Частина 4. Курсова робота

Загальні положення про курсову роботу

Виконання практичної частини курсової роботи

Загальні положення

Мета і задачі курсової роботи

Курсова робота з інформатики - підсумкова самостійна робота студента, яка завершує вивчення дисципліни "Інформатика" і призначена для поглиблення знань у галузі обчислювальної техніки та інформаційних технологій через написання реферату по темі, зазначеній у додатку Г або узгодженій із викладачем, і отримання практичних навичок розробки програмного забезпечення для вирішення задач інженерної механіки.
Метою пропонованої курсової роботи є створення програмної системи для розрахунку і моделювання плоского механізму. Програмна система реалізується в будь-якому середовищі (рекомендується використовувати Pascal, Delphi, Visual Basic, C, Flash тощо) з використанням додаткових модулів та бібліотек або без них. Для досягнення поставленої мети необхідно:

1) Проаналізувати відомості про механізм (відповідно до індивідуального варіанту завдання - додаток Д), які включають:
- Креслення і спрощену схему з параметрами;
- Розрахункові залежності, які визначають взаємозв'язок параметрів механізму;
- Розрахункові залежності, які визначають функціонування механізму (функції положення і передаточного відношення).
Примітка. На компакт-диску розміщені креслення механізмів (папка `Kurs` файли `*.wmf`), а електронна версія навчального посібника містить й анімовані тривимірні моделі механізмів, які ілюструють його роботу;

2) Виявити основні (визначають роботу механізму) і допоміжні (слугують конструктивному оформленню) параметри механізму та можливі діапазони їхньої зміни;

3) Реалізувати код власних функцій, які визначатимуть положення ланок механізму та передаточне відношення;

4) Реалізувати код власних процедур, які призначатимуться для:
- введення та розрахунку параметрів механізму;
- виведення таблиці положень механізму і передаточної функції, таблиці граничних значень, реалізації запису результатів у текстовий файл;
- виведення в графічному режимі спрощеного параметричного ескізу механізму;
- виведення в графічному режимі повного параметричного ескізу механізму;
- імітацію роботи механізму (спрощеного, повного або обох) з можливістю регулювання швидкості і напрямку перед викликом процедури або під час її роботи;
- виведення графіків залежності положення залежної ланки механізму та передаточного відношення - від положення привідної ланки при встановлених параметрах механізму;
- виведення інформації про програмну систему;
- виведення інформації про розробника системи;
Примітка 1. Для реалізації вищезазначених процедур мовою Pascal рекомендується використовувати типи, константи, змінні, процедури і функції допоміжних модулів (додаток Е), які при створенні програмного забезпечення дозволяють:
- Модуль `PVEdit` - використовувати процедури коректного введення даних рядкового (`ReadString`), цілочислового (`ReadInteger`) та дійсного типів (`ReadReal`), а також логічну функцію, яка обробляє натискання клавіш `Y/N` (`YesOrNo`);
- Модуль `PVGr` - використовувати дві процедури і нестандартний тип даних, які в графічному режимі дозволяють вивести систему координат (`BuildCoord`) і графік (`BuildChart`);
- Модуль `PVMath` - використовувати нестандартні математичні функції `arcsin, arcctg, arccos`;
- Модуль `PVMech` - використовувати допоміжні функції для малювання в графічному режимі спрощеної кінематичної схеми механізму (`Kulisa, KulisaPaz, NapravHor, NapravVer, Opora, PolzunAng, PolzunHor, PolzunVer, RazmerAngular, RazmerLinear, StoykaHor, StpykaVer, Uzel`).
- Модуль `PVMenu` - використовувати процедуру (`SMenu`) і нестандартні типи, які дозволяють в текстовому режимі реалізувати діалогове меню;
- Модуль `PVServis` - використовувати процедури управління видимістю текстового курсору (`CursorOn, CursorOff`), визначення кольорів в текстовому режимі (`Colors`) та створення текстових вікон (`BWindow`);
Примітка 2: При реалізації програмного забезпечення у середовищі Delphi для побудови спрощеного ескізу механізму можна скористатись процедурами `Kulisa, KulisaPaz, NapravHor, NapravVer, Opora, PolzunAng, PolzunHor, PolzunVer, RazmerAngular, RazmerLinear, StoykaHor, StpykaVer, Uzel` модулю `Eskiz`.

5) Скласти програмну систему, яка за допомогою системи меню організовуватиме сумісну роботу всіх власних процедур і функцій. Для реалізації програмної системи мовою Pascal рекомендується використовувати процедуру `SMenu` з допоміжного модулю `PVMenu` (додаток Е)

6) У випадку використання допоміжних модулів, ретельно ознайомитись із алгоритмічною і програмною реалізацією їхніх процедур і функцій.

Вимоги до оформлення звіту з курсової роботи

Звіт з курсової роботи повинен бути виконаний українською мовою комп'ютерним способом і надрукований на аркушах формату А4 (210х297 мм) з однієї сторони. Розміри полів: лівого - 30 мм; правого, верхнього і нижнього - по 20 мм. Усі сторінки звіту повинні бути пронумерованими у правому нижньому куті.
Розроблене програмне забезпечення представляється для захисту на будь-якому комп'ютерному носії інформації (дискеті, компакт-диску, Flash-пам'ять), який після захисту

роботи повертається студентові.

Обов'язковими складовими елементами звіту є:

- Титульна сторінка (див. рис. 2, файл бланку завдання зберігається на компакт-диску в папці `Kurs` під ім'ям `Zavdanya.doc`).
- Заповнений і підписаний бланк завдання до курсової роботи (див. рис. 1, файл бланку завдання зберігається на компакт-диску в папці `Kurs` під ім'ям `Titul.doc`);
- Реферат на визначену тему (рекомендований об'єм реферату 5-8 стор., за умови оформлення реферату у текстовому редакторі Microsoft Word шрифтом розміром 14 пт та з полуторною міжрядковою відстанню);
- Результати розробки програмної системи (постановка задачі, алгоритми програмної системи в цілому та окремих блоків, вихідні коди власних процедур; копії з екранів, що демонструють роботу окремих блоків);
- Висновки.

Національний технічний університет України

"Київський політехнічний інститут"

Механіко–машинобудівний **інститут** **Кафедра** технології машинобудування

Спеціальність: 7.090.202 – технологія машинобудування

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ З ІНФОРМАТИКИ

Студентові:

(група, прізвище, ім'я, по-батькові)

1. Тема реферату:

2. Завдання практичної частини

3. Термін здачі студентом закінченої роботи " 25 " травня 200__р.

4. Обов'язкові елементи звіту з курсової роботи:

- Реферат;
- Вимоги до програми;
- Алгоритм програми в цілому і алгоритми вирішення окремих частин програми;
- Фрагменти роботи програми;
- Висновки

5. Дата видачі завдання "___" _____ 200__ р.

Студент _____
(підпис)

Керівник роботи _____
(підпис)

Рис. 1. Бланк загального завдання до курсової роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет України

"Київський політехнічний інститут"

Кафедра технології машинобудування

КУРСОВА РОБОТА

з курсу

"Інформатика"

Виконав: студент гр. _____ **Керівник**: _____

_____ _____

(Прізвище І. Б.) (Прізвище І. Б.)

Робота захищена з оцінкою "_____"

"___"_____ 200__ р. _____

(підпис)

Київ 200__

Рис. 2. Бланк титульної сторінки звіту з курсової роботи

Порядок захисту курсової роботи

Курсові роботи захищаються у протягом трьох тижнів, які передують екзаменаційній сесії. Захист курсової роботи у період екзаменаційної сесії заборонений. Студенти, які своєчасно не захистили курсову роботу до іспиту з курсу „Інформатика" не допускаються. На захист представляється оформлений і скріплений звіт з курсової роботи, а також програмне забезпечення. Під час захисту студент повинен продемонструвати роботу програми. Викладач, який приймає курсову роботу розглядає представлені матеріали і задає питання як по змісту курсової роботи, так і по курсу „Інформатика".

Результати захисту оцінюються оцінками „відмінно", „добре" та „задовільно", які проставляються у відомість і залікову книжку студента.

Критерії оцінювання курсової роботи

При оцінюванні курсової роботи враховуються такі фактори:

- Своєчасність виконання, оформлення і представлення роботи;
- Коректність отриманих результатів;
- Якість і оригінальність алгоритмів вирішення задач;
- Якість оформлення звіту з курсової роботи.

Студент отримує оцінку **задовільно**, якщо:

- У встановлений термін він представив звіт з курсової роботи;
- Програмне забезпечення має блочну структуру із викликом кожного блоку через систему меню
- В програмі коректно працюють процедури введення даних і виведення результатів, виведення відомостей про авторів програми і про роботу з нею;
- Коректно відображається спрощений ескіз механізму;
- Коректно працює процедура кінематичного моделювання механізму (спрощена структура);
- Студент при захисті роботи може пояснити загальну структуру програми, призначення використаних модулів і процедур без детального аналізу їхньої роботи.

Студент отримує оцінку **добре**, якщо:

- Він виконав вимоги до оцінки "задовільно";
- Програма має якісний інтерфейс, аналізує введену інформацію і не дозволяє користувачу вводити некоректні вихідні дані:
- Коректно відображається повний параметричний ескіз механізму;
- Студент при захисті роботи може детально проаналізувати роботу власних модулів і процедур, які використані у програмі.

Студент отримує оцінку **відмінно**, якщо:

- Він виконав вимоги до оцінок "задовільно" і "добре";
- У програмі наявний повний параметричний ескіз механізму;
- Реалізована процедура кінематичного моделювання механізму (ускладнена схема);
- В програмі коректно працює процедура побудови графіків функцій;
- Студент при захисті роботи може детально проаналізувати роботу усіх використаних в програмі модулів і процедур, виявляє при цьому ґрунтовні знання;
- У програмі використані власні оригінальні алгоритми.

