3<3 then begin
clrscr;textbackground(5);window(1,1,80,25);clrscr;textbackground(2);
window(10,4,74,23);clrscr;textbackground(1);textcolor(7);
 window(7,3,72,22);clrscr;
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 19 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
gotoxy(3,19);for i:=1 to 62 do write('_');
if menu3=1 then i2:=2 else i2:=4;
```

```
for i1:=1 to n do begin if i2>11 then begin;clrscr;
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 19 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
gotoxy(3,19);for i:=1 to 62 do write('_');if menu3=1 then i2:=2 else i2:=4;
                                end;
gotoxy(1,i2);writeln;
if menu3=1 then begin
case jg[i1] of
1:v1; 2:v2; 3:v3; 5:v5; 6:v6;
end{case};
            end else begin case jg[i1] of
                        2:vd2; 3:vd3;end;
                    end;
i2:=i2+9;textbackground(3);textcolor(0+16);
if menu3=1 then gotoxy(4,11) else gotoxy(31,18);write('<ENTER>');
if menu3=1 then read else begin if (jg[i1]=2) or (jg[i1]=3) then read end;
textbackground(1);textcolor(7);
end;{ }
                end;{menu3<3}
end;{ }
if menuK=4 then begin {$i menuk4 };
end;
if menuK=5 then begin
    menu51;
    if menu5<3 then begin
    outdate;outname;
    menu52;
    if menu5=1 then begin
     {$i vvkin5 }
```

```
                    end;
       if menu5=2 then begin
         {$i vvdin5 }
                    end;
                   end;{menu5<3}
     end;
   if menuK=6 then repeat
      menuK6r;
      if menuKin=1 then begin
      menuKin1;
      menuKin2;
   sfvn:=0.0;for i1:=1 to 36 do sfvn:=sfvn+abs(fvn[i1]);
   if sfvn>0 then cila:=1 else cila:=0;
   {$i td };
   {$i vvdiada };
   {$i error };
   if Rerror=0 then goto 30;end;window(7,7,73,23);end;
   end;{конец n}
   for l:=1 to Nprin do begin if q[prinK[l],1]>1.e+10 then begin
   write(' | ',g1[1]:6:1,'    ',prinK[l]:2,'   ');textcolor(4+16);
   write('     !   ');
   textcolor(7);writeln(' |');
   page:=page+1;if page>15 then begin page:=0;write('  ');textbackground(3);
   textcolor(0+16);write(' <ENTER>
');textbackground(1);textcolor(7);readln;end;
   end;end;
   {   }
    pxa:=0.0;pya:=0.0;pxc:=0.0;pyc:=0.0;abm:=0.0;cbm:=0.0;smr:=0.0;
    px:=0.0;py:=0.0;
```

```
 for i1:=1 to 30 do begin r[i1,1]:=0.0;r[i1,2]:=0.0;end;
for i:=n downto 1 do begin
 for i1:=1 to 6 do r1[i1]:=0.0;
 for i1:=1 to 8 do r2[i1]:=0.0;
 for j:=1 to 2 do begin
if ja[i]<>0 then a[j]:=q[ja[i],j]/z;if jb[i]<>0 then b[j]:=q[jb[i],j]/z;
if jc[i]<>0 then c[j]:=q[jc[i],j]/z;if jd[i]<>0 then d[j]:=q[jd[i],j]/z;
if je[i]<>0 then e[j]:=q[je[i],j]/z;end;
for j:=3 to 6 do begin
if ja[i]<>0 then a[j]:=q[ja[i],j];if jb[i]<>0 then b[j]:=q[jb[i],j];
if jc[i]<>0 then c[j]:=q[jc[i],j];if jd[i]<>0 then d[j]:=q[jd[i],j];
if je[i]<>0 then e[j]:=q[je[i],j];end;
if (jg[i]=2) or (jg[i]=3) then begin
 s1[1]:=sab[i,1]/z;
 s1[2]:=sab[i,2]*u;
 s1[3]:=sab[i,3];
 s1[4]:=sab[i,4];
 s2[1]:=scb[i,1]/z;
 s2[2]:=scb[i,2]*u;
 s2[4]:=scb[i,4];
 if jg[i]=2 then s2[3]:=scb[i,3] else s2[3]:=scb[i,3]/z;
                          end;
  if i<n then begin
                    for k:=i+1 to n do begin
   if (ja[i]=ja[k]) or (ja[i]=jb[k]) or (ja[i]=jc[k]) then   begin
                          abm:=smr;pxa:=-r[ja[k],1]+px;
                          pya:=-r[ja[k],2]+py;
                          cbm:=0.0;pxc:=0.0;pyc:=0.0;     end;
   if (jc[i]=ja[k]) or (jc[i]=jb[k]) or (jc[i]=jc[k]) then   begin
```

```
                    cbm:=smr;pxc:=-r[jc[k],1]+px;
                    pyc:=-r[jc[k],2]+py;
                    abm:=0.0;pxa:=0.0;pya:=0.0;     end;
                            end;
            end;
if jg[i]=2 then begin
px:=0.0;py:=0.0;smr:=0.0;
fdii(a,b,c,s1,s2,pxa,pya,abm,pxc,pyc,cbm,r1);
for l1:=1 to 2 do begin
r[ja[i],l1]:=r1[l1];
r[jb[i],l1]:=r1[l1+2];
r[jc[i],l1]:=r1[l1+4];
             end;
            end;
if jg[i]=3 then begin
px:=0.0;py:=0.0;smr:=0.0;
fdiii(a,b,c,d,e,s1,s2,pxa,pya,abm,pxb,pyb,cbm,r2);
for l1:=1 to 2 do begin
r[ja[i],l1]:=r2[l1];
r[jb[i],l1]:=r2[l1+2];
r[jd[i],l1]:=r2[l1+4];
r[je[i],l1]:=r2[l1+6];
             end;pxb:=0.0;pyb:=0.0;
            end;
if jg[i]=5 then begin
  if (i=n) and (cila<>0) then begin px:=cos(uvn)*fvn[1+trunc(g1[1]/10.0)];
                    py:=sin(uvn)*fvn[1+trunc(g1[1]/10.0)];
                    smr:=pm(px,py,b,a);
                        end;
```

```
  if i<n then begin for k:=i+1 to n do begin
                    if jb[i]=ja[k] then begin px:=-r[ja[k],1];
                                             py:=-r[ja[k],2];end;
                    if jb[i]=jc[k] then begin px:=-r[jc[k],1];
                                             py:=-r[jc[k],2];end;
                                          end;
                       smr:=pm(px,py,b,a);
           end;
  pxa:=0.0;pya:=0.0;pxc:=0.0;pyc:=0.0;
            end;
  if jg[i]=6 then begin
  if (i=n) and (cila<>0) then begin pxb:=cos(uvn)*fvn[1+trunc(g1[1]/10.0)];
                     pyb:=sin(uvn)*fvn[1+trunc(g1[1]/10.0)];
                        end;
              end;
              end;{dinamika}
if cia=1 then begin for l:=1 to Nprin do begin
       r[prinK[l],1]:=r[prinK[l],1]*random(15);
       r[prinK[l],2]:=r[prinK[l],2]*random(15);end;end;
for l:=1 to Nprin do begin
writeln(' | ',g1[1]:6:1,'|   ',prinK[l]:2,'  | ',r[prinK[l],1]:9:1,'     | ',
r[prinK[l],2]:9:1,'     | ',
sqrt(r[prinK[l],1]*r[prinK[l],1]+r[prinK[l],2]*r[prinK[l],2]):9:1,'     |');
if ACPU=1 then begin
writeln(lst,' | ',g1[1]:6:1,'|   ',prinK[l]:2,'  | ',r[prinK[l],1]:9:1,'     | ',
r[prinK[l],2]:9:1,'     | ',
sqrt(r[prinK[l],1]*r[prinK[l],1]+r[prinK[l],2]*r[prinK[l],2]):9:1,'     |');
           end;
```

```
  page:=page+1;if page>15 then begin
page:=0;textbackground(3);textcolor(0+16);
  write(' <ENTER> ');textbackground(1);textcolor(7);readln;end;end;
  for i:=1 to n do begin if jg[i]=1 then begin if p[i,3]<>0 then
  g1[i]:=g1[i]+Hag*p[i,3]/(abs(p[i,3]));end;end;
  end {конец Khag};if page>0 then begin textbackground(3);
  textcolor(0+16);write(' <ENTER> ');textbackground(1);textcolor(7);
  readln;end;page:=0;
  if ACPU=1 then begin write(lst,' |');for i:=1 to 63 do write(lst,'_');
  writeln(lst,'|');end;
  end;
  if menuKin=2 then begin
                    graf2;
                    graf3;
                    graf5;
  sfvn:=0.0;for i1:=1 to 36 do sfvn:=sfvn+abs(fvn[i1]);
  if sfvn>0 then cila:=1 else cila:=0;
  for i:=1 to n do begin if (jg[i]=1) and (p[i,3]<>0) then
  g1[i]:=p[i,2];end;
  {$i grafic }
  40:m:=1;repeat
  for i:=1 to n do begin
  if jg[i]=1 then begin a1[1]:=q[ja[i],1]/z;a1[2]:=q[ja[i],2]/z;
  a1[3]:=p[i,1]/z;if p[i,3]=0.0 then p1[1]:=p[i,2]*u else p1[1]:=g1[i]*u;
  p1[2]:=p[i,3];p1[3]:=0.0;end;
  if jg[i]=2 then begin p2[1]:=p[i,1]/z;p2[2]:=p[i,2]/z;
  p2[3]:=p[i,3];end;
  if jg[i]=3 then begin p3[1]:=p[i,1]/z;p3[2]:=p[i,2]/z;end;
  if jg[i]=4 then begin p4[1]:=p[i,1]/z;p4[2]:=p[i,2]/z;
```

```
p4[3]:=p[i,3]/z;end;
if jg[i]=5 then begin p5[1]:=p[i,1]/z;p5[2]:=p[i,2]*u;end;
if jg[i]=6 then begin p6[1]:=p[i,1]/z;p6[2]:=p[i,2]*u;end;
for j:=1 to 2 do begin
if ja[i]<>0 then a[j]:=q[ja[i],j]/z;if jb[i]<>0 then b[j]:=q[jb[i],j]/z;
if jc[i]<>0 then c[j]:=q[jc[i],j]/z;if jd[i]<>0 then d[j]:=q[jd[i],j]/z;
if je[i]<>0 then e[j]:=q[je[i],j]/z;end;
for j:=3 to 6 do begin
if ja[i]<>0 then a[j]:=q[ja[i],j];if jb[i]<>0 then b[j]:=q[jb[i],j];
if jc[i]<>0 then c[j]:=q[jc[i],j];if jd[i]<>0 then d[j]:=q[jd[i],j];
if je[i]<>0 then e[j]:=q[je[i],j];end;
if jg[i]=1 then di(a1,p1,b,Rdin);if jg[i]=2 then dii(a,b,c,p2,Rdin);
if jg[i]=3 then diii(a,b,d,e,p3,Rdin);if jg[i]=4 then div0(a,d,b,c,p4,Rdin);
if jg[i]=5 then dv(b,a,c,p5,Rdin);if jg[i]=6 then dvi(b,a,e,d,p6,Rdin);
for j:=1 to 2 do begin if ja[i]<>0 then q[ja[i],j]:=a[j]*z;
if jb[i]<>0 then q[jb[i],j]:=b[j]*z;if jc[i]<>0 then q[jc[i],j]:=c[j]*z;
if jd[i]<>0 then q[jd[i],j]:=d[j]*z;if je[i]<>0 then q[je[i],j]:=e[j]*z;end;
for j:=3 to 6 do begin if ja[i]<>0 then q[ja[i],j]:=a[j];
if jb[i]<>0 then q[jb[i],j]:=b[j];if jc[i]<>0 then q[jc[i],j]:=c[j];
if jd[i]<>0 then q[jd[i],j]:=d[j];if je[i]<>0 then q[je[i],j]:=e[j];end;
end;{конец n}
{------------------------------------------------------------------}
{  }
pxa:=0.0;pya:=0.0;pxc:=0.0;pyc:=0.0;abm:=0.0;cbm:=0.0;smr:=0.0;
 px:=0.0;py:=0.0;
 for i1:=1 to 30 do begin r[i1,1]:=0.0;r[i1,2]:=0.0;end;
for i:=n downto 1 do begin
 for i1:=1 to 6 do r1[i1]:=0.0;
 for i1:=1 to 8 do r2[i1]:=0.0;
```

```
 for j:=1 to 2 do begin
if ja[i]<>0 then a[j]:=q[ja[i],j]/z;if jb[i]<>0 then b[j]:=q[jb[i],j]/z;
if jc[i]<>0 then c[j]:=q[jc[i],j]/z;if jd[i]<>0 then d[j]:=q[jd[i],j]/z;
if je[i]<>0 then e[j]:=q[je[i],j]/z;end;
for j:=3 to 6 do begin
if ja[i]<>0 then a[j]:=q[ja[i],j];if jb[i]<>0 then b[j]:=q[jb[i],j];
if jc[i]<>0 then c[j]:=q[jc[i],j];if jd[i]<>0 then d[j]:=q[jd[i],j];
if je[i]<>0 then e[j]:=q[je[i],j];end;
if (jg[i]=2) or (jg[i]=3) then begin
 s1[1]:=sab[i,1]/z;
 s1[2]:=sab[i,2]*u;
 s1[3]:=sab[i,3];
 s1[4]:=sab[i,4];
 s2[1]:=scb[i,1]/z;
 s2[2]:=scb[i,2]*u;
 s2[4]:=scb[i,4];
 if jg[i]=2 then s2[3]:=scb[i,3] else s2[3]:=scb[i,3]/z;
                          end;
  if i<n then begin
                    for k:=i+1 to n do begin
   if (ja[i]=ja[k]) or (ja[i]=jb[k]) or (ja[i]=jc[k]) then   begin
                        abm:=smr;pxa:=-r[ja[k],1]+px;
                        pya:=-r[ja[k],2]+py;
                        cbm:=0.0;pxc:=0.0;pyc:=0.0;     end;
   if (jc[i]=ja[k]) or (jc[i]=jb[k]) or (jc[i]=jc[k]) then   begin
                        cbm:=smr;pxc:=-r[jc[k],1]+px;
                        pyc:=-r[jc[k],2]+py;
                        abm:=0.0;pxa:=0.0;pya:=0.0;     end;
                              end;
```

```
            end;
if jg[i]=2 then begin
px:=0.0;py:=0.0;smr:=0.0;
fdii(a,b,c,s1,s2,pxa,pya,abm,pxc,pyc,cbm,r1);
for l1:=1 to 2 do begin
r[ja[i],l1]:=r1[l1];
r[jb[i],l1]:=r1[l1+2];
r[jc[i],l1]:=r1[l1+4];
              end;
             end;
if jg[i]=3 then begin
px:=0.0;py:=0.0;smr:=0.0;
fdiii(a,b,c,d,e,s1,s2,pxa,pya,abm,pxb,pyb,cbm,r2);
for l1:=1 to 2 do begin
r[ja[i],l1]:=r2[l1];
r[jb[i],l1]:=r2[l1+2];
r[jd[i],l1]:=r2[l1+4];
r[je[i],l1]:=r2[l1+6];
              end;pxb:=0.0;pyb:=0.0;
             end;
if jg[i]=5 then begin
  if (i=n) and (cila<>0) then begin px:=cos(uvn)*fvn[1+trunc(g1[1]/10.0)];
                    py:=sin(uvn)*fvn[1+trunc(g1[1]/10.0)];
                    smr:=pm(px,py,b,a);
                        end;
  if i<n then begin for k:=i+1 to n do begin
                   if jb[i]=ja[k] then begin px:=-r[ja[k],1];
                                       py:=-r[ja[k],2];end;
                   if jb[i]=jc[k] then begin px:=-r[jc[k],1];
```

```
                                    py:=-r[jc[k],2];end;
                                    end;
                            smr:=pm(px,py,b,a);
          end;
  pxa:=0.0;pya:=0.0;pxc:=0.0;pyc:=0.0;
           end;
  if jg[i]=6 then begin
  if (i=n) and (cila<>0) then begin pxb:=cos(uvn)*fvn[1+trunc(g1[1]/10.0)];
                  pyb:=sin(uvn)*fvn[1+trunc(g1[1]/10.0)];
                     end;
              end;
              end;{dinamika}
if cia=1 then begin for l:=1 to Nprin do begin
      r[prinK[l],1]:=r[prinK[l],1]*random(15);
      r[prinK[l],2]:=r[prinK[l],2]*random(15);end;end;
for l:=1 to Nprin do
begin
      ii:=trunc((r[prinK[l],1])/mgr)+160;
      jj:=trunc((r[prinK[l],2])/mgr)+100;
      plot(ii,200-jj,3);
end;
for i:=1 to n do begin if jg[i]=1 then begin if p[i,3]<>0 then
g1[i]:=g1[i]+Hag*p[i,3]/(abs(p[i,3]));end;end;delay(mult*50);gotoxy(2,3);
if abs(g1[1])>=360 then g1[1]:=0.0;
writeln('Угол=',g1[1]:6:1);
if Nprin=1 then begin gotoxy(30,2);writeln('x=',r[prinK[1],1]:7:2);
                gotoxy(30,3);writeln('y=',r[prinK[1],2]:7:2);
                end;
if Nprin=2 then begin gotoxy(30,2);writeln('x1=',r[prinK[1],1]:7:2);
```

```
                gotoxy(30,3);writeln('y1=',r[prinK[1],2]:7:2);
                gotoxy(29,23);writeln('x2=',r[prinK[2],1]:7:2);
                gotoxy(29,24);writeln('y2=',r[prinK[2],2]:7:2);
                end;
read(kbd,ch);m:=m+1;until (m>Khag) or (ch=#115);
gotoxy(30,2);writeln('x=       ');gotoxy(30,3);writeln('y=       ');
gotoxy(29,23);writeln('Xx=       ');gotoxy(29,24);writeln('Yy=       ');
ii:=160;jj:=100;
cv:=getdotcolor(ii,200-jj);
x:=0;y:=0;eoj:=false;cont:=2;
while not eoj do begin read(kbd,ch);
z0:=0;case ch of
#56:begin x:=0;y:=sp;end;
#50:begin x:=0;y:=-sp;end;
#52:begin x:=-sp;y:=0;end;
#54:begin x:=sp;y:=0;end;
#57:begin x:=sp;y:=sp;end;
#51:begin x:=sp;y:=-sp;end;
#55:begin x:=-sp;y:=sp;end;
#49:begin x:=-sp;y:=-sp;end;
#43:begin draw(ii,(200-jj),ii+40,(200-jj),2);
        draw(ii,(200-jj),ii,(200-jj-40),2);end;
#48:begin
       x1:=(ii-160)*mgr+x0;y1:=(jj-100)*mgr+y0;
       z0:=1;
   end;
#53:begin
       x2:=(ii-160)*mgr+x0;y2:=(jj-100)*mgr+y0;
       z0:=2;
```

```
   end;
#109:sp:=1;#98:sp:=5;
#118:begin cont:=cont+1;end;
#112:goto 40;#113:palette(2);#119:palette(3);
#110:begin eoj:=true;end;end;
if(ch<#49)or(ch=#53)or(ch>#57)
then ch:=ch else begin if (cont mod 2)=0 then plot(ii,(200-jj),cv);
ii:=ii+x;jj:=jj+y;
if ii<5 then ii:=5;if ii>310 then ii:=310;
if jj<7 then jj:=7;if jj>191 then jj:=191;cv:=getdotcolor(ii,200-jj);end;
if z0=0 then plot(ii,(200-jj),3);
                gotoxy(32,2);writeln(((ii-160)*mgr):7:2);
                gotoxy(32,3);writeln(((jj-100)*mgr):7:2);
if z0=2 then begin gotoxy(32,23);writeln((x2-x1):7:2);
              gotoxy(32,24);writeln((y2-y1):7:2);end;
end;{while}
end;
textmode;
    until menuKin>2;{ }
if menuK=7 then begin
            model;
            end;
  until menuK>7;{ };
   program culak(input,output);
  {$i graph.p}
  {$i intg }
  {$i rea }
  label 40;
  const
```

```
    maxnumber = 361;
  type
    stroka=record
             number:integer;
             s:real;
          end;
  var
    strokafile:file of stroka;
    strokarec:stroka;
    nameF:string[14];
    s0,s1,sa,sb,sc,mgr,sg,x1,x2,y1,y2
    :real;
    i,filed,n,menuK,j,x,y,i0,fnew,menuKin,u1,u2,Khag,ACPU,page,
    ia,ib,ic,ii,jj,cv,cont,z0,sp
    :integer;
    ch:char;
    OK,eoj:boolean;
  begin
  n:=0;menuK:=1;menuKin:=2;u1:=0;u2:=1;Khag:=1;ACPU:=0;page:=0;
  sp:=1;
repeat
  textbackground(5);window(1,1,80,25);clrscr;
  textbackground(0);window(13,8,72,16);clrscr;
  textbackground(2);window(11,7,70,15);clrscr;textcolor(1);writeln;
  write('          : ');
  textbackground(7);textcolor(0);writeln(' ',n:2,' ');
  textbackground(2);textcolor(1);
  writeln('                    );
  writeln(.');
```

```
 writeln('         .');
 writeln(.');
   write('         4.');textbackground(4);textcolor(7);
 writeln(.');textbackground(2);textcolor(1);
   write('                    : ');
 textbackground(7);window(12,9,18,13);clrscr;textcolor(4);
 write(' актив.');for i:=1 to menuK-1 do writeln;textcolor(0+16);
 write(' ---->');
 textbackground(2);window(50,13,70,14);clrscr;textbackground(7);
 textcolor(0);gotoxy(1,2);write('   ');gotoxy(2,2);menuK:=intg(menuK);
 if menuK=1 then
              begin
 repeat
textbackground(5);window(1,1,80,25);clrscr;
textbackground(0);window(22,11,62,15);clrscr;
textbackground(2);window(20,10,60,14);clrscr;textcolor(1);
writeln('         _____________________  ');
writeln('         ');
writeln(');
  write(: ');textbackground(7);textcolor(0);
  write('             ');
gotoxy(23,4);readln(nameF);
assign(strokafile,nameF);
{$I-} reset(strokafile);{$I+}
OK :=(IOresult=0);
if not OK then filed:=1 else begin textbackground(5);
window(1,1,80,25);clrscr;textbackground(0);window(27,11,57,15);clrscr;
textbackground(4);window(25,10,55,14);clrscr;textcolor(7);
writeln;
```

```
writeln('        Имя файла данных  ');
 write('   ');textcolor(0+16);writeln('П ?');
textcolor(7);
write('    (: ');filed:=0;
write(filed:1);gotoxy(wherex-1,wherey);
filed:=intg(filed);end;
until filed>0;
 begin
 rewrite(strokafile);
  with strokarec do
   begin
     s:=-1.0;
     for i:=1 to maxnumber do
                        begin
                          number:=i;
                          write(strokafile,strokarec);
                        end;
   end;
 close(strokafile);
 end;
 i:=0;i0:=0;
 clrscr;
 assign(strokafile,nameF);reset(strokafile);
 textbackground(5);window(1,1,80,25);clrscr;
 textbackground(0);
 window(9,7,74,23);clrscr;textbackground(3);textcolor(0);
 window(7,6,72,22);clrscr;
 write('  ');for j:=1 to 62 do write('_');write(' ');
 for j:=2 to 16 do begin gotoxy(2,j);write('|');gotoxy(65,j);write('|');end;
```

```
gotoxy(3,16);for j:=1 to 62 do write('_');
gotoxy(20,15);textbackground(7);textcolor(1);
writeln(' - <ENTER>');textbackground(1);textcolor(7);
gotoxy(18,3);
writeln('  ');
textbackground(3);textcolor(1);
gotoxy(10,7);
write('1. ');textbackground(4);textcolor(0);
write('(-1 -)');textbackground(3);textcolor(1);write('  : ');
textbackground(7);textcolor(1);
x:=wherex;y:=wherey;
write(i:3);gotoxy(x,y);textcolor(0);
i:=intg(i);textbackground(3);textcolor(1);
while (i>-1) and (i<maxnumber) do
 begin
  seek(strokafile,i);
  read(strokafile,strokarec);
  with strokarec do
   begin
     gotoxy(10,10);textbackground(3);textcolor(1);
     write('2.     : ');
     textbackground(7);textcolor(1);
     x:=wherex;y:=wherey;
     write(s:6:2);
     gotoxy(x,y);textcolor(0);
     s:=rea(s);s1:=s;
     if i>i0+1 then
                 begin  }
                   fnew:=0;
```

```
                gotoxy(15,12);textbackground(1);textcolor(7);
                write(' ');
                gotoxy(15,13);
                write('  угла от ',i0:3,' до ',i:3,' (Да-1/Нет-0)  ?  : ');
                x:=wherex;y:=wherey;
                write(fnew:1);
                gotoxy(x,y);
                fnew:=intg(fnew);
                textbackground(3);
                gotoxy(15,12);write('                                    ');
                gotoxy(15,13);write('                                    ');
                if fnew>0 then
                begin
                 for j:=i0 to i do
                          begin
                           number:=j;
                           s:=(s1-s0)/(i-i0)*(j-i0)+s0;
                           seek(strokafile,j);
                           write(strokafile,strokarec);
                          end;
                 end;
               end;
    number:=i;
    i0:=i;s0:=s;
  end;
 seek(strokafile,i);
 write(strokafile,strokarec);
textbackground(3);textcolor(1);
gotoxy(10,7);
```

```
 write('1. ');textbackground(4);textcolor(0);
 write('(-)');textbackground(3);textcolor(1);write('  : ');
 textbackground(7);textcolor(1);
 x:=wherex;y:=wherey;
 write('   ');gotoxy(x,y);textcolor(0);
 i:=intg(i);textbackground(3);textcolor(1);
  end;
 close(strokafile);
              end;{конец menuK=1}
 if menuK=2 then
              begin
 repeat
textbackground(5);window(1,1,80,25);clrscr;
textbackground(0);window(22,11,62,15);clrscr;
textbackground(2);window(20,10,60,14);clrscr;textcolor(1);
writeln('        _____________________________  ');
writeln('          ');
writeln(');
  write('         : ');textbackground(7);textcolor(0);
  write('              ');
gotoxy(23,4);readln(nameF);
assign(strokafile,nameF);
{$I-} reset(strokafile);{$I+}
OK :=(IOresult=0);
if not OK then  begin
textbackground(0);window(35,14,65,18);clrscr;
textbackground(4);window(33,13,63,17);clrscr;textcolor(7);
writeln;
writeln('    );
```

```
write(- ');textcolor(0+16);writeln(!');
textcolor(7);
write('   <ENTER>');read(kbd,ch);
              end;until OK;
 i:=0;i0:=0;
 clrscr;
 assign(strokafile,nameF);reset(strokafile);
 textbackground(5);window(1,1,80,25);clrscr;
 textbackground(0);
 window(9,7,74,23);clrscr;textbackground(3);textcolor(0);
 window(7,6,72,22);clrscr;
 write('  ');for j:=1 to 62 do write('_');write(' ');
 for j:=2 to 16 do begin gotoxy(2,j);write('|');gotoxy(65,j);write('|');end;
 gotoxy(3,16);for j:=1 to 62 do write('_');
 gotoxy(20,15);textbackground(7);textcolor(1);
 writeln(' - <ENTER>');textbackground(1);textcolor(7);
 gotoxy(18,3);
 writeln('  ');
 textbackground(3);textcolor(1);
 gotoxy(10,7);
 write('1. ');textbackground(4);textcolor(0);
 write('(-1)');textbackground(3);textcolor(1);write('  : ');
 textbackground(7);textcolor(1);
 x:=wherex;y:=wherey;
 write(i:3);gotoxy(x,y);textcolor(0);
 i:=intg(i);textbackground(3);textcolor(1);
 while (i>-1) and (i<maxnumber) do
  begin
   seek(strokafile,i);
```

```
read(strokafile,strokarec);
with strokarec do
 begin
   gotoxy(10,10);textbackground(3);textcolor(1);
   write('2.: ');
   textbackground(7);textcolor(1);
   x:=wherex;y:=wherey;
   write(s:6:2);
   gotoxy(x,y);textcolor(0);
   s:=rea(s);s1:=s;
   if i>i0+1 then
              begin  { }
                fnew:=0;
                gotoxy(15,12);textbackground(1);textcolor(7);
                write(' ');
                gotoxy(15,13);
                write('  угла от ',i0:3,' до ',i:3,' (Да-1/Нет-0)  ?  : ');
                x:=wherex;y:=wherey;
                write(fnew:1);
                gotoxy(x,y);
                fnew:=intg(fnew);
                textbackground(3);
                gotoxy(15,12);write('                                    ');
                gotoxy(15,13);write('                                    ');
                if fnew>0 then
                begin
                 for j:=i0 to i do
                            begin
                             number:=j;
```

```
                                s:=(s1-s0)/(i-i0)*(j-i0)+s0;
                                seek(strokafile,j);
                                write(strokafile,strokarec);
                               end;
                    end;
                   end;
      number:=i;
      i0:=i;s0:=s;
    end;
   seek(strokafile,i);
   write(strokafile,strokarec);
 textbackground(3);textcolor(1);
 gotoxy(10,7);
 write('1. ');textbackground(4);textcolor(0);
 write(')');textbackground(3);textcolor(1);write('  : ');
 textbackground(7);textcolor(1);
 x:=wherex;y:=wherey;
 write('   ');gotoxy(x,y);textcolor(0);
 i:=intg(i);textbackground(3);textcolor(1);
  end;
 close(strokafile);
              end;{конец menuK=2}
 if menuK=3 then
           begin
              repeat
textbackground(5);window(1,1,80,25);clrscr;
textbackground(0);window(22,11,62,15);clrscr;
textbackground(2);window(20,10,60,14);clrscr;textcolor(1);
writeln('        _____  ');
```

```
writeln('          ');
writeln(');
  write(: ');textbackground(7);textcolor(0);
  write('              ');
gotoxy(23,4);readln(nameF);
assign(strokafile,nameF);
{$I-} reset(strokafile);{$I+}
OK :=(IOresult=0);
if not OK then  begin
textbackground(0);window(35,14,65,18);clrscr;
textbackground(4);window(33,13,63,17);clrscr;textcolor(7);
writeln;
writeln('    );
write(- ');textcolor(0+16);writeln(' !');
textcolor(7);
write('   <ENTER>');read(kbd,ch);
              end;until OK;
               i:=0;
               repeat
               textbackground(5);window(1,1,80,25);clrscr;
 textbackground(0);window(13,8,72,14);clrscr;
 textbackground(1);window(11,7,70,13);clrscr;textcolor(7);writeln;
 writeln(:  ');
 writeln(.');
 writeln(.');
   write('         3.');textbackground(4);textcolor(7);
 writeln('.');textbackground(1);textcolor(7);
   write('                      : ');
 textbackground(7);window(12,8,18,12);clrscr;textcolor(4);
```

```
   write(' ');for i:=1 to menuKin-1 do writeln;textcolor(0+16);
   write(' ---->');
   textbackground(1);window(50,11,70,13);clrscr;textbackground(7);
   textcolor(0);gotoxy(1,2);write('   ');gotoxy(2,2);menuKin:=intg(menuKin);
         if menuKin<3 then
                        begin
textbackground(0);window(9,12,74,23);clrscr;
textbackground(7);textcolor(0);
 window(7,11,72,22);clrscr;
write('  ');for i:=1 to 62 do write('_');write(' ');
for i:=2 to 11 do begin gotoxy(2,i);write('|');gotoxy(65,i);write('|');end;
gotoxy(3,11);for i:=1 to 62 do write('_');
gotoxy(20,10);textbackground(1);textcolor(7);
writeln(' Нет корректировки - <ENTER>');textbackground(7);textcolor(1);
gotoxy( (1);textcolor(7);
writeln('   ИСХОДНЫЕ ДАННЫЕ    ');writeln;
textbackground(7);textcolor(1);
gotoxy(8,5);write('1. НАЧАЛЬНОЕ значение угла радиус-вектора   : ');
writeln(u1:3);   gotoxy(8,6);write('2: ');
writeln(u2:3);
gotoxy(8,7);write('3. ');textcolor(0);write('ШАГА');
textcolor(1);write('               : ');
writeln(Khag:3);textcolor(1);
gotoxy(8,8);write('4 ');textcolor(0);
write();
textcolor(1);write('     : ');
writeln(ACPU:2);textbackground(7);textcolor(0);
gotoxy(54,5);u1:=intg(u1);textbackground(7);textcolor(0);
gotoxy(54,6);u2:=intg(u2);textbackground(7);textcolor(0);
```

```
 gotoxy(54,7);Khag:=intg(Khag);textbackground(7);textcolor(0);
 gotoxy(55,8);ACPU:=intg(ACPU);
                          end;
       if menuKin=1 then
                    begin
 textbackground(5);window(1,1,80,25);clrscr;
 textbackground(2);window(9,2,75,24);clrscr;textbackground(1);textcolor(7);
 window(7,1,73,6);clrscr;
 write('  _______________');textbackground(7);textcolor(0);
 write(' ');textbackground(1);
 textcolor(7);writeln('_______________');
writeln(' | |');
writeln(' | |');
writeln(' |');
writeln(')  |');
  write(' |-----------------|    |-----------------|    |-----------------|');
if ACPU=1 then begin
writeln(lst,'                ');
writeln(lst,'  _____________________    _____________________
_____________________');
writeln(lst,' | |');
writeln(lst,' | - |');
writeln(lst,' |);
writeln(lst,' | ');
writeln(lst,' |-----------------|    |-----------------|    |-----------------|');
end;
textbackground(1);window(7,7,73,23);clrscr;textcolor(7);
     assign(strokafile,nameF);
     reset(strokafile);
```

```
  i:=u1;
repeat
  ia:=0;ib:=0;ic:=0;sa:=0;sb:=0;sc:=0;

  if (i>-1) and (i<maxnumber) then
  begin
   seek(strokafile,i);
   read(strokafile,strokarec);
   with strokarec do
                  begin
                    ia:=i;sa:=s;
                  end;
  end;
  i:=i+Khag;
  if (i>-1) and (i<maxnumber) then
  begin
   seek(strokafile,i);
   read(strokafile,strokarec);
   with strokarec do
                  begin
                    ib:=i;sb:=s;
                  end;
  end;
  i:=i+Khag;
  if (i>-1) and (i<maxnumber) then
  begin
   seek(strokafile,i);
   read(strokafile,strokarec);
   with strokarec do
```

```
                    begin
                      ic:=i;sc:=s;
                    end;
      end;
         begin
  writeln(' |  ',ia:3,'  | ',sa:6:2,'  |','    |  ',ib:3,'  | ',sb:6:2,'  |','    |  ',ic:3,'  | ',sc:6:2,'  |');
         end;
  if ACPU=1 then begin
  writeln(lst,' |  ',ia:3,'  | ',sa:6:2,'  |','    |  ',ib:3,'  | ',sb:6:2,'  |','    |  ',ic:3,'  | ',sc:6:2,'
|');
           end;
  page:=page+1;if page>15 then begin
page:=0;textbackground(3);textcolor(0+16);
  write(' <ENTER> ');textbackground(1);textcolor(7);readln;
                        end;
    i:=Khag+i;
    until i>u2;
    close(strokafile);
  textbackground(3);
  textcolor(0+16);write(' <ENTER> ');textbackground(1);textcolor(7);
  read(kbd,ch);page:=0;
   if ACPU=1 then begin write(lst,' |');for i:=1 to 63 do write(lst,'_');
               writeln(lst,'|');
             end;
                    end;
       if menuKin=2 then
                    begin
                    sg:=0;
                    assign(strokafile,nameF);
```

```
                    reset(strokafile);
                     i:=0;
                     while (i>-1) and (i<maxnumber) do
                    begin
                     seek(strokafile,i);
                     read(strokafile,strokarec);
                     with strokarec do
                                  begin
                                   sa:=s;
                                  end;
                        if sg<sa then sg:=sa;
                        i:=i+1;
                    end;
                    close(strokafile);
               textbackground(0);window(27,11,57,15);clrscr;
  textbackground(2);window(25,10,55,14);clrscr;textcolor(0);
  writeln;
  writeln('              ');
  writeln('    ');
    write('          : ');
  write((sg*2):6:2);gotoxy(wherex-6,wherey);
  sg:=rea(sg);
    textbackground(0);window(1,1,80,25);clrscr;
textmode;clrscr;
graphcolormode;clearscreen;
palette(3);graphbackground(1);
clearscreen;mgr:=2*sg/200.;
draw(0,0,319,0,1);draw(319,0,319,199,1);
draw(319,199,0,199,1);draw(0,199,0,0,1);
```

```
draw(3,3,316,3,1);draw(316,3,316,196,1);
draw(316,196,3,196,1);draw(3,196,3,3,1);
gotoxy(2,2);writeln();
gotoxy(19,2);writeln('Y');gotoxy(39,14);writeln('X');
draw(160,10,160,189,2);draw(10,100,306,100,2);
draw(157,13,160,10,2);draw(163,13,160,10,2);
draw(303,103,306,100,2);draw(303,97,306,100,2);
    40:assign(strokafile,nameF);
      reset(strokafile);
       i:=u1;
       while ((i>=u1) and (i<=u2)) do
           begin
            seek(strokafile,i);
            read(strokafile,strokarec);
            with strokarec do
                        begin
                           sa:=s;
                        end;
            ii:=trunc(sa*cos(i*3.1415926/180.0)/mgr)+160;
            jj:=trunc(sa*sin(i*3.1415926/180.0)/mgr)+100;
            plot(ii,200-jj,3);
            gotoxy(2,3);writeln('Угол=',i:3);
            gotoxy(30,2);writeln('x=',(sa*cos(i*3.1415926/180.0)):7:2);
            gotoxy(30,3);writeln('y=',(sa*sin(i*3.1415926/180.0)):7:2);
            gotoxy(2,24);writeln('R=',sa:6:2);
            read(kbd,ch);if ch=#115 then begin u1:=i;i:=u2+1;end
                              else i:=i+Khag;
           end;
           close(strokafile);
```

```
gotoxy(30,2);writeln('x=       ');gotoxy(30,3);writeln('y=       ');
gotoxy(2,24);writeln('R=       ');
gotoxy(29,23);writeln('Xx=       ');gotoxy(29,24);writeln('Yy=       ');
ii:=160;jj:=100;
cv:=getdotcolor(ii,200-jj);
x:=0;y:=0;eoj:=false;cont:=2;
while not eoj do begin read(kbd,ch);
z0:=0;case ch of
#56:begin x:=0;y:=sp;end;
#50:begin x:=0;y:=-sp;end;
#52:begin x:=-sp;y:=0;end;
#54:begin x:=sp;y:=0;end;
#57:begin x:=sp;y:=sp;end;
#51:begin x:=sp;y:=-sp;end;
#55:begin x:=-sp;y:=sp;end;
#49:begin x:=-sp;y:=-sp;end;
#43:begin draw(ii,(200-jj),ii+40,(200-jj),2);
        draw(ii,(200-jj),ii,(200-jj-40),2);end;
#48:begin x1:=(ii-160)*mgr;y1:=(jj-100)*mgr;z0:=1;end;
#53:begin x2:=(ii-160)*mgr;y2:=(jj-100)*mgr;z0:=2;end;
#109:sp:=1;#98:sp:=5;
#118:begin cont:=cont+1;end;
#112:goto 40;
#113:palette(2);#119:palette(3);
#110:begin eoj:=true;end;end;
if(ch<#49)or(ch=#53)or(ch>#57)
then ch:=ch else begin if (cont mod 2)=0 then plot(ii,(200-jj),cv);
ii:=ii+x;jj:=jj+y;
```

```
if ii<5 then ii:=5;if ii>310 then ii:=310;
if jj<7 then jj:=7;if jj>191 then jj:=191;cv:=getdotcolor(ii,200-jj);end;
if z0=0 then plot(ii,(200-jj),3);
                gotoxy(32,2);writeln(((ii-160)*mgr):7:2);
                gotoxy(32,3);writeln(((jj-100)*mgr):7:2);
                gotoxy(5,24);writeln((sqrt(
                ((ii-160)*mgr)*((ii-160)*mgr)+
                ((jj-100)*mgr)*((jj-100)*mgr))):6:2);
if z0=2 then begin gotoxy(32,23);writeln((x2-x1):7:2);
              gotoxy(32,24);writeln((y2-y1):7:2);end;
end;{while}
textmode;
                                    until menuKin>2;
            end;
  until menuK>3;
            textbackground(5);window(1,1,80,25);clrscr;
            for i:=20 to 59 do begin delay(50);
            textbackground(0);window(i+2,11,i+3,15);clrscr;
            textbackground(2);window(i,10,i+1,14);clrscr;
                        end;
            textbackground(2);window(20,10,60,14);clrscr;
            textcolor(1);
            writeln('       _____  ');
            writeln('           ');
            writeln(');
              write('         : ');textbackground(7);textcolor(0+16);
              write('   NEW.COM     ');
              delay(2500);
    end.
```

ЛІСТИНГ ПРОГРАМ ПРИКЛАДНИХ ПРОГРАМ

Програма PUG для визначення кутів для роторного натягувача нитки

```
unit UG1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;
type
  TfrmUG1 = class(TForm)
    lbl1UG1: TLabel;
    lbl2UG1: TLabel;
    lbl3UG1: TLabel;
    btn1UG1: TButton;
    img1UG1: TImage;
    lbl4UG1: TLabel;
    lbl5UG1: TLabel;
    procedure btn1UG1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmUG1: TfrmUG1;
implementation
uses UG2;
{$R *.dfm}
procedure TfrmUG1.btn1UG1Click(Sender: TObject);
```

```
begin
frmUG1.Hide;
frmUG2.Show;
end;
end.
unit UG2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,Math,Menus, TeEngine, Series, ExtCtrls, TeeProcs, Chart, StdCtrls,
  Printers;
type
  TfrmUG2 = class(TForm)
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    lbl1UG2: TLabel;
    Memo1: TMemo;
    mem1UG2: TMemo;
    lbl2UG2: TLabel;
    chr1UG2: TChart;
```

```
    lbl3UG2: TLabel;
    lbl4UG2: TLabel;
    Series1: TLineSeries;
    chr2UG2: TChart;
    Series2: TLineSeries;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    N12: TMenuItem;
    N13: TMenuItem;
    N11: TMenuItem;
    procedure N5Click(Sender: TObject);
    procedure N6Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure N8Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure N12Click(Sender: TObject);
    procedure N13Click(Sender: TObject);
    procedure N11Click(Sender: TObject);
      private
    { Private declarations }
  public
    { Public declarations }
  end;
  type xf=array[1..50]of Real;
var
  frmUG2: TfrmUG2;
  s1,RR,i,j,k,ug:integer;
  h,x1,g,Fi:Real;
```

```
  A:array[1..50,1..50] of Real;
  B:array[1..50]of Real;
  F,X:xf;
  x0,xx:array[1..50] of string;
  ii,ss,xxx,ugi,LL1:string;
  ugol1,ugol2:array[1..50]of Real;
  h1,d1,h2,d2,L1:array[1..50]of Real;
implementation
  uses UG1,Synt,UErrors,UG3,UG4,UG5;
{$R *.dfm}
 procedure v(var F:xf; X:xf;r4,r2,r1,R,L,Fi:Real);
       var i:Integer;
    begin
        SetData('r4',r4);
        SetData('r2',r2);
        SetData('r1',r1);
        SetData('R',R);
        SetData('L',L);
        SetData('Fi',Fi);
        SetData('X[1]',X[1]);
        SetData('X[2]',X[2]);
        SetData('X[3]',X[3]);
        SetData('X[4]',X[4]);
        SetData('X[5]',X[5]);
        SetData('X[6]',X[6]);
        SetData('X[7]',X[7]);
        SetData('X[8]',X[8]);
        SetData('X[9]',X[9]);
        SetData('X[10]',X[10]);
```

```
 SetData('X[11]',X[11]);
 SetData('X[12]',X[12]);
 SetData('X[13]',X[13]);
 SetData('X[14]',X[14]);
 SetData('X[15]',X[15]);
 SetData('X[16]',X[16]);
 SetData('X[17]',X[17]);
 SetData('X[18]',X[18]);
 SetData('X[19]',X[19]);
 SetData('X[20]',X[20]);
 SetData('X[21]',X[21]);
 SetData('X[22]',X[22]);
 SetData('X[23]',X[23]);
 SetData('X[24]',X[24]);
 for i:=1 to N do
  begin
Calculate(F[i]);
  end;
 GetData('F[1]',F[1]);
 GetData('F[2]',F[2]);
 GetData('F[3]',F[3]);
 GetData('F[4]',F[4]);
 GetData('F[5]',F[5]);
 GetData('F[6]',F[6]);
 GetData('F[7]',F[7]);
 GetData('F[8]',F[8]);
 GetData('F[9]',F[9]);
 GetData('F[10]',F[10]);
 GetData('F[11]',F[11]);
```

```
    GetData('F[12]',F[12]);
    GetData('F[13]',F[13]);
    GetData('F[14]',F[14]);
    GetData('F[15]',F[15]);
    GetData('F[16]',F[16]);
    GetData('F[17]',F[17]);
    GetData('F[18]',F[18]);
    GetData('F[19]',F[19]);
    GetData('F[20]',F[20]);
    GetData('F[21]',F[21]);
    GetData('F[22]',F[22]);
    GetData('F[23]',F[23]);
    GetData('F[24]',F[24]);
  end;
procedure TfrmUG2.N5Click(Sender: TObject);
begin
frmUG1.Close;
end;
procedure TfrmUG2.N6Click(Sender: TObject);
begin
frmUG5.Show;
frmUG3.show;
end;
procedure TfrmUG2.N7Click(Sender: TObject);
begin
frmUG4.Show;
end;
procedure TfrmUG2.N8Click(Sender: TObject);
begin
```

```
  s1:=0;
    for i:=1 to N do
    begin
    ii:= format('%2.0d',[i]);
    mem1UG2.Lines.Add(+ii+'-го корня');
    x0[i]:=InputBox(X0(i)','Значение корня X0('+ii+')=','');
    Val(x0[i],x[i],code);
    xx[i]:=format('%12.5f',[x[i]]);
    mem1UG2.Lines.Add('x('+ii+')='+xx[i]);
    end;
    for ug:=1 to 19 do
begin
   Fi:=10*(ug-1)*Pi/180;
      //
  if (FErrors <> nil) then FErrors.Close;
  //
  if not CreatePZ(Memo1.Text)
  then
  begin
  //
  Application.CreateForm(TFErrors, FErrors);
  FErrors.LBErrors.Items.Assign(ErrorList);
  FErrors.Show;
  exit;
  end;
      repeat
          v(F,X,r4,r2,r1,R,L,Fi);
          for i:=1 to N do
           begin
```

```
 B[i]:=-F[i];
 end;
for j:=1 to N do
 begin
 x1:=x[j]; h:=e*abs(x1);
 x[j]:=x1+h;
   v(F,X,r4,r2,r1,R,L,Fi);
 for i:=1 to N do
   begin
   A[i,j]:=(F[i]+B[i])/h;
   end;
 x[j]:=x1;
 end;
  s1:=s1+1;
    if s1=M+1 then
          begin
          ss:=format('%3.0d',[s1]);
          mem1UG2.Lines.Add('Число итераций s='+ss);
          Break;
          end;
 for i:=1 to N-1 do
   begin
     for j:=i+1 to N do
       begin
       A[j,i]:=-A[j,i]/A[i,i];
         for k:=i+1 to N do
           begin
             A[j,k]:=A[j,k]+A[j,i]*A[i,k];
           end;
```

```
        B[j]:=B[j]+A[j,i]*B[i];
        end;
    end;
  F[N]:=B[N]/A[N,N];
    for i:=N-1 downto 1 do
        begin
        h:=B[i];
          for j:=i+1 to N do
             begin
             h:=h-F[j]*A[i,j];
             end;
        F[i]:=h/A[i,i];
        end;
   RR:=0;
     for i:=1 to N do
         begin
         x[i]:=x[i]+F[i];
           if abs(F[i]/x[i])>e then RR:=1;
         end;
     if RR=1 then Continue;
     mem1UG2.Lines.Add();
        ugi:=format('%4.0d',[(ug-1)*10]);
        mem1UG2.Lines.Add(='+ugi);
      for i:=1 to N do
       begin
       ii:= format('%2.0d',[i]);
       xxx:=format('%11.5f',[x[i]]);
       mem1UG2.Lines.Add('x('+ii+')='+xxx);
       end;
```

```
                ugol1[ug]:=x[1];
                ugol2[ug]:=x[2];
              h1[ug]:=R*sin(Fi)+r1*cos(ugol1[ug])-r4-r2*(1-cos(ugol1[ug]));
              d1[ug]:=L-R*cos(Fi)-sin(ugol1[ug])*(r1+r2);
              h2[ug]:=R*sin(Fi)+r1*cos(ugol2[ug])-r4-r2*(1-cos(ugol2[ug]));
              d2[ug]:=L+R*cos(Fi)-sin(ugol2[ug])*(r1+r2);
1[ug]:=(r1+r2)*(ugol1[ug]+ugol2[ug])+sqrt(h1[ug]*h1[ug]+d1[ug]*d1[ug])+sqr
t(h2[ug]*h2[ug]+d2[ug]*d2[ug])-2*L;
              LL1:=format('%11.5f',[L1[ug]]);
              mem1UG2.Lines.Add('Подача L равна='+LL1);
               ss:=format('%3.0d',[s1]);
               mem1UG2.Lines.Add(s='+ss);
               Break;
       until false;
end;
procedure TfrmUG2.N10Click(Sender: TObject);
begin
 with Series1 do
       begin
        for ug:=1 to 19 do
        begin
       Fi:=10*(ug-1)*Pi/180;
       Series1.AddXY(Fi,L1[ug],",clRed);
       Series2.AddXY(Fi,Ugol1[ug],",clBlue);
       Series3.AddXY(Fi,Ugol2[ug],",clGreen);
       Series4.AddXY(arcsin((r4-r1)/R),L1[ug],",clBlack);
       Series5.AddXY(arcsin((r4-r1)/R),Ugol1[ug],",clBlack);
        end;
       end;
```

```
end;
procedure TfrmUG2.N12Click(Sender: TObject);
    Var
      Prn:TextFile;
      k:Integer;
  begin
      AssignPrn(Prn);
      Rewrite(Prn);
      Printer.Canvas.Font:=mem1UG2.Font;
      for k:=0 to mem1UG2.Lines.Count-1 do
      WriteLn(Prn,mem1UG2.Lines[k]);
      CloseFile(Prn);
 end;
procedure TfrmUG2.N13Click(Sender: TObject);
begin
    chr1UG2.Print;
    chr2UG2.Print;
end;
procedure TfrmUG2.N11Click(Sender: TObject);
begin
mem1UG2.Clear;
end;
end.
unit UG3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
```

```
 TfrmUG3 = class(TForm)
   lbl2UG3: TLabel;
   lbl3UG3: TLabel;
   lbl4UG3: TLabel;
   lbl5UG3: TLabel;
   edt1UG3: TEdit;
   edt2UG3: TEdit;
   edt3UG3: TEdit;
   edt4UG3: TEdit;
   btn1UG3: TButton;
   lbl1UG3: TLabel;
   edt5UG3: TEdit;
   procedure btn1UG3Click(Sender: TObject);
 private
   { Private declarations }
 public
   { Public declarations }
 end;
var
 frmUG3: TfrmUG3;
 r4,r2,r1,R,L:Real;
 code:Integer;
implementation
  uses UG2;
{$R *.dfm}
procedure TfrmUG3.btn1UG3Click(Sender: TObject);
begin
 Val(edt1UG3.Text,r4,code);
 Val(edt2UG3.Text,r2,code);
```

```
 Val(edt3UG3.Text,r1,code);
 Val(edt4UG3.Text,R,code);
 Val(edt5UG3.Text,L,code);
frmUG3.Hide;
end;
end.
unit UG4;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TfrmUG4 = class(TForm)
    lbl1UG4: TLabel;
    lbl2UG4: TLabel;
    lbl3UG4: TLabel;
    btn1UG4: TButton;
    procedure btn1UG4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmUG4: TfrmUG4;
implementation
  uses UG2;
{$R *.dfm}
procedure TfrmUG4.btn1UG4Click(Sender: TObject);
```

```
begin
frmUG4.Close;
end;
end.
unit UG5;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TfrmUG5 = class(TForm)
    lbl1UG5: TLabel;
    lbl2UG5: TLabel;
    lbl3UG5: TLabel;
    edt1UG5: TEdit;
    edt2UG5: TEdit;
    edt3UG5: TEdit;
    btn1UG5: TButton;
    procedure btn1UG5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmUG5: TfrmUG5;
  N,M,code:Integer;
  e:Real;
implementation
```

```
  uses UG2;
{$R *.dfm}
procedure TfrmUG5.btn1UG5Click(Sender: TObject);
begin
     Val(edt1UG5.Text,N,code);
     Val(edt2UG5.Text,M,code);
     Val(edt3UG5.Text,e,code);
  frmUG5.Hide;
end;
end.
unit Synt;
interface
uses classes;
type
 TData = record
   Name: string;
   Data:real;
 end;
var
 NConst: integer = 100;
 ErrorList: TStringList;
 PZ: array of integer;
 DataList: array of TData;
const
 MConst = 2;
procedure SyntItem(S:string; First:boolean=false; Pos:Integer=1);
function CreatePZ(S:string):boolean;
function Calculate(var R:real):boolean;
function SetData(Name:string; Data:real):boolean;
```

```
function GetData(Name:string; var Data:real):boolean;
implementation
uses Sysutils, Math, Dialogs,UG2,UErrors;
type
 TType = (None,  Ident, Func, Part, All);
 TSynt = record
    mode: TType;
    Number:real;
    Ident:string;
    Error:boolean;
    Pos1,Pos2:integer;
 end;
const
    SetNum: set of char=['0'..'9', ','];
    SetDiv: set of char=[';', '(', ')', '=', '+', '-', '/', '*', '^',
                         '{', ' ', #13];
    SetChar: set of char=['a'..'z','A'..'Z','_'];
    NFunc = 11;
    Functions: array[1..NFunc] of string =
      ('exp','sin','cos','sqrt','abs','ln','tg','arctan','arccos','sqr','arcsin');
var
 SItem: TSynt;
 TrStack: array of char;
 ConstList: array of real;
 Position: Integer;
procedure SyntItem(S:string;First:boolean=false;Pos:Integer=1);
var i:integer;
begin
 if (S = '') then begin
```

```
 SItem.mode := All;
 exit;
end;
if(First) then Position := Pos;
repeat
 if (S[Position] = '{')
  then begin
   repeat
    Inc(Position)
   until (Position >= Length(S)) or (S[Position] = '}');
   Inc(Position);
  end;
 if(Position <= Length(S)) then
  while ((S[Position] = ' ')or
        (S[Position] = #13)or
        (S[Position] = #10)or
        (S[Position] = #0))
  do Inc(Position);
until (S[Position] <> '{');
SItem.Error:=false;
SItem.Pos1:=Position;
if(Position > Length(S)) then begin
 SItem.mode := All;
 exit;
end;
SItem.Ident := S[Position];
if (S[Position] in SetChar)
    then SItem.mode := Ident
else if (S[Position] in SetNum)
```

```
   then SItem := Number
else if (S[Position] in SetDiv)
   then begin
    if (S[Position] <> ';')
     then SItem.mode := Divider
     else SItem.mode := Part;
    Inc(Position);
    exit;
   end
else begin
 SItem.mode := None;
 Inc(Position);
 exit;
end;
repeat
 Inc(Position);
 if (SItem.mode = Number)and
   ((S[Position] = '-')or(S[Position] = '+'))and
   (UpCase(S[Position-1])='E')
  then SItem.Ident := SItem.Ident + S[Position]
 else if ((Position > Length(S))or(S[Position] in SetDiv))
  then begin
   if(SItem.mode = Number)
     then try
       SItem.Number := StrToFloat(SItem)
     except
      on EConvertError do SItem.Error := true;
     end;
   for i:=1 to NFunc do
```

```
     if (LowerCase(SItem.Ident) = Functions[i])
       then begin
        SItem.mode:=Func;
        SItem.Number:=i;
        break;
       end;
     SItem.Pos2:=Position-1;
     exit;
    end
    else SItem.Ident := SItem.Ident + S[Position];
 until false;
end;
procedure ClearPZ;
begin
 ErrorList.Clear;
 SetLength(ConstList,0);
 SetLength(DataList,MConst);
 SetLength(PZ,0);
end;
function CreatePZ(S:string):boolean;
var
    lend:boolean;
    i:integer;
    Assign:boolean;
    Adress: integer;
    OldMode: TType;
    OldS: char;
procedure code;
begin
```

```
 SetLength(PZ,High(PZ)+2);
 case TrStack[High(TrStack)] of
  '+': PZ[High(PZ)] := -1;
  '-': PZ[High(PZ)] := -2;
  '*': PZ[High(PZ)] := -3;
  '/': PZ[High(PZ)] := -4;
  '^': PZ[High(PZ)] := -5;
  'M': PZ[High(PZ)] := -6;
 end;
end;
procedure proc1;
begin
 SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc2;
begin
 code;
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc3;
begin
 code;
 SetLength(TrStack,High(TrStack));
 lend:=false;
end;
procedure proc4;
begin
 SetLength(TrStack,High(TrStack));
```

```
end;
procedure proc5;
begin
 SetLength(TrStack,High(TrStack));
 TrStack[High(TrStack)] := Chr(127+Round(SItem.Number));
end;
procedure proc6;
begin
 SetLength(PZ,High(PZ)+2);
 PZ[High(PZ)] := -Ord(TrStack[High(TrStack)]);
 SetLength(TrStack,High(TrStack));
end;
begin
 ClearPZ;
 SetLength(TrStack,1);
 TrStack[0] := '0';
 OldMode := None;
 OldS := ' ';
 Assign := true;
 Adress := 0;
 SyntItem(S,true);
 if (SItem.mode = All)
  then begin
   ErrorList.Add();
   Result := false;
   exit;
  end;
 repeat
  if ((OldMode = Func)and(SItem.Ident[1] <> '('))
```

```
 then ErrorList.Add('+IntToStr(SItem.Pos1));
case SItem.mode of
 Number: begin
         if((OldMode <> Divider)and(OldMode <> None)and
           (OldMode <> Part))
          then ErrorList.Add('+IntToStr(SItem.Pos1)+');
         if (SItem.Error)
          then ErrorList.Add('+IntToStr(SItem.Pos1)+
                          ' - '+IntToStr(SItem.Pos2))
          else begin
           SetLength(ConstList,High(ConstList)+2);
           ConstList[High(ConstList)] := SItem.Number;
           SetLength(PZ,High(PZ)+2);
           PZ[High(PZ)] := High(ConstList);
          end;
         Assign:=false;
        end;
 Ident:  begin
         if((OldMode <> Divider)and(OldMode <> None)and
           (OldMode <> Part))
          then ErrorList.Add(ии '+IntToStr(SItem.Pos1)+' ');
         for i:=0 to High(DataList) do
          begin
           if (UpperCase(SItem.Ident) = DataList[i].Name)
            then begin
             SetLength(PZ,High(PZ));
             PZ[High(PZ)] := NConst+i;
             break;
            end;
```

```
          if(i = High(DataList)) then
           begin
            SetLength(DataList,High(DataList)+2);
            DataList[High(DataList)].Name:=UpperCase(SItem.Ident);
            DataList[High(DataList)].Data:=0;
            SetLength(PZ,High(PZ)+2);
            PZ[High(PZ)] := NConst+High(DataList);
           end;
         end;
        end;
All,Part:begin
        repeat
         lend:=true;
         case TrStack[High(TrStack)] of
          '0': begin
               if (Adress <> 0) then begin
                 SetLength(PZ,High(PZ)+3);
                 PZ[High(PZ)-1] := -7;
                 PZ[High(PZ)] := Adress;
                 Adress := 0;
                end;
               break;
               end;
          '(': ErrorList.Add();
          else proc3;
         end;
        until lend;
        if (ErrorList.Count = 0)
         then Result:=true
```