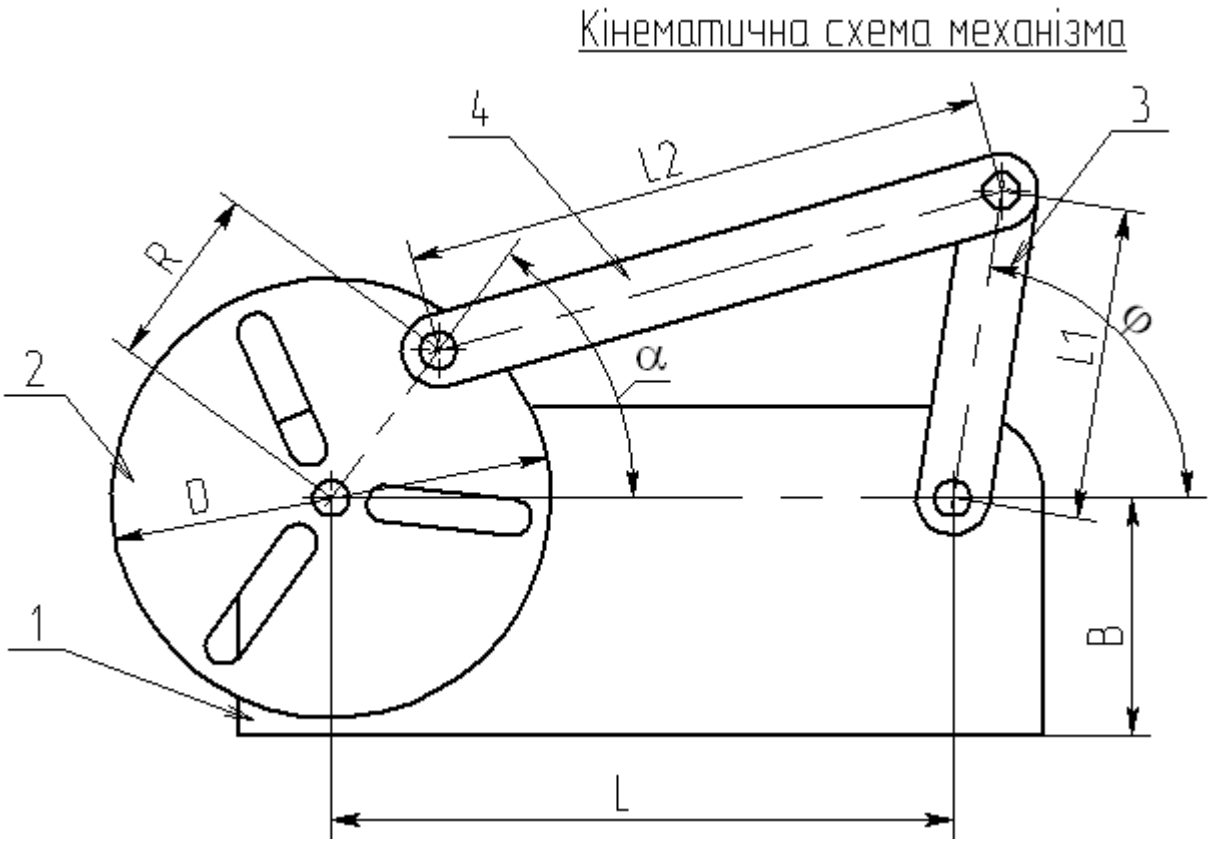
Приклад виконання практичної частини курсової роботи

Виконання курсової роботи будемо проводити відповідно до рекомендацій щодо розробки програмних систем.

- Постановка задачі
- Аналіз задачі
- Розробка алгоритму вирішення задачі
- Проектування загальної структури програми і структури окремих блоків
- Кодування
- Налагодження і верифікація програми
- Отримання результату і його інтерпретація
- Передача замовнику результату роботи
- Супровід системи

Постановка задачі

Розробити діалогову систему розрахунку і моделювання кривошипно-коромислового механізму. Така система повинна забезпечувати введення даних про довжини ланок механізму, розраховувати кут нахилу коромисла і передаточне відношення при будь-якому положенні кривошипу, відображати ескіз механізму, імітувати його роботу, виводити графіки залежності кута нахилу коромисла та передаточного відношення від кута повороту кривошипу.



Привідним ланцюгом є кривошип 2, який обертається навколо вісі корпусу 1.

Основними геометричними розмірами механізму є:

- L - відстань між центрами обертання;
- R - відстань між віссю обертання кривошипу та шипом на ньому;
- L1 - довжина коромисла;
- L2 - довжина шатуна.

Додатковими геометричними розмірами механізму є:

- D - діаметр кривошипу;
- B - відстань від основи корпусу до осей обертання кривошипу та коромисла.

Співвідношення розмірів механізму, які визначають можливість конструктивного виконання механізму, наведені далі.

Конструктивні елементи і розміри:

1. Корпус L = від ... до ... мм

B > D/2

2. Кривошип R = від 10 до L–20 мм

D = 2*R+10 мм

3. Коромисло L1 >= R+10 мм

4. Шатун $L2^2=(L-R)^2+L1^2$

Положення механізму визначається кутом обертання кривошипу (α) від якого, а також від основних геометричних розмірів, залежить кут нахилу коромисла (φ). Розрахункові формули наведені далі.

Розрахункові формули:

Функція положення коромисла :

$$\varphi = \pi - \arcsin\frac{R\cdot\sin\alpha}{Lk} + \arccos\frac{L1^2 + Lk^2 - L2^2}{2\cdot L1\cdot Lk},$$

де $Lk = \sqrt{R^2 + L^2 - 2\cdot R\cdot L\cdot\cos\alpha}$

Передаточна функція :

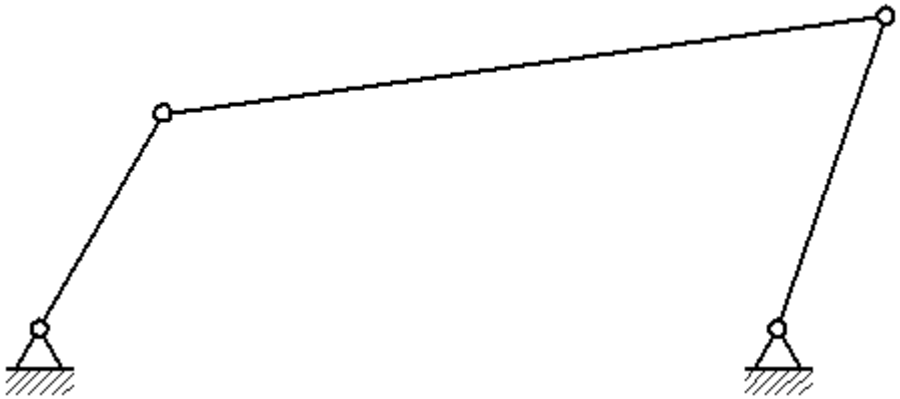
$$u = \frac{R\cdot\sin(\alpha-\mu)}{L1\cdot\sin(\varphi-\mu)},$$

де $\mu = \arctan\left(\frac{-R\cdot\sin\alpha}{L - R\cdot\cos\alpha}\right) + \arccos\frac{L2^2 + Lk^2 - L1^2}{2\cdot L2\cdot Lk},$

де $Lk = \sqrt{R^2 + L^2 - 2\cdot R\cdot L\cdot\cos\alpha}$

Спрощена кінематична схема символізує той самий механізм, у якому відповідні ланцюги замінені ланками, що з'єднують певні вузлові точки механізму.

Спрощена кінематична схема механізма:



Зазначимо, що в учбовій курсовій роботі немає сенсу давати відповіді на запитання, чи має система комерційний інтерес, чим вона відрізняється від існуючих тощо, але важливо з'ясувати які алгоритми і модулі ми можемо запозичити для вирішення поставленого завдання. Тут, зокрема, можемо використовувати стандартні модулі Turbo Pascal або компоненти Delphi, а також допоміжні модулі.

Аналіз задачі

Привідним ланцюгом механізму є кривошип 2, який обертється навколо вісі, закріпленої у корпусі 1. Із кривошипом 2 та коромислом 3 з'єднаний шатун 4, який передає рух. Таким чином при обертальному русі кривошипу 2 коромисло 3 виконує гойдальні рухи.

У постановці задачі до курсової роботи значення діапазонів довжин вказуються викладачем. Домовимось, що діапазон міжцентрової відстані L знаходиться в межах від 30 до 100 мм. Значення решти розмірів або розраховуються точно (D), або уводяться з клавіатури у діапазоні, якій суттєво залежить від міжцентрової відстані L. Відмітимо, що програмна система, яка розробляється повинна забезпечити суворий контроль уведеної інформації, адже невірні дані унеможливлюють отримання коректного результату.

Аналіз розрахункових залежностей потрібно проводити з точки зору виникнення небажаних ситуацій під час розрахунків. До таких ситуацій у першу чергу належить ділення на нуль та знаходження кореня від'ємного числа. З цього боку слід забезпечити наступне:
$L1 \neq 0,\ Lk \neq 0,\ \varphi \neq \mu,\ L > R,\ L2 \neq 0.$
Після аналізу розмірів механізму стає зрозумілим, що контроль введеної інформації унеможливлює ситуації ділення на нуль або знаходження кореня відє'много числа, тобто відповідні функції не потребують додаткових заходів аналізу даних.

Подальший аналіз в учбовій курсовій роботі можна не проводити, адже постановка задачі достатньо формалізована і містить математичну модель механізму.

Розробка алгоритму вирішення задачі

З огляду на розрахункові залежності, які визначають модель механізму, у даному випадку потрібно спиратися на класичні розгалужені і циклічні алгоритми, а також алгоритми знаходження найменшого і найбільшого значення. Доцільно розбити задачу на окремі процедури, алгоритми яких будуть суттєво простішими.

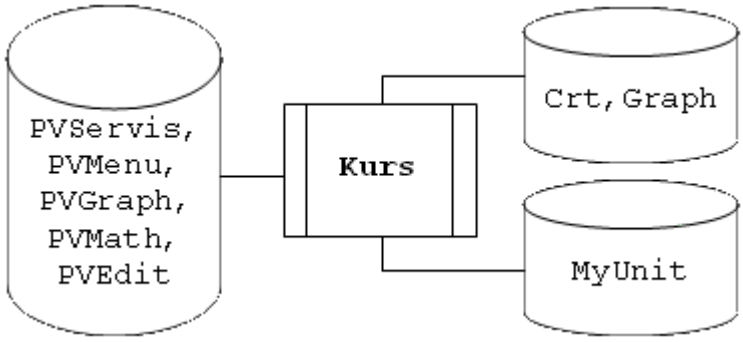
Проектування загальної структури програми

Зміст

1. Загальна структура програми

Діалогова система повинна забезпечувати багаторазове виконання певних дій, тому доцільно створити загальну оболонку системи, а певні частини програми реалізувати у вигляді окремих процедур. Наша система буде використовувати стандартні модулі (`Crt, Graph`), додаткові модулі (`PVServis, PVMenu, PVGraph, PVMath, PVEdit`) і власний модуль (`MyUnit`).

Представимо загальну структуру програми.



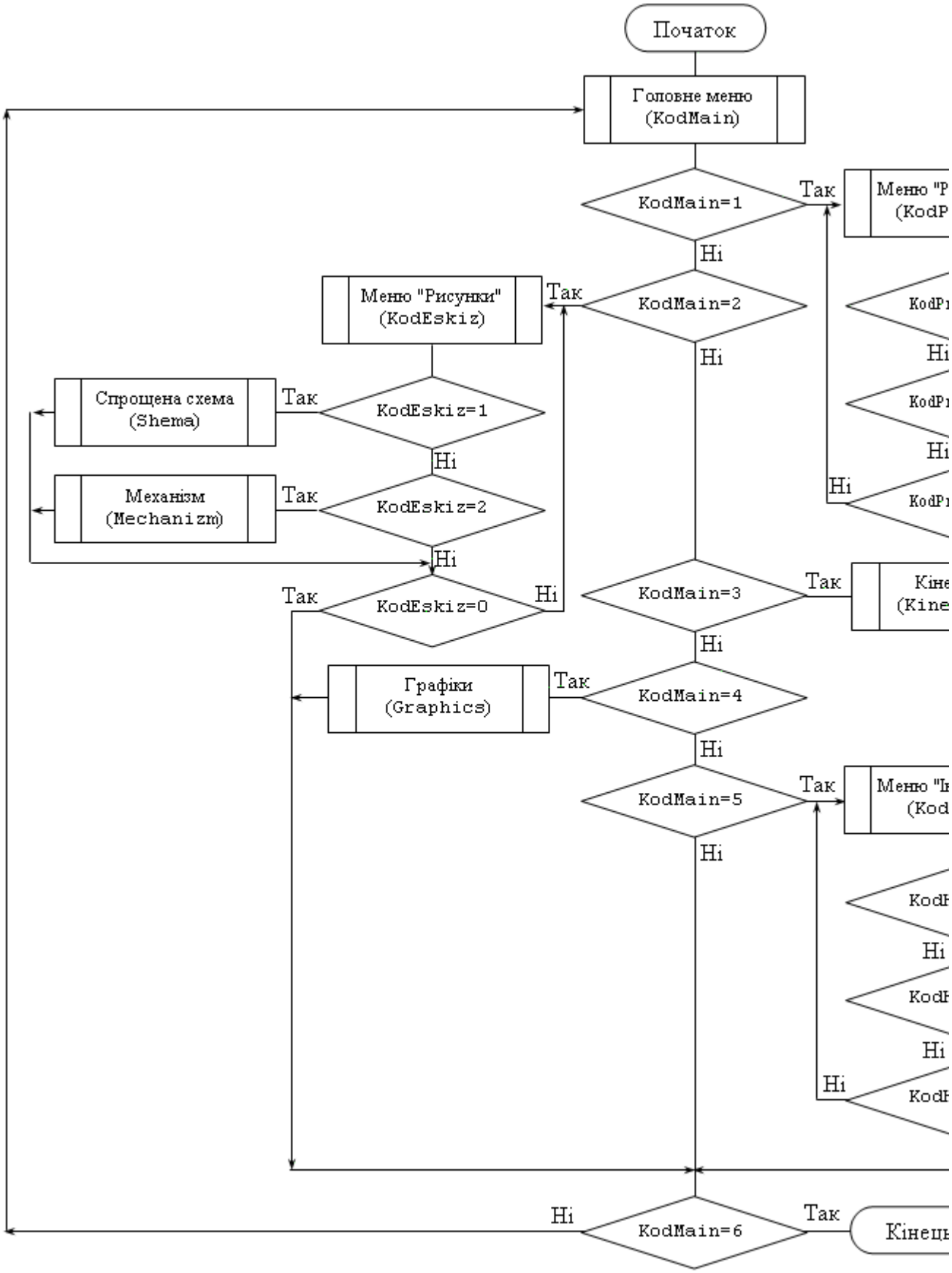
Призначення програми `kurs` - організація всієї системи у єдине ціле з можливістю багаторазового виконання певних дій.
Призначення модулю `MyUnit` - збереження певних процедур і функцій, які реалізують структурно відокремлені, простіші дії, такі як: введення початкових даних, виведення результатів, виведення спрощеної схеми механізму, виведення повного ескізу механізму, імітація роботи механізму, виведення графіків функцій, виведення інформації про програму та про автора.

Така структуризація дозволить поступово і незалежно одна від одної розробляти відповідні процедури, після чого підключати їх до загальної програми.

На початок сторінки

2. Структура основної програми

Структура основної програми `kurs` представлена на рисунку.



Основою алгоритму є цикл, завершення якого відбувається при значенні змінної `KodMain=6`. Своє значення ця змінна отримує під час роботи процедури виведення головного меню. Сама процедура `SMenu`, яка реалізує виведення меню запозичена з модулю `PVMenu`. Аналіз значення змінної `KodMain` дозволяє спрямувати алгоритм по одному з п'яти

напрямків:

Перший напрямок реалізує виведення підменю "Розрахунки". Результатом роботи цієї процедури є змінна `KodProces`, в залежності від значення якої можна викликати або процедуру введення вихідних даних (`InputDatas`) або виведення результатів (`Result`).

Другий напрямок реалізує виведення підменю "Рисунки". Результатом роботи цієї процедури є змінна `KodEskiz`, в залежності від значення якої можна викликати або процедуру відображення спрощеного ескізу механізму (`Shema`) або повного зображення механізму (`Mechanizm`).

Третій напрямок дозволяє викликати процедуру імітації роботи механізму (`Kinematika`).

Четвертий напрямок дозволяє викликати процедуру виведення графіків функцій (`Graphics`).

П'ятий напрямок реалізує виведення підменю "Інформація". Результатом роботи цієї процедури є змінна `KodHelp`, в залежності від значення якої можна викликати або виведення інформацію про механізм та можливості програми (`AboutProgram`) або процедуру виведення інформації про автора програми (`AboutAuthor`).

Розроблена загальна структура програми дозволяє усі основні дії реалізувати через окремі процедури, викликаючи їх скільки завгодно разів і у будь-якій послідовності.

На початок сторінки Текст основної програми

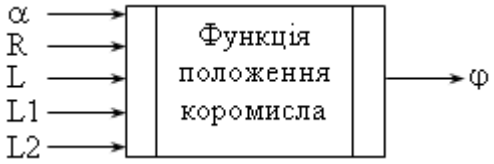
3. Структура елементів власного модулю

3.1. Структура функції положення коромисла

Для однозначного визначення кута φ положення коромисла функція повинна отримувати для розрахунку такі параметри:

- кут повороту кривошипу - α;
- радіус кривошипу - R;
- відстань між опорами - L;
- довжину коромисла - L1;
- довжину шатуна - L2.

Схематично функцію можна відобразити так.

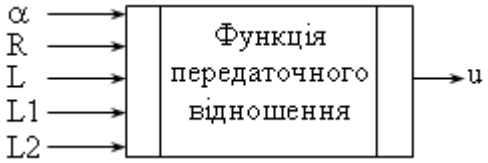


На початок сторінки Текст функції Ffi

3.2. Структура функції передаточного відношення

Для однозначного визначення передаточного відношення `u` функція повинна отримувати для розрахунку такі саме, як і для функції положення коромисла, параметри. Схематично

функцію можна відобразити так.



На початок сторінки Текст функції FU

3.3. Структура процедури введення вихідних даних

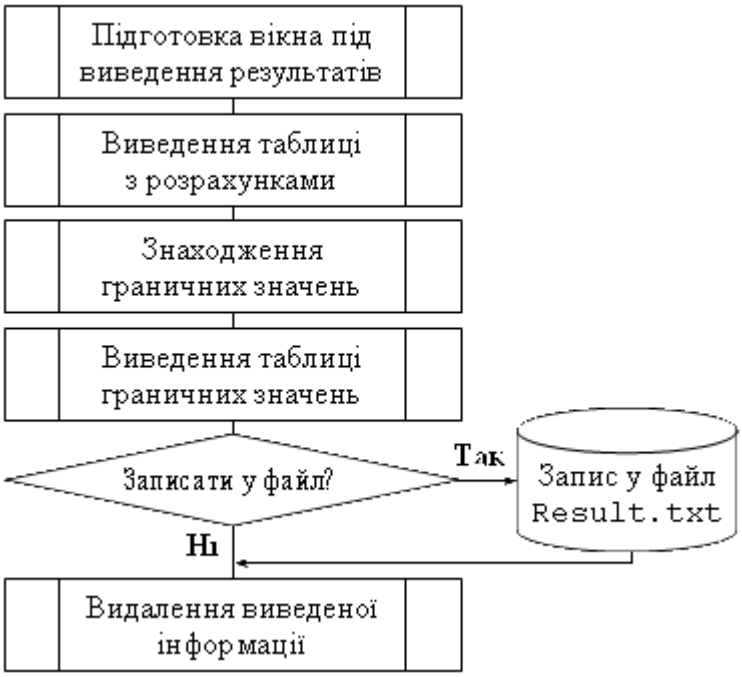
Для організації введення початкових даних в окремому вікні слід всю процедуру розбити на декілька етапів, які наведені на рисунку. Зазначимо, що дані розділяються на такі які слід вводити з клавіатури (`L, R, L1`) і такі, значення яких розраховується (`L2`). Після введення і розрахунку усіх даних на екран слід вивести контрольну інформацію і, після підтвердження користувача (натискання будь-якої клавіші), очищення вікна введення даних.

У тому випадку, коли необхідно організувати безпомилкове введення початкових даних, кожне із них повинно вводитися окремо у відповідності до схеми, зображеної нижче. Суть такого підходу полягає у тому, що перед початком циклу введення даного (адже введення може бути багаторазовим) запам'ятовується поточне положення курсору. Це робиться для того, щоб повторне введення проводилось у тій самій позиції. У циклі курсор переводиться у зафіксовану позицію і всі інформація з цього рядка видаляється. Далі виводиться коментар, який пояснює, що і у яких межах слід вводити, після чого відбувається саме введення даного. Зазначимо, що це введення відбувається спочатку як послідовність символів (щоб унеможливити аварійне завершення програми через помилку введення), а згодом відбувається перетворення цієї послідовності символів у число. Цикл введення завершується тільки у тому випадку, якщо перетворення послідовності символів у число відбулося коректно а саме число потрапило у діапазон припустимих значень. У протилежному випадку цикл введення повторюється.