```
        else Result:=false;
       if (SItem.mode = All) then exit
       else begin
        Assign := true;
        SItem.mode := None;
       end
      end;
Divider:begin
      if((OldMode = Divider)and
       ((SItem.Ident[1]<>'=')and
       (SItem.Ident[1]<>'(')and
       (SItem.Ident[1] <> ')'))and
       ((OldS <> '(')and
       (OldS <> ')')and
       (OldS <> '=')))
       then begin
        ErrorList.Add(+IntToStr(SItem.Pos1)+
                          ':);
        break;
       end;
      repeat
       lend:=true;
       case SItem.Ident[1] of
        '=': if Assign and (OldMode = Ident)
             then begin
              Adress := PZ[High(PZ)];
              SetLength(PZ,High(PZ));
              SItem.mode := None;
             end
```

```
      else ErrorList.Add('Позиция '+IntToStr(SItem.Pos1)+
                         ':');
'(': if(OldMode = Ident) or (OldMode = Number)
     then ErrorList.Add('+IntToStr(SItem.Pos1))
     else  proc1;
'+','-','M': begin
     if((OldMode = None)or(OldS = '('))
      then if (SItem.Ident[1] = '+')
       then break
      else SItem.Ident[1] := 'M';
     case TrStack[High(TrStack)] of
      '0','(': proc1;
      '+','-','M': proc2;
      '*','/','^': proc3;
     end;
     end;
'*','/':
     if OldS = '('
     then ErrorList.Add('+IntToStr(SItem.Pos1))
     else
     case TrStack[High(TrStack)] of
      '0','(','+','-','M': proc1;
      '*','/': proc2;
      '^': proc3;
     end;
'^':
     if OldS = '('
     then ErrorList.Add('+IntToStr(SItem.Pos1))
     else
```

```
            case TrStack[High(TrStack)] of
             '0','(','+','-','*','/','M': proc1;
             '^': proc2;
            end;
         ')':
            case TrStack[High(TrStack)] of
             '0': ErrorList.Add();
             '(': begin
                   proc4;
                   if (Ord(TrStack[High(TrStack)]) > 127)
                    then proc6;
                  end;
             '+','-','*','/','^','M': proc3;
            end;
        end;
       until lend;
       Assign:=false;
       end;
 Func:   begin
       repeat
        lend:=true;
        proc5
       until lend;
       Assign:=false;
       end;
 None:   ErrorList.Add('+IntToStr(SItem.Pos1));
end;
OldMode := SItem.mode;
OldS := SItem.Ident[1];
```

```
 SyntItem(S);
 until false;
 if(ErrorList.Count = 0)
 then Result := true
 else Result := false;
end;
function Calculate(var R:real):boolean;
var Stack: array of real;
    i:integer;
begin
  for i:=0 to High(PZ) do begin
   if (i > 0) then
    if (PZ[i-1] = -7) and (i < High(PZ)) then Continue;
   if PZ[i] < -100
    then begin
     try
      case -PZ[i]-100 of
       1: Stack[High(Stack)]:=Exp(Stack[High(Stack)]);
       2: Stack[High(Stack)]:=Sin(Stack[High(Stack)]);
       3: Stack[High(Stack)]:=Cos(Stack[High(Stack)]);
       4: Stack[High(Stack)]:=Sqrt(Stack[High(Stack)]);
       5: Stack[High(Stack)]:=Abs(Stack[High(Stack)]);
       6: Stack[High(Stack)]:=Ln(Stack[High(Stack)]);
       7: Stack[High(Stack)]:=Tan(Stack[High(Stack)]);
       8: Stack[High(Stack)]:=ArcTan(Stack[High(Stack)]);
       9: Stack[High(Stack)]:=ArcCos(Stack[High(Stack)]);
       10: Stack[High(Stack)]:=Sqr(Stack[High(Stack)]);
       11: Stack[High(Stack)]:=ArcSin(Stack[High(Stack)]);
      end
```

```
 except
  Result := false;
  exit;
 end;
 if(FloatToStr(Stack[High(Stack)]) = 'NA') or
   (FloatToStr(Stack[High(Stack)]) = 'INF') or
   (FloatToStr(Stack[High(Stack)]) = '-INF')
  then begin
   Result := false;
   exit;
  end
end
else if PZ[i] < 0
then begin
try
 case -PZ[i] of
  1: Stack[High(Stack)-1]:=
     Stack[High(Stack)-1]+Stack[High(Stack)];
  2: Stack[High(Stack)-1]:=
     Stack[High(Stack)-1]-Stack[High(Stack)];
  3: Stack[High(Stack)-1]:=
     Stack[High(Stack)-1]*Stack[High(Stack)];
  4: Stack[High(Stack)-1]:=
       Stack[High(Stack)-1]/Stack[High(Stack)];
  5: Stack[High(Stack)-1]:=
       Power(Stack[High(Stack)-1],Stack[High(Stack)]);
  6: Stack[High(Stack)]:= -Stack[High(Stack)];
  7: DataList[PZ[i+2]-NConst].Data := Stack[High(Stack)];
 end;
```

```
   except
    Result := false;
    exit;
   end;
    if (PZ[i] <> -6)
      then SetLength(Stack,High(Stack));
   end
   else begin
    SetLength(Stack,High(Stack));
    if (PZ[i] < NConst)
     then Stack[High(Stack)]:=ConstList[PZ[i]]
     else Stack[High(Stack)]:=DataList[PZ[i]-NConst].Data;
   end;
 end;
 Result := true;
 R :=Stack[High(Stack)];
end;
function SetData(Name:string; Data:real):boolean;
var i:integer;
begin
 for i:=MConst to High(DataList) do
  if (UpperCase(Name) = DataList[i].Name)
   then  begin
    DataList[i].Data := Data;
    Result:=true;
    exit;
   end;
 Result := false;
end;
```

```
function GetData(Name:string; var Data:real):boolean;
var i:integer;
begin
 for i:=0 to High(DataList) do
  if (UpperCase(Name) = DataList[i1].Name)
   then  begin
    Data := DataList[i].Data;
    Result:=true;
    exit;
   end;
 Result := false;
end;
initialization
 SetLength(DataList,MConst);
 DataList[0].Name:='PI';
 DataList[0].Data:=Pi;
 DataList[1].Name:='E';
 DataList[1].Data:=2.71828183;
 ErrorList := TStringList.Create;
finalization
 ErrorList.Free;
end.
unit UErrors;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TFErrors = class(TForm)
```

```
    LBErrors: TListBox;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FErrors: TFErrors;
implementation
{$R *.dfm}
unit RFP1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, jpeg, ExtCtrls, StdCtrls;
type
  TfrmRFP1 = class(TForm)
    imgRFP1: TImage;
    lbl1RFP1: TLabel;
    lbl2RFP1: TLabel;
    lbl3RFP1: TLabel;
    lbl4RFP1: TLabel;
    lbl5RFP1: TLabel;
    btn1RFP1: TButton;
    procedure btn1RFP1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
 end;
var
 frmRFP1: TfrmRFP1;
implementation
    uses RFP2;
{$R *.dfm}
procedure TfrmRFP1.btn1RFP1Click(Sender: TObject);
begin
frmRFP1.Hide;
frmRFP2.Show;
end;
end.
unit RFP2;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs,Math, Menus, ExtCtrls, TeeProcs, TeEngine, Chart,
 StdCtrls, Series,Printers;
type
 TfrmRFP2 = class(TForm)
   MainMenu1: TMainMenu;
   N1: TMenuItem;
   N2: TMenuItem;
   N3: TMenuItem;
   N4: TMenuItem;
   N5: TMenuItem;
   N6: TMenuItem;
   N7: TMenuItem;
   N8: TMenuItem;
```

```
N9: TMenuItem;
N10: TMenuItem;
N11: TMenuItem;
N12: TMenuItem;
N13: TMenuItem;
N14: TMenuItem;
mem1RFP2: TMemo;
mem2RFP2: TMemo;
cht1RFP2: TChart;
cht2RFP2: TChart;
Series1: TLineSeries;
Series2: TLineSeries;
Series3: TLineSeries;
Series4: TLineSeries;
Series5: TLineSeries;
Series6: TLineSeries;
Series7: TLineSeries;
Series8: TLineSeries;
Series9: TLineSeries;
Series10: TLineSeries;
Series11: TLineSeries;
Series12: TLineSeries;
Series13: TLineSeries;
Series14: TLineSeries;
Series15: TLineSeries;
procedure N6Click(Sender: TObject);
procedure N2Click(Sender: TObject);
procedure N12Click(Sender: TObject);
procedure N8Click(Sender: TObject);
```

```
    procedure N13Click(Sender: TObject);
    procedure N14Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmRFP2: TfrmRFP2;
  i,t,k:Integer;
  a:array[1..50,1..50] of Real;
   ras,sk,L,s:Real;
  aa,aaa:array[1..50] of Real;
  kk,kkk:string;
implementation
      uses RFP1,RFP3,RFP4;
{$R *.dfm}
procedure TfrmRFP2.N6Click(Sender: TObject);
begin
frmRFP1.Close;
end;
procedure v(var L:Real;l1,l2,aa,ras:Real;i:Integer);
begin
  L:=sqrt(sqr(l1+(i-1)*ras)+sqr(30*sin(1.57*100*aa/14)))+ sqrt(sqr(l2-(i-
1)*ras)+sqr(30*sin(1.57*100*aa/14)))-(l1+l2);
end;
```

```
 procedure TfrmRFP2.N2Click(Sender: TObject);
begin
frmRFP3.Show;
end;
procedure TfrmRFP2.N12Click(Sender: TObject);
begin
frmRFP4.Show;
end;
procedure TfrmRFP2.N8Click(Sender: TObject);
begin
            ras:=12;
            sk:=214.3;
            for t:=1 to 29 do
            begin
            aa[t]:=(t-1)/100;
            end;
            for i:=1 to r do
               begin
              for t:=1 to 29 do
                   begin
               v(L,l1,l2,aa[t],ras,i);
               a[i,t]:=L;
               kk:=format('%4.2f',[aa[t]]);
               kkk:=format('%7.2f',[L]);
               mem1RFP2.Lines.Add('L('+kk+')='+kkk);
                   end;
              mem1RFP2.Lines.Add('');
               end;
end;
```

```
procedure TfrmRFP2.N13Click(Sender: TObject);
begin

                for t:=1 to 29 do
               begin
                  s:=0;
                  for i:=1 to r do
                   begin
                   s:=s+ a[i,t];
                   end;
                aaa[t]:=s/r;
               kk:=format('%4.2f',[aa[t]]);
               kkk:=format('%7.2f',[aaa[t]]);
               mem2RFP2.Lines.Add('L('+kk+')='+kkk);
               end;
end;
procedure TfrmRFP2.N14Click(Sender: TObject);
begin
          with Series1 do
       begin
        for t:=1 to 29 do
        begin
       Series1.AddXY(aa[t],aaa[t],'',clRed);
        end;
       end;
end;
procedure TfrmRFP2.N10Click(Sender: TObject);
begin
       with Series2 do
```

```
        begin
         for t:=1 to 29 do
         begin
        Series2.AddXY(aa[t],a[1,t],",clRed);
        Series3.AddXY(aa[t],a[2,t],",clRed);
        Series4.AddXY(aa[t],a[3,t],",clRed);
        Series5.AddXY(aa[t],a[4,t],",clRed);
        Series6.AddXY(aa[t],a[5,t],",clRed);
        Series7.AddXY(aa[t],a[6,t],",clRed);
        Series8.AddXY(aa[t],a[7,t],",clRed);
        Series9.AddXY(aa[t],a[8,t],",clRed);
        Series10.AddXY(aa[t],a[9,t],",clRed);
        Series11.AddXY(aa[t],a[10,t],",clRed);
        Series12.AddXY(aa[t],a[11,t],",clRed);
        Series13.AddXY(aa[t],a[12,t],",clRed);
        Series14.AddXY(aa[t],a[13,t],",clRed);
        Series15.AddXY(aa[t],a[14,t],",clRed);
         end;
        end;
end;
procedure TfrmRFP2.N3Click(Sender: TObject);
   Var
      Prn:TextFile;
      k:Integer;
begin
  begin
      AssignPrn(Prn);
      Rewrite(Prn);
      Printer.Canvas.Font:=mem1RFP2.Font;
```

```
      for k:=0 to mem1RFP2.Lines.Count-1 do
      WriteLn(Prn,mem1RFP2.Lines[k]);
      CloseFile(Prn);
  end;
end;
procedure TfrmRFP2.N4Click(Sender: TObject);
begin
        begin
    cht1RFP2.Print;
    cht2RFP2.Print;
end;
end;
end.
unit RFP3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;
type
  TfrmRFP3 = class(TForm)
    btn1RFP3: TButton;
    lbl1RFP3: TLabel;
    lbl2RFP3: TLabel;
    lbl3RFP3: TLabel;
    lbl4RFP3: TLabel;
    edt1RFP3: TEdit;
    edt2RFP3: TEdit;
    edt3RFP3: TEdit;
    edt4RFP3: TEdit;
```

```
  procedure btn1RFP3Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 frmRFP3: TfrmRFP3;
 l1,l2,h:Real;
 r,code:Integer;
implementation
    uses RFP2;
{$R *.dfm}
procedure TfrmRFP3.btn1RFP3Click(Sender: TObject);
begin
     Val(edt1RFP3.Text,l1,code);
     Val(edt2RFP3.Text,l2,code);
     Val(edt3RFP3.Text,r,code);
     Val(edt4RFP3.Text,h,code);
     frmRFP3.Hide;
end;
end.
unit RFP4;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, Menus, StdCtrls;
type
 TfrmRFP4 = class(TForm)
```

```
    lbl1RFP4: TLabel;
    lbl2RFP4: TLabel;
    lbl3RFP4: TLabel;
    btn1RFP4: TButton;
    procedure btn1RFP4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmRFP4: TfrmRFP4;
implementation
      uses RFP2;
{$R *.dfm}
procedure TfrmRFP4.btn1RFP4Click(Sender: TObject);
begin
frmRFP4.Close;
end;
```

Програма для визначення приведеного коефіцієнту тертя при реалізації алгоритму дихотомії для трансцендентних рівнянь

```
 program Project1;
uses
 Forms,   TUDI1 in 'TUDI1.pas' {frmTUDI1},   TUDI2 in 'TUDI2.pas'
{frmTUDI2},   Unit1 in 'Unit1.pas' {Form1},   Unit2 in 'Unit2.pas' {Form2},
Unit3 in 'Unit3.pas' {Form3};
{$R *.res}  begin   Application.Initialize;
Application.CreateForm(TfrmTUDI1, frmTUDI1);
```

```
Application.CreateForm(TfrmTUDI2, frmTUDI2);
Application.CreateForm(TForm1, Form1);
 Application.CreateForm(TForm2, Form2);
Application.CreateForm(TForm3, Form3);
Application.Run; end. unit TUDI2;
interface  uses   Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls,ComObj , Forms,   Dialogs, StdCtrls, TeEngine, Series, ExtCtrls,
TeeProcs, Chart,Math,   Grids,Jpeg,Printers, ShellAPI;
 type   TfrmTUDI2 = class(TForm)
btn21TUDI2: TButton;
btn23TUDI2: TButton;
StringGrid1: TStringGrid;
Button1: TButton;
Button2: TButton;
Label1: TLabel;
procedure btn23TUDI2Click(Sender: TObject);
procedure btn21TUDI2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
private  { Private declarations }
public  { Public declarations }   end;
var
frmTUDI2: TfrmTUDI2;
a,b,e,X,g,k,vn,teks,rad,vnp,rH,sn,rn,ktren,ug,ktrp,Ktrenie,R:Real;
m1,m2,m3,m4:array[1..9,1..11] of Real;
  code,n,j1,j2:Integer;   X1,n1,ug1,ktrp1,ktren1:string;
implementation
uses TUDI1,Unit1; {$R *.dfm}  procedure
TfrmTUDI2.btn21TUDI2Click(Sender: TObject);
```

```
begin
StringGrid1.Cells[0,0]:='rH';
StringGrid1.Cells[1,0]:='R=1.5';
 StringGrid1.Cells[2,0]:='R=2';
  StringGrid1.Cells[3,0]:='R=2.5';
 StringGrid1.Cells[4,0]:='R=3';
StringGrid1.Cells[5,0]:='R=3.5';
 StringGrid1.Cells[6,0]:='R=4';
 StringGrid1.Cells[7,0]:='R=4.5';
StringGrid1.Cells[8,0]:='R=5';
 for j1:=1 to 8  do
begin
R:=0.5+j1; for j2:=1 to 10 do
 begin
 rH:=j2/10;    a:=0.0001;    b:=1;    e:=0.0001; n:=0;
 repeat
X:=a; g:=3.14*sqr(rH)-sqr(R)*arccos((R-x)/R)+(R-x)*sqrt(2*R*x-sqr(x));     if
g>=0 then
begin
k:=1;
end
 else
 begin
 k:=-1;   end;
X:=(a+b)/2;  g:=3.14*sqr(rH)-sqr(R)*arccos((R-x)/R)+(R-x)*sqrt(2*R*x-
sqr(x));  g:=k*g;     if g>0 then
 begin
a:=X;   end    else
begin
```

```
 b:=X;
end;
 n:=n+1;
until b-a<e;
m1[j1,j2]:=x;
 m2[j1,j2]:=2*arccos((R-x)/R); m4[j1,j2]:=rH;
  end;
  end;
  for j1:=1 to 8  do
 begin
 for j2:=1 to 10 do
begin
StringGrid1.Cells[j1,j2]:=format('%8.4f',[m1[j1,j2]]);
end;
end; for j2:=1 to 10 do
begin
StringGrid1.Cells[0,j2]:=format('%4.1f',[m4[1,j2]]);
end;
end;
procedure XlsWriteCellLabel(XlsStream: TStream;
const ACol, ARow: Word;
 const AValue: string);
var
 L: Word;
 const    {$J+}    CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);    {$J-}
begin
L := Length(AValue);    CXlsLabel[1] := 8 + L;    CXlsLabel[2] := ARow;
CXlsLabel[3] := ACol;    CXlsLabel[5] := L;
```

```
XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
XlsStream.WriteBuffer(Pointer(AValue) , L);
 end;
  function SaveAsExcelFile(AGrid: TStringGrid;
AFileName: string): Boolean;
const    {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0);
{$J-}    CXlsEof: array[0..1] of Word = ($0A, 00);
var
FStream: TFileStream;    I, J: Integer;  begin    Result := False;    FStream :=
TFileStream.Create(PChar(AFileName), fmCreate or fmOpenWrite);
try
CXlsBof[4] := 0;   FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
for i := 0 to AGrid.ColCount - 1 do     for j := 0 to AGrid.RowCount - 1 do
XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
Result := True;
  finally   FStream.Free;
 end;
 end;
procedure TfrmTUDI2.Button1Click(Sender: TObject);
begin
 if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls')
 then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
 {ShowMessage('StringGrid saved!')};
end;
```

```
procedure TfrmTUDI2.btn23TUDI2Click(Sender: TObject); begin
frmTUDI1.Close; end;
  procedure TfrmTUDI2.Button2Click(Sender: TObject); begin Form1.Show;
frmTUDI2.Hide;
end;
end.
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,   Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
Grids,Jpeg,Printers, ShellAPI;
 type   TForm1 = class(TForm)  Button1: TButton;
StringGrid1: TStringGrid;
Button2: TButton;
Button3: TButton;
Button4: TButton;
 Label1: TLabel;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
 procedure Button4Click(Sender: TObject);
private  { Private declarations }   public  { Public declarations }   end;
var
Form1: TForm1;
a,b,e,X,g,k,vn,teks,rad,vnp,rH,sn,rn,ktren,ug,ktrp,Ktrenie,R:Real;
m1,m2,m3,m4:array[1..9,1..11] of Real;
  code,n,j1,j2:Integer;   X1,n1,ug1,ktrp1,ktren1:string;  implementation  uses
TUDI2,Unit2;
```

```
{$R *.dfm}  procedure TForm1.Button1Click(Sender: TObject); begin
Form1.Hide; frmTUDI2.Show;
 end;
  procedure TForm1.Button2Click(Sender: TObject);
begin
 StringGrid1.Cells[0,0]:='rH';
StringGrid1.Cells[1,0]:='R=1.5';
StringGrid1.Cells[2,0]:='R=2';
StringGrid1.Cells[3,0]:='R=2.5';
StringGrid1.Cells[4,0]:='R=3';
StringGrid1.Cells[5,0]:='R=3.5';
 StringGrid1.Cells[6,0]:='R=4';
StringGrid1.Cells[7,0]:='R=4.5';
StringGrid1.Cells[8,0]:='R=5';
for j1:=1 to 8  do
 begin
R:=0.5+j1; for j2:=1 to 10 do
begin
rH:=j2/10;    a:=0.0001;    b:=1;    e:=0.0001; n:=0;
repeat     X:=a; g:=3.14*sqr(rH)-sqr(R)*arccos((R-x)/R)+(R-x)*sqrt(2*R*x-
sqr(x));
if g>=0 then
begin
 k:=1;   end
else
begin
 k:=-1;   end; X:=(a+b)/2;
 g:=3.14*sqr(rH)-sqr(R)*arccos((R-x)/R)+(R-x)*sqrt(2*R*x-sqr(x));
g:=k*g;     if g>0 then
```

```
begin
 a:=X;   end
 else
begin
 b:=X;   end; n:=n+1;
  until b-a<e;
  m1[j1,j2]:=x;
 m2[j1,j2]:=2*arccos((R-x)/R);
 m4[j1,j2]:=rH;
  end;
 end;
 for j1:=1 to 8  do
begin
for j2:=1 to 10 do
begin
StringGrid1.Cells[j1,j2]:=format('%8.4f',[m2[j1,j2]]);
end;
 end;
for j2:=1 to 10 do
begin
StringGrid1.Cells[0,j2]:=format('%4.1f',[m4[1,j2]]);
end;
end;
 procedure XlsWriteCellLabel(XlsStream: TStream;
const ACol, ARow: Word;    const AValue: string);
var    L: Word;
const    {$J+}    CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);    {$J-}
begin    L := Length(AValue);    CXlsLabel[1] := 8 + L;    CXlsLabel[2] :=
ARow;    CXlsLabel[3] := ACol;    CXlsLabel[5] := L;
```

```
XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
XlsStream.WriteBuffer(Pointer(AValue) , L);
end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
const    {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
CXlsEof: array[0..1] of Word = ($0A, 00);  var    FStream: TFileStream;    I, J:
Integer;  begin    Result := False;    FStream :=
TFileStream.Create(PChar(AFileName), fmCreate or fmOpenWrite);
try   CXlsBof[4] := 0;
FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
 for i := 0 to AGrid.ColCount - 1 do
for j := 0 to AGrid.RowCount - 1 do    XlsWriteCellLabel(FStream, I, J,
AGrid.cells[i, j]);
FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
 Result := True;
 finally   FStream.Free;
 end;
end;
  procedure TForm1.Button3Click(Sender: TObject);
 begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
{ShowMessage('StringGrid saved!')};
end;
 procedure TForm1.Button4Click(Sender: TObject);
 begin
Form1.Hide; Form2.Show;
```

```
 end;
end. unit Unit2;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,   Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
Grids,Jpeg,Printers, ShellAPI;
type
TForm2 = class(TForm)  Button1: TButton;
Button2: TButton;
StringGrid1: TStringGrid;
Button3: TButton;
Button4: TButton;
Label1: TLabel;
Edit1: TEdit;
 Label2: TLabel;
procedure Button1Click(Sender: TObject);
 procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
 private  { Private declarations }   public  { Public declarations }   end;
var
Form2: TForm2;
a,b,e,X,g,k,vn,teks,rad,vnp,rH,sn,rn,ktren,ug,ktrp,Ktrenie,R:Real;
m1,m2,m3,m4:array[1..9,1..11] of Real;
  code,n,j1,j2:Integer;   X1,n1,ug1,ktrp1,ktren1:string;
 implementation
uses
```

```
Unit1,Unit3; {$R *.dfm}  procedure TForm2.Button1Click(Sender: TObject);
begin Form1.Show; Form2.Hide; end;
procedure TForm2.Button2Click(Sender: TObject);
begin
val(Edit1.Text,Ktrenie,code);
StringGrid1.Cells[0,0]:='rH';
StringGrid1.Cells[1,0]:='R=1.5';
StringGrid1.Cells[2,0]:='R=2';
StringGrid1.Cells[3,0]:='R=2.5';
StringGrid1.Cells[4,0]:='R=3';
 StringGrid1.Cells[5,0]:='R=3.5';
  StringGrid1.Cells[6,0]:='R=4';
StringGrid1.Cells[7,0]:='R=4.5';
StringGrid1.Cells[8,0]:='R=5';
for j1:=1 to 8  do
begin
R:=0.5+j1;
 for j2:=1 to 10 do
 begin
rH:=j2/10;
a:=0.0001;    b:=1;
    e:=0.0001; n:=0;
 repeat
X:=a; g:=3.14*sqr(rH)-sqr(R)*arccos((R-x)/R)+(R-x)*sqrt(2*R*x-sqr(x));     if
g>=0 then
begin
k:=1;
end
else
```

```
 begin
k:=-1;   end; X:=(a+b)/2;
  g:=3.14*sqr(rH)-sqr(R)*arccos((R-x)/R)+(R-x)*sqrt(2*R*x-sqr(x));  g:=k*g;
if g>0 then
begin
a:=X;   end    else
begin
 b:=X;
 end;
n:=n+1;     until b-a<e;    m1[j1,j2]:=x;
 m2[j1,j2]:=2*arccos((R-x)/R);
    m3[j1,j2]:=4*Ktrenie*sin(arccos((R-x)/R))/ 2*arccos((R-
x)/R)+sin(2*arccos((R-x)/R));
  m4[j1,j2]:=rH;
end;
 end;
 for j1:=1 to 8  do
 begin
  for j2:=1 to 10 do
begin
StringGrid1.Cells[j1,j2]:=format('%8.4f',[m3[j1,j2]]);
  end;
  end;
for j2:=1 to 10 do
begin
StringGrid1.Cells[0,j2]:=format('%4.1f',[m4[1,j2]]);
end;
 end;
    procedure XlsWriteCellLabel(XlsStream: TStream;
```

```
const ACol, ARow: Word;    const AValue: string);  var    L: Word;  const
{$J+}    CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);    {$J-}  begin
L := Length(AValue);    CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;    CXlsLabel[3] := ACol;
 CXlsLabel[5] := L;
XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
XlsStream.WriteBuffer(Pointer(AValue) , L);
 end;
   function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
const    {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
CXlsEof: array[0..1] of Word = ($0A, 00);  var    FStream: TFileStream;    I, J:
Integer;
begin
 Result := False;
 FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try   CXlsBof[4] := 0;
FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
  for i := 0 to AGrid.ColCount - 1 do     for j := 0 to AGrid.RowCount - 1 do
XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
Result := True;
 finally   FStream.Free;
   end;
 end;
procedure TForm2.Button3Click(Sender:
TObject); begin if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
```

```
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
{ShowMessage('StringGrid saved!')};
end;
procedure TForm2.Button4Click(Sender: TObject);
 begin
 Form2.Hide; Form3.Show;
end;
  end.
unit Unit3;
 interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,   Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
Grids,Jpeg,Printers, ShellAPI;
 type
TForm3 = class(TForm)
Button1: TButton;
Label1: TLabel;
 Edit1: TEdit;
Label2: TLabel;
Edit2:TEdit;
Label3: TLabel;
Label4: TLabel;
Edit3: TEdit;
 Label5: TLabel;
 Edit4: TEdit;
GroupBox1:
TGroupBox;
```

```
Label6: TLabel;
Label7: TLabel;
 Label8: TLabel;
 Label9: TLabel;
  Label10: TLabel;
 Label11: TLabel;
 GroupBox2: TGroupBox;
 Label12: TLabel;
 Label13: TLabel;
 Label14: TLabel;
 Label15: TLabel;
Edit5: TEdit;
Button2: TButton;
Memo1: TMemo;
Button3: TButton;
 Chart1: TChart;
Chart2: TChart;
Chart3: TChart;
Chart4: TChart;
Series1: TBarSeries;
 Series2: TBarSeries;
Series3: TBarSeries;
Series4: TBarSeries;
 Series5: TBarSeries;
Series6: TBarSeries;
Series7: TBarSeries;
 procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
 procedure Button3Click(Sender: TObject);
```

```
 private  { Private declarations }   public  { Public declarations }   end;
var
 Form3: TForm3;   a,b,e,X,g,k,vn,teks,rad,vnp,rn,sn,ktren,ug,ktrp:Real;
code,n:Integer;   X1,n1,ug1,ktrp1,ktren1:string;
 implementation
 uses Unit2; {$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
 begin
Form2.Show; Form3.Hide;
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
val(edit1.Text,vn,code);
 val(edit2.Text,teks,code);
val(edit3.Text,rad,code);
val(edit4.Text,vnp,code);
 a:=0.0001;   b:=1;   e:=0.0001;
 if vn=1 then
 begin
rn:=0.0395*sqrt(teks)/2;
sn:=3.14*sqr(rn);  end     else
begin
if vn=2 then
 begin
rn:=0.0427*sqrt(teks)/2;   sn:=3.14*sqr(rn);
end
 else
 begin
if vn=3 then  begin  rn:=0.0411*sqrt(teks)/2;  sn:=3.14*sqr(rn);
```

```
 end
 else
 begin
 if vn=4 then
begin
rn:=0.0411*sqrt(teks)/2;  sn:=3.14*sqr(rn);
end
 else
begin
  if vn=5 then
begin
 rn:=0.0386*sqrt(teks)/2;     sn:=3.14*sqr(rn);
 end
else
begin
rn:=0.0411*sqrt(teks)/2;
 sn:=3.14*sqr(rn);
 end;
 end;
end;
end;
 end;
 if vnp=1 then
begin
if vn=1 then
begin
 ktren:=0.169;
 end
else
```

```
begin
if vn=2 then
begin   ktren:=0.143;
end
else
begin
if vn=3 then
begin
ktren:=0.121;
 end
  else
 begin
if vn=4 then
begin
ktren:=0.152;
 end
else
begin
  if vn=5 then
begin
  ktren:=0.196;
 end
else
begin
 ktren:=0.159;
  end;
end;
 end;
 end;
```

```
 end;
end
else
begin
if vn=1 then
begin  k
tren:=0.27;
end
else
begin
if vn=2 then
begin
ktren:=0.24;
 end
  else
 begin
if vn=3 then
begin
ktren:=0.2;
 end
 else
begin
if vn=4 then
begin
 ktren:=0.25;
end
  else
begin
 if vn=5 then
```

```
begin
  ktren:=0.31;
end
 else
begin
 ktren:=0.26;
end;
end;
end;
 end;
end;
  end;
n:=0;
repeat     X:=a; g:=sn-sqr(rad)*arccos((rad-x)/rad)+(rad-x)*sqrt(2*rad*x-sqr(x));
if g>=0 then
begin
 k:=1;
 end
else
begin
k:=-1;
end;
X:=(a+b)/2;  g:=sn-sqr(rad)*arccos((rad-x)/rad)+(rad-x)*sqrt(2*rad*x-sqr(x));
g:=k*g;     if g>0 then
begin
a:=X;
 end
 else
  begin
```

```
 b:=X;   end; n:=n+1;
   until b-a<e;   X1:=format('%10.4f',[X]);
 n1:=format('%5.0d',[n]);
 Memo1.Lines.Add(' h='+X1);
  Edit5.Text:=n1;   ug:=2*arccos((rad-x)/rad);
ktrp:=4*ktren*sin(ug/2)/(ug+sin(ug));
 ug1:=format('%10.4f',[ug]);
 ktrp1:=format('%10.4f',[ktrp]);
  ktren1:=format('%10.4f',[ktren]);
 Memo1.Lines.Add('K трен.='+ktren1);
 Memo1.Lines.Add('Кут рад.охоп.='+ug1);
 Memo1.Lines.Add('K трен.прив.='+ktrp1);
 end;
procedure TForm3.Button3Click(Sender: TObject);
 begin
Series1.Add(teks,'T',clRed);   Series2.Add(rad,'R',clGreen);
Series3.Add(rn,'Rн',clRed);  Series4.Add(sn,'Sн',clBlue);
Series5.Add(x,'h',clYellow);   Series6.Add(ktren,'Kt',clYellow);
Series7.Add(ktrp,'Ktp',clBlue); end;
 end.
```

Програма для визначення значення натягу при взаємодії нитки з напрямною з використанням зворотних польських записів

```
program PA;
 uses
Forms,   NN1 in 'NN1.PAS' {frm1NN1},   NN2 in 'NN2.pas' {frm1NN2},
NN3 in 'NN3.pas' {frm1NN3},   NN4 in 'NN4.pas' {frm1NN4},
Synt in 'SYNT.PAS',
 UErrors in 'UERRORS.PAS' {FErrors};  {$R *.res}
begin
```

```
Application.Initialize;  Application.CreateForm(Tfrm1NN1, frm1NN1);
Application.CreateForm(Tfrm1NN2, frm1NN2);
Application.CreateForm(Tfrm1NN3, frm1NN3);
Application.CreateForm(Tfrm1NN4, frm1NN4);
Application.CreateForm(TFErrors, FErrors);
Application.Run;
end. unit NN1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs,Math, StdCtrls, jpeg, ExtCtrls;
type
Tfrm1NN1 = class(TForm)
btn1NN1: TButton;
lbl1NN1: TLabel;
lbl2NN1: TLabel;
lbl3NN1: TLabel;
 lbl4NN1: TLabel;
Image1: TImage;
procedure btn1NN1Click(Sender: TObject);
 private { Private declarations }  public { Public declarations }  end;
var
frm1NN1: Tfrm1NN1;
implementation  u
ses NN2,NN3,NN4; {$R *.dfm}
procedure Tfrm1NN1.btn1NN1Click(Sender: TObject);
begin
frm1NN1.Hide; frm1NN2.Show;
end;
```

```
 end.
unit NN2;
interface  uses   Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,   Dialogs,Math, StdCtrls, Menus, TeEngine, Series, ExtCtrls,
TeeProcs,   Chart,Printers;
type
Tfrm1NN2 = class(TForm) MainMenu1:
TMainMenu;
n1NN2: TMenuItem;
n2NN2: TMenuItem;
n3NN2: TMenuItem;
n4NN2: TMenuItem;
n5NN2: TMenuItem;
n6NN2: TMenuItem;
 n7NN2: TMenuItem;
n8NN2: TMenuItem;
n9NN2: TMenuItem;
n10NN2: TMenuItem;
n11NN2: TMenuItem;
n12NN2: TMenuItem;
n13NN2: TMenuItem;
lbl1NN2: TLabel;
lbl2NN2: TLabel;
 lbl3NN2: TLabel;
lbl4NN2: TLabel;
edt1NN2: TEdit;
edt2NN2: TEdit;
edt3NN2: TEdit;
edt4NN2: TEdit;
```

```
lbl5NN2: TLabel;
Memo1: TMemo;
lbl6NN2: TLabel;
mem1NN2: TMemo;
cht1NN2: TChart;
Series1: TLineSeries;
 lbl7NN2: TLabel;
edt5NN2: TEdit;
 procedure btn1NN2Click(Sender: TObject);
procedure n11NN2Click(Sender: TObject);
 procedure n5NN2Click(Sender: TObject);
 procedure n2NN2Click(Sender: TObject);
procedure n7NN2Click(Sender: TObject);
procedure n8NN2Click(Sender: TObject);
 procedure n9NN2Click(Sender: TObject);
procedure n3NN2Click(Sender: TObject);
procedure n4NN2Click(Sender: TObject);
private { Private declarations }   public { Public declarations }   end;
var
 frm1NN2: Tfrm1NN2;
x,F,a,b,e,h,c,R1,kt,d0,d,V1,SS0,SS1,GG0,GG1,USS0,USS1,UGG0,UGG1,L1,L
2,L3,ktp,V1P,L1P,L2P,V11P,L3P:Real;
  k,code,R11,i:Integer;
 aa:array[1..50]of Real;
  kk,kkk,kkk1:String;
w:Integer;
 implementation
uses NN1,Synt,UErrors,NN3,NN4; {$R *.dfm}
 procedure v(var F:Real;X,P0,b1,r,E1,B0,FP,a2,b2,R1,d0,kt,V1:Real);
```

```
 begin
 SetData('P0',P0);
SetData('b1',b1);
  SetData('r',r);
  SetData('E1',E1);
  SetData('B0',B0);
   SetData('FP',FP);
  SetData('a2',a2);
 SetData('b2',b2);
  SetData('R1',R1);
  SetData('d0',d0);
 SetData('kt',kt);
  SetData('V1',V1);
 SetData('X',X);
  Calculate(F);
end;
 procedure Tfrm1NN2.btn1NN2Click(Sender: TObject);
begin
frm1NN1.Close; end;
  procedure Tfrm1NN2.n11NN2Click(Sender: TObject);
begin
frm1NN3.Show;
end;
procedure Tfrm1NN2.n5NN2Click(Sender: TObject);
begin
frm1NN1.Close;
end;
 procedure Tfrm1NN2.n2NN2Click(Sender: TObject);
 begin
```

```
 frm1NN4.Show
 end;
  procedure Tfrm1NN2.n7NN2Click(Sender: TObject);
 begin
  Val(edt1NN2.Text,a,code);
   Val(edt2NN2.Text,b,code);
  Val(edt3NN2.Text,e,code);
  Val(edt4NN2.Text,h,code);
for R11:=2 to 28 do
begin
 R1:=R11/4;    i:=R11-1;        if (FErrors <> nil) then FErrors.Close;    if not
CreatePZ(Memo1.Text)  then  begin  Application.CreateForm(TFErrors,
FErrors);  FErrors.LBErrors.Items.Assign(ErrorList);  FErrors.Show;
 exit;
end;
 c:=h; k:=0; x:=a;
   kt:=a2/exp(b2*ln(P0/R1));
   d0:=P0*(R1+r)/(r*P0+E1*b1*sqr(R1+r));
   if d0>1 then d0:=1;
    d:=d0*exp(kt*FP);
   if d>1 then d:=1;
   SS0:=1-d0*sqr(2*r/R1);
 SS1:=1-d*sqr(2*r/R1);
    GG0:=1-(B0/(2*P0*sqr(R1+r)));
   GG1:=1-(B0/(2*X*sqr(R1+r)));
   USS0:=1.57-arctan(SS0/sqrt(1-SS0*SS0));
  USS1:=1.57-arctan(SS1/sqrt(1-SS1*SS1));
    UGG0:=1.57-arctan(GG0/sqrt(1-GG0*GG0));
  UGG1:=1.57-arctan(GG1/sqrt(1-GG1*GG1));
```

```
 V1:=FP+USS0+USS1-UGG0-UGG1;
V(F,X,P0,b1,r,E1,B0,FP,a2,b2,R1,d0,kt,V1);
  w:= Trunc(F/abs(F));
 repeat      x:=x+c;
 if x-c>=b then Break;
V(F,X,P0,b1,r,E1,B0,FP,a2,b2,R1,d0,kt,V1);
  if F*w/c>0 then Continue;
   c:=-c/4;
  if abs(c)>(e/4) then Continue;
    k:=k+1;
     kk:=format('%5.2f',[R1]);
   kkk:=format('%17.8f',[x]);
 aa[i]:=x;
  mem1NN2.Lines.Add('P('+kk+')='+kkk);
 c:=h;
 w:=-w;
until False;
 end; end;
procedure Tfrm1NN2.n8NN2Click(Sender: TObject);
begin
with Series1 do  for i:=1 to 27 do
 begin
R1:=(i+1)/4;  series1.AddXY(R1,aa[i],'',clRed);
end;
end;
 procedure V2(var F:Real;X:Real);
begin
  kt:=a2/exp(b2*ln(P0/X));
 d0:=P0*(X+r)/(r*P0+E1*b1*sqr(X+r));
```

```
  if d0>1 then d0:=1;
 d:=d0*exp(kt*FP);
 if d>1 then d:=1;
  SS0:=1-d0*sqr(2*r/X);
 SS1:=1-d*sqr(2*r/X);
USS0:=1.57-arctan(SS0/sqrt(1-SS0*SS0));
USS1:=1.57-arctan(SS1/sqrt(1-SS1*SS1));
V1:=FP+USS0+USS1;
L1:=X+r; L2:=r*d0; L3:=exp(kt*V1);
ktp:=a2*b2*power(X/P0,b2)/X;
   L1P:=1;
 L2P:=r*P0*(((r*P0+E1*b1*sqr(X+r))-
2*E1*b1*sqr(X+r))/sqr(r*P0+E1*b1*sqr(X+r)));
 V11P:=(4*r/sqr(X))*((L2P-(L2*2/X))/sqrt(1-sqr(1-
4*L2*r/sqr(X)))+exp(kt*FP)*(L2P+L2*(ktp*FP-(2/X)))/sqrt(1-sqr(1-
(4*L2*r*exp(kt*FP)/sqr(X)))));
L3P:=exp(kt*V1)*(ktp*V1+kt*V11P);
 F:=L3P*sqr(L1)-L1P*L3*L2-L1*L3P*L2+L2P*L1*L3;
  end;
procedure Tfrm1NN2.n9NN2Click(Sender: TObject);
begin
  a:=0.6; b:=5;
e:=0.01; h:=0.1;
c:=h; k:=0; x:=a;
   V2(F,X);
  w:= Trunc(F/abs(F));
repeat      x:=x+c;
    if x-c>=b then Break;
  V2(F,X);
```

```
    if F*w/c>0 then Continue;
   c:=-c/4;
  if abs(c)>(e/4) then Continue;
     k:=k+1;
    kkk1:=format('%17.8f',[x/2]);
  edt5NN2.Text:=('R='+kkk1);
   c:=h;       w:=-w;
  until False;
 end;
  procedure Tfrm1NN2.n3NN2Click(Sender: TObject);
  var
  Prn:TextFile;
  k:Integer;  begin     AssignPrn(Prn);
  Rewrite(Prn);
  Printer.Canvas.Font:=mem1NN2.Font;
   for k:=0 to mem1NN2.Lines.Count-1 do     WriteLn(Prn,mem1NN2.Lines[k]);
CloseFile(Prn);
end;
procedure Tfrm1NN2.n4NN2Click(Sender: TObject);
begin
cht1NN2.Print;
end;
 end.
 unit NN3;
  interface
 uses   Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,   Dialogs,Math, StdCtrls;
type
Tfrm1NN3 = class(TForm)
```

```
lbl1NN3: TLabel;
 lbl2NN3: TLabel;
lbl3NN3: TLabel;
lbl4NN3: TLabel;
lbl5NN3: TLabel;
btn1NN3: TButton;
Label1: TLabel;
procedure btn1NN3Click(Sender: TObject);
  private { Private declarations }
public { Public declarations }
  end;
 var
frm1NN3: Tfrm1NN3;  implementation
uses NN2; {$R *.dfm}  procedure
Tfrm1NN3.btn1NN3Click(Sender: TObject);
begin
frm1NN3.Close;
 end;
end.
unit NN4;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs,Math, StdCtrls;
type
Tfrm1NN4 = class(TForm)
lbl2NN4: TLabel;
lbl3NN4: TLabel;
lbl4NN4: TLabel;
```

```
lbl5NN4: TLabel;
lbl6NN4: TLabel;
lbl7NN4: TLabel;
lbl8NN4: TLabel;
lbl9NN4: TLabel;
edt2NN4: TEdit;
dt3NN4: TEdit;
 edt4NN4: TEdit;
edt5NN4: TEdit;
edt6NN4: TEdit;
edt7NN4: TEdit;
edt8NN4: TEdit;
 edt9NN4: TEdit;
btn1NN4: TButton;
procedure btn1NN4Click(Sender: TObject);
private { Private declarations }   public { Public declarations }   end;
var
 frm1NN4: Tfrm1NN4;
code:Integer;
 P0,b1,r,E1,B0,FP,a2,b2:Real;  implementation uses NN2,Synt,UErrors; {$R
*.dfm}  procedure Tfrm1NN4.btn1NN4Click(Sender: TObject);
begin  val(edt2NN4.Text,P0,code);
 val(edt3NN4.Text,b1,code);
 val(edt4NN4.Text,r,code);
 val(edt5NN4.Text,E1,code);
   val(edt6NN4.Text,B0,code);
 val(edt7NN4.Text,FP,code);
  val(edt8NN4.Text,a2,code);
  val(edt9NN4.Text,b2,code);
```

```
frm1NN4.Hide;
 end;
 end.
unit Synt;
interface  uses classes;
  type  TData = record    Name: string;
  Data:real;  end;  NConst: integer = 100;
  ErrorList: TStringList;
 PZ: array of integer;   DataList:
array of TData;  const  MConst = 2;
 procedure SyntItem(S:string;
First:boolean=false; Pos:Integer=1);
 function CreatePZ(S:string):boolean;
  {Расчет по польской записи} function Calculate(var R:real):boolean;
function SetData(Name:string; Data:real):boolean
  function GetData(Name:string;
var
Data:real):boolean;
 implementation
uses Sysutils, Math, Dialogs,NN2;
  type
TType = (None, Number, Divider, Ident, Func, Part, All);
TSynt = record mode: TType;   Number:real;
  Ident:string;  Error:boolean;
Pos1,Pos2:integer;
 SetNum: set of char=['0'..'9', ','];
SetDiv: set of char=[';', '(', ')', '=', '+', '-', '/', '*', '^',      '{', ' ', #13];
SetChar: set of char=['a'..'z','A'..'Z','_' NFunc = 10 Functions: array[1..NFunc] of
string =   ('exp','sin','cos','sqrt','abs','ln','tg','arctan','arccos','sqr');
```

```
var
SItem: TSynt;
  TrStack: array of char;
  ConstList: array of real;
  Position: Integer;  procedure
SyntItem(S:string;First:boolean=false;Pos:Integer=1);
 First=true var i:integer;
  begin
 if (S = '') then begin   SItem.mode := All;
 exit;
 end;
Position = if(First) then Position := Pos;
repeat if (S[Position] = '{')    then begin repeat  Inc(Position) until (Position >=
Length(S)) or (S[Position] = '}');
Inc(Position);
  end;
if(Position <= Length(S)) then    while ((S[Position] = ' ')or   (S[Position] =
#13)or   (S[Position] = #10)or   (S[Position] = #0))    do Inc(Position);  until
(S[Position] <> '{');
 SItem.Error:=false;  SItem.Pos1:=Position;  Position > mode = All  if(Position
> Length(S)) then begin   SItem.mode := All;
exit;
 end;
Ident  SItem.Ident := S[Position];
  if (S[Position] in SetChar)  then SItem.mode := Ident  else if (S[Position] in
SetNum)  then Item.mode := Number  else if (S[Position] in SetDiv)  then if
(S[Position] <> ';')    then SItem.mode := Divider    else SItem.mode := Part
Inc(Position);   exit;  end  else begin   SItem.mode := None;   Inc(Position);
exit;  end Inc(Position);
```

```
 if (SItem.mode = Number)and  ((S[Position] = '-')or(S[Position] = '+'))and
(UpCase(S[Position-1])='E') then SItem.Ident := SItem.Ident + S[Position]   else
if ((Position > Length(S))or(S[Position] in SetDiv)) then begin if(SItem.mode =
Number)    then try  SItem.Number := StrToFloat(SItem.Ident)    except     on
EConvertError do SItem.Error := true;
   end;
for i:=1 to NFunc do   if (LowerCase(SItem.Ident) = Functions[i])     then
 begin
SItem.mode:=Func;
SItem.Number:=i;
 break;
  end;  SItem.Pos2:=Position-1;
exit; end else SItem.Ident := SItem.Ident + S[Position];
  until false;
 end;
procedure ClearPZ;
 begin
 ErrorList.Clear;
 SetLength(ConstList,0);
SetLength(DataList,MConst);
  SetLength(PZ,0);
end;
function CreatePZ(S:string):boolean;
var
 lend:boolean; i:integer;
Assign:boolean;
Adress: integer;
OldMode: TType;
  OldS: char;
```

```
 procedure code;
begin  SetLength(PZ,High(PZ)+2);
 case TrStack[High(TrStack)] of   '+': PZ[High(PZ)] := -1;
   '-': PZ[High(PZ)] := -2;
  '*': PZ[High(PZ)] := -3;
  '/': PZ[High(PZ)] := -4;
  '^': PZ[High(PZ)] := -5;
 'M': PZ[High(PZ)] := -6;
 end; end;  procedure proc1;
begin
SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
 procedure proc2;
begin
  code;  TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc3;
begin
 code;  SetLength(TrStack,High(TrStack));
 lend:=false;
end;
  procedure proc4;
begin
  SetLength(TrStack,High(TrStack));
end;
 procedure proc5;
//  SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := Chr(127+Round(SItem.Number));
```

```
 end;
procedure proc6; //
SetLength(PZ,High(PZ)+2);  PZ[High(PZ)] := -
Ord(TrStack[High(TrStack)])+27;
SetLength(TrStack,High(TrStack));
 end;
begin
ClearPZ;
 SetLength(TrStack,1);
TrStack[0] := '0';
OldMode := None;
 OldS := ' ';
Assign := true;
Adress := 0;
//
SyntItem(S,true);  if (SItem.mode = All)   then begin    ErrorList.Add();
   Result := false;
 exit;
  end;
repeat   if ((OldMode = Func)and(SItem.Ident[1] <> '('))    then
ErrorList.Add('+IntToStr(SItem.Pos1));
  case SItem.mode of    Number: begin     if((OldMode <>
Divider)and(OldMode <> None)and    (OldMode <> Part))      then
ErrorList.Add('+IntToStr(SItem.Pos1)+');
   if (SItem.Error)      then ErrorList.Add(' '+IntToStr(SItem.Pos1)+         ' -
'+IntToStr(SItem.Pos2))
 else
begin
SetLength(ConstList,High(ConstList)+2);
```

```
ConstList[High(ConstList)] := SItem.Number;
  SetLength(PZ,High(PZ)+2);
PZ[High(PZ)] := High(ConstList);
    end;
 Assign:=false;
 end;
  Ident:
 begin
 if((OldMode <> Divider)and(OldMode <> None)and    (OldMode <> Part))
then ErrorList.Add('В позиции '+IntToStr(SItem.Pos1)+');
 for i:=0 to High(DataList) do
begin
if (UpperCase(SItem.Ident) = DataList[i].Name)    then
begin
 SetLength(PZ,High(PZ)+2);     PZ[High(PZ)] := NConst+i;
   break;
 end;
 if(i = High(DataList)) then
begin
SetLength(DataList,High(DataList)+2);
DataList[High(DataList)].Name:=UpperCase(SItem.Ident);
DataList[High(DataList)].Data:=0;
  SetLength(PZ,High(PZ)+2);
   PZ[High(PZ)] := NConst+High(DataList);
 end;
    end;
   end;
All,Part:begin     repeat      lend:=true;
  case TrStack[High(TrStack)] of   '0':
```

```
begin
if (Adress <> 0) then
begin
 SetLength(PZ,High(PZ)+3);       PZ[High(PZ)-1] := -7;
    PZ[High(PZ)] := Adress;
     Adress := 0;
  end;
   break;
 end;
 '(': ErrorList.Add(');   else proc3;
     end;
  until lend;
   if (ErrorList.Count = 0)      then Result:=true      else Result:=false;     if
(SItem.mode = All) then exit     else begin      Assign := true;      SItem.mode :=
None;
 end
    end;
  Divider:begin    if((OldMode = Divider)and      ((SItem.Ident[1]<>'=')and
(SItem.Ident[1]<>'(')and      (SItem.Ident[1] <> ')'))and      ((OldS <> '(')and
(OldS <> ')')and      (OldS <> '='))     then begin      ErrorList.Add('
'+IntToStr(SItem.Pos1)+          ':);
   break;
   end;
   repeat     lend:=true;     case SItem.Ident[1] of      '=': if Assign and (OldMode
= Ident)    then
begin
Adress := PZ[High(PZ)];     SetLength(PZ,High(PZ));     SItem.mode := None;
end
else
```

```
ErrorList.Add(' '+IntToStr(SItem.Pos1)+          ': ');      '(': if(OldMode = Ident)
or (OldMode = Number)       then ErrorList.Add('+IntToStr(SItem.Pos1))
else  proc1;
    '+','-','M':
begin
 if((OldMode = None)or(OldS = '('))   then if (SItem.Ident[1] = '+' then break
else SItem.Ident[1] := 'M';
     case TrStack[High(TrStack)] of    '0','(': proc1;
  '+','-','M': proc2;
   '*','/','^': proc3;
   end;
   end;
   '*','/':       if OldS = '('       then ErrorList.Add('+IntToStr(SItem.Pos1))       else
case TrStack[High(TrStack)] of    '0','(','+','-','M': proc1;    '*','/': proc2;    '^':
proc3;       end;      '^':
      if OldS = '('       then ErrorList.Add('+IntToStr(SItem.Pos1))       else
case TrStack[High(TrStack)] of    '0','(','+','-','*','/','M': proc1;    '^': proc2;
end;
    ')':       case TrStack[High(TrStack)] of    '0': ErrorList.Add();    '(':
begin
    proc4;      if (Ord(TrStack[High(TrStack)]) > 127)
   then proc6;
 end;
    '+','-','*','/','^','M': proc3;
   end;
    end;
   until lend;
  Assign:=false;
  end;
```

```
Func:
begin
  repeat
lend:=true;
 proc5
until lend;    Assign:=false;
 end;
   None:   ErrorList.Add('+IntToStr(SItem.Pos1));   end;   OldMode :=
SItem.mode;   OldS := SItem.Ident[1];   SyntItem(S);  until false;
if(ErrorList.Count = 0)  then Result := true  else Result := false; end;  function
Calculate(var R:real):boolean; var Stack: array of real; i:integer; begin   for i:=0
to High(PZ) do begin    if (i > 0) then if (PZ[i-1] = -7) and (i < High(PZ)) then
Continue;    if PZ[i] < -100 then begin  try   case -PZ[i]-100 of    1:
Stack[High(Stack)]:=Exp(Stack[High(Stack)]);    2:
Stack[High(Stack)]:=Sin(Stack[High(Stack)]);    3:
Stack[High(Stack)]:=Cos(Stack[High(Stack)]);    4:
Stack[High(Stack)]:=Sqrt(Stack[High(Stack)]);    5:
Stack[High(Stack)]:=Abs(Stack[High(Stack)]);    6:
Stack[High(Stack)]:=Ln(Stack[High(Stack)]);    7:
Stack[High(Stack)]:=Tan(Stack[High(Stack)]);    8:
Stack[High(Stack)]:=ArcTan(Stack[High(Stack)]);    9:
Stack[High(Stack)]:=ArcCos(Stack[High(Stack)]);    10:
(FloatToStr(Stack[High(Stack)]) = 'INF') or    (FloatToStr(Stack[High(Stack)])
= '-INF')   then begin    Result := false;    exit;   end end else if PZ[i] < 0 then
begin try  case -PZ[i] of   1: Stack[High(Stack)-1]:=  Stack[High(Stack)-
1]+Stack[High(Stack)];   2: Stack[High(Stack)-1]:=  Stack[High(Stack)-1]-
Stack[High(Stack)];   3: Stack[High(Stack)-1]:=  Stack[High(Stack)-
1]*Stack[High(Stack)];   4: Stack[High(Stack)-1]:=    Stack[High(Stack)-
1]/Stack[High(Stack)];   5: Stack[High(Stack)-1]:=    Power(Stack[High(Stack)-
```

```
1],Stack[High(Stack)]);   6: Stack[High(Stack)]:= -Stack[High(Stack)];   7:
DataList[PZ[i+1]-NConst].Data := Stack[High(Stack)];  end; except  Result :=
false;  exit; end;  if (PZ[i] <> -6)  then SetLength(Stack,High(Stack)); end else
begin  SetLength(Stack,High(Stack)+2);  if (PZ[i] < NConst)   then
Stack[High(Stack)]:=ConstList[PZ[i]]   else
Stack[High(Stack)]:=DataList[PZ[i]-NConst].Data; end;  end; Result := true;  R
:=Stack[High(Stack)]; end;  function SetData(Name:string; Data:real):boolean;
var i:integer; begin  for i:=MConst to High(DataList) do   if (UpperCase(Name)
= DataList[i].Name)    then
 begin
DataList[i].Data := Data;
Result:=true;
exit;
end;
Result := false;
end;
 function GetData(Name:string;
 var
Data:real):boolean;
var
 i:integer;
begin
for i:=0 to High(DataList) do   if (UpperCase(Name) = DataList[i].Name)
 then
begin
 Data := DataList[i].Data;
Result:=true; exit;
  end;
Result := false;
```

```
end;
 initialization  SetLength(DataList,MConst);
 DataList[0].Name:='PI';  DataList[0].Data:=Pi;
DataList[1].Name:='E';
DataList[1].Data:=2.71828183;
ErrorList := TStringList.Create; finalization  ErrorList.Free; end. unit UErrors;
interface
 uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls;  type   TFErrors = class(TForm) LBErrors: TListBox;
private { Private declarations }   public { Public declarations }   end;  var
FErrors: TFErrors;  implementation   uses NN2; {$R *.dfm}  end.
```

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

системи розрахунку динаміки маятникових пристроїв контактного намотування нитки

```
nterface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm3 = class(TForm)
    Button1: TButton;
    StringGrid1: TStringGrid;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;
```