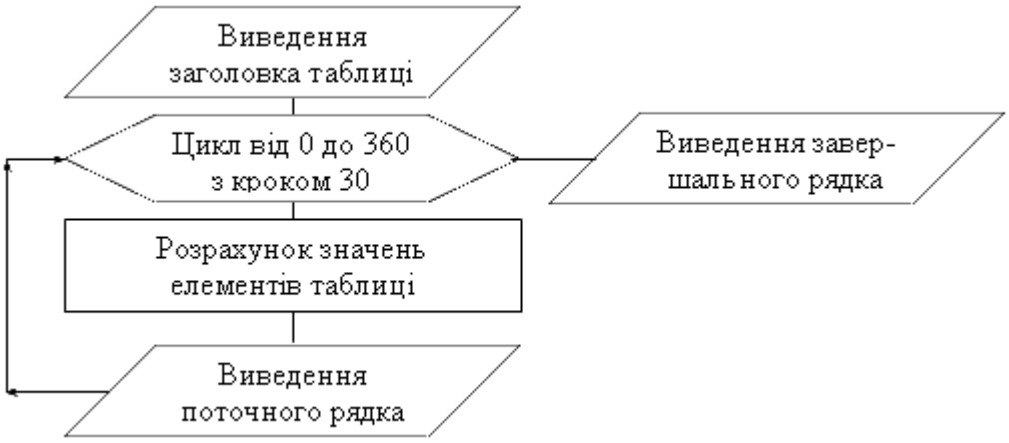
На початок сторінки Текст процедури InputDatas

3.4. Структура процедури виведення результатів

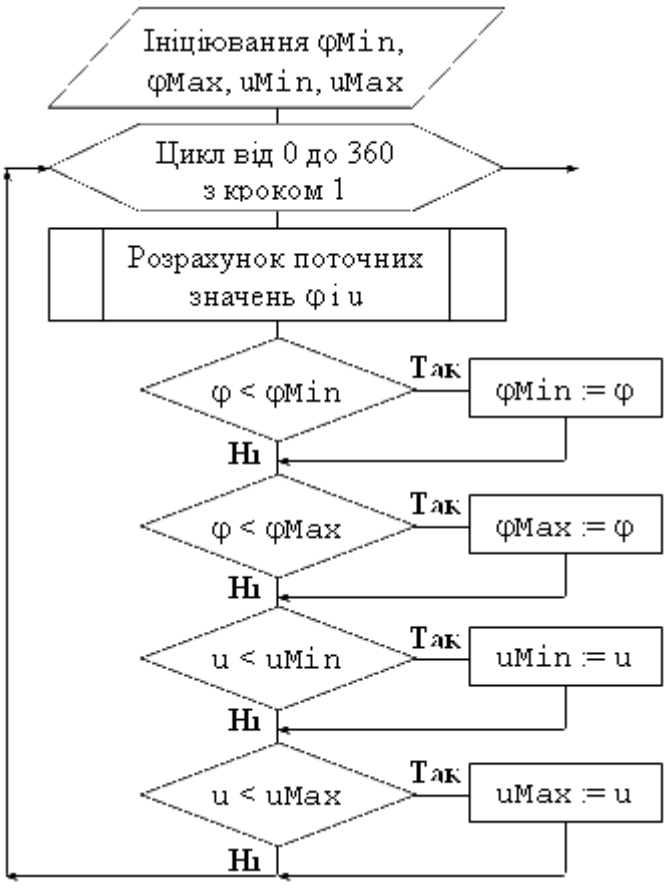
Процедура виведення результатів передбачає послідовне виведення таблиці із розрахунками, таблиці граничних значень і, можливо, запис результатів у файл. Виконуватись такі дії повинні відповідно до загальної схеми виведення результатів, яка зображена нижче.



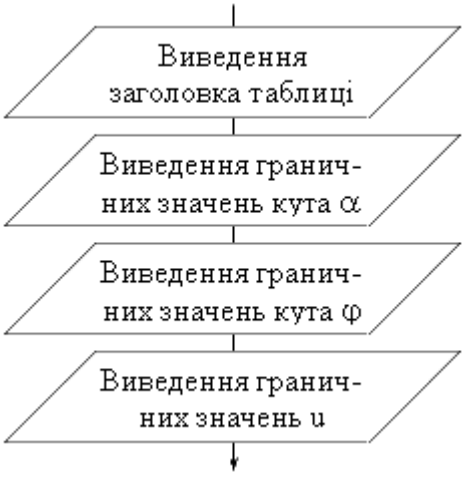
Виведення таблиці з розрахунками складається із виведення заголовка таблиці, розрахунку і виведення елементів таблиці (це відбувається у циклі) та виведення завершального рядка. Нижче наведений алгоритм виведення такої таблиці.



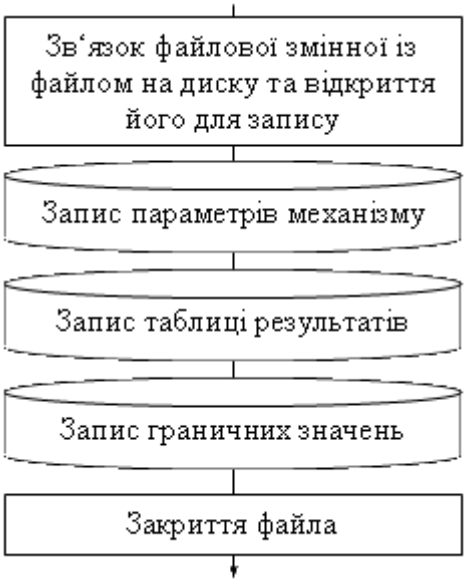
Розрахунок граничних значень будемо виконувати за стандартною схемою, коли спочатку ініціюються початкові граничні значення, а далі у циклі розраховуються поточні значення відповідних величин і проводиться їх порівняння із граничними. У випадку, коли поточне значення виявляється меншим за мінімальне або більшим за максимальне граничне значення, відповідним граничним значенням присвоюються поточні значення. Далі наведений алгоритм пошуку граничних значень.



Після того, як граничні значення знайдені, їх виведення на є проблемою. Послідовність виведення представлена на схемі.



У випадку, коли користувач забажає записати результати розрахунків у файл, зробити це слід у послідовності, яка наведена нижче. Спочатку слід встановити зв'язок файлової змінної із конкретним файлом на магнітному диску і відкрити цей файл для запису. Запис параметрів механізму є аналогічним до виведення контрольної інформації процедури введення початкових даних, запис таблиці результатів і запис граничних значень є аналогічними відповідним блокам процедури виведення результатів.



Результатом роботи усієї процедури виведення результатів є представлена на екрані і, за бажанням користувача, записана у файл інформація про поточні і граничні значення параметрів механізму.

На початок сторінки Текст процедури Result

3.5. Структура процедури виведення спрощеного ескізу

Спрощений ескіз передбачає виведення спрощеної кінематичної схеми, розміри ланок якої є пропорційними до реальних розмірів механізму. Ескіз повинен мати написи біля вузлових точок і текстові позначення лінійних і кутових величин, які визначають вигляд і положення механізму. Виведення самого ескізу передбачає перехід із текстового у графічний режим. Якщо ініціювання такого режиму відбулося без помилок, виконуються блоки, що реалізують розрахунок геометричних параметрів механізму і виведення на екран ескізу механізму. Після того як користувач ознайомився із ескізом і натисканням будь-якої клавіші продовжив

виконання процедури, відбувається закриття графічного режиму, повернення у текстовий режим і завершення процедури виведення спрощеного ескізу механізму. Загальна структура цієї процедури наведена на рисунку.

На початок сторінки Текст процедури Shema

3.6. Структура процедури виведення повного ескізу механізму

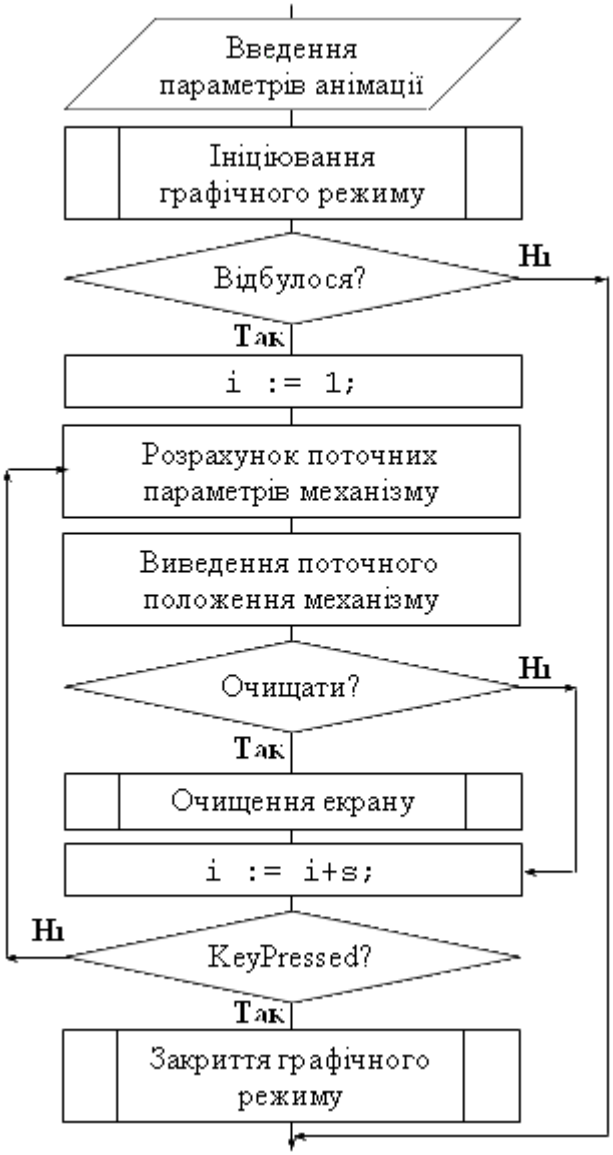
Структура виведення повної схеми механізму практично повністю повторює структуру попередньої процедури. Відмінність полягає у більш детальному пророблені зображення механізму, для чого може знадобитись і, в нашому випадку це було доречним, створення внутрішньої процедури. Загальна структура цієї процедури наведена на рисунку.

Структура процедури виведення ланки наведена на рисунку.