```
  Label1: TLabel;
  Button4: TButton;
  Button5: TButton;
  Chart1: TChart;
  Series1: TLineSeries;
  Series2: TLineSeries;
  Series3: TLineSeries;
  Chart2: TChart;
  Series4: TLineSeries;
  Series5: TLineSeries;
  Series6: TLineSeries;
  Chart3: TChart;
  Series7: TLineSeries;
  Series8: TLineSeries;
  Series9: TLineSeries;
  Edit2: TEdit;
  Label2: TLabel;
  Series10: TLineSeries;
  Series11: TLineSeries;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure Button5Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

```
var
  Form3: TForm3;
  i,j,n,code: Integer;
  U1,xA,yA,vAx,vAy,wAx,wAy:Real;
  l1,l2,U2,xB,US2,UU2:Real;
  vBx,wBx,xAB,yAB:Real;
  vABx,vABy,wABx,wABy:Real;
  mxA,myA,mvAx,mvAy,mwAx,mwAy: array [0..360] of Real;
  mU1,mU2,mxB,mxAB,myAB,mUS2,mUU2: array [0..360] of Real;
 mvBx,mvABx,mvABy,mwBx,mwABx,mwABy: array [0..360] of Real;
implementation
 uses unit2;
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
Form3.Hide;
Form2.Show;
end;
procedure TForm3.Button4Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
```

```
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;

for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mxB[j],'',clBlue);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,mxAB[j],'',clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,myAB[j],'',clYellow);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mvBx[j],'',clWhite);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mvABx[j],'',clGreen);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mvABy[j],'',clAqua);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,mUS2[j],'',clRed);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mwBx[j],'',clBlue);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mwABx[j],'',clGreen);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mwABy[j],'',clRed);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mUU2[j],'',clYellow);
end;
end;
procedure TForm3.Button5Click(Sender: TObject);
unit Unit2;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
```

```
TForm2 = class(TForm)
  Button1: TButton;
  Button2: TButton;
  StringGrid1: TStringGrid;
  Button3: TButton;
  Button4: TButton;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Edit1: TEdit;
  Edit2: TEdit;
  Edit3: TEdit;
  Button5: TButton;
  Chart1: TChart;
  Series1: TLineSeries;
  Series2: TLineSeries;
  Series3: TLineSeries;
  Chart2: TChart;
  Series5: TLineSeries;
  Series6: TLineSeries;
  Chart3: TChart;
  Series7: TLineSeries;
  Series9: TLineSeries;
  Button6: TButton;
  Edit4: TEdit;
  Label4: TLabel;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
```

```
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
  i,j,n,code: Integer;
  omg,s,Tbav,Tvov,tol:Real;
  R,qsbav,vpbav,mpbav:Real;
  sqsbav,vobav,vHbav:Real;
  ktgbav,qsvov,dbav,dvov:Real;
  mpvov,sqsvov,H,R1:Real;
  vovov,vHvov,gambav,gamvov:Real;
  ktgvov:Real;
  mR,mqsbav,mvpbav,mmpbav: array [0..360] of Real;
  msqsbav,mvobav,mvHbav,mktgbav,mqsvov: array [0..360] of Real;
  mmpvov,msqsvov, mvovov,mvHvov,mktgvov: array [0..360] of Real;
implementation
 uses unit1, unit3;
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
Form1.Close;
end;
```

```
procedure TForm2.Button2Click(Sender: TObject);
begin
Form2.Hide;
Form3.Show;
end;
procedure TForm2.Button6Click(Sender: TObject);
begin
Chart1.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\БУРЯК_В_В\Буряк_В_В\Example\1.bmp');
Chart2.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\БУРЯК_В_В\Буряк_В_В\Example\2.bmp');
Chart3.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\БУРЯК_В_В\Буряк_В_В\Example\3.bmp');
end;
procedure TForm2.Button5Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mR[j],mqsvov[j]/30,'',clRed);
Chart1.SeriesList[1].AddXY(mR[j],mvHvov[j],'',clGreen);
Chart1.SeriesList[2].AddXY(mR[j],mktgvov[j],'',clYellow);
```

```
Chart2.SeriesList[0].AddXY(mR[j],msqsvov[j],'',clGreen);
Chart2.SeriesList[1].AddXY(mR[j],mvovov[j],'',clRed);
Chart3.SeriesList[0].AddXY(mR[j],mvpbav[j],'',clBlue);
Chart3.SeriesList[1].AddXY(mR[j],mmpvov[j],'',clYellow);
end;
end;
procedure TForm2.Button3Click(Sender: TObject);
begin
        Val(Edit1.text,omg,code);
        Val(Edit2.text,s,code);
        Val(Edit3.text,Tbav,code);
        Val(Edit4.text,Tvov,code);
                begin
        StringGrid1.cells[0,0]:='R';
        StringGrid1.cells[1,0]:='1T';
        StringGrid1.cells[2,0]:='1.5T';
        StringGrid1.cells[3,0]:='1.83T';
        StringGrid1.cells[4,0]:='1.94T';
        StringGrid1.cells[5,0]:='0.78T';
        StringGrid1.cells[6,0]:='2.1T';
        StringGrid1.cells[7,0]:='2.3T';
        StringGrid1.cells[8,0]:='0.95T';
        StringGrid1.cells[9,0]:='2.6T';
        StringGrid1.cells[10,0]:='3.44T';
        StringGrid1.cells[11,0]:='3.98T';
        StringGrid1.cells[12,0]:='4.2T';
        StringGrid1.cells[13,0]:='2.78';
                end;
```

```
R:=15.4;
tol:=1;
for i:=0 to 360 do
begin
R:=(42/360)+R;
qsvov:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(0.95
*tol))));
vHvov:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(4.2*
tol))));
ktgvov:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(2.78
*tol))));
sqsvov:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(3.44
*tol))));
vovov:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(3.98
*tol))));
vpbav:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(1.5*
tol))));
mpvov:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(2.6*
tol))));
qsbav:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(1*to
l))));
```

```
mpbav:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(1.83
*tol))));
sqsbav:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(1.94
*tol))));
vobav:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(0.78
*tol))));
vHbav:=28.54*((exp(0.546*ln(R))*
exp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(2.1*
tol))));
ktgbav:=28.54*((exp(0.546*ln(R))*
xp(0.446*ln(s))*exp(0.96*ln(Tbav)))/(exp(0.546*ln(Tvov))*exp(1.912*ln(2.3*t
ol))));
        mR[i]:=R;
        mqsbav[i]:=qsbav;
        mvpbav[i]:=vpbav;
        mmpbav[i]:=mpbav;
        msqsbav[i]:=sqsbav;
        mvobav[i]:=vobav;
        mvHbav[i]:=vHbav;
        mktgbav[i]:=ktgbav;
        mqsvov[i]:=qsvov;
        mmpvov[i]:=mpvov;
        msqsvov[i]:=sqsvov;
        mvovov[i]:=vovov;
        mvHvov[i]:=vHvov;
        mktgvov[i]:=ktgvov;
```

```
        end;
            for n:=0 to 360 do
              begin
          StringGrid1.cells[0,n+1]:=format('%9.4f',[mR[n]]);
          StringGrid1.cells[1,n+1]:=format('%9.4f',[mqsbav[n]]);
          StringGrid1.cells[2,n+1]:=format('%9.4f',[mvpbav[n]]);
          StringGrid1.cells[3,n+1]:=format('%9.4f',[mmpbav[n]]);
          StringGrid1.cells[4,n+1]:=format('%9.4f',[msqsbav[n]]);
          StringGrid1.cells[5,n+1]:=format('%9.4f',[mvobav[n]]);
          StringGrid1.cells[6,n+1]:=format('%9.4f',[mvHbav[n]]);
          StringGrid1.cells[7,n+1]:=format('%9.4f',[mktgbav[n]]);
          StringGrid1.cells[8,n+1]:=format('%9.4f',[mqsvov[n]]);
          StringGrid1.cells[9,n+1]:=format('%9.4f',[mmpvov[n]]);
          StringGrid1.cells[10,n+1]:=format('%9.4f',[msqsvov[n]]);
          StringGrid1.cells[11,n+1]:=format('%9.4f',[mvovov[n]]);
          StringGrid1.cells[12,n+1]:=format('%9.4f',[mvHvov[n]]);
          StringGrid1.cells[13,n+1]:=format('%9.4f',[mktgvov[n]]);
              end;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
```

```
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
var
  FStream: TFileStream;
  I, J: Integer;
begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
```

```
 end;
procedure TForm2.Button4Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm3 = class(TForm)
    Button1: TButton;
    StringGrid1: TStringGrid;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Button4: TButton;
    Button5: TButton;
    Chart1: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
```

```
    Series3: TLineSeries;
    Chart2: TChart;
    Series4: TLineSeries;
    Series5: TLineSeries;
    Series6: TLineSeries;
    Chart3: TChart;
    Series7: TLineSeries;
    Series8: TLineSeries;
    Series9: TLineSeries;
    Edit2: TEdit;
    Label2: TLabel;
    Series10: TLineSeries;
    Series11: TLineSeries;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form3: TForm3;
  i,j,n,code: Integer;
  U1,xA,yA,vAx,vAy,wAx,wAy:Real;
  l1,l2,U2,xB,US2,UU2:Real;
  vBx,wBx,xAB,yAB:Real;
```

```
 vABx,vABy,wABx,wABy:Real;
 mxA,myA,mvAx,mvAy,mwAx,mwAy: array [0..360] of Real;
 mU1,mU2,mxB,mxAB,myAB,mUS2,mUU2: array [0..360] of Real;
mvBx,mvABx,mvABy,mwBx,mwABx,mwABy: array [0..360] of Real;
implementation
 uses unit2;
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
Form3.Hide;
Form2.Show;
end;
procedure TForm3.Button4Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mxB[j],'',clBlue);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,mxAB[j],'',clGreen);
```

```
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,myAB[j],'',clYellow);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mvBx[j],'',clWhite);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mvABx[j],'',clGreen);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mvABy[j],'',clAqua);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,mUS2[j],'',clRed);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mwBx[j],'',clBlue);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mwABx[j],'',clGreen);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mwABy[j],'',clRed);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mUU2[j],'',clYellow);
end;
end;
procedure TForm3.Button5Click(Sender: TObject);
begin
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
        Val(Edit1.text,l1,code);
        Val(Edit2.text,l2,code);
           begin
       StringGrid1.cells[0,0]:='U1';
       StringGrid1.cells[1,0]:='U2';
       StringGrid1.cells[2,0]:='xB';
       StringGrid1.cells[3,0]:='xAB';
       StringGrid1.cells[4,0]:='yAB';
       StringGrid1.cells[5,0]:='US2';
       StringGrid1.cells[6,0]:='UU2';
       StringGrid1.cells[7,0]:='vBx';
       StringGrid1.cells[8,0]:='vABx';
       StringGrid1.cells[9,0]:='vABy';
```

```
StringGrid1.cells[10,0]:='wBx';
StringGrid1.cells[11,0]:='wABx';
StringGrid1.cells[12,0]:='wABy';
        end;
for i:=0 to 360 do
begin
U1:=i*Pi/180;
xA:=l1*cos(U1);
yA:=l1*sin(U1);
vAx:=-omg*l1*sin(U1);
vAy:=omg*l1*cos(U1);
wAx:=-sqr(omg)*l1*cos(U1);
wAy:=-sqr(omg)*l1*sin(U1);
mxA[i]:=xA;
myA[i]:=yA;
mvAx[i]:=vAx;
mvAy[i]:=vAy;
mwAx[i]:=wAx;
mwAy[i]:=wAy;
end;
  for i:=0 to 360 do
 begin
 U1:=i*Pi/180;
 U2:=arcTan(((-myA[i])/l2)/sqrt(1-sqr(((-myA[i])/l2))));
 xB:=mxA[i]+l2*cos(U2);
 xAB:=mxA[i]+(l2/4)*cos(U2);
 yAB:=myA[i]+(l2/4)*sin(U2);
 US2:=-mvAy[i]/sqrt(sqr(l2)-sqr(-myA[i]));
```

```
        UU2:=(sqr(omg)*l1*sin(U1)/(l2*cos(U2)))-
(omg*l1*US2*cos(U1)*sin(U2)/(l2*sqr(cos(U2))));
        vBx:=mvAx[i]-l2*US2*sin(U2);
        wBx:=-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2)+mwAx[i];
        vABx:=mvAx[i]-(l2/4)*US2*sin(U2);
        vABy:=mvAy[i]+(l2/4)*US2*cos(U2);
        wABx:=mwAx[i]-(l2/4)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
        wABy:=mwAy[i]+(l2/4)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
        mU1[i]:=U1;
        mU2[i]:=U2;
        mxB[i]:=xB;
        mxAB[i]:=xAB;
        myAB[i]:=yAB;
        mUS2[i]:=US2;
        mUU2[i]:=UU2;
        mvBx[i]:=vBx;
        mvABx[i]:=vABx;
        mvABy[i]:=vABy;
        mwBx[i]:=wBx;
        mwABx[i]:=wABx;
        mwABy[i]:=wABy;
        end;
              for n:=0 to 360 do
               begin
          StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
          StringGrid1.cells[1,n+1]:=format('%9.4f',[mU2[n]]);
          StringGrid1.cells[2,n+1]:=format('%9.4f',[mxB[n]]);
          StringGrid1.cells[3,n+1]:= format('%9.4f',[mxAB[n]]);
          StringGrid1.cells[4,n+1]:=format('%9.4f',[myAB[n]]);
```

```
            StringGrid1.cells[5,n+1]:=format('%9.4f',[mUS2[n]]);
            StringGrid1.cells[6,n+1]:= format('%9.4f',[mUU2[n]]);
            StringGrid1.cells[7,n+1]:=format('%9.4f',[mvBx[n]]);
            StringGrid1.cells[8,n+1]:=format('%9.4f',[mvABx[n]]);
            StringGrid1.cells[9,n+1]:=format('%9.4f',[mvABy[n]]);
            StringGrid1.cells[10,n+1]:=format('%9.4f',[mwBx[n]]);
            StringGrid1.cells[11,n+1]:=format('%9.4f',[mwABx[n]]);
            StringGrid1.cells[12,n+1]:=format('%9.4f',[mwABy[n]]);
                end;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
 function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
```

```
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
   CXlsBof[4] := 0;
   FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
   for i := 0 to AGrid.ColCount - 1 do
    for j := 0 to AGrid.RowCount - 1 do
     XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
   FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
   Result := True;
  finally
   FStream.Free;
  end;
 end;
procedure TForm3.Button3Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
   {ShowMessage('StringGrid saved!')};
```

```
end;
```

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

системи визначення натягу нитки при осьовому змотуванні з пакування конічної форми

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm1 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;
implementation
 uses unit2;
{$R *.dfm}
procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Hide;
Form2.Show;
end;
unit Unit2;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm2 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    StringGrid1: TStringGrid;
    Button3: TButton;
    Button4: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label1: TLabel;
    Label2: TLabel;
```

```
Label3: TLabel;
Chart1: TChart;
Button5: TButton;
Button6: TButton;
Series1: TLineSeries;
Series2: TLineSeries;
Series3: TLineSeries;
Chart2: TChart;
Series4: TLineSeries;
Series5: TLineSeries;
Series6: TLineSeries;
Chart3: TChart;
Series7: TLineSeries;
Series8: TLineSeries;
Series9: TLineSeries;
Chart4: TChart;
Series10: TLineSeries;
Series11: TLineSeries;
Series12: TLineSeries;
Series13: TLineSeries;
Series14: TLineSeries;
Series15: TLineSeries;
Series16: TLineSeries;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
```

```
 private
   { Private declarations }
 public
   { Public declarations }
 end;
var
 Form2: TForm2;
 i,j,n,code: Integer;
t,omg,i2,mopr,b11,b22,b33,b44:Real;
a1,b1,c1,a2,b2,c2:Real;
ak1,bk1,ck1,ek1,ook1,dk1:Real;
ak2,bk2,ck2,ek2,ook2,dk2:Real;
omg11,fi11,eps11,omg21,fi21,eps21,mpux11,mfri11:Real;
omg12,fi12,eps12,omg22,fi22,eps22,mpux12,mfri12:Real;
mpO,masO,masU,mzsV,mHts: array [0..360] of Real;
mt,momg11,mfi11,meps11,momg21,mfi21,meps21,mmpux11,mmfri11: array
[0..360] of Real;
momg12,mfi12,meps12,momg22,mfi22,meps22,mmpux12,mmfri12 : array
[0..360] of Real;
implementation
uses unit1, unit3;
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
Form1.Close;
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
```

```
Form2.Hide;
Form3.Show;
end;
procedure TForm2.Button5Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
Chart4.SeriesList[0].Clear;
Chart4.SeriesList[1].Clear;
Chart4.SeriesList[2].Clear;
Chart4.SeriesList[3].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mt[j],momg11[j],",clRed);
Chart1.SeriesList[1].AddXY(mt[j],momg21[j],",clGreen);
Chart1.SeriesList[2].AddXY(mt[j],momg12[j],",clYellow);
Chart1.SeriesList[3].AddXY(mt[j],momg22[j],",clBlue);
Chart2.SeriesList[0].AddXY(mt[j],mfi11[j],",clWhite);
Chart2.SeriesList[1].AddXY(mt[j],mfi21[j],",clGreen);
```

```
Chart2.SeriesList[2].AddXY(mt[j],mfi12[j],",clRed);
Chart2.SeriesList[3].AddXY(mt[j],mfi22[j],",clBlue);
Chart3.SeriesList[0].AddXY(mt[j],meps11[j],",clBlue);
Chart3.SeriesList[1].AddXY(mt[j],meps21[j],",clGreen);
Chart3.SeriesList[2].AddXY(mt[j],meps12[j],",clYellow);
Chart3.SeriesList[3].AddXY(mt[j],meps22[j],",clRed);
Chart4.SeriesList[0].AddXY(mt[j],mmpux11[j],",clGreen);
Chart4.SeriesList[1].AddXY(mt[j],mmfri11[j],",clRed);
Chart4.SeriesList[2].AddXY(mt[j],mmpux12[j],",clYellow);
Chart4.SeriesList[3].AddXY(mt[j],mmfri12[j],",clBlue);
end;
end;
procedure TForm2.Button6Click(Sender: TObject);
begin
Chart1.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\ПРИХОДЬКО_ В_Ю\Приходько В Ю\Example\1.bmp');
Chart2.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\ПРИХОДЬКО_ В_Ю\Приходько В Ю\Example\2.bmp');
Chart3.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\ПРИХОДЬКО_ В_Ю\Приходько В Ю\Example\3.bmp');
Chart4.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\ПРИХОДЬКО_ В_Ю\Приходько В Ю\Example\4.bmp');
end;
procedure TForm2.Button3Click(Sender: TObject);
begin
        Val(Edit1.text,omg,code);
        Val(Edit2.text,i2,code);
        Val(Edit3.text,mopr,code);
```

```
        begin
StringGrid1.cells[0,0]:='R';
StringGrid1.cells[1,0]:='I/5';
StringGrid1.cells[2,0]:='E/5';
StringGrid1.cells[3,0]:='rt/5';
StringGrid1.cells[4,0]:='I/3';
StringGrid1.cells[5,0]:='E/3';
StringGrid1.cells[6,0]:='rt/3';
StringGrid1.cells[7,0]:='mt/5';
StringGrid1.cells[8,0]:='mt/3';
StringGrid1.cells[9,0]:='I/2';
StringGrid1.cells[10,0]:='E/2';
StringGrid1.cells[11,0]:='rt/2';
StringGrid1.cells[12,0]:='I/1';
StringGrid1.cells[13,0]:='E/1';
StringGrid1.cells[14,0]:='rt/1';
StringGrid1.cells[15,0]:='mt/2';
StringGrid1.cells[16,0]:='mt/1';
        end;
 t:=20;
 b11:=0.02;
 b22:=0.03;
 b33:=0.05;
 b44:=0.08;
 for i:=0 to 360 do
 begin
 t:= t+(60/360);
```

```
 omg11:=5.1+b11*(Pi*(i2/5)*mopr)*(1.1*sqr(t)-sqr(20));
 omg21:=6.2+b11*(Pi*(i2/3)*mopr)*(1.2*sqr(t)-sqr(20));
 omg12:=7.05+b11*(Pi*(i2/2)*mopr)*(1.3*sqr(t)-sqr(20));
 omg22:=8.1+b11*(Pi*i2*mopr)*(1.2*sqr(t)-sqr(20));

fi11:=1.4+0.4*b22*sqr(10)*(2*Pi*(i2/500)*mopr*sqr(t));
 fi21:=1.2+0.4*b22*sqr(10)*(2*Pi*(i2/300)*mopr*sqr(t));
 fi12:=1.3+0.4* b22*sqr(10)*(2*Pi*(i2/200)*mopr*sqr(t));
 fi22:=1.6+ 0.4*b22*sqr(10)*(2*Pi*(i2/100)*mopr*sqr(t));

eps11:=1.6+ b33*(10)*(2*Pi*(i2/50)*mopr*sqr(t));
 eps21:=1.4+ b33*(10)*(2*Pi*(i2/30)*mopr*sqr(t));
 eps12:=1.2+ b33*(10)*(2*Pi*(i2/20)*mopr*sqr(t));
 eps22:=1.7+ b33*(10)*(2*Pi*(i2/10)*mopr*t);

mpux11:=b44*t+b22*sqrt(10)*(2*Pi*(i2/50)*mopr*sqr(t));
 mfri11:=b44*t+b22*sqrt(10)*(2*Pi*(i2/30)*mopr*sqr(t));
 mpux12:=b44*t+b22*sqrt(10)*(2*Pi*(i2/20)*mopr*sqr(t));
 mfri12:=b44*t+b22*sqrt(10)*(2*Pi*(i2/10)*mopr*sqr(t));
          mt[i]:=t;
 momg11[i]:=omg11;
 mfi11[i]:=fi11;
 meps11[i]:=eps11;
 momg21[i]:=omg21;
 mfi21[i]:=fi21;
 meps21[i]:=eps21;
 mmpux11[i]:=mpux11;
 mmfri11[i]:=mfri11;
 momg12[i]:=omg12;
```

```
mfi12[i]:=fi12;
meps12[i]:=eps12;
momg22[i]:=omg22;
mfi22[i]:=fi22;
meps22[i]:=eps22;
mmpux12[i]:=mpux12;
mmfri12[i]:=mfri12;
end;
     for n:=0 to 360 do
        begin
  StringGrid1.cells[0,n+1]:=format('%9.4f',[mt[n]]);
  StringGrid1.cells[1,n+1]:=format('%9.4f',[momg11[n]]);
  StringGrid1.cells[2,n+1]:=format('%9.4f',[mfi11[n]]);
  StringGrid1.cells[3,n+1]:=format('%9.4f',[meps11[n]]);
  StringGrid1.cells[4,n+1]:=format('%9.4f',[momg21[n]]);
  StringGrid1.cells[5,n+1]:=format('%9.4f',[mfi21[n]]);
  StringGrid1.cells[6,n+1]:=format('%9.4f',[meps21[n]]);
  StringGrid1.cells[7,n+1]:=format('%9.4f',[mmpux11[n]]);
  StringGrid1.cells[8,n+1]:=format('%9.4f',[mmfri11[n]]);
  StringGrid1.cells[9,n+1]:=format('%9.4f',[momg12[n]]);
  StringGrid1.cells[10,n+1]:=format('%9.4f',[mfi12[n]]);
  StringGrid1.cells[11,n+1]:=format('%9.4f',[meps12[n]]);
  StringGrid1.cells[12,n+1]:=format('%9.4f',[momg22[n]]);
  StringGrid1.cells[13,n+1]:=format('%9.4f',[mfi22[n]]);
  StringGrid1.cells[14,n+1]:=format('%9.4f',[meps22[n]]);
  StringGrid1.cells[15,n+1]:=format('%9.4f',[mmpux12[n]]);
  StringGrid1.cells[16,n+1]:=format('%9.4f',[mmfri12[n]]);
        end;
```

```
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
 function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
```

```
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
 end;
procedure TForm2.Button4Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm3 = class(TForm)
```

```
Button1: TButton;
StringGrid1: TStringGrid;
Button2: TButton;
Button3: TButton;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Chart1: TChart;
Chart2: TChart;
Button4: TButton;
Series1: TLineSeries;
Series2: TLineSeries;
Series3: TLineSeries;
Series4: TLineSeries;
Series5: TLineSeries;
Series6: TLineSeries;
Series7: TLineSeries;
Series8: TLineSeries;
Series9: TLineSeries;
Series10: TLineSeries;
Series11: TLineSeries;
Series12: TLineSeries;
Button5: TButton;
Edit4: TEdit;
Label4: TLabel;
procedure Button1Click(Sender: TObject);
```

```
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form3: TForm3;
  i,j,n:Integer;
 l1,l2,l3,l4,xA,yA,U1,A1,AC,p,r,A2,U3,U2,xB,yB,xAB,yAB:Real;
  US2,UU2,US3,UU3:Real;
  vAx,vAy,wAx,wAy,vBx,vBy,wBx,wBy:Real;
  vABx,vABy,wABx,wABy:Real;
  mxA,myA,mU1,mxB,myB,mxAB,myAB,mvAx,mvAy,mwAx,mwAy:array
[0..361] of Real;
  mUS2,mUU2,mUS3,mUU3:array [0..361] of Real;
  mvBx,mvBy,mwBx,mwBy,mvABx,mvABy,mwABx,mwABy:array [0..361]
of Real;
implementation
 uses unit2;
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
Form3.Hide;
Form2.Show;
```

```
end;
rocedure TForm3.Button4Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart1.SeriesList[4].Clear;
Chart1.SeriesList[5].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart2.SeriesList[4].Clear;
Chart2.SeriesList[5].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mxB[j],",clRed);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,myB[j],",clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mxAB[j],",clYellow);
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,myAB[j],",clBlue);
Chart1.SeriesList[4].AddXY(mU1[j]*180/Pi,mUS2[j]/100,",clWhite);
Chart1.SeriesList[5].AddXY(mU1[j]*180/Pi,mUS3[j]/100,",clAqua);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mvBx[j],",clWhite);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mvBy[j],",clGreen);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mvABx[j],",clRed);
Chart2.SeriesList[3].AddXY(mU1[j]*180/Pi,mvABy[j],",clBlue);
Chart2.SeriesList[4].AddXY(mU1[j]*180/Pi,mwABx[j]/100,",clYellow);
Chart2.SeriesList[5].AddXY(mU1[j]*180/Pi,mwABy[j]/100,",clAqua);
```

```
end;
end;
procedure TForm3.Button5Click(Sender: TObject);
begin
Chart1.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\ПРИХОДЬКО_ В_Ю\Приходько В Ю\Example\5.bmp');
Chart2.SaveToBitmapFile('C:\Users\Вова\Documents\Диплом_2020\Стаціона
р\1 КОГОРТА\ПРИХОДЬКО_ В_Ю\Приходько В Ю\Example\6.bmp');
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
        Val(Edit1.text,l1,code);
        Val(Edit2.text,l2,code);
        Val(Edit3.text,l3,code);
        Val(Edit4.text,l4,code);
            begin
       StringGrid1.cells[0,0]:='U1';
       StringGrid1.cells[1,0]:='xB';
       StringGrid1.cells[2,0]:='yB';
       StringGrid1.cells[3,0]:='xAB';
       StringGrid1.cells[4,0]:='yAB';
       StringGrid1.cells[5,0]:='US2';
       StringGrid1.cells[6,0]:='US3';
       StringGrid1.cells[7,0]:='UU2';
       StringGrid1.cells[8,0]:='UU3';
       StringGrid1.cells[9,0]:='vBx';
       StringGrid1.cells[10,0]:='vBy';
       StringGrid1.cells[11,0]:='vABx';
       StringGrid1.cells[12,0]:='vABy';
```

```
 StringGrid1.cells[13,0]:='wBx';
 StringGrid1.cells[14,0]:='wBy';
 StringGrid1.cells[15,0]:='wABx';
 StringGrid1.cells[16,0]:='wABy';
            end;
        for i:=0 to 360 do
begin
U1:=i*Pi/180;
xA:=l1*cos(U1);
yA:=l1*sin(U1);
vAx:=-omg*l1*sin(U1);
vAy:=omg*l1*cos(U1);
wAx:=-sqr(omg)*l1*cos(U1);
wAy:=-sqr(omg)*l1*sin(U1);
mxA[i]:=xA;
myA[i]:=yA;
mvAx[i]:=vAx;
mvAy[i]:=vAy;
mwAx[i]:=wAx;
mwAy[i]:=wAy;
end;
     for i:=0 to 360 do
begin
   U1:=i*Pi/180;
  A1:=arcTan((myA[i])/(l4-mxA[i]));
    if A1<>0 then
       begin
    AC:=(myA[i])/sin(A1);
       end
```

```
            else
          begin
          AC:=l4-mxA[i];
          end;
       p:=(l2+l3+AC)/2;
       r:=sqrt((p-l2)*(p-l3)*(p-AC)/p);
       A2:=2*arcTan(r/(p-l2));
          U3:=2*Pi-(A1+A2);
       xB:=l4-l3*cos(U3);
       yB:=-l3*sin(U3);
  U2:=arcTan(sqrt(sqr(-l3*sin(U3)-myA[i]))/sqrt(sqr(l4-l3*cos(U3)-mxA[i])));
      xAB:=mxA[i]+(l2/2)*cos(U2); yAB:=myA[i]+(l2/2)*sin(U2);
      US3:=(-mvAx[i]*cos(U2)-mvAy[i]*sin(U2))/(l3*(cos(U3)*sin(U2)-
cos(U2)*sin(U3)));
      US2:= (mvAx[i]/(l2*sin(U2)))-((US3*l3*sin(U3))/(l2*sin(U2)));
      UU2:= (-l2*US2*US2*cos(U2)*cos(U3)-
l3*US3*US3*cos(U3)*sin(U3)+mwAx[i]*cos(U3)+mwAy[i]*sin(U3)-
l2*US2*US2*sin(U2)*sin(U3))/(l2*(sin(U2)*cos(U3)-cos(U2)*sin(u3)));
      UU3:=(mwAx[i]-l2*UU2*sin(U2)-l2*sqr(US2)*cos(U2)-
l3*sqr(US3)*cos(U3))/(l3*sin(U3));
      vBx:=l3*US3*sin(U3); vBy:=-l3*US3*cos(U3);
      vABx:=mvAx[i]-(l2/2)*US2*sin(U2);
vABy:=mvAy[i]+(l2/2)*US2*cos(U2);
      wBx:=l3*UU3*sin(U3)+l3*sqr(US3)*cos(U3);
      wBy:=-l3*UU3*cos(U3)+l3*sqr(US3)*sin(U3);
      wABx:=mwAx[i]-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
      wABy:=mwAy[i]+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
```

```
  mU1[i]:=U1;
  mxB[i]:=xB;
  myB[i]:=yB;
  mxAB[i]:=xAB;
  myAB[i]:=yAB;
  mUS2[i]:=US2;
  mUS3[i]:=US3;
  mUU2[i]:=UU2;
  mUU3[i]:=UU3;
  mvBx[i]:=vBx;
  mvBy[i]:=vBy;
  mvABx[i]:=vABx;
  mvABy[i]:=vABy;
  mwBx[i]:=wBx;
  mwBy[i]:=wBy;
  mwABx[i]:=wABx;
  mwABy[i]:=wABy;

end;
      for n:=0 to 360 do
        begin
    StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
    StringGrid1.cells[1,n+1]:=format('%9.4f',[mxB[n]]);
    StringGrid1.cells[2,n+1]:= format('%9.4f',[myB[n]]);
    StringGrid1.cells[3,n+1]:= format('%9.4f',[mxAB[n]]);
    StringGrid1.cells[4,n+1]:=format('%9.4f',[myAB[n]]);
    StringGrid1.cells[5,n+1]:=format('%9.4f',[mUS2[n]]);
    StringGrid1.cells[6,n+1]:=format('%9.4f',[mUS3[n]]);
    StringGrid1.cells[7,n+1]:= format('%9.4f',[mUU2[n]]);
```

```
            StringGrid1.cells[8,n+1]:=format('%9.4f',[mUU3[n]]);
            StringGrid1.cells[9,n+1]:=format('%9.4f',[mvBx[n]]);
            StringGrid1.cells[10,n+1]:=format('%9.4f',[mvBy[n]]);
            StringGrid1.cells[11,n+1]:=format('%9.4f',[mvABx[n]]);
            StringGrid1.cells[12,n+1]:=format('%9.4f',[mvABy[n]]);
            StringGrid1.cells[13,n+1]:=format('%9.4f',[mwBx[n]]);
            StringGrid1.cells[14,n+1]:=format('%9.4f',[mwBy[n]]);
            StringGrid1.cells[15,n+1]:=format('%9.4f',[mwABx[n]]);
            StringGrid1.cells[16,n+1]:=format('%9.4f',[mwABy[n]]);
                end;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
```