На початок сторінки Текст процедури Mechanizm

3.7. Структура процедури імітації роботи механізму

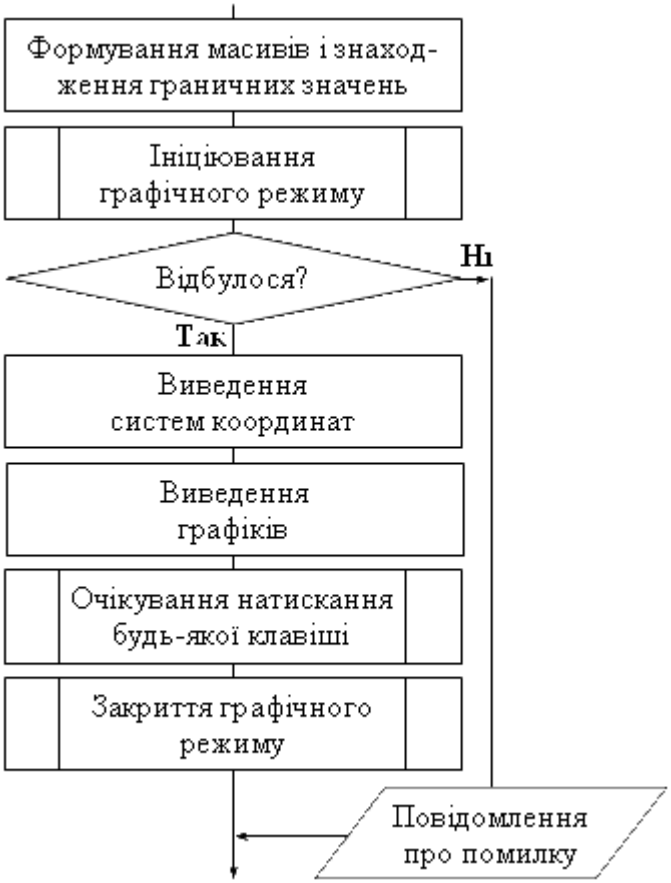
Реалізація анімації роботи механізму потребує початкового введення параметрів анімації. До таких параметрів належать швидкість обертання кривошипу (додатна або від'ємна) та відповідь на питання, потрібно чи ні очищувати екран перед виведенням наступного положення механізму.
Після цього ініціюється графічний режим і, у випадку коректного ініціювання, продовжується виконання процедури. Для відображення поточного положення механізму спочатку встановлюється початкове значення лічильника циклу. Тіло циклу складається із кількох блоків. Спочатку розраховуються поточні параметри механізму. Усі вони залежать від глобальних параметрів - розмірів ланок механізму та локальних параметрів - кутів повороту кривошипу і коромисла, які також залежать від параметру циклу. Наступним блоком є виведення поточного положення механізму і, якщо при входженні у процедуру була дана позитивна відповідь на питання, чи слід очищати екран, відбувається таке очищення. Далі відбувається збільшення (при додатному `s`) або зменшення (при від'ємному `s`) лічильника циклу і, якщо до цього часу не була натиснута жодна з клавіш на клавіатурі, цикл починається заново. Нижче представлена структура процедури імітації роботи механізму.



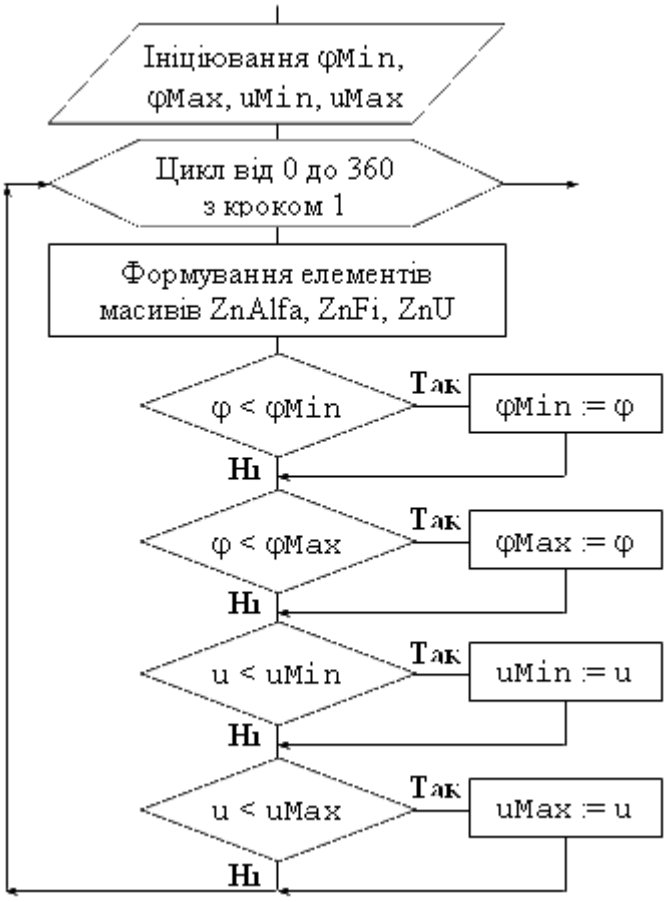
На початок сторінки Текст процедури Kinematika

3.8. Структура процедури виведення графіків

Робота процедури виведення графіків починається із формування масивів значень по яких будуватимуться графіки і знаходженні граничних значень цих елементів. Детальніше робота цього блоку буде розглянута далі. Наступним етапом є ініціювання графічного режиму за умови вдалого виконання якого починається формування графіків. Тут спочатку виводяться системи координат із осями, градуюваннями і поясненнями осей. Далі виводяться безпосередньо графіки, які "накладаються" на системи координат. Після натискання користувачем будь-якої клавіші графічний режим закривається і звершується робота всієї процедури. Так саме завершується робота процедури і після невдалого ініціювання графічного режиму, що супроводжується відповідним повідомленням. Загальна структура процедури виведення графіків наведена на рисунку.



Формування масивів елементів та знаходження граничних значень фактично повторює аналогічну частину процедури виведення результатів, але тут значення не тільки розраховуються, а й додатково записуються як елементи відповідних масивів.



На початок сторінки Текст процедури Graphics

3.9. Структура процедури виведення інформації про програму

Виведення інформації про програму повинно проводитися в окремому вікні. Сама інформація повинна включати відомості про основні можливості програми і про те як ними скористатися. Після отримання підтвердження від користувача (натискання будь-якої клавіші) слід видалити з екрану всю виведену інформацію. Структура дій по відображенню інформації про програму наведена нижче.

На початок сторінки Текст процедури AboutProgram

3.10. Структура процедури виведення інформації про автора

Структура процедури виведення інформації про автора нічим не відрізняється від попередньої процедури. Відмінним є лише зміст цієї інформації (див. рис.)

На початок сторінки Текст процедури AboutAuthor

Код програми

Зміст

1. Основна програма

```
uses Crt,Graph,MyUnit,PVMath,PVServis,PVEdit,PVMenu,PVGr,PVMech;
BEGIN
 repeat
   TextAttr := $07;
   clrscr;
   CursorOff;
   SMenu(KodMain,1,1,6,MenuMain,g,$07,$07,$70,2,0);
   case KodMain of
        1: repeat
             SMenu(KodProces,1,4,2,MenuProces,v,$07,$07,$70,1,0);
```

```
              case KodProces of
                   1: InputDatas;
                   2: Result;
              end; {case}
            until KodProces=0;
         2: repeat
              SMenu(KodEskiz,18,4,2,MenuEskiz,v,$07,$07,$70,1,0);
              case KodEskiz of
                   1: Shema;
                   2: Mechanizm;
              end; {case}
            until KodEskiz=0;
         3: Kinematika;
         4: Graphics;
         5: repeat
              SMenu(KodHelp,52,4,2,MenuHelp,v,$07,$07,$70,1,0);
              case KodHelp of
                   1: AboutProgram;
                   2: AboutAuthors;
              end; {case}
            until KodHelp=0;
    end; {case}
  until KodMain = 6;
  CursorOn;
  clrscr;
END.
```

Коментарі до коду програми

Основна програма зберігається на компакт диску в папці `Kurs` під ім'ям `Kurs.pas`.

У секції `uses` перелічені усі модулі, які необхідні для виконання програми. Зовнішню частину програми складає цикл `repeat ... until KodMain=6`. Умова завершення циклу відповідає вибору шостого елементу головного меню програми, тобто пункту `"Вихід"`.
На початку циклу встановлюється основний колір програми (`$07` - на чорному фоні символи світло-сірого кольору), очищується екран і відключається мерехтіння курсору (`CursorOff`).
Процедура `SMenu` потребує таких параметрів: `KodMain` - ім'я змінної, яка зберігатиме код вибраного пункту меню; `1,1` - координати розташування лівого верхнього кута меню; `6` - кількість пунктів головного меню; `MenuMain` - ім'я типізованої константи з модулю `MyUnit`, яка зберігає тексти пунктів меню; `g` - ознака горизонтального типу меню; `$07,$07,$70` - атрибути кольорів, відповідно, на який колір накладається меню (для відновлення початкового стану екрану), яким кольором виводити пункти меню, яким кольором відображати вибраний пункт меню; `2` - обводити головне меню подвійною рамкою; `0` - не імітувати тінь під меню.
Після завершення роботи процедури `SMenu` (а завершує вона свою роботу у випадку натискання клавіші `Enter` або `Esc`) оператором `case` аналізується код обраного пункту меню. Для деяких пунктів (3 і 4) викликаються процедури, дії яких відповідають текстовим написам пунктів головного меню, а для інших (1,2 і 5) - створюються вкладені цикли, що реалізують спадаючі меню. Ці цикли також містять всередині процедури `SMenu` із аналогічними параметрами (уточнити значення параметрів можна у модулі `PVMenu`). Далі у цих циклах аналізується коди відповідних меню і викликаються відповідні процедури.
Вихід із підпорядкованих меню відбувається натисканням клавіші `Esc`, завершення програми в цілому - вибором шостого пункту головного меню.

На початок сторінки Структура основної програми

2. Власний модуль

Власний модуль є місцем збереження усіх змінних, констант, функцій і процедур, які автор програми самостійно розробляє для реалізації програми. Повний вихідний текст модулю розміщений на компакт-диску в папці `Kurs` під ім'ям `MyUnit.pas`.

2.1. Секція оголошень

```
unit MyUnit;
interface
uses Crt,Graph,PVMath,PVServis,PVEdit,PVMenu,PVGr, PVMech ;
const
 MenuMain: MenuArr = (' Розрахунки ',
                      ' Рисунки    ',
                      ' Рух        ',
                      ' Графіки    ',
                      ' Інформація ',
                      ' Вихід      ',
                      '','','','');
 MenuProces:MenuArr= (' Введення даних ',
                      ' Результати     ',
                      '','','','','','','','');
 MenuEskiz:Menuarr = (' Схема    ',
                      ' Механізм ',
                      '','','','','','','','');
 MenuHelp: MenuArr = (' Робота з програмою ',
                      ' Про автора         ',
                      '','','','','','','','');
var
  KodMain,KodHelp,KodEskiz,KodGraph,KodProces:byte;
  L2,alfa,fi,u,mju,Lk,fiMin,fiMax,UMin,UMax:real;
  i:integer;
const { Ініціювання початкових даних }
   L  :real = 180.0;
   R  :real = 60;
   D  :real = 130;
   B  :real = 100;
   L1 :real = 120;

   function Ffi(alfa,R,L,L1,L2:real):real;
   function Fu(alfa,R,L,L1,L2:real):real;
   procedure InputDatas;
   procedure Result;
   procedure Shema;
   procedure Mechanizm;
   procedure Kinematika;
   Procedure Graphics;
   Procedure AboutProgram;
   Procedure AboutAuthors;
implementation
. . . . . .
begin
   L2:=sqrt(sqr(L-R)+sqr(L1));
end.
```

Коментарі до коду програми

В інтерфейсній частині у секції `uses` перелічуються інші модулі, які необхідні для реалізації власного модуля. Далі описуються, як типізовані константи `MenuMain, MenuProces, MenuEskiz, MenuHelp`, що містять тексти відповідних меню. Загальна кількість елементів

кожного меню, включаючи й пусті, повинна складати 10. Для покращення зовнішнього вигляду меню рекомендується першим і останнім символом тексту пункту меню ставити пробіл, крім того, довжину пунктів слід робити однаковою (по найдовшому пункту).
Далі описуються глобальні змінні, які визначатимуть розміри механізму, його положення і граничні значення. Тут саме визначаються службові змінні, які необхідні для організації циклів, меню тощо.
Далі знову описуються типізовані константи, що визначають початкові дані, які будуть присвоєні відповідним змінним від початку роботи програми. Нагадаємо, що значення цих змінних можна буде змінювати під час роботи програми. У цій секції описуються ті змінні, значення яких повинні вводитись користувачем. Ті ж змінні, значення яких повинні розраховуватись, визначаються як звичайні змінні, але в ініціюючій секції модулю слід розрахувати їхні значення (див. передостанній рядок коду програми).
Також в інтерфейсній секції описуються заголовки та параметри процедур і функцій.
Секція реалізації складається із реалізацій відповідних процедур і функції, що буде розглянуті далі.

На початок сторінки

2.2. Функція положення коромисла

```
function Ffi(alfa,R,L,L1,L2:real):real;
begin
  Lk:=sqrt(sqr(R)+sqr(L)-2*R*L*cos(alfa));
  Ffi:=Pi-(arcsin(R*sin(alfa)/Lk)+
       arccos((sqr(L1)+sqr(Lk)-sqr(L2))/(2*L1*Lk)));
end;
```

Коментарі до коду програми

У функцію передаються параметри, які однозначно дозволяють визначити положення (кут повороту) коромисла. Першим рядком функції розраховується значення Lk, а другим - безпосередньо значення функції. При створення функції обов'язково слід чітко визначити у яких одиницях передаються значення параметрів до неї і у яких одиницях функція видає результат. У нашому випадку кутові величини (alfa) передаються в радіанах, лінійні (R, L, L1, L2) - у міліметрах. Результат функції (fi) - кутова величина у радіанах.

На початок сторінки Структура функції Ffi

2.3. Функція передаточного відношення

```
function Fu(alfa,R,L,L1,L2:real):real;
begin
  Lk:=sqrt(sqr(R)+sqr(L)-2+R*L*cos(alfa));
  mju:=arctan(-R*sin(alfa)/(L-R*cos(alfa))+
       arccos((sqr(L1)+sqr(Lk)-sqr(L2))/(2*L1*Lk)));
  Fu:=R*sin(alfa-mju)/(L1*sin(Ffi(alfa,R,L,L1,L2)-mju));
end;
```

Коментарі до коду програми

Функція передаточного відношення вимагає передачі до неї тих саме параметрів, що й попередня функція, виклик якої, до речі, відбувається всередині тіла функції. Відмітимо, що передаточне відношення не має розмірності.

На початок сторінки Структура функції FU

2.4. Процедура введення вихідних даних

```
Procedure InputDatas;
var
  x,y,code:integer;
  s  :string;

begin
  BWindow(7,8,74,24,1,1,$07); {формування вікна введення даних}
  writeln('         Уведіть параметри механізму:         ');
  CursorOn;

  { Організація безпомилкового введення відстані між центрами обертання }
  x:=whereX; y:=whereY; {Фіксація початкового положення курсору}
  repeat {Початок циклу обробки помилок введення }
    GotoXY(x,y);  DelLine; {Переведення курсору і очищення поточного рядка}
    write(' Відстань між центрами обертання L (100..250mm)       = ');
    readln(s); {Зчитування даних як рядка символів}
    val(s,L,Code); {Перетворення рядка у число}
  until (L>=100) and (L<=250) and (Code=0); {умова завершення циклу}

  { Організація безпомилкового введення радіусу кривошипа }
  x:=whereX; y:=whereY;
  repeat
    GotoXY(x,y); DelLine;
    {Найбільша довжина кривошипу розраховується}
    write(' Радіус кривошипа R (10..',L-20:6:2,'mm) = ');
    readln(s);
    val(s,R,Code);
  until (R>=10) and (R<=L-20) and (Code=0);

  { Розрахунок діаметра кривошипа }
  D:=2*R+10;

  { Організація безпомилкового введення висоти розташування опор }
  x:=whereX; y:=whereY;
  repeat
    GotoXY(x,y); DelLine;
    {Мінімальна висота розташування опор розраховується}
    write(' Висота розташування опор B (',D/2:6:2,'.. mm) = ');
    readln(s);
    val(s,B,Code);
  until (B>=D/2) and (Code=0);

  { Організація безпомилкового введення довжини коромисла }
  x:=whereX; y:=whereY;
  repeat
    GotoXY(x,y); DelLine;
    {Мінімальна довжина коромисла розраховується}
    write(' Довжина коромисла L1(',R+10:6:2,'.. mm) = ');
    readln(s);
    val(s,L1,Code);
  until (L1>=R+10) and (Code=0);

  { Розрахунок довжини шатуна L2}
  L2:=sqrt(sqr(L-R)+sqr(L1));

  { Виведення контрольної інформації }
  writeln(' Контрольна інформація:');
  writeln(' Відстань між центрами обертання L  = ',L:6:2,' мм');
  writeln(' Радіус кривошипа                R  = ',R:6:2,' мм ');
```

```
  writeln(' Діаметр диска кривошипа          D  = ',D:6:2,' мм ');
  writeln(' Висота розташування опор         B  = ',B:6:2,' мм ');
  writeln(' Довжина коромисла                L1 = ',L1:6:2,' мм ');
  writeln(' Довжина шатуна                   L2 = ',L2:6:2,' мм ');
  write  (' Для завершення введення натисніть будь-яку клавішу ...');
  CursorOff;

  repeat until KeyPressed;
  window(7,8,75,24);
  TextAttr:=$07;
  clrscr;
end;{Procedure InputDatas;}
```

Коментарі до коду програми

Всередині процедури визначаються локальні змінні (`x, y`), які потрібні для запам'ятовування поточного розташування курсору і рядкова змінна (`s`) для організації безпомилкового введення даних.
Кожне дане вводиться окремо і організовується його введення спочатку як послідовність символів, що згодом перетворюється у число. У випадку виникнення помилок введення (використовуються неприпустимі символи або число виходить за межі припустимого діапазону) введення даного повторюється.
Порядок введення даних має значення, адже граничні значення деяких даних визначаються попередньо введеними значеннями інших параметрів.
Ті параметри, які чітко залежать від інших розраховуються.
Після введення усіх значень на екран виводиться контрольна інформація, у якій присутні усі параметри - як такі, що вводились з клавіатури, так і ті, значення яких розраховувались.
Відмітимо, що перед початком введення даних слід визначити вікно (виклик процедури `BWindow`) і активізувати мерехтіння курсору (виклик процедури `CursorOn`), а після завершення введення - відключити мерехтіння курсору (виклик процедури `CursorOff`) і очистити зону введення даних.

На початок сторінки Структура процедури InputDatas

2.5. Процедура виведення результатів

Через невідповідність символів кодової таблиці MS-DOS і Windows символи псевдографіки, з яких формуються елементи оформлення таблиць мають інший вигляд.

```
Procedure Result;
   var
   Ch:char;
   f:text;
begin
   Window(5,7,42,25);
   TextAttr:=$07;
   { Розрахунок і виведення таблиці значень }
   writeln('----------T---------T---------¬');
   writeln('¦   alfa  ¦    fi   ¦    u    ¦');
   writeln('+---------+---------+---------+');
   for i := 0 to 12 do begin
     { Розрахунок кута повороту кривошипу (у радіанах) }
      alfa:=i*30*pi/180;
     { Розрахунок кута повороту коромисла (через виклик функції)}
     fi:=Ffi(alfa,R,L,L1,l2);
     { Розрахунок передаточного відношення (через виклик функції)}
     u:= Fu(alfa,R,L,L1,l2);
     { Виведення розрахованих значень (кутові величини переводяться у градуси }
```

```
  writeln('¦ ',i*30:5,' ¦ ',fi*180/pi:7:2,' ¦',u:8:3,' ¦');
end;
write('L---------+---------+----------');
{ Знаходження мінімальних і максимальних значень }
FiMax:=-9999; FiMin:=9999;
UMax :=-9999; UMin :=9999;
for i:= 0 to 360 do begin
  alfa:=i*Pi/180;
  fi:=Ffi(alfa,R,L,L1,l2);
  u:= Fu(alfa,R,L,L1,l2);
  if fi<fiMin then fiMin:=fi;
  if fi>fiMax then fiMax:=fi;
  if u<uMin   then uMin :=u;
  if u>uMax   then uMax :=u;
end;
{ Виведення таблиці граничних значень }
Window(43,7,75,25);
writeln('-----------------------------¬');
writeln('¦ Граничні значення          ¦');
writeln('+-----------------------------+');
writeln('¦ Кут повороту кривошипу     ¦');
writeln('¦ Мінімальний :   0 град.    ¦');
writeln('¦ Максимальний: 360 град.    ¦');
writeln('+-----------------------------+');
writeln('¦ Кут повороту коромисла     ¦');
writeln('¦ Мінімальний : ',FiMin*180/pi:6:2,' град.¦');
writeln('¦ Максимальний: ',FiMax*180/pi:6:2,' град.¦');
writeln('+-----------------------------+');
writeln('¦ Передаточне відношення     ¦');
writeln('¦ Мінімальне : ',UMin:7:3,'     ¦');
writeln('¦ Максимальне: ',UMax:7:3,'     ¦');
writeln('L-----------------------------');
writeln('Зберегти результати у файлі');
writeln('result.txt (Y/N)?');
Ch:=readkey;
{ Реалізація запису у файл при позитивній відповіді }
if Ch in ['Y','y','Н','н']
then begin
  assign(f,'result.txt');{ Зв'язати файлову змінну f із файлом 'result.txt'}
  rewrite(f);{ Відкрити файл для запису даних }
  { Запис параметрів механізму }
  writeln(f,' Параметри механізма:');
  writeln(f,' Відстань між центрами обертання L  = ',L:6:2,' мм');
  writeln(f,' Радіус кривошипа                R  = ',R:6:2,' мм');
  writeln(f,' Діаметр диска кривошипа         D  = ',D:6:2,' мм');
  writeln(f,' Висота розташування опор        B  = ',B:6:2,' мм');
  writeln(f,' Довжина коромисла               L1 = ',L1:6:2,' мм');
  writeln(f,' Довжина шатуна                  L2 = ',L2:6:2,' мм');
  writeln(f,' Таблиця розрахункових значень:');
  writeln(f,' alfa fi u');
  for i := 0 to 12 do begin
    alfa:=i*30*pi/180;
    fi:=Ffi(alfa,R,L,L1,l2);
    u:= Fu(alfa,R,L,L1,l2);
    { Запис розрахованих значень }
    writeln(f,i*30:5,fi*180/pi:10:2,u:10:2);
  end;
  { Запис граничних значень }
  writeln(f,' Граничні значення: ');
  writeln(f,'Кут повороту кривошипа');
  writeln(f,'Мінімальний :   0 град.');
  writeln(f,'Максимальний: 360 град.');
  writeln(f,'Кут повороту коромисла ');
  writeln(f,'Мінімальний : ',FiMin*180/pi:6:2,' град.');
```

```
      writeln(f,'Максимальний: ',FiMax*180/pi:6:2,' град.');
      writeln(f,'Передаточне відношення ');
      writeln(f,'Мінімальне : ',UMin:7:3);
      writeln(f,'Максимальне: ',UMax:7:3);
      { Закриття файла }
      close(f);
    end;
    { Завершальне очищення екрану }
    window(5,4,76,25);
    TextAttr:=$07;
    clrscr;
end; {procedure Result}
```