```
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
const
  {$J+}   CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
var
  FStream: TFileStream;
  I, J: Integer;
begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
end;
procedure TForm3.Button3Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
```

```
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
```

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

системи розрахунку динамічної моделі об'ємного накопичувача

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm3 = class(TForm)
    Button1: TButton;
    StringGrid1: TStringGrid;
    Button2: TButton;
    Button3: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Label4: TLabel;
    Edit4: TEdit;
    Button4: TButton;
    Chart1: TChart;
```

```
  Chart2: TChart;
  Series1: TLineSeries;
  Series2: TLineSeries;
  Series3: TLineSeries;
  Series4: TLineSeries;
  Series6: TLineSeries;
  Series7: TLineSeries;
  Series8: TLineSeries;
  Button5: TButton;
  Chart3: TChart;
  Chart4: TChart;
  Series5: TLineSeries;
  Series9: TLineSeries;
  Series10: TLineSeries;
  Series11: TLineSeries;
  Series12: TLineSeries;
  Series13: TLineSeries;
  Series14: TLineSeries;
  Series15: TLineSeries;
  Label5: TLabel;
  Label6: TLabel;
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure Button5Click(Sender: TObject);
private
  { Private declarations }
public
```

```
   { Public declarations }
  end;
var
  Form3: TForm3;
  i,j,n,code: Integer;
  l1,l3,l4,lD3,US1,U1,xA,yA,U3,k1,k2:Real;
  vAx,vAy,wAx,wAy,US3,UU3:Real;
  xlD3,ylD3,vlD3x,vlD3y:Real;
  wlD3x,wlD3y:Real;
  mU1,mxA,myA,mU3,mxlD3,mylD3:array [1..361] of Real;
mvAx,mvAy,mwAx,mwAy,mUS3,mUU3,mvlD3x,mvlD3y,mwlD3x,mwlD3y:a
rray [1..361] of Real;
 implementation
 uses unit2;
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
Form3.Hide;
Form2.Show;
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
   Val(Edit1.text,l1,code);
   Val(Edit2.text,lD3,code);
   Val(Edit3.text,l4,code);
   Val(Edit4.text,US1,code);
          begin
        StringGrid1.cells[0,0]:='uU1';
        StringGrid1.cells[1,0]:='uxA';
```

```
StringGrid1.cells[2,0]:='uyA';
StringGrid1.cells[3,0]:='uxlD3';
StringGrid1.cells[4,0]:='uylD3';
StringGrid1.cells[5,0]:='uU3';
StringGrid1.cells[6,0]:='uvAx';
StringGrid1.cells[7,0]:='uvAy';
StringGrid1.cells[8,0]:='uwAx';
StringGrid1.cells[9,0]:='uwAy';
StringGrid1.cells[10,0]:='uUS3';
StringGrid1.cells[11,0]:='uUU3';
StringGrid1.cells[12,0]:='uvlD3x';
StringGrid1.cells[13,0]:='uvlD3y';
StringGrid1.cells[14,0]:='uwlD3x';
StringGrid1.cells[15,0]:='uwlD3y';

 end;
 for i:=0 to 360 do
 begin
 U1:=i*Pi/180;
 xA:=l1*cos(U1);
 yA:=l1*sin(U1);
 vAx:= -US1*l1*sin(U1);
 vAy:=US1*l1*cos(U1);
 wAx:=-sqr(US1)*l1*cos(U1);
 wAy:=-sqr(US1)*l1*sin(U1);
 if yA-l4<0 then
    begin
    U3:=arcTan(xA/(yA-l4));
    end
```

```
          else
      begin
         if yA-l4>0 then
      begin
       U3:=Pi+arcTan(xA/(yA-l4));
      end
                  else
       begin
         if xA>0 then
       begin
        U3:=3*Pi/2;
       end
                 else
       begin
         U3:=Pi/2;
       end;
    end;
  end;
if U3<>0 then
 begin
 l3:=-xA/sin(U3);
 end
        else
 begin
  l3:=l4-yA;
 end;
xlD3:=(lD3/2)*sin(U3);
ylD3:=-(lD3/2)*cos(U3);
k1:=l4-yA;
```

```
        k2:=(sqr(xA)/sqr(k1))+1;
        US3:= ((vAx/k1)+(xA*vAy/sqr(k1)))/k2;
        vlD3x:=(lD3/2)*US3*cos(U3);
        vlD3y:=(lD3/2)*US3*sin(U3);
        UU3:=
(((wAx/k1)+(vAy*vAx/sqr(k1))+(wAy*xA/sqr(k1))+(vAx*vAy/sqr(k1))+(2*v
Ay*xA*vAy/(k1*sqr(k1))))/k2)-
((((2*xA*vAx/sqr(k1))+(2*sqr(xA)*vAy/(k1*sqr(k1))))*((vAx/k1)+vAy*xA/sq
r(k1)))/sqr(k2));
        wlD3x:=(lD3/2)*UU3*cos(U3)-(lD3/2)*sqr(US3)*sin(U3);
        wlD3y:=(lD3/2)*UU3*sin(U3)+(lD3/2)*sqr(US3)*cos(U3);
        mU1[i]:=U1;
        mxA[i]:=xA;
        myA[i]:=yA;
        mxlD3[i]:=xlD3;
        mylD3[i]:=ylD3;
        mU3[i]:=U3;
        mvAx[i]:=vAx;
        mvAy[i]:=vAy;
        mwAx[i]:=wAx;
        mwAy[i]:=wAy;
        mUS3[i]:=US3;
        mUU3[i]:=UU3;
        mvlD3x[i]:=vlD3x;
        mvlD3y[i]:=vlD3y;
        mwlD3x[i]:=wlD3x;
        mwlD3y[i]:=wlD3y;
        end;
           for n:=0 to 360 do
```

```
            begin
        StringGrid1.cells[0,n+1]:=format('%9.4f',[mU1[n]]);
        StringGrid1.cells[1,n+1]:=format('%9.4f',[mxA[n]]);
        StringGrid1.cells[2,n+1]:=format('%9.4f',[myA[n]]);
        StringGrid1.cells[3,n+1]:=format('%9.4f',[mxlD3[n]]);
        StringGrid1.cells[4,n+1]:=format('%9.4f',[mylD3[n]]);
        StringGrid1.cells[5,n+1]:=format('%9.4f',[mU3[n]]);
        StringGrid1.cells[6,n+1]:=format('%9.4f',[mvAx[n]]);
        StringGrid1.cells[7,n+1]:=format('%9.4f',[mvAy[n]]);
        StringGrid1.cells[8,n+1]:=format('%9.4f',[mwAx[n]]);
        StringGrid1.cells[9,n+1]:=format('%9.4f',[mwAy[n]]);
        StringGrid1.cells[10,n+1]:=format('%9.4f',[mUS3[n]]);
        StringGrid1.cells[11,n+1]:= format('%9.4f',[mUU3[n]]);
        StringGrid1.cells[12,n+1]:=format('%9.4f',[mvlD3x[n]]);
        StringGrid1.cells[13,n+1]:=format('%9.4f',[mvlD3y[n]]);
        StringGrid1.cells[14,n+1]:=format('%9.4f',[mwlD3x[n]]);
        StringGrid1.cells[15,n+1]:=format('%9.4f',[mwlD3y[n]]);
            end;
end;
procedure TForm3.Button4Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
```

```
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
Chart4.SeriesList[0].Clear;
Chart4.SeriesList[1].Clear;
Chart4.SeriesList[2].Clear;
Chart4.SeriesList[3].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mU1[j]*180/Pi,mxA[j],",clRed);
Chart1.SeriesList[1].AddXY(mU1[j]*180/Pi,myA[j],",clGreen);
Chart1.SeriesList[2].AddXY(mU1[j]*180/Pi,mxlD3[j],",clYellow);
Chart1.SeriesList[3].AddXY(mU1[j]*180/Pi,mylD3[j],",clBlue);
Chart2.SeriesList[0].AddXY(mU1[j]*180/Pi,mU3[j],",clBlue);
Chart2.SeriesList[1].AddXY(mU1[j]*180/Pi,mUS3[j],",clWhite);
Chart2.SeriesList[2].AddXY(mU1[j]*180/Pi,mUU3[j],",clAqua);
Chart3.SeriesList[0].AddXY(mU1[j]*180/Pi,mvAx[j],",clRed);
Chart3.SeriesList[1].AddXY(mU1[j]*180/Pi,mvAy[j],",clGreen);
Chart3.SeriesList[2].AddXY(mU1[j]*180/Pi,mvlD3x[j],",clYellow);
Chart3.SeriesList[3].AddXY(mU1[j]*180/Pi,mvlD3y[j],",clBlue);
Chart4.SeriesList[0].AddXY(mU1[j]*180/Pi,mwAx[j],",clWhite);
Chart4.SeriesList[1].AddXY(mU1[j]*180/Pi,mwAy[j],",clGreen);
Chart4.SeriesList[2].AddXY(mU1[j]*180/Pi,mwlD3x[j],",clYellow);
Chart4.SeriesList[3].AddXY(mU1[j]*180/Pi,mwlD3y[j],",clBlue);
end;
end;
procedure TForm3.Button5Click(Sender: TObject);
begin
```

```
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
```

```
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
 end;
procedure TForm3.Button3Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
end;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm2 = class(TForm)
    Button1: TButton;
```

Button2: TButton;

StringGrid1: TStringGrid;

Button3: TButton;

Button4: TButton;

Chart1: TChart;

Button5: TButton;

Series1: TLineSeries;

Series2: TLineSeries;

Series3: TLineSeries;

Series4: TLineSeries;

Edit1: TEdit;

Edit2: TEdit;

Edit3: TEdit;

Edit4: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Chart2: TChart;

Series5: TLineSeries;

Series6: TLineSeries;

Series7: TLineSeries;

Chart3: TChart;

Series9: TLineSeries;

Series10: TLineSeries;

Series11: TLineSeries;

Button6: TButton;

Edit5: TEdit;

Label5: TLabel;

```
    Edit6: TEdit;
    Label6: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
  i,j,n,code: Integer;
  l,b,b1,a,a1,s:Real;
  ro,gam,alf:Real;
  MK,NK,f,Z,Y:Real;
  PZ,PY,Vabs,sigm:Real;
  mro,mgam,malf: array [0..360] of Real;
  mMK,mNK,mf,mZ,mY: array [0..360] of Real;
  mPZ,mPY,mVabs,msigm: array [0..360] of Real;
implementation
 uses unit1, unit3;
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
Form1.Close;
```

```
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
Form2.Hide;
Form3.Show;
end;
procedure TForm2.Button5Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
  for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mro[j],mgam[j],'',clRed);
Chart1.SeriesList[1].AddXY(mro[j],malf[j],'',clGreen);
Chart1.SeriesList[2].AddXY(mro[j],mMK[j],'',clYellow);
Chart1.SeriesList[3].AddXY(mro[j],mNK[j],'',clBlue);
Chart2.SeriesList[0].AddXY(mro[j],mf[j],'',clWhite);
Chart2.SeriesList[1].AddXY(mro[j],mZ[j],'',clGreen);
Chart2.SeriesList[2].AddXY(mro[j],mY[j],'',clRed);
```

```
Chart3.SeriesList[0].AddXY(mro[j],mPZ[j],'',clBlue);
Chart3.SeriesList[1].AddXY(mro[j],mPY[j],'',clGreen);
Chart3.SeriesList[2].AddXY(mro[j],mVabs[j],'',clYellow);


end;
end;
procedure TForm2.Button6Click(Sender: TObject);
begin
end;
procedure TForm2.Button3Click(Sender: TObject);
begin
        Val(Edit1.text,l,code);
        Val(Edit2.text,b,code);
        Val(Edit3.text,b1,code);
        Val(Edit4.text,a,code);
        Val(Edit5.text,a1,code);
        Val(Edit6.text,s,code);
                begin
       StringGrid1.cells[0,0]:='do';
       StringGrid1.cells[1,0]:='vssbv';
       StringGrid1.cells[2,0]:='sbv03';
       StringGrid1.cells[3,0]:='vibv7';
       StringGrid1.cells[4,0]:='vsbv4';
       StringGrid1.cells[5,0]:='lsvi1';
       StringGrid1.cells[6,0]:='jsv7';
       StringGrid1.cells[7,0]:='ksv4';
       StringGrid1.cells[8,0]:='lsl1';
       StringGrid1.cells[9,0]:='bsl7';
       StringGrid1.cells[10,0]:='bsl4';
```

```
StringGrid1.cells[11,0]:='bs03';
        end;
 ro:=0;
 for i:=0 to 360 do
 begin
 ro:=ro+(12/360);
 alf:=0.96*(l/b)*exp(0.2*ln(ro));
 gam:=0.94*(l/b)*exp(0.3*ln(ro));
 MK:= 0.98*(l/b)*exp(0.5*ln(ro));
 NK:= 0.97*(l/b)*exp(0.9*ln(ro));
 f:= 0.91*(b1/a)*exp(0.2*ln(ro));
 Z:= 0.99*(b1/a)*exp(0.5*ln(ro));
 Y:= 0.92*(b1/a)*exp(0.9*ln(ro));
 PZ:= 0.93*(a1/s)*exp(0.2*ln(ro));
 PY:= 0.97*(a1/s)*exp(0.7*ln(ro));
 Vabs:= 0.94*(a1/s)*exp(0.9*ln(ro));
 sigm:=(a1/s)*exp(1.8*ln(ro));
 mro[i]:=ro;
 mgam[i]:=gam;
 malf[i]:=alf;
 mMK[i]:=MK;
 mNK[i]:=NK;
 mf[i]:=f;
 mZ[i]:=Z;
 mY[i]:=Y;
 mPZ[i]:=PZ;
 mPY[i]:=PY;
 mVabs[i]:=Vabs;
 msigm[i]:=sigm;
```

```
        end;
            for n:=0 to 360 do
              begin
          StringGrid1.cells[0,n+1]:=format('%9.4f',[mro[n]]);
          StringGrid1.cells[1,n+1]:=format('%9.4f',[malf[n]]);
          StringGrid1.cells[2,n+1]:=format('%9.4f',[mgam[n]]);
          StringGrid1.cells[3,n+1]:=format('%9.4f',[mMK[n]]);
          StringGrid1.cells[4,n+1]:=format('%9.4f',[mNK[n]]);
          StringGrid1.cells[5,n+1]:=format('%9.4f',[mf[n]]);
          StringGrid1.cells[6,n+1]:=format('%9.4f',[mZ[n]]);
          StringGrid1.cells[7,n+1]:=format('%9.4f',[mY[n]]);
          StringGrid1.cells[8,n+1]:=format('%9.4f',[mPZ[n]]);
          StringGrid1.cells[9,n+1]:=format('%9.4f',[mPY[n]]);
          StringGrid1.cells[10,n+1]:=format('%9.4f',[mVabs[n]]);
          StringGrid1.cells[11,n+1]:=format('%9.4f',[msigm[n]]);
                        end;
        end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
var
  L: Word;
const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
```

```
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
  finally
    FStream.Free;
  end;
end;
```

```
procedure TForm2.Button4Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
```

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

системи розрахунку навантажень в жорсткій на вигин нитки постійної кривини

```
unit Unit2;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm2 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    StringGrid1: TStringGrid;
    Button3: TButton;
    Button4: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
```

```
Edit4: TEdit;
Edit5: TEdit;
Edit6: TEdit;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Button5: TButton;
Chart1: TChart;
Series1: TLineSeries;
Series2: TLineSeries;
Chart2: TChart;
Series3: TLineSeries;
Series4: TLineSeries;
Series5: TLineSeries;
Chart3: TChart;
Series7: TLineSeries;
Series8: TLineSeries;
Series9: TLineSeries;
Series10: TLineSeries;
Button6: TButton;
Series11: TLineSeries;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
```

```
  procedure Button6Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 Form2: TForm2;
 i,j,n,code: Integer;
 r,rb,L,kpua,a,b,c,ro:Real;
 D,EPS,sig1,NR:Real;
 AK,BK,KK,UK,QP,FI,PFI,MR1,PSI:Real;
 KK1,UK1,FI1,PFI1:Real;
mr,msig1,mNR : array [0..360] of Real;
mAK,mBK,mKK,mUK,mQP,mFI,mPFI,mMR1,mPSI: array [0..360] of Real;
 mKK1,mUK1,mFI1,mPFI1: array [0..360] of Real;
implementation
 uses unit1, unit3;
{$R *.dfm}
procedure TForm2.Button1Click(Sender: TObject);
begin
Form1.Close;
end;
procedure TForm2.Button2Click(Sender: TObject);
begin
Form2.Hide;
Form3.Show;
end;
```

```
procedure TForm2.Button5Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
Chart3.SeriesList[3].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(mr[j],msig1[j],",clRed);
Chart1.SeriesList[1].AddXY(mr[j],mFI1[j],",clGreen);
Chart1.SeriesList[2].AddXY(mr[j],mPFI1[j],",clYellow);
Chart2.SeriesList[0].AddXY(mr[j],mNR[j],",clWhite);
Chart2.SeriesList[1].AddXY(mr[j],mAK[j],",clGreen);
Chart2.SeriesList[2].AddXY(mr[j],mMR1[j],",clRed);
Chart3.SeriesList[0].AddXY(mr[j],mUK1[j],",clBlue);
Chart3.SeriesList[1].AddXY(mr[j],mKK1[j],",clGreen);
Chart3.SeriesList[2].AddXY(mr[j],mQP[j],",clYellow);
Chart3.SeriesList[3].AddXY(mr[j],mPSI[j],",clRed);
end;
end;
procedure TForm2.Button6Click(Sender: TObject);
begin
```

```
end;
procedure TForm2.Button3Click(Sender: TObject);
begin
        Val(Edit1.text,rb,code);
        Val(Edit2.text,L,code);
        Val(Edit3.text,kpua,code);
        Val(Edit4.text,a,code);
        Val(Edit5.text,b,code);
        Val(Edit6.text,c,code);
                begin
       StringGrid1.cells[0,0]:='kr';
       StringGrid1.cells[1,0]:='psig';
       StringGrid1.cells[2,0]:='pNR';
       StringGrid1.cells[3,0]:='pAK';
       StringGrid1.cells[4,0]:='pKK';
       StringGrid1.cells[5,0]:='p1UK';
       StringGrid1.cells[6,0]:='pQP';
       StringGrid1.cells[7,0]:='pFI';
       StringGrid1.cells[8,0]:='pPFI';
       StringGrid1.cells[9,0]:='pMR';
       StringGrid1.cells[10,0]:='pPSI';
                        end;
        r:=0.4;
        for i:=0 to 360 do
       begin
       r:=r+(12/360);
       ro:=(r/rb);
```

```
{31}    UK1:=rb*exp(((r-0.2)/r)*L*kpua);
{32}    KK1:=UK1/r;
{33}    QP:=((a*b)/c)*KK1;
{34}    PSI:=UK1*12/(2*r);
 {11}     sig1:=rb*exp(((r-0.4)/r)*L*kpua);
 {12}     FI1:=sig1/r;
 {13}     PFI1:=((a*b)/c)*FI1;
 {21}     NR:=rb*exp(((r-0.6)/r)*L*kpua);
  {22}      AK:=NR/r;
 {23}     MR1:=((a*b)/c)*AK;
        D:=r*exp(6*ln(10));
        EPS:=r*8.158;
        BK:=sqrt(r)*(a/2);
        KK:=r*sin(r)*(b/(3*r));
        UK:=sqr(a)*c/(4*sqr(r));
        FI:=(AK/10)*sin(r/2);
        PFI:=cos(BK*KK)*(rb/r)+UK1;
      mr[i]:=r;
      msig1[i]:=sig1;
      mNR[i]:=NR;
      mAK[i]:=AK;
      mBK[i]:=BK;
      mKK[i]:=KK;
      mKK1[i]:=KK1;
      mUK[i]:=UK;
      mUK1[i]:=UK1;
      mQP[i]:=QP;
      mFI[i]:=FI;
      mFI1[i]:=FI1;
```

```
        mPFI[i]:=PFI;
        mPFI1[i]:=PFI1;
        mMR1[i]:=MR1;
        mPSI[i]:=PSI;

      end;

          for n:=0 to 360 do
            begin
        StringGrid1.cells[0,n+1]:=format('%9.4f',[mr[n]]);
        StringGrid1.cells[1,n+1]:=format('%9.4f',[msig1[n]]);
        StringGrid1.cells[2,n+1]:=format('%9.4f',[mNR[n]]);
        StringGrid1.cells[3,n+1]:=format('%9.4f',[mAK[n]]);
        StringGrid1.cells[4,n+1]:=format('%9.4f',[mKK1[n]]);
        StringGrid1.cells[5,n+1]:=format('%9.4f',[mUK1[n]]);
        StringGrid1.cells[6,n+1]:=format('%9.4f',[mQP[n]]);
        StringGrid1.cells[7,n+1]:=format('%9.4f',[mFI1[n]]);
        StringGrid1.cells[8,n+1]:=format('%9.4f',[mPFI1[n]]);
        StringGrid1.cells[9,n+1]:=format('%9.4f',[mMR1[n]]);
        StringGrid1.cells[10,n+1]:=format('%9.4f',[mPSI[n]]);
            end;
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
```

```
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
    CXlsBof[4] := 0;
    FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
    for i := 0 to AGrid.ColCount - 1 do
      for j := 0 to AGrid.RowCount - 1 do
        XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
    FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
    Result := True;
```

```
  finally
    FStream.Free;
  end;
 end;
procedure TForm2.Button4Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
    {ShowMessage('StringGrid saved!')};
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,ComObj ,
Forms,
  Dialogs, StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart,Math,
  Grids,Jpeg,Printers, ShellAPI;
type
  TForm3 = class(TForm)
    Button1: TButton;
    StringGrid1: TStringGrid;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Chart1: TChart;
    Chart2: TChart;
```

Chart3: TChart;

Button4: TButton;

Series1: TLineSeries;

Series2: TLineSeries;

Series3: TLineSeries;

Series4: TLineSeries;

Series5: TLineSeries;

Series6: TLineSeries;

Series7: TLineSeries;

Series8: TLineSeries;

Series9: TLineSeries;

Series10: TLineSeries;

Button5: TButton;

Edit3: TEdit;

Label1: TLabel;

Edit4: TEdit;

Label4: TLabel;

Edit5: TEdit;

Label5: TLabel;

Edit6: TEdit;

Label6: TLabel;

Series11: TLineSeries;

Series12: TLineSeries;

Series13: TLineSeries;

Series14: TLineSeries;

Series15: TLineSeries;

Series16: TLineSeries;

Series17: TLineSeries;

Series18: TLineSeries;

```
    Series19: TLineSeries;
    Series20: TLineSeries;
    Series21: TLineSeries;
    Series22: TLineSeries;
    Series23: TLineSeries;
    Series24: TLineSeries;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form3: TForm3;
  i,j,n,code: Integer;
l1,l2,l3,l4,l5,A1,AC,p,r,A2,U3,US1,U1,xA,yA,U2,xB,yB,xAB,yAB,xCB,yCB:
Real;
  vAx,vAy,wAx,wAy,US2,UU2,US3,UU3,k1,k2:Real;
  vBx,vBy,wBx,wBy,vCBx,vCBy:Real;
  vABx,vABy,wABx,wABy,wCBx,wCBy:Real;
  mU1,mxA,myA,mU2,mU3,mxB,myB,mxAB,myAB,mxCB,myCB:array
[1..361] of Real;
  mvAx,mvAy,mwAx,mwAy,mUS2,mUU2,mUS3,mUU3:array [1..361] of
Real;
```

```
 mvBx,mvBy,mwBx,mwBy,mvABx,mvABy,mwABx,mwABy:array [1..361]
of Real;
 mvCBx,mvCBy,mwCBx,mwCBy:array [1..361] of Real;
implementation
 uses unit2;
{$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
Form3.Hide;
Form2.Show;
end;
procedure TForm3.Button2Click(Sender: TObject);
begin
        Val(Edit1.text,US1,code);
        Val(Edit2.text,l1,code);
        Val(Edit3.text,l2,code);
        Val(Edit4.text,l3,code);
        Val(Edit5.text,l4,code);
        Val(Edit6.text,l5,code);
            begin
       StringGrid1.cells[0,0]:='kU1';
       StringGrid1.cells[1,0]:='kU2';
       StringGrid1.cells[2,0]:='kU3';
       StringGrid1.cells[3,0]:='kxB';
       StringGrid1.cells[4,0]:='kyB';
       StringGrid1.cells[5,0]:='kxAB';
       StringGrid1.cells[6,0]:='kyAB';
       StringGrid1.cells[7,0]:='kxCB';
       StringGrid1.cells[8,0]:='kyCB';
```

```
StringGrid1.cells[9,0]:='kUS2';
StringGrid1.cells[10,0]:='kUS3';
StringGrid1.cells[11,0]:='kUU2';
StringGrid1.cells[12,0]:='kUU3';
StringGrid1.cells[13,0]:='kvBx';
StringGrid1.cells[14,0]:='kvBy';
StringGrid1.cells[15,0]:='kvABx';
StringGrid1.cells[16,0]:='kvABy';
StringGrid1.cells[17,0]:='kvCBx';
StringGrid1.cells[18,0]:='kvCBy';
StringGrid1.cells[19,0]:='kwBx';
StringGrid1.cells[20,0]:='kwBy';
StringGrid1.cells[21,0]:='kwABx';
StringGrid1.cells[22,0]:='kwABy';
StringGrid1.cells[23,0]:='kwCBx';
StringGrid1.cells[24,0]:='kwCBy';
                end;
 for i:=0 to 360 do
 begin
 U1:=i*Pi/180;
 xA:=l1*cos(U1);
 yA:=l1*sin(U1);
 if l4-xA<>0 then
    begin
 A1:=arcTan((yA-l5)/(l4-xA));
    end
         else
    begin
  if yA>0 then
```

```
    begin
 A1:=Pi/2;
    end
       else
    begin
 A1:=3*Pi/2;
     end;
 end;
 if A1<>0 then
     begin
  AC:=(yA-l5)/sin(A1);
     end
         else
   if A1=0 then
     begin
     AC:=l4-xA;
  end;
p:=(l2+l3+AC)/2;
r:=sqrt((p-l2)*(p-l3)*(p-AC)/p);
A2:=2*arcTan(r/(p-l2));
U3:=2*Pi-(A1+A2);
xB:=l4-l3*cos(U3);
if l5<=0 then
     begin
 yB:=-l5-l3*sin(U3);
     end
        else
      begin
 yB:=l5-l3*sin(U3);
```

```
            end;
          if xB>xA then
              begin
           U2:=arcTan((yB-yA)/(xB-xA));
              end
                else
           if xB<xA then
              begin
           U2:=Pi+arcTan((yB-yA)/(xB-xA));
              end
                else
              begin
           U2:=Pi/2;
            end;
        xAB:=xA+(l2/2)*cos(U2); yAB:=yA+(l2/2)*sin(U2);
        xCB:=l4+(l3/2)*cos(U3); yCB:=(l3/2)*sin(U3)+l5;
        US2:=((sin(U1)/tan(U3))+cos(U1))*US1*l1/(l2*(cos(U2)-
(sin(U2)/tan(U3))));
        US3:=(-l1*US1*sin(U1)-l2*US2*sin(U2))/(l3*sin(U3));
        k1:=-l1*sqr(US1)*cos(U1)-l2*sqr(US2)*cos(U2)-l3*sqr(US3)*cos(U3);
        k2:=-l1*sqr(US1)*sin(U1)-l2*sqr(US2)*sin(U2)-l3*sqr(US3)*sin(U3);
        UU2:=(k2+(k1/tan(U3)))/(l2*((sin(U2)/tan(U3))-cos(U2)));
        UU3:=(k1-l2*UU2*sin(U2))/(l3*sin(U3));
        vAx:=-l1*US1*sin(U1); vAy:=l1*US1*cos(U1);
        vBx:=-l3*US3*sin(U3); vBy:=-l3*US3*cos(U3);
        vABx:=vAx-(l2/2)*US2*sin(U2); vABy:=vAy+(l2/2)*US2*cos(U2);
        vCBx:=-(l3/2)*US3*sin(U3); vCBy:=(l3/2)*US3*cos(U3);
        wAx:=-l1*sqr(US1)*cos(U1); wAy:=-l1*sqr(US1)*sin(U1);
        wBx:=-l3*UU3*sin(U3)-l3*sqr(US3)*cos(U3);
```

```
wBy:=l3*UU3*cos(U3)-l3*sqr(US3)*cos(U3);
wABx:=wAx-(l2/2)*UU2*sin(U2)-(l2/2)*sqr(US2)*cos(U2);
wABy:=wAy+(l2/2)*UU2*cos(U2)-(l2/2)*sqr(US2)*sin(U2);
wCBx:=-(l3/2)*UU3*sin(U3)-(l3/2)*sqr(US3)*cos(U3);
wCBy:=(l3/2)*UU3*cos(U3)-(l3/2)*sqr(US3)*sin(U3);
 mU1[i]:=U1;
 mxA[i]:=xA;
 myA[i]:=yA;
 mU2[i]:=U2;
 mU3[i]:=U3;
 mxB[i]:=xB;
 myB[i]:=yB;
 mxAB[i]:=xAB;
 myAB[i]:=yAB;
 mxCB[i]:=xCB;
 myCB[i]:=yCB;
 mUS2[i]:=US2;
 mUS3[i]:=US3;
 mUU2[i]:=UU2;
 mUU3[i]:=UU3;
 mvAx[i]:=vAx;
 mvAy[i]:=vAy;
 mvBx[i]:=vBx;
 mvBy[i]:=vBy;
 mvABx[i]:=vABx;
 mvABy[i]:=vABy;
 mvCBx[i]:=vCBx;
 mvCBy[i]:=vCBy;
 mwAx[i]:=wAx;
```

```
mwAy[i]:=wAy;
mwBx[i]:=wBx;
mwBy[i]:=wBy;
mwABx[i]:=wABx;
mwABy[i]:=wCBx;
mwCBx[i]:=wCBy;
mwCBy[i]:=wCBy;
end;
    for n:=0 to 360 do
        begin
  StringGrid1.cells[0,n+1]:=format('%3.0d',[n]);
  StringGrid1.cells[1,n+1]:=format('%9.4f',[mU2[n]]);
  StringGrid1.cells[2,n+1]:=format('%9.4f',[mU3[n]]);
  StringGrid1.cells[3,n+1]:=format('%9.4f',[mxB[n]]);
  StringGrid1.cells[4,n+1]:= format('%9.4f',[myB[n]]);
  StringGrid1.cells[5,n+1]:= format('%9.4f',[mxAB[n]]);
  StringGrid1.cells[6,n+1]:=format('%9.4f',[myAB[n]]);
  StringGrid1.cells[7,n+1]:=format('%9.4f',[mxCB[n]]);
  StringGrid1.cells[8,n+1]:=format('%9.4f',[myCB[n]]);
  StringGrid1.cells[9,n+1]:=format('%9.4f',[mUS2[n]]);
  StringGrid1.cells[10,n+1]:=format('%9.4f',[mUS3[n]]);
  StringGrid1.cells[11,n+1]:= format('%9.4f',[mUU2[n]]);
  StringGrid1.cells[12,n+1]:=format('%9.4f',[mUU3[n]]);
  StringGrid1.cells[13,n+1]:=format('%9.4f',[mvBx[n]]);
  StringGrid1.cells[14,n+1]:=format('%9.4f',[mvBy[n]]);
  StringGrid1.cells[15,n+1]:=format('%9.4f',[mvABx[n]]);
  StringGrid1.cells[16,n+1]:=format('%9.4f',[mvABy[n]]);
  StringGrid1.cells[17,n+1]:= format('%9.4f',[mvCBx[n]]);
  StringGrid1.cells[18,n+1]:= format('%9.4f',[mvCBy[n]]);
```

```
            StringGrid1.cells[19,n+1]:=format('%9.4f',[mwBx[n]]);
            StringGrid1.cells[20,n+1]:=format('%9.4f',[mwBy[n]]);
            StringGrid1.cells[21,n+1]:=format('%9.4f',[mwABx[n]]);
            StringGrid1.cells[22,n+1]:=format('%9.4f',[mwABy[n]]);
            StringGrid1.cells[23,n+1]:=format('%9.4f',[mwCBx[n]]);
            StringGrid1.cells[24,n+1]:= format('%9.4f',[mwCBy[n]]);
                        end;
end;
procedure TForm3.Button4Click(Sender: TObject);
begin
Chart1.SeriesList[0].Clear;
Chart1.SeriesList[1].Clear;
Chart1.SeriesList[2].Clear;
Chart1.SeriesList[3].Clear;
Chart1.SeriesList[4].Clear;
Chart1.SeriesList[5].Clear;
Chart1.SeriesList[6].Clear;
Chart1.SeriesList[7].Clear;
Chart2.SeriesList[0].Clear;
Chart2.SeriesList[1].Clear;
Chart2.SeriesList[2].Clear;
Chart2.SeriesList[3].Clear;
Chart2.SeriesList[4].Clear;
Chart2.SeriesList[5].Clear;
Chart2.SeriesList[6].Clear;
Chart2.SeriesList[7].Clear;
Chart3.SeriesList[0].Clear;
Chart3.SeriesList[1].Clear;
Chart3.SeriesList[2].Clear;
```

```
Chart3.SeriesList[3].Clear;
Chart3.SeriesList[4].Clear;
Chart3.SeriesList[5].Clear;
Chart3.SeriesList[6].Clear;
Chart3.SeriesList[7].Clear;
for j:=0 to 360 do
 begin
Chart1.SeriesList[0].AddXY(j,mxB[j],'',clYellow);
Chart1.SeriesList[1].AddXY(j,myB[j],'',clBlue);
Chart1.SeriesList[2].AddXY(j,mxAB[j],'',clAqua);
Chart1.SeriesList[3].AddXY(j,myAB[j],'',clBlack);
Chart1.SeriesList[4].AddXY(j,mxCB[j],'',clWhite);
Chart1.SeriesList[5].AddXY(j,myCB[j],'',clGreen);
Chart1.SeriesList[6].AddXY(j,mU2[j]/10,'',clRed);
Chart1.SeriesList[7].AddXY(j,mU3[j]/10,'',clGreen);
Chart2.SeriesList[0].AddXY(j,mvBx[j],'',clYellow);
Chart2.SeriesList[1].AddXY(j,mvBy[j],'',clBlue);
Chart2.SeriesList[2].AddXY(j,mvABx[j],'',clAqua);
Chart2.SeriesList[3].AddXY(j,mvABy[j],'',clBlack);
Chart2.SeriesList[4].AddXY(j,mvCBx[j],'',clWhite);
Chart2.SeriesList[5].AddXY(j,mvCBy[j],'',clGreen);
Chart2.SeriesList[6].AddXY(j,mUS2[j]/10,'',clRed);
Chart2.SeriesList[7].AddXY(j,mUS3[j]/10,'',clGreen);
Chart3.SeriesList[0].AddXY(j,mwBx[j],'',clYellow);
Chart3.SeriesList[1].AddXY(j,mwBy[j],'',clBlue);
Chart3.SeriesList[2].AddXY(j,mwABx[j],'',clAqua);
Chart3.SeriesList[3].AddXY(j,mwABy[j],'',clBlack);
Chart3.SeriesList[4].AddXY(j,mwCBx[j],'',clWhite);
Chart3.SeriesList[5].AddXY(j,mwCBy[j],'',clGreen);
```

```
Chart3.SeriesList[6].AddXY(j,mUU2[j]/100,'',clRed);
Chart3.SeriesList[7].AddXY(j,mUU3[j]/100,'',clGreen);
end;
end;
procedure TForm3.Button5Click(Sender: TObject);
begin
end;
procedure XlsWriteCellLabel(XlsStream: TStream; const ACol, ARow: Word;
  const AValue: string);
 var
  L: Word;
 const
  {$J+}
  CXlsLabel: array[0..5] of Word = ($204, 0, 0, 0, 0, 0);
  {$J-}
 begin
  L := Length(AValue);
  CXlsLabel[1] := 8 + L;
  CXlsLabel[2] := ARow;
  CXlsLabel[3] := ACol;
  CXlsLabel[5] := L;
  XlsStream.WriteBuffer(CXlsLabel, SizeOf(CXlsLabel));
  XlsStream.WriteBuffer(Pointer(AValue)^, L);
 end;
function SaveAsExcelFile(AGrid: TStringGrid; AFileName: string): Boolean;
 const
  {$J+} CXlsBof: array[0..5] of Word = ($809, 8, 00, $10, 0, 0); {$J-}
  CXlsEof: array[0..1] of Word = ($0A, 00);
 var
```

```
  FStream: TFileStream;
  I, J: Integer;
 begin
  Result := False;
  FStream := TFileStream.Create(PChar(AFileName), fmCreate or
fmOpenWrite);
  try
   CXlsBof[4] := 0;
   FStream.WriteBuffer(CXlsBof, SizeOf(CXlsBof));
   for i := 0 to AGrid.ColCount - 1 do
    for j := 0 to AGrid.RowCount - 1 do
     XlsWriteCellLabel(FStream, I, J, AGrid.cells[i, j]);
   FStream.WriteBuffer(CXlsEof, SizeOf(CXlsEof));
   Result := True;
  finally
   FStream.Free;
  end;
 end;
procedure TForm3.Button3Click(Sender: TObject);
begin
if SaveAsExcelFile(StringGrid1,  extractfilepath(paramstr(0))+
'\TableForPrint.xls') then
ShellExecute(Application.Handle,PChar('open'),PChar(extractfilepath(paramstr(
0))+ '\TableForPrint.xls'),Nil,Nil,SW_SHOW);
   {ShowMessage('StringGrid saved!')};
```

ПРОГРАМА

для чисельного інтегрування системи диференційних рівнянь

```
unit SDU2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,Math;
type
  TfrmSDU2 = class(TForm)
    btn1SDU2: TButton;
    lbl1SDU2: TLabel;
    lbl2SDU2: TLabel;
    lbl3SDU2: TLabel;
    lbl4SDU2: TLabel;
    lbl5SDU2: TLabel;
    lbl6SDU2: TLabel;
    lbl7SDU2: TLabel;
    lbl8SDU2: TLabel;
    Memo1: TMemo;
    edt1SDU2: TEdit;
    edt2SDU2: TEdit;
    edt3SDU2: TEdit;
    edt4SDU2: TEdit;
    edt5SDU2: TEdit;
    mem1SDU2: TMemo;
    btn2SDU2: TButton;
    btn3SDU2: TButton;
    procedure btn1SDU2Click(Sender: TObject);
    procedure btn3SDU2Click(Sender: TObject);
    procedure btn2SDU2Click(Sender: TObject);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;
 type mm=array[1..50]of Real;
var
  frmSDU2: TfrmSDU2;
  N,j,d1,code,i:Integer;
  x,h,xm,e1,e2,e3:Real;
  y,w,c,F,a,d,e:mm;
  Y0,xx,yy:array[1..50]of String;
  ii,xxx,jj:String;
implementation
  uses SDU1,Synt,UErrors;
{$R *.dfm}
     procedure V1(var F:mm; X:Real;y:mm);
    begin
       SetData('X',X);
     SetData('Y[1]',Y[1]);
     SetData('Y[2]',Y[2]);
     SetData('Y[3]',Y[3]);
     SetData('Y[4]',Y[4]);
     SetData('Y[5]',Y[5]);
     SetData('Y[6]',Y[6]);
     SetData('Y[7]',Y[7]);
     SetData('Y[8]',Y[8]);
     SetData('Y[9]',Y[9]);
     SetData('Y[10]',Y[10]);
     SetData('Y[11]',Y[11]);
```

```
SetData('Y[12]',Y[12]);
SetData('Y[13]',Y[13]);
SetData('Y[14]',Y[14]);
SetData('Y[15]',Y[15]);
SetData('Y[16]',Y[16]);
SetData('Y[17]',Y[17]);
SetData('Y[18]',Y[18]);
SetData('Y[19]',Y[19]);
SetData('Y[20]',Y[20]);
SetData('Y[21]',Y[21]);
SetData('Y[22]',Y[22]);
SetData('Y[23]',Y[23]);
SetData('Y[24]',Y[24]);
  for i:=1 to N do
    begin
 Calculate(F[i]);
    end;
  GetData('F[1]',F[1]);
  GetData('F[2]',F[2]);
  GetData('F[3]',F[3]);
  GetData('F[4]',F[4]);
  GetData('F[5]',F[5]);
  GetData('F[6]',F[6]);
  GetData('F[7]',F[7]);
  GetData('F[8]',F[8]);
  GetData('F[9]',F[9]);
  GetData('F[10]',F[10]);
  GetData('F[11]',F[11]);
  GetData('F[12]',F[12]);
```

```
    GetData('F[13]',F[13]);
    GetData('F[14]',F[14]);
    GetData('F[15]',F[15]);
    GetData('F[16]',F[16]);
    GetData('F[17]',F[17]);
    GetData('F[18]',F[18]);
    GetData('F[19]',F[19]);
    GetData('F[20]',F[20]);
    GetData('F[21]',F[21]);
    GetData('F[22]',F[22]);
    GetData('F[23]',F[23]);
    GetData('F[24]',F[24]);
  end;
procedure TfrmSDU2.btn1SDU2Click(Sender: TObject);
begin
frmSDU1.Close;
end;
procedure TfrmSDU2.btn3SDU2Click(Sender: TObject);
begin
edt1SDU2.Clear;
edt2SDU2.Clear;
edt3SDU2.Clear;
edt4SDU2.Clear;
edt5SDU2.Clear;
Memo1.Clear;
mem1SDU2.Clear;
end;
procedure TfrmSDU2.btn2SDU2Click(Sender: TObject);
```

```
begin
      Val(edt1SDU2.Text,N,code);
      Val(edt2SDU2.Text,x,code);
      Val(edt3SDU2.Text,xm,code);
      Val(edt4SDU2.Text,e1,code);
      Val(edt5SDU2.Text,h,code);
      for i:=1 to N do
      begin
      ii:= format('%2.0d',[i]);
      mem1SDU2.Lines.Add('Начальное значение y0'+ii+'-й функции y');
      Y0[i]:=InputBox('Введение начальных значений функции
Y0(i)','Значение функции Y0('+ii+')=','');
      Val(Y0[i],w[i],code);
      y[i]:=w[i];
      xx[i]:=format('%12.5f',[w[i]]);
      mem1SDU2.Lines.Add('Y0('+ii+')='+xx[i]);
      end;
    if (FErrors <> nil) then FErrors.Close;
       if not CreatePZ(Memo1.Text)
    then
    begin
    Application.CreateForm(TFErrors, FErrors);
    FErrors.LBErrors.Items.Assign(ErrorList);
    FErrors.Show;
    exit;
    end;
     repeat
        e3:=0;
        V1(F,X,Y);
```

```
d1:=0;
for j:=1 to N do
  begin
  a[j]:=F[j]*h;
  y[j]:=w[j]+(a[j]/3);
  end;
  x:=x+(h/3);
  V1(F,X,Y);
  for j:=1 to N do
   begin
   y[j]:=w[j]+((a[j]+F[j]*h)/6);
   end;
   V1(F,X,Y);
   for j:=1 to N do
    begin
    c[j]:=F[j]*h;
    y[j]:=w[j]+(a[j]/8)+0.375*c[j];
    end;
    x:=x+(h/6);
    V1(F,X,Y);
    for j:=1 to N do
    begin
    d[j]:=F[j]*h;
    y[j]:=w[j]+(a[j]/2)-1.5*c[j]+2*d[j];
    end;
    x:=x+h/2;
    V1(F,X,Y);
    for j:=1 to N do
   begin
```

```
  e[j]:=F[j]*h;
  y[j]:=w[j]+(a[j]+4*d[j]+e[j])/6;
  e2:=(abs(-2*a[j]+9*c[j]-8*d[j]+e[j]))/30;
  if e2<=e1 then
              begin
               if e2<(e1/20) then
                        begin
                        d1:=d1+1;
                        end;
              end
            else
              begin
              e3:=1;
              end;
end;
   if e3=0 then
             begin
               if d1=N then
                     begin
                      h:=h+h;
                     end;
  xxx:=Format('%17.8f',[x]);
  mem1SDU2.Lines.Add('X='+xxx);

  for j:=1 to N do
       begin
      jj:=format('%2.0d',[j]);
      yy[j]:=format('%17.8f',[y[j]]);
      mem1SDU2.Lines.Add('Y['+jj+']='+yy[j]);
```

```
                w[j]:=y[j];
                end;
                    end
                  else
                    begin
                    x:=x-h;
                     for j:=1 to N do
                          begin
                          y[j]:=w[j];
                          end;
                    h:=h/2;
                     end;
      until x>xm;
end;
unit Synt;
interface
uses Classes;
 type
 { }
 TData = record
   Name: string;
   Data:real;
 end;
var
 //
 NConst: integer = 100;
 //
 ErrorList: TStringList;
```

```
 //
 PZ: array of integer;
 //
 DataList: array of TData;
const
 //
 MConst = 2;
{}
procedure SyntItem(S:string; First:boolean=false; Pos:Integer=1);
{}
function CreatePZ(S:string):boolean;
function Calculate(var R:real):boolean;
function SetData(Name:string; Data:real):boolean;
function GetData(Name:string; var Data:real):boolean;
implementation
 uses Sysutils, Math, Dialogs,STUMN2,UErrors;
type
 //
 TType = (None, Number, Divider, Ident, Func, Part, All);
 //
 TSynt = record
    mode: TType;
    Number:real;
    Ident:string;
    Error:boolean;
    Pos1,Pos2:integer;
end;
const
       SetNum: set of char=['0'..'9', ','];
```

```
   SetDiv: set of char=[';', '(', ')', '=', '+', '-', '/', '*', '^',
                        '{', ' ', #13];
   { }
   SetChar: set of char=['a'..'z','A'..'Z','_'];
      NFunc = 9;
   Functions: array[1..NFunc] of string =
    ('exp','sin','cos','sqrt','abs','ln','tg','arctan','arccos');
var
  SItem: TSynt;
TrStack: array of char;
 ConstList: array of real;
 Position: Integer;
procedure SyntItem(S:string;First:boolean=false;Pos:Integer=1);
var i:integer;
begin
 if (S = '') then begin
  SItem.mode := All;
  exit;
 end;
 if(First) then Position := Pos;
repeat
  if (S[Position] = '{')
   then begin
    repeat
     Inc(Position)
    until (Position >= Length(S)) or (S[Position] = '}');
    Inc(Position);
   end;
  if(Position <= Length(S)) then
```

```
  while ((S[Position] = ' ')or
        (S[Position] = #13)or
        (S[Position] = #10)or
        (S[Position] = #0))
  do Inc(Position);
until (S[Position] <> '{');
SItem.Error:=false;
SItem.Pos1:=Position;
if(Position > Length(S)) then begin
 SItem.mode := All;
 exit;
end;
SItem.Ident := S[Position];
 if (S[Position] in SetChar)
    then SItem.mode := Ident
 else if (S[Position] in SetNum)
    then SItem.mode := Number
 else if (S[Position] in SetDiv)
    then begin
          if (S[Position] <> ';')
      then SItem.mode := Divider
       else SItem.mode := Part;
     Inc(Position);
          exit;
    end
     else begin
 SItem.mode := None;
 Inc(Position);
 exit;
```

```
 end;
repeat
    Inc(Position);
  if (SItem.mode = Number)and
     ((S[Position] = '-')or(S[Position] = '+'))and
     (UpCase(S[Position-1])='E')
    then SItem.Ident := SItem.Ident + S[Position]
  else if ((Position > Length(S))or(S[Position] in SetDiv))
    then begin
         if(SItem.mode = Number)
       then try
         SItem.Number := StrToFloat(SItem.Ident)
       except
        on EConvertError do SItem.Error := true;
       end;
         for i:=1 to NFunc do
      if (LowerCase(SItem.Ident) = Functions[i])
        then begin
         SItem.mode:=Func;
         SItem.Number:=i;
         break;
        end;
     SItem.Pos2:=Position-1;
     exit;
    end
    else SItem.Ident := SItem.Ident + S[Position];
until false;
end;
procedure ClearPZ;
```

```
begin
 ErrorList.Clear;
 SetLength(ConstList,0);
 SetLength(DataList,MConst);
 SetLength(PZ,0);
end;
function CreatePZ(S:string):boolean;
var
      lend:boolean;
   i:integer;
   Assign:boolean;
   Adress: integer;
   OldMode: TType;
   OldS: char;
procedure code;
begin
 SetLength(PZ,High(PZ)+2);
 case TrStack[High(TrStack)] of
  '+': PZ[High(PZ)] := -1;
  '-': PZ[High(PZ)] := -2;
  '*': PZ[High(PZ)] := -3;
  '/': PZ[High(PZ)] := -4;
  '^': PZ[High(PZ)] := -5;
  'M': PZ[High(PZ)] := -6;
 end;
end;
procedure proc1;
begin
 SetLength(TrStack,High(TrStack)+2);
```

```
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc2;
begin
 code;
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc3;
begin
 code;
 SetLength(TrStack,High(TrStack));
 lend:=false;
end;
procedure proc4;
begin
 SetLength(TrStack,High(TrStack));
end;
procedure proc5;
begin
 SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := Chr(127+Round(SItem.Number));
end;
procedure proc6;
begin
 SetLength(PZ,High(PZ)+2);
 PZ[High(PZ)] := -Ord(TrStack[High(TrStack)])+27;
 SetLength(TrStack,High(TrStack));
end;
begin
```

```
ClearPZ;
SetLength(TrStack,1);
TrStack[0] := '0';
OldMode := None;
OldS := ' ';
Assign := true;
Adress := 0;
SyntItem(S,true);
if (SItem.mode = All)
 then begin
  ErrorList.Add();
  Result := false;
  exit;
 end;
repeat
 if ((OldMode = Func)and(SItem.Ident[1] <> '('))
  then ErrorList.Add('+IntToStr(SItem.Pos1));
 case SItem.mode of
  Number: begin
          if((OldMode <> Divider)and(OldMode <> None)and
            (OldMode <> Part))
           then ErrorList.Add('+IntToStr(SItem.Pos1)+');
          if (SItem.Error)
           then ErrorList.Add('+IntToStr(SItem.Pos1)+
                        ' - '+IntToStr(SItem.Pos2))
           else begin
            SetLength(ConstList,High(ConstList)+2);
            ConstList[High(ConstList)] := SItem.Number;
            SetLength(PZ,High(PZ)+2);
```

```
          PZ[High(PZ)] := High(ConstList);
         end;
        Assign:=false;
       end;
Ident:  begin
        if((OldMode <> Divider)and(OldMode <> None)and
          (OldMode <> Part))
         then ErrorList.Add('+IntToStr(SItem.Pos1)+');
        for i:=0 to High(DataList) do
         begin
          if (UpperCase(SItem.Ident) = DataList[i].Name)
           then begin
            SetLength(PZ,High(PZ)+2);
            PZ[High(PZ)] := NConst+i;
            break;
           end;
          if(i = High(DataList)) then
           begin
            SetLength(DataList,High(DataList)+2);
            DataList[High(DataList)].Name:=UpperCase(SItem.Ident);
            DataList[High(DataList)].Data:=0;
            SetLength(PZ,High(PZ)+2);
            PZ[High(PZ)] := NConst+High(DataList);
           end;
         end;
        end;
All,Part:begin
        repeat
         lend:=true;
```

```
        case TrStack[High(TrStack)] of
         '0': begin
              if (Adress <> 0) then begin
                                SetLength(PZ,High(PZ)+3);
                PZ[High(PZ)-1] := -7;
                PZ[High(PZ)] := Adress;
                Adress := 0;
               end;
              break;
              end;
         '(': ErrorList.Add();
          else proc3;
        end;
       until lend;
       if (ErrorList.Count = 0)
        then Result:=true
        else Result:=false;
       if (SItem.mode = All) then exit
       else begin
        Assign := true;
        SItem.mode := None;
       end
      end;
Divider:begin
      if((OldMode = Divider)and
       ((SItem.Ident[1]<>'=')and
       (SItem.Ident[1]<>'(')and
       (SItem.Ident[1] <> ')'))and
       ((OldS <> '(')and
```

```
 (OldS <> ')')and
 (OldS <> '=')))
 then begin
  ErrorList.Add('+IntToStr(SItem.Pos1)
  break;
 end;
repeat
 lend:=true;
 case SItem.Ident[1] of
  '=': if Assign and (OldMode = Ident)
       then begin
                    Adress := PZ[High(PZ)];
        SetLength(PZ,High(PZ));
        SItem.mode := None;
       end
       else ErrorList.Add('+IntToStr(SItem.Pos1)+
                    ':);
  '(': if(OldMode = Ident) or (OldMode = Number)
      then ErrorList.Add('+IntToStr(SItem.Pos1))
      else  proc1;
  '+','-','M': begin
      if((OldMode = None)or(OldS = '('))
       then if (SItem.Ident[1] = '+')
        then break
       // 'M'
       else SItem.Ident[1] := 'M';
      case TrStack[High(TrStack)] of
       '0','(': proc1;
       '+','-','M': proc2;
```

```
      '*','/','^': proc3;
    end;
    end;
'*','/':
    if OldS = '('
    then ErrorList.Add('+IntToStr(SItem.Pos1))
    else
    case TrStack[High(TrStack)] of
     '0','(','+','-','M': proc1;
     '*','/': proc2;
     '^': proc3;
    end;
'^':
    if OldS = '('
    then ErrorList.Add('+IntToStr(SItem.Pos1))
    else
    case TrStack[High(TrStack)] of
     '0','(','+','-','*','/','M': proc1;
     '^': proc2;
    end;
')':
    case TrStack[High(TrStack)] of
     '0': ErrorList.Add();
     '(': begin
          proc4;
          if (Ord(TrStack[High(TrStack)]) > 127)
           then proc6;
         end;
     '+','-','*','/','^','M': proc3;
```

```
            end;
          end;
         until lend;
         Assign:=false;
         end;
  Func:   begin
         repeat
          lend:=true;
          proc5
         until lend;
         Assign:=false;
         end;
   None:   ErrorList.Add(и '+IntToStr(SItem.Pos1));
  end;
  OldMode := SItem.mode;
  OldS := SItem.Ident[1];
  SyntItem(S);
 until false;
 if(ErrorList.Count = 0)
 then Result := true
 else Result := false;
end;
function Calculate(var R:real):boolean;
var Stack: array of real;
    i:integer;
begin
  for i:=0 to High(PZ) do begin
   if (i > 0) then
    if (PZ[i-1] = -7) and (i < High(PZ)) then Continue;
```

```
  if PZ[i] < -100
// функция
   then begin
    try
     case -PZ[i]-100 of
      1: Stack[High(Stack)]:=Exp(Stack[High(Stack)]);
      2: Stack[High(Stack)]:=Sin(Stack[High(Stack)]);
      3: Stack[High(Stack)]:=Cos(Stack[High(Stack)]);
      4: Stack[High(Stack)]:=Sqrt(Stack[High(Stack)]);
      5: Stack[High(Stack)]:=Abs(Stack[High(Stack)]);
      6: Stack[High(Stack)]:=Ln(Stack[High(Stack)]);
      7: Stack[High(Stack)]:=Tan(Stack[High(Stack)]);
      8: Stack[High(Stack)]:=ArcTan(Stack[High(Stack)]);
      9: Stack[High(Stack)]:=ArcCos(Stack[High(Stack)]);
     end
    except
     Result := false;
     exit;
    end;
    if(FloatToStr(Stack[High(Stack)]) = 'NAN') or
      (FloatToStr(Stack[High(Stack)]) = 'INF') or
      (FloatToStr(Stack[High(Stack)]) = '-INF')
     then begin
      Result := false;
      exit;
     end
   end
   else if PZ[i] < 0
//
```

```
then begin
try
 case -PZ[i] of
  1: Stack[High(Stack)-1]:=
     Stack[High(Stack)-1]+Stack[High(Stack)];
  2: Stack[High(Stack)-1]:=
     Stack[High(Stack)-1]-Stack[High(Stack)];
  3: Stack[High(Stack)-1]:=
     Stack[High(Stack)-1]*Stack[High(Stack)];
  4: Stack[High(Stack)-1]:=
       Stack[High(Stack)-1]/Stack[High(Stack)];
  5: Stack[High(Stack)-1]:=
       Power(Stack[High(Stack)-1],Stack[High(Stack)]);
  6: Stack[High(Stack)]:= -Stack[High(Stack)];
  7: DataList[PZ[i+1]-NConst].Data := Stack[High(Stack)];
 end;
except
 Result := false;
 exit;
end;
 if (PZ[i] <> -6)  //
   then SetLength(Stack,High(Stack));
end
else begin
 SetLength(Stack,High(Stack)+2);
 if (PZ[i] < NConst)
  then Stack[High(Stack)]:=ConstList[PZ[i]]
  else Stack[High(Stack)]:=DataList[PZ[i]-NConst].Data;
end;
```

```
 end;
 Result := true;
 R :=Stack[High(Stack)];
end;
function SetData(Name:string; Data:real):boolean;
var i:integer;
begin
 for i:=MConst to High(DataList) do
  if (UpperCase(Name) = DataList[i].Name)
   then  begin
    DataList[i].Data := Data;
    Result:=true;
    exit;
   end;
 Result := false;
end;
function GetData(Name:string; var Data:real):boolean;
var i:integer;
begin
 for i:=0 to High(DataList) do
  if (UpperCase(Name) = DataList[i].Name)
   then  begin
    Data := DataList[i].Data;
    Result:=true;
    exit;
   end;
 Result := false;
end;
initialization
```

```
 SetLength(DataList,MConst);
 DataList[0].Name:='PI';
 DataList[0].Data:=Pi;
 DataList[1].Name:='E';
 DataList[1].Data:=2.71828183;
 ErrorList := TStringList.Create;
finalization
 ErrorList.Free;
```