Коментарі до коду програми

Після підготовки вікна під виведення інформації відбувається виведення таблиці значень. Тут особливу увагу слід звернути на узгодження розмірностей параметрів, які передаються у функції, які повертаються ними і на те, яку розмірність мають результати, що виводяться на екран. Особливо це стосується кутових величин, які можуть представлятися у градусах та радіанах.
Основна частина таблиці результатів виводиться у циклі, через параметр якого спочатку розраховується кут повороту кривошипу (`alfa`) та інші параметри.
Перед виведенням граничних значень слід їх віднайти. Для цього у циклі змінюється кут повороту кривошипу і розраховуються відповідні параметри. Якщо поточні значення виявились меншими за мінімальне значення або більшими за максимальне, тоді відповідні граничні значення перевизначаються. Після цього можна вивести таблицю з такими граничними значеннями. Тут слід звернути увагу на те, що максимальні значення цієї таблиці можуть дещо перевищувати а мінімальні значення бути меншими за результати, які відображаються у попередній таблиці результатів. Це може бути лише у тому випадку, якщо екстремум (-ми) функції відповідають куту повороту кривошипу, що не є кратним 30.
Для обробки відповіді, чи потрібно записувати результати у файл використовується функція `ReadKey` зі стандартного модулю `Crt`. Частина коду програми, яка реалізує запис у файл виконується тільки у тому випадку, коли користувач натиснув на клавіатурі клавішу "`Y`" у будь-якому регістрі (в тому числі якщо клавіатура перебувала у стані введення символів кирилиці - латинському символу "`Y`" на клавіатурі відповідає символ кирилиці "`Н`").
Нагадаємо, що робота із файлами розглядалася у темі 18.
Запис результатів у файл фактично є копією фрагментів процедури введення даних (частина, яка реалізує виведення контрольної інформації) і процедури виведення результатів (частини, які реалізують виведення таблиці розрахункових значень і таблиці граничних значень). Відмінність полягає у тому, що в операторах `writeln` першим параметром вказується ім'я файлової змінної. Відсутня також частина розрахунку граничних значень, оскільки вони щойно розраховувались для виведення на екран.
Завершує процедуру очищення тієї частини екрану на яку виводилась інформація.

На початок сторінки Структура процедури Result

2.6. Процедура виведення спрощеного ескізу

```
procedure Shema;
var
  i,gD,gM,EC,
  x0,y0,xA,yA,xB,yB,xC,yC:integer;
begin
   { Ініціювання графічного режиму }
   gD := Detect;
   InitGraph(gD, gM,'');
   EC := GraphResult;
```

```
  if EC <> grOk then Halt(1);
  { Розрахунок геометричних параметрів механізму }
  alfa:=45*Pi/180; {Встановлення кута повороту кривошипа}
  x0:=250; y0:=200; {Встановлення початкової точки механізму - точки обертання к
  xC:=x0+round(L); yC:=Y0;{Розрахунок координат точки обертання коромисла}
  {Розрахунок положення рухомої точки кривошипа}
  xA:=x0+round(R*cos(alfa));
  yA:=y0-round(R*sin(alfa));
  fi:=Ffi(alfa,R,L,L1,L2); {Розрахунок кута повороту коромисла}
  { Розрахунок положення рухомої точки коромисла }
  xB:=xC+round(L1*cos(fi));
  yB:=yC-round(L1*sin(fi));
    SetTextJustify(CenterText,TopText);
  OutTextXY(GetMaxX div 2,0,'Press <Esc> to exit');
  { Виведення опор }
  Opora(x0,y0,270);
  Opora(xC,yC,270);
    SetLineStyle(SolidLn,0,ThickWidth);
  { Виведення ланок механізму }
  MoveTo(x0,y0); LineTo(xA,yA);
  MoveTo(xA,yA); LineTo(xB,yB);
  MoveTo(xB,yB); LineTo(xC,yC);
  { Виведення вузлів з іменами }
  Uzel(x0,y0,'0');
  Uzel(xA,yA,'A');
  Uzel(xB,yB,'B');
  Uzel(xC,yC,'C');
    SetColor(Green);
  { Виведення лінійних розмірів }
  RazmerLinear(x0,y0,xc,yc,'L');
  RazmerLinear(x0,y0,xa,ya,'R');
  RazmerLinear(xa,ya,xb,yb,'L2');
  RazmerLinear(xc,yc,xb,yb,'L1');
  { Виведення кутових розмірів }
  RazmerAngular(x0,y0,xc,yc,xa,ya,'alfa');
  RazmerAngular(xc,yc,xc+1,yc,xb,yb,'fi');
  repeat until KeyPressed;
  closeGraph; {Завершення роботи у графічному режимі}
end;
```

Коментарі до коду програми

Виведення спрощеного ескізу механізму вимагає переходу із текстового у графічний режим. Тому першою секцією процедури є секція ініціювання графічного режиму (детально про ініціювання графічного режиму можна переглянути у темі 20).
У випадку вдалого ініціювання графічного режиму спочатку розраховуються координати базових точок механізму в системі координат екрану (про систему координат екрану у графічному режимі див. тему 20). У нашому випадку такими точками є точки `O,A,B,C`, причому точкою початку локальної системи координат механізму є точка `O`, від її положення залежить положення усіх інших базових точок.
При виведення самого ескізу використовуються процедури `SetTextJustify`, `OutTextXY`, `SetLineStyle`, `MoveTo`, `LineTo`, `SetColor` зі стандартного модулю `Graph` та процедури `Opora`, `Uzel`, `RazmerLinear`, `RazmerAngular` з додаткового модулю `PVMech`.
Завершення відображення спрощеного ескізу закриття графічного режиму та повернення у текстовий режим відбувається після того як користувач натисне на будь-яку клавішу.

На початок сторінки Структура процедури Shema

2.7. Процедура виведення повного ескізу механізму

```
procedure Mechanizm;
var
   i,gD,gM,EC,
   rMax,rMin,x0,y0,xA,yA,xB,yB,xC,yC:integer;
   xp,yp:array[1..3,1..2] of integer;
   s:string;

   { Внутрішня процедура рисування ланки механізму
   x1,y1 - координати початкової точки ланки
   x2,y2 - координати кінцевої точки ланки
   B - половина ширини ланки }
   procedure Zveno(x1,y1,x2,y2,B:longint);
   var L:real;
   p,xa,ya,xb,yb,xc,yc,xd,yd:integer;
   begin
      SetLineStyle(SolidLn,0,ThickWidth);
      L:=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)); {довжина ланки}
      {Розрахунок координат базових точок ланки}
      xa:=x1+round((y2-y1)*B/L); ya:=y1-round((x2-x1)*B/L);
      xb:=x2+round((y2-y1)*B/L); yb:=y2-round((x2-x1)*B/L);
      xc:=x1-round((y2-y1)*B/L); yc:=y1+round((x2-x1)*B/L);
      xd:=x2-round((y2-y1)*B/L); yd:=y2+round((x2-x1)*B/L);
      { Виведення ліній ланки }
      Line(xa,ya,xb,yb); Line(xc,yc,xd,yd);
      { Виведення кіл ланки }
      circle(x1,y1,B);   circle(x2,y2,B);
   end; { procedure Zveno }

begin {Див. також коментарі до процедури Shema }
   gD := Detect;
   InitGraph(gD, gM,'');
   EC := GraphResult;
   if EC <> grOk then Halt(1);
   { Встановлення ы розрахунок параметрыв механізму }
   alfa:=45*Pi/180;
   x0:=250; y0:=200;
   xC:=x0+round(L); yC:=Y0;
   xA:=x0+round(R*cos(alfa));
   yA:=y0-round(R*sin(alfa));
   rMax:=round(R);
   rMin:=round(0.2*R);
   fi:=Ffi(alfa,R,L,L1,L2);
   xB:=xC+round(L1*cos(fi));
   yB:=yC-round(L1*sin(fi));
   { Виведення тексту }
     SetTextJustify(CenterText,TopText);
   OutTextXY(GetMaxX div 2,0,'Press <Esc> to exit');
   { Виведення вузлів }
   Uzel(x0,y0,'0');
   Uzel(xA,yA,'A');
   Uzel(xB,yB,'B');
   Uzel(xC,yC,'C');
   { Рисування диска кривошипа із пазами }
   Circle(x0,y0,round(d/2));
   for i:= 1 to 3 do begin
     xp[i,1]:=x0+round(rMax*cos(alfa+Pi+2*Pi/3*(i-1)));
     yp[i,1]:=y0-round(rMax*sin(alfa+Pi+2*Pi/3*(i-1)));
     xp[i,2]:=x0+round(rMin*cos(alfa+Pi+2*Pi/3*(i-1)));
     yp[i,2]:=y0-round(rMin*sin(alfa+Pi+2*Pi/3*(i-1)));
     Zveno(xp[i,1],yp[i,1],xp[i,2],yp[i,2],2);
   end;
   { Рисування корпуса }
```

```
   arc(x0,y0,90,180,20);
   arc(xc,yc,0,90,20);
   MoveTo(x0,Y0-20);          LineTo(Xc,Yc-20);
   MoveTo(x0-20,y0);          LineTo(x0-20,y0+round(B));
   MoveTo(xC+20,yC);          LineTo(xC+20,yC+round(B));
   MoveTo(xC+20,yC+round(B));LineTo(x0-20,y0+round(B));
   { Рисування ланок механізму }
   Zveno(xa,ya,xb,yb,10);
   Zveno(xc,yc,xb,yb,10);
   { Виведення лінійних і кутових розмірів.
     Перед виведенням відбувається перетворення числа у рядок }
   SetColor(Green);
   str(L:6:2,s);  RazmerLinear(x0,y0,xc,yc,s);
   str(R:6:2,s);  RazmerLinear(x0,y0,xa,ya,s);
   str(L2:6:2,s); RazmerLinear(xa,ya,xb,yb,s);
   str(L1:6:2,s); RazmerLinear(xc,yc,xb,yb,s);
   str(alfa*180/Pi:6:2,s); RazmerAngular(x0,y0,xc,yc,xa,ya,s);
   str(fi*180/Pi:6:2,s);   RazmerAngular(xc,yc,xc+1,yc,xb,yb,s);
   SetColor(Blue);
   str(B:6:2,s); RazmerLinear(x0,y0,x0,y0+round(B),s);
   str(D:6:2,s); RazmerLinear(x0-round(D/2),y0,x0+round(D/2),y0,s);
   repeat until KeyPressed;
   closeGraph;
end;
```