ПРОГРАМА

для розв'язання системи трансцендентних рівнянь

```
unit STUMN2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs,Math, StdCtrls;
type
  TfrmSTUMN2 = class(TForm)
    btn1STUMN2: TButton;
    lbl1STUMN2: TLabel;
    edt1STUMN2: TEdit;
    mem1STUMN2: TMemo;
    btn2STUMN2: TButton;
    lbl2STUMN2: TLabel;
    edt2STUMN2: TEdit;
    lbl3STUMN2: TLabel;
    edt3STUMN2: TEdit;
    Memo1: TMemo;
    lbl4STUMN2: TLabel;
    btn3STUMN2: TButton;
```

```
    lbl5STUMN2: TLabel;
    edt4STUMN2: TEdit;
    lbl7STUMN2: TLabel;
    procedure btn1STUMN2Click(Sender: TObject);
    procedure btn2STUMN2Click(Sender: TObject);
    procedure btn3STUMN2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

type
   xf=array [1..50]of Real;
  var
  frmSTUMN2: TfrmSTUMN2;
  N,code,s1,M,R:integer;
  e,h,x1,g:Real;
  A:array[1..50,1..50] of Real;
  B:array[1..50]of Real;
  F,X:xf;
  x0,xx:array[1..50] of string;
  ii,ss,xxx:string;
implementation
  uses STUMN1,Synt,UErrors;
{$R *.dfm}
  procedure v(var F:xf; X:xf);
      var i:Integer;
    begin
```

```
  SetData('X[1]',X[1]);
  SetData('X[2]',X[2]);
  SetData('X[3]',X[3]);
  SetData('X[4]',X[4]);
  SetData('X[5]',X[5]);
  SetData('X[6]',X[6]);
  SetData('X[7]',X[7]);
  SetData('X[8]',X[8]);
  SetData('X[9]',X[9]);
  SetData('X[10]',X[10]);
  SetData('X[11]',X[11]);
  SetData('X[12]',X[12]);
  SetData('X[13]',X[13]);
  SetData('X[14]',X[14]);
  SetData('X[15]',X[15]);
  SetData('X[16]',X[16]);
  SetData('X[17]',X[17]);
  SetData('X[18]',X[18]);
  SetData('X[19]',X[19]);
  SetData('X[20]',X[20]);
  SetData('X[21]',X[21]);
  SetData('X[22]',X[22]);
  SetData('X[23]',X[23]);
  SetData('X[24]',X[24]);
 for i:=1 to N do
   begin
Calculate(F[i]);
   end;
 GetData('F[1]',F[1]);
```

```
    GetData('F[2]',F[2]);
    GetData('F[3]',F[3]);
    GetData('F[4]',F[4]);
    GetData('F[5]',F[5]);
    GetData('F[6]',F[6]);
    GetData('F[7]',F[7]);
    GetData('F[8]',F[8]);
    GetData('F[9]',F[9]);
    GetData('F[10]',F[10]);
    GetData('F[11]',F[11]);
    GetData('F[12]',F[12]);
    GetData('F[13]',F[13]);
    GetData('F[14]',F[14]);
    GetData('F[15]',F[15]);
    GetData('F[16]',F[16]);
    GetData('F[17]',F[17]);
    GetData('F[18]',F[18]);
    GetData('F[19]',F[19]);
    GetData('F[20]',F[20]);
    GetData('F[21]',F[21]);
    GetData('F[22]',F[22]);
    GetData('F[23]',F[23]);
    GetData('F[24]',F[24]);
  end;
procedure TfrmSTUMN2.btn1STUMN2Click(Sender: TObject);
begin
frmSTUMN1.Close;
end;
```

```
procedure TfrmSTUMN2.btn2STUMN2Click(Sender: TObject);
   Var
   i,j,k:Integer;
begin
        s1:=0;
       Val(edt1STUMN2.Text,N,code);
       Val(edt2STUMN2.Text,M,code);
       Val(edt3STUMN2.Text,e,code);
      for i:=1 to N do
       begin
       ii:= format('%2.0d',[i]);
       mem1STUMN2.Lines.Add(+ii+'-го корня');
       x0[i]:=InputBox(X0(i)', X0('+ii+')=','0');
       Val(x0[i],x[i],code);
       xx[i]:=format('%12.5f',[x[i]]);
       mem1STUMN2.Lines.Add('x('+ii+')='+xx[i]);
       end;
    if (FErrors <> nil) then FErrors.Close;
    if not CreatePZ(Memo1.Text)
    then
    begin
        Application.CreateForm(TFErrors, FErrors);
    FErrors.LBErrors.Items.Assign(ErrorList);
    FErrors.Show;
    exit;
    end;
         repeat

              v(F,X);
```

```
for i:=1 to N do
 begin
 B[i]:=-F[i];
 end;
for j:=1 to N do
 begin
 x1:=x[j]; h:=e*abs(x1);
 x[j]:=x1+h;
   v(F,X);
 for i:=1 to N do
   begin
   A[i,j]:=(F[i]+B[i])/h;
   end;
 x[j]:=x1;
 end;
  s1:=s1+1;
    if s1=M+1 then
          begin
          ss:=format('%3.0d',[s1]);
          edt4STUMN2.Text:=ss;
          Break;
          end;
 for i:=1 to N-1 do
   begin
     for j:=i+1 to N do
       begin
       A[j,i]:=-A[j,i]/A[i,i];
         for k:=i+1 to N do
           begin
```

```
            A[j,k]:=A[j,k]+A[j,i]*A[i,k];
          end;
        B[j]:=B[j]+A[j,i]*B[i];
        end;
    end;
F[N]:=B[N]/A[N,N];
   for i:=N-1 downto 1 do
        begin
        h:=B[i];
          for j:=i+1 to N do
            begin
            h:=h-F[j]*A[i,j];
            end;
        F[i]:=h/A[i,i];
        end;
  R:=0;
    for i:=1 to N do
        begin
        x[i]:=x[i]+F[i];
          if abs(F[i]/x[i])>e then R:=1;
        end;
    if R=1 then Continue;
    mem1STUMN2.Lines.Add(');
     for i:=1 to N do
      begin
      ii:= format('%2.0d',[i]);
      xxx:=format('%11.5f',[x[i]]);
      mem1STUMN2.Lines.Add('x('+ii+')='+xxx);
      end;
```

```
                ss:=format('%3.0d',[s1]);
                edt4STUMN2.Text:=ss;
                Break;
        until false;
end;
procedure TfrmSTUMN2.btn3STUMN2Click(Sender: TObject);
begin
edt1STUMN2.Clear;
edt2STUMN2.Clear;
edt3STUMN2.Clear;
edt4STUMN2.Clear;
Memo1.Clear;
mem1STUMN2.Clear;
end;
unit Synt;
interface
uses Classes;
 type
 { }
 TData = record
   Name: string;
   Data:real;
 end;
var
 //
 NConst: integer = 100;
 //
 ErrorList: TStringList;
 //
```

```
 PZ: array of integer;
 //
 DataList: array of TData;
const
 //
 MConst = 2;
{}
procedure SyntItem(S:string; First:boolean=false; Pos:Integer=1);
{}
function CreatePZ(S:string):boolean;
function Calculate(var R:real):boolean;
function SetData(Name:string; Data:real):boolean;
function GetData(Name:string; var Data:real):boolean;
implementation
 uses Sysutils, Math, Dialogs,STUMN2,UErrors;
type
 //
 TType = (None, Number, Divider, Ident, Func, Part, All);
 //
 TSynt = record
    mode: TType;
    Number:real;
    Ident:string;
    Error:boolean;
    Pos1,Pos2:integer;
end;
const
        SetNum: set of char=['0'..'9', ','];
    SetDiv: set of char=[';', '(', ')', '=', '+', '-', '/', '*', '^',
```

```
                    '{', ' ', #13];
  { }
  SetChar: set of char=['a'..'z','A'..'Z','_'];
     NFunc = 9;
  Functions: array[1..NFunc] of string =
   ('exp','sin','cos','sqrt','abs','ln','tg','arctan','arccos');
var
  SItem: TSynt;
TrStack: array of char;
 ConstList: array of real;
 Position: Integer;
procedure SyntItem(S:string;First:boolean=false;Pos:Integer=1);
var i:integer;
begin
 if (S = '') then begin
  SItem.mode := All;
  exit;
 end;
 if(First) then Position := Pos;
repeat
  if (S[Position] = '{')
   then begin
    repeat
     Inc(Position)
    until (Position >= Length(S)) or (S[Position] = '}');
    Inc(Position);
   end;
  if(Position <= Length(S)) then
   while ((S[Position] = ' ')or
```

```
        (S[Position] = #13)or
        (S[Position] = #10)or
        (S[Position] = #0))
  do Inc(Position);
until (S[Position] <> '{');
SItem.Error:=false;
SItem.Pos1:=Position;
if(Position > Length(S)) then begin
 SItem.mode := All;
 exit;
end;
SItem.Ident := S[Position];
if (S[Position] in SetChar)
   then SItem.mode := Ident
else if (S[Position] in SetNum)
   then SItem.mode := Number
else if (S[Position] in SetDiv)
   then begin
          if (S[Position] <> ';')
     then SItem.mode := Divider
     else SItem.mode := Part;
    Inc(Position);
          exit;
   end
    else begin
 SItem.mode := None;
 Inc(Position);
 exit;
end;
```

```
repeat
   Inc(Position);
 if (SItem.mode = Number)and
    ((S[Position] = '-')or(S[Position] = '+'))and
    (UpCase(S[Position-1])='E')
   then SItem.Ident := SItem.Ident + S[Position]
 else if ((Position > Length(S))or(S[Position] in SetDiv))
   then begin
        if(SItem.mode = Number)
      then try
        SItem.Number := StrToFloat(SItem.Ident)
      except
       on EConvertError do SItem.Error := true;
      end;
        for i:=1 to NFunc do
     if (LowerCase(SItem.Ident) = Functions[i])
       then begin
        SItem.mode:=Func;
        SItem.Number:=i;
        break;
       end;
    SItem.Pos2:=Position-1;
    exit;
   end
   else SItem.Ident := SItem.Ident + S[Position];
until false;
end;
procedure ClearPZ;
begin
```

```
 ErrorList.Clear;
 SetLength(ConstList,0);
 SetLength(DataList,MConst);
 SetLength(PZ,0);
end;
function CreatePZ(S:string):boolean;
var
      lend:boolean;
   i:integer;
   Assign:boolean;
   Adress: integer;
   OldMode: TType;
   OldS: char;
procedure code;
begin
 SetLength(PZ,High(PZ)+2);
 case TrStack[High(TrStack)] of
  '+': PZ[High(PZ)] := -1;
  '-': PZ[High(PZ)] := -2;
  '*': PZ[High(PZ)] := -3;
  '/': PZ[High(PZ)] := -4;
  '^': PZ[High(PZ)] := -5;
  'M': PZ[High(PZ)] := -6;
 end;
end;
procedure proc1;
begin
 SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := SItem.Ident[1];
```

```
end;
procedure proc2;
begin
 code;
 TrStack[High(TrStack)] := SItem.Ident[1];
end;
procedure proc3;
begin
 code;
 SetLength(TrStack,High(TrStack));
 lend:=false;
end;
procedure proc4;
begin
 SetLength(TrStack,High(TrStack));
end;
procedure proc5;
begin
 SetLength(TrStack,High(TrStack)+2);
 TrStack[High(TrStack)] := Chr(127+Round(SItem.Number));
end;
procedure proc6;
begin
 SetLength(PZ,High(PZ)+2);
 PZ[High(PZ)] := -Ord(TrStack[High(TrStack)])+27;
 SetLength(TrStack,High(TrStack));
end;
begin
 ClearPZ;
```

```
SetLength(TrStack,1);
TrStack[0] := '0';
OldMode := None;
OldS := ' ';
Assign := true;
Adress := 0;
SyntItem(S,true);
if (SItem.mode = All)
 then begin
  ErrorList.Add();
  Result := false;
  exit;
 end;
repeat
 if ((OldMode = Func)and(SItem.Ident[1] <> '('))
  then ErrorList.Add('+IntToStr(SItem.Pos1));
 case SItem.mode of
  Number: begin
          if((OldMode <> Divider)and(OldMode <> None)and
            (OldMode <> Part))
           then ErrorList.Add('+IntToStr(SItem.Pos1)+');
          if (SItem.Error)
           then ErrorList.Add('+IntToStr(SItem.Pos1)+
                        ' - '+IntToStr(SItem.Pos2))
           else begin
            SetLength(ConstList,High(ConstList)+2);
            ConstList[High(ConstList)] := SItem.Number;
            SetLength(PZ,High(PZ)+2);
            PZ[High(PZ)] := High(ConstList);
```

```
        end;
       Assign:=false;
      end;
Ident:  begin
       if((OldMode <> Divider)and(OldMode <> None)and
         (OldMode <> Part))
        then ErrorList.Add('+IntToStr(SItem.Pos1)+');
       for i:=0 to High(DataList) do
        begin
         if (UpperCase(SItem.Ident) = DataList[i].Name)
          then begin
           SetLength(PZ,High(PZ)+2);
           PZ[High(PZ)] := NConst+i;
           break;
          end;
         if(i = High(DataList)) then
          begin
           SetLength(DataList,High(DataList)+2);
           DataList[High(DataList)].Name:=UpperCase(SItem.Ident);
           DataList[High(DataList)].Data:=0;
           SetLength(PZ,High(PZ)+2);
           PZ[High(PZ)] := NConst+High(DataList);
          end;
        end;
       end;
All,Part:begin
       repeat
        lend:=true;
        case TrStack[High(TrStack)] of
```

```
          '0': begin
               if (Adress <> 0) then begin
                                SetLength(PZ,High(PZ)+3);
                 PZ[High(PZ)-1] := -7;
                 PZ[High(PZ)] := Adress;
                 Adress := 0;
                end;
               break;
               end;
           '(': ErrorList.Add();
            else proc3;
          end;
         until lend;
         if (ErrorList.Count = 0)
          then Result:=true
          else Result:=false;
         if (SItem.mode = All) then exit
         else begin
          Assign := true;
          SItem.mode := None;
         end
        end;
Divider:begin
        if((OldMode = Divider)and
          ((SItem.Ident[1]<>'=')and
          (SItem.Ident[1]<>'(')and
          (SItem.Ident[1] <> ')'))and
          ((OldS <> '(')and
          (OldS <> ')')and
```

```
 (OldS <> '=')))
 then begin
  ErrorList.Add(''+IntToStr(SItem.Pos1)
  break;
 end;
repeat
 lend:=true;
 case SItem.Ident[1] of
  '=': if Assign and (OldMode = Ident)
        then begin
                      Adress := PZ[High(PZ)];
         SetLength(PZ,High(PZ));
         SItem.mode := None;
        end
        else ErrorList.Add('+IntToStr(SItem.Pos1)+
                    ':);
  '(': if(OldMode = Ident) or (OldMode = Number)
       then ErrorList.Add('+IntToStr(SItem.Pos1))
       else  proc1;
  '+','-','M': begin
       if((OldMode = None)or(OldS = '('))
        then if (SItem.Ident[1] = '+')
         then break
        // 'M'
        else SItem.Ident[1] := 'M';
       case TrStack[High(TrStack)] of
        '0','(': proc1;
        '+','-','M': proc2;
        '*','/','^': proc3;
```

```
    end;
    end;
'*','/':
    if OldS = '('
    then ErrorList.Add('+IntToStr(SItem.Pos1))
    else
    case TrStack[High(TrStack)] of
     '0','(','+','-','M': proc1;
     '*','/': proc2;
     '^': proc3;
    end;
'^':
    if OldS = '('
    then ErrorList.Add('+IntToStr(SItem.Pos1))
    else
    case TrStack[High(TrStack)] of
     '0','(','+','-','*','/','M': proc1;
     '^': proc2;
    end;
')':
    case TrStack[High(TrStack)] of
     '0': ErrorList.Add();
     '(': begin
           proc4;
           if (Ord(TrStack[High(TrStack)]) > 127)
            then proc6;
          end;
     '+','-','*','/','^','M': proc3;
    end;
```

```
          end;
         until lend;
         Assign:=false;
         end;
  Func:   begin
         repeat
          lend:=true;
          proc5
         until lend;
         Assign:=false;
         end;
   None:   ErrorList.Add(и '+IntToStr(SItem.Pos1));
  end;
  OldMode := SItem.mode;
  OldS := SItem.Ident[1];
  SyntItem(S);
 until false;
 if(ErrorList.Count = 0)
 then Result := true
 else Result := false;
end;
function Calculate(var R:real):boolean;
var Stack: array of real;
    i:integer;
begin
  for i:=0 to High(PZ) do begin
   if (i > 0) then
    if (PZ[i-1] = -7) and (i < High(PZ)) then Continue;
   if PZ[i] < -100
```

```
// функция
   then begin
    try
     case -PZ[i]-100 of
      1: Stack[High(Stack)]:=Exp(Stack[High(Stack)]);
      2: Stack[High(Stack)]:=Sin(Stack[High(Stack)]);
      3: Stack[High(Stack)]:=Cos(Stack[High(Stack)]);
      4: Stack[High(Stack)]:=Sqrt(Stack[High(Stack)]);
      5: Stack[High(Stack)]:=Abs(Stack[High(Stack)]);
      6: Stack[High(Stack)]:=Ln(Stack[High(Stack)]);
      7: Stack[High(Stack)]:=Tan(Stack[High(Stack)]);
      8: Stack[High(Stack)]:=ArcTan(Stack[High(Stack)]);
      9: Stack[High(Stack)]:=ArcCos(Stack[High(Stack)]);
     end
    except
     Result := false;
     exit;
    end;
    if(FloatToStr(Stack[High(Stack)]) = 'NAN') or
      (FloatToStr(Stack[High(Stack)]) = 'INF') or
      (FloatToStr(Stack[High(Stack)]) = '-INF')
     then begin
      Result := false;
      exit;
     end
   end
   else if PZ[i] < 0
//
   then begin
```

```
  try
   case -PZ[i] of
    1: Stack[High(Stack)-1]:=
       Stack[High(Stack)-1]+Stack[High(Stack)];
    2: Stack[High(Stack)-1]:=
       Stack[High(Stack)-1]-Stack[High(Stack)];
    3: Stack[High(Stack)-1]:=
       Stack[High(Stack)-1]*Stack[High(Stack)];
    4: Stack[High(Stack)-1]:=
         Stack[High(Stack)-1]/Stack[High(Stack)];
    5: Stack[High(Stack)-1]:=
         Power(Stack[High(Stack)-1],Stack[High(Stack)]);
    6: Stack[High(Stack)]:= -Stack[High(Stack)];
    7: DataList[PZ[i+1]-NConst].Data := Stack[High(Stack)];
   end;
  except
   Result := false;
   exit;
  end;
   if (PZ[i] <> -6)  //
     then SetLength(Stack,High(Stack));
  end
  else begin
   SetLength(Stack,High(Stack)+2);
   if (PZ[i] < NConst)
    then Stack[High(Stack)]:=ConstList[PZ[i]]
    else Stack[High(Stack)]:=DataList[PZ[i]-NConst].Data;
  end;
end;
```

```
 Result := true;
 R :=Stack[High(Stack)];
end;
function SetData(Name:string; Data:real):boolean;
var i:integer;
begin
 for i:=MConst to High(DataList) do
  if (UpperCase(Name) = DataList[i].Name)
   then  begin
    DataList[i].Data := Data;
    Result:=true;
    exit;
   end;
 Result := false;
end;
function GetData(Name:string; var Data:real):boolean;
var i:integer;
begin
 for i:=0 to High(DataList) do
  if (UpperCase(Name) = DataList[i].Name)
   then  begin
    Data := DataList[i].Data;
    Result:=true;
    exit;
   end;
 Result := false;
end;
initialization
 SetLength(DataList,MConst);
```

```
 DataList[0].Name:='PI';
 DataList[0].Data:=Pi;
 DataList[1].Name:='E';
 DataList[1].Data:=2.71828183;
 ErrorList := TStringList.Create;
finalization
 ErrorList.Free;
```

ПРОГРАМА

для визначення коренів полінома

```
unit KRPO2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,Math;
type
  TfrmKRPO2 = class(TForm)
    lbl1KRPO2: TLabel;
    lbl2KRPO2: TLabel;
    edt1KRPO2: TEdit;
    edt2KRPO2: TEdit;
    lbl3KRPO2: TLabel;
    mem1KRPO2: TMemo;
    btn1KRPO2: TButton;
    btn2KRPO2: TButton;
    btn3KRPO2: TButton;
    procedure btn2KRPO2Click(Sender: TObject);
    procedure btn3KRPO2Click(Sender: TObject);
    procedure btn1KRPO2Click(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmKRPO2: TfrmKRPO2;
  e,C,T,A1,S,R,P,Q,H,M,L,U,D,V,Y,Z,X,B,W,F:Real;
  N,j,K,code,vv:Integer;
  A:array[1..50]of Real;
  vvv,KK,MM,PP,QQ:String;
  aa:array[1..50]of String;
  xx:array[1..50]of String;
implementation
   uses KRPO1;
{$R *.dfm}
procedure V1;
   begin
        N:=N-2;A1:=0;
     if sqrt((S-(R*C/2))*(S-(R*C/2))+R*R*abs(H))<=e then
                                    begin
                                    A1:=1;
                                    end;
      B:=0;
      if N>=2 then
            begin
            B:=1;
            end;
   end;
```

```
procedure V2;
begin
      P:=-C/2; Q:=sqrt(abs(H));
      if H>=0 then
              begin
              M:=P+Q;P:=P-Q;Q:=0;
              end
            else
              begin
              M:=P;
              end;
end;
procedure V3;
label M1;
begin
      begin
      repeat
         begin
         repeat
M1:                    if M<>10 then
                     begin
                     M:=M+1; H:=0;Q:=A[1];
                     P:=A[2]-C*Q; L:=Q;
                     end
                   else
                     begin
                     P:=C; M:=0; Q:=D;
                     C:=U; D:=V;
                     U:=P;V:=Q;Y:=C;
```

```
            Z:=D; F:=-F;
            M:=M+1; H:=0; Q:=A[1];
            P:=A[2]-C*Q; L:=Q;
            end;
    for j:=3 to N do
            begin
            R:=P; P:=A[j]-C*R-D*Q;
            Q:=R; R:=L; L:=Q-C*R-H*D;
            H:=R;
            end;
     Q:=A[N+1]-D*Q; S:=L+C*R;
     if T=0 then
            begin
            X:=D*R; H:=R*X+S*L;
             if H=0 then
                    begin

                    goto M1;
                    end;
            end;
until True;
end;
      C:=C+((P*S-Q*R)/H); D:=D+((P*X+Q*L)/H);
         if (C-Y+D-Z)<>0 then
                    begin
                    end
                   else
                    begin
                    if F=-W then
```

```
                                begin
                                 InputBox();

                                Exit;
                                end
                               else
                                begin
                                W:=-F;
                                end;
                          end;
                    H:=(C*C/4)-D;
      if sqrt((Q-(P*C/2))*(Q-(P*C/2))+P*P*abs(H))>e/N then
                                        begin
                                        goto M1;
                                        end;
    until True;
    end;
              T:=0; A[2]:=A[2]-C*A[1];
              for j:=3 to N-1 do
                   begin
              A[j]:=A[j]-C*A[j-1]-D*A[j-2];
                   end;
                   V2;
end;
procedure V4;
begin
              for j:=3 to N do
                 begin
                 R:=P; P:=A[j]-C*R-D*Q;
```

```
        Q:=R; R:=L; L:=Q-C*R-H*D;
        H:=R;
        end;
Q:=A[N+1]-D*Q; S:=L+C*R;
if T=0 then
      begin
      X:=D*R; H:=R*X+S*L;
       if H=0 then
            begin
            V3;
            end;
      end;
 C:=C+((P*S-Q*R)/H); D:=D+((P*X+Q*L)/H);
   if (C-Y+D-Z)<>0 then
             begin
             end
            else
             begin
             if F=-W then
                  begin
                 InputBox();
                  Exit;
                  end
                 else
                  begin
                  W:=-F;
                  end;
             end;
        H:=(C*C/4)-D;
```

```
        if sqrt((Q-(P*C/2))*(Q-(P*C/2))+P*P*abs(H))>e/N then
                                        begin
                                        V3;
                                        end;
                T:=0; A[2]:=A[2]-C*A[1];
                for j:=3 to N-1 do
                    begin
                A[j]:=A[j]-C*A[j-1]-D*A[j-2];
                    end;
                    V2;
end;
procedure TfrmKRPO2.btn2KRPO2Click(Sender: TObject);
begin
edt1KRPO2.Clear;
edt2KRPO2.Clear;
mem1KRPO2.Clear;
end;
procedure TfrmKRPO2.btn3KRPO2Click(Sender: TObject);
begin
frmKRPO1.Close;
end;
procedure TfrmKRPO2.btn1KRPO2Click(Sender: TObject);
begin
      Val(edt1KRPO2.Text,e,code);
      Val(edt2KRPO2.Text,N,code);
      for j:=1 to N+1 do
       begin
      vv:=N+1-j;
      vvv:=format('%3.0d',[vv]);
```

```
 aa[j]:=InputBox(A['+vvv+']=','');
 Val(aa[j],A[j],code);
  xx[j]:=format('%12.5f',[A[j]]);
mem1KRPO2.Lines.Add('A('+vvv+')='+xx[j]);
  end;
         K:=1;
 repeat
         T:=1;C:=A[2]/A[1];
       if N=1 then
             begin
             P:=-C; Q:=0;
             KK:=format('%3.0d',[K]);
             MM:=format('%12.5f',[M]);
             PP:=format('%12.5f',[P]);
             QQ:=format('%12.5f',[Q]);
             mem1KRPO2.Lines.Add('X('+KK+')='+PP+'+j*('+'-'+QQ+')');
             K:=K+1;
             if T<>0 then Break;
             V1;
             end
            else
             begin
             if N=2 then
                   begin
                   H:=(C*C/4)-(A[3]/A[1]);
                   V2;
                   KK:=format('%3.0d',[K]);
                   MM:=format('%12.5f',[M]);
                   PP:=format('%12.5f',[P]);
```

```
                QQ:=format('%12.5f',[Q]);
              mem1KRPO2.Lines.Add('X('+KK+')='+MM+'+j*('+QQ+')');
                K:=K+1;
                KK:=format('%3.0d',[K]);
                mem1KRPO2.Lines.Add('X('+KK+')='+PP+'+j*('+-
'+QQ+')');
                K:=K+1;
                if T<>0 then Break;
                V1;
                end
               else
                begin
                M:=10; C:=4; D:=8; U:=4;
                V:=8; F:=1; W:=2; T:=0;
                V3;
                KK:=format('%3.0d',[K]);
                MM:=format('%12.5f',[M]);
                PP:=format('%12.5f',[P]);
                QQ:=format('%12.5f',[Q]);
              mem1KRPO2.Lines.Add('X('+KK+')='+MM+'+j*('+QQ+')');
                K:=K+1;
                KK:=format('%3.0d',[K]);
                mem1KRPO2.Lines.Add('X('+KK+')='+PP+'+j*('+-
'+QQ+')');
                K:=K+1;
                if T<>0 then Break;
                V1;
                end;
          end;
```

```
            if A1+B=2 then
                    begin
                    T:=1;
                    V4;
                    KK:=format('%3.0d',[K]);
                    MM:=format('%12.5f',[M]);
                    PP:=format('%12.5f',[P]);
                    QQ:=format('%12.5f',[Q]);
                    mem1KRPO2.Lines.Add('X('+KK+')='+MM+'+j*('+QQ+')');
                    K:=K+1;
                    KK:=format('%3.0d',[K]);
                    mem1KRPO2.Lines.Add('X('+KK+')='+PP+'+j*('+-
'+QQ+')');
                    K:=K+1;
                    if T<>0 then Break;
                    V1;
                    end;
      until False;
end;
```

ПРОГРАМА

для розв'язання системи лінійних рівнянь

```
unit SLGA2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,Math;
type
  TfrmSLGA2 = class(TForm)
```

```
    lbl1SLGA2: TLabel;
    edt1SLGA2: TEdit;
    lbl2SLGA: TLabel;
    mem1SLGA2: TMemo;
    btn1SLGA2: TButton;
    btn2SLGA2: TButton;
    btn3SLGA2: TButton;
    procedure btn3SLGA2Click(Sender: TObject);
    procedure btn2SLGA2Click(Sender: TObject);
    procedure btn1SLGA2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmSLGA2: TfrmSLGA2;
  N,i,j,u,m,k,z,code:Integer;
  v,v1,H:Real;
  A:array[1..50,1..50]of Real;
  B:array[1..50]of Real;
  X:array[1..50]of Real;
  aa:array[1..50,1..50]of String;
  bb:array[1..50]of String;
  ii,jj,ww:String;
  xx:array[1..50,1..50]of String;
  xxx:array[1..50]of String;
implementation
   uses SLGA1;
```

```
{$R *.dfm}
procedure TfrmSLGA2.btn3SLGA2Click(Sender: TObject);
begin
frmSLGA1.Close;
end;
procedure TfrmSLGA2.btn2SLGA2Click(Sender: TObject);
begin
edt1SLGA2.Clear;
mem1SLGA2.Clear;
end;
procedure TfrmSLGA2.btn1SLGA2Click(Sender: TObject);
begin
      Val(edt1SLGA2.Text,N,code);
      for i:=1 to N do
         begin
           for j:=1 to N do
             begin
       ii:= format('%3.0d',[i]);
       jj:= format('%3.0d',[j]);
       aa[i,j]:=InputBox(a['+ii+','+jj+']=','0');
       Val(aa[i,j],A[i,j],code);
        xx[i,j]:=format('%12.5f',[A[i,j]]);
      mem1SLGA2.Lines.Add('A('+ii+','+jj+')='+xx[i,j]);
              end;
           begin
       ii:= format('%3.0d',[i]);
       bb[i]:=InputBox(b['+ii+']=','0');
       Val(bb[i],B[i],code);
       xxx[i]:=format('%12.5f',[B[i]]);
```

```
mem1SLGA2.Lines.Add('B('+ii+')='+xxx[i]);
      end;
    end;
 for i:=1 to N-1 do
        begin
        for j:=i+1 to N do
        begin
        A[j,i]:=-A[j,i]/A[i,i];
        for k:=i+1 to N do
        begin
        A[j,k]:=A[j,k]+A[j,i]*A[i,k];
        END;
        B[j]:=B[j]+A[j,i]*B[i];
        END;
        END;
          X[N]:=B[N]/A[N,N];
          for i:=N-1 downto 1 do
          begin
          H:=B[i];
          for j:=i+1 to N do
          begin
          H:=H-X[j]*A[i,j];
          END;
          X[i]:=H/A[i,i];
          END;
        for i:=1 to N do
        begin
        ii:= format('%3.0d',[i]);
        ww:=format('%12.6f',[x[i]]);
```

```
            mem1SLGA2.Lines.Add('x('+ii+')='+ww);
            END;
end;
```

ЩЕРБАНЬ В.Ю., КОЛИСКО О.З., ЩЕРБАНЬ Ю.Ю.,
ВОЛЯНИК О.Ю., ЧУПРИНКА Н.В., МЕЛЬНИК Г.В.,
ГОЛЬДБЕРГ М.І., КИРИЧЕНКО А.М.,
КАЛАШНИК В.Ю.

МАТЕМАТИЧНИЙ ТА КОМП'ЮТЕРНИЙ АНАЛІЗ СИСТЕМ І ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ

Том 2

Математичне та програмне забезпечення для аналізу механічних систем та прикладних питань математичних моделей

Підписано до друку 24.06.2024
Формат 60Х84/8.
Папір офсетний. Друк офсетний.
Ум. друк. арк. 40,86.
Тираж 200 пр.

ТОВ «Фастбінд Україна»
вул. Верхній Вал 62 оф. 4, Київ, 04071
Свідоцтво суб'єкта видавничої справи
ДК 6324 від 31.07.2018 р.