Коментарі до коду програми

Структурно процедура виведення повного ескізу відрізняється від спрощеного додатковою реалізацією внутрішньої процедури `Zveno`, яке рисує ланку механізму (коромисло, шатун) або виріз на диску кривошипу. Така процедура вимагає передачі до неї координат двох вузлових точок ланки і половини її ширини. Пази на диску кривошипу рисуються у циклі. Додатково у цій процедурі використовуються процедури `Circle`, `Arc` зі стандартного модулю `Graph`, а перед використанням процедур `RazmerLinear` і `RazmerAngular` за допомогою стандартної процедури `Str` виконується перетворення числових параметрів довжин ланок і кутів повороту вдані рядкового типу. Це дозволяє у повному ескізі механізму виводити не позначення розмірів, а конкретні значення, що були введені користувачем або розраховані.

На початок сторінки Структура процедури Mechanizm

2.8. Процедура імітації роботи механізму

```
procedure Kinematika;
   var
   i,n,gD,gM,EC,x,y,Code,
   x0,y0,xA,yA,xB,yB,xC,yC:integer;
   ClearFlag:boolean;
   c:char; s:string;

begin
   { Запити на параметри анімації механізму }
   Window(12,10,68,12);
   CursorOn;
   { Запит швидкості обертання механізму }
   x:=WhereX; y:=WhereY;
   repeat
     GotoXY(x,y); DelLine;
     write('Уведіть швидкість обертання механізму (-90..90)  = ');readln(s);
     val(s,n,Code);
   until (Code=0) and (n>=-90) and (n<=90);
```

```
   { Запит на очищення екрану }
   write('Очищувати екран перед виведенням наступного положення (Y/N)?');readln(c
   CursorOff;
   ClearFlag:=(c='Y')or(c='y');
   { Ініціювання графічного режиму }
   gD := Detect;
   InitGraph(gD, gM,'');
   EC := GraphResult;
   if EC <> grOk then Halt(1);
   i:=0;
   repeat { Початок циклу анімації механізму }
      alfa:=i*Pi/180; { Розрахунок кута повороту кривошипа через параметр циклу }
      { Розрахунок положень базових точок механізму }
      x0:=250; y0:=200;
      xC:=x0+round(L); yC:=Y0;
      xA:=x0+round(R*cos(alfa));
      yA:=y0-round(R*sin(alfa));
      fi:=Ffi(alfa,R,L,L1,L2);
      xB:=xC+round(L1*cos(fi));
      yB:=yC-round(L1*sin(fi));
      if ClearFlag then ClearViewPort; {При необхідності очистити екран}
      SetTextJustify(CenterText,TopText);
      OutTextXY(GetMaxX div 2,0,'Press <Esc> to exit');
      { Виведення механізму в певному положенні }
      Opora(x0,y0,270);
      Opora(xC,yC,270);
      SetLineStyle(SolidLn,0,ThickWidth);
      MoveTo(x0,y0); LineTo(xA,yA);
      MoveTo(xA,yA); LineTo(xB,yB);
      MoveTo(xB,yB); LineTo(xC,yC);
      Uzel(x0,y0,'');
      Uzel(xA,yA,'');
      Uzel(xB,yB,'');
      Uzel(xC,yC,'');
      i:=i+n; {Збільшити (зменшити при від'ємному значенні n) лічильник циклу}
   until KeyPressed; {Завершити цикл при натискання на будь-яку клавішу}
   closeGraph; {Завершити роботу в графічному режимі}
end;
```

Коментарі до коду програми

Суть імітації роботи механізму полягає у послідовному виведенні його положень, що змінюються.
Перед початком анімації користувач може ввести швидкість обертання кривошипу, але фактично це є приростом кута повороту, а не швидкістю у звичайному розумінні. Значення швидкості може бути додатним або від'ємним (тоді привідна ланка обертатиметься у протилежному напрямку).
Користувач також може вказати, очищувати чи ні екран перед виведенням наступного положення механізму. Якщо екран очищувати,- буде імітуватися робота механізму, якщо не очищувати,- можна буде оцінити розміри зони, яка потрібна механізмові для роботи.
Після ініціювання графічного режиму, починається цикл анімації механізму. Завершити цей цикл можна у будь-який момент натиснувши будь-яку клавішу. Тіло циклу повністю відповідає виведенню спрощеного ескізу механізму без нанесення розмірів і текстових написів. Наприкінці циклу екран може в залежності від стану логічної змінної `ClearFlag` очищуватися або ні.

На початок сторінки Структура процедури Kinematika

2.9. Процедура виведення графіків

```
Procedure graphics;
   var
   i,gD,gM,EC : integer;
   ZnAlfa,ZnFi,ZnU : ArrReal;
   Dx : real;
begin
   { Формування масивів і знаходження граничних значень }
   fiMin:=9999; fiMax:=-9999; uMin:=9999; uMax:-9999;
   for i := 0 to NumbOfPoint do begin
      { Формування масиву кутів повороту кривошипа (в градусах) }
      ZnAlfa[i] := i;
      alfa:=i*Pi/180; { Розрахунок кута повороту кривошипа (в радіанах) }
      { Формування масиву кутів повороту коромисла (в градусах) }
      ZnFi[i] := Ffi(alfa,R,L,L1,L2)*180/pi;
      { Формування масиву передаточних відношень механізму }
      ZnU[i] := FU (alfa,R,L,L1,L2);
      {Відбір мінімального кута повороту коромисла}
      if ZnFi[i]<FiMin then FiMin:=ZnFi[i];
      {Відбір максимального кута повороту коромисла}
      if ZnFi[i]>FiMax then FiMax:=ZnFi[i];
      {Відбір мінімального передаточного відношення}
      if ZnU[i]<UMin then UMin:=ZnU[i];
      {Відбір максимального передаточного відношення}
      if ZnU[i]>UMax then UMax:=ZnU[i];{ }
   end;
   { Ініціювання графічного режиму}
   gD := Detect;
   InitGraph(gD, gM,'');
   EC := GraphResult;
   if EC = grOk then
   begin
      { Побудова системи координат графіка залежності
      кута повороту коромисла від кута повороту кривошипа }
      BuildCoord(0.1,0.45,0.9,0.1,
                 1,0,1,
                 13,2,0,13,6,2,
                 0,360,FiMin,FiMax,
                 'Fi = f(alfa)','Fi','alfa',
                 2);
      { Побудова системи координат графіка залежності
      передаточного відношення від кута повороту кривошипа }
      BuildCoord(0.1,0.9,0.9,0.5,
                 1,0,1,
                 13,2,0,13,6,2,
                 0,360,UMin,UMax,
                 'U = f(alfa)','U','alfa',
                 2);
      { Побудова графіка залежності
      кута повороту коромисла від кута повороту кривошипа }
      BuildChart(0.1,0.45,0.9,0.1,
                 NumbOfPoint+1,
                 ZnAlfa[0],0,ZnAlfa[NumbOfPoint],360,
                 ZnFi[0],FiMin,ZnFi[NumbOfPoint],FiMax,
                 Red,SolidLn,ThickWidth,
                 ZnAlfa,ZnFi);
      { Побудова графіка залежності
      передаточного відношення від кута повороту кривошипа }
      BuildChart(0.1,0.9,0.9,0.5,
                 NumbOfPoint+1,
                 ZnAlfa[0],0,ZnAlfa[NumbOfPoint],360,
                 ZnU[0],UMin,ZnU[NumbOfPoint],UMax,
                 Red,SolidLn,ThickWidth,
                 ZnAlfa,ZnU);
      Readln;
```

```
      CloseGraph;
   end
   else
      Writeln('Graphics error:', GraphErrorMsg(EC));
end; {graphics}
```

Коментарі до коду програми

Коректна побудова графіків із використанням процедур BuildCoord і BuildChart із додаткового модулю PVGr вимагає формування трьох масивів значень - ZnAlfa, ZnFi, ZnU. Ці масиви будуть передаватися всередину процедур, як параметри і саме вони визначатимуть вид і форму графіків.
Формування цих масивів, а також визначення граничних значень кожного із них, відбувається у циклі. Відмітимо, що представлення інформації про кути повороту в градусах, вимагає і формування значень масивів у градусах, але слід пам'ятати, що розрахунки залежних значень через функції вимагають передавати параметри у радіанах.
Після ініціювання графічного режиму відбувається виведення двох систем координат (процедури BuildCoord). Перша чотири параметри визначають координати (x,y) положення лівого нижнього і правого верхнього кутів графіка у частинах екрану. Тому ці координати є додатними числами в межах від 0 до 1. Наступні три параметри визначають кольори, якими слід заповнити робочу площину графіка, виводити лінії координатних осей і сітки, виводити градуювання осей. Ці кольори можуть передаватися як цифри або як іменовані константи. Наступні дві трійки параметрів визначають для вісі X і для вісі Y кількість точок розбиття, загальну довжину представлення чисел у градуюванні вісі і кількість цифр дробової частини числа такого градуювання. Дальші дві пари параметрів визначають для вісі X і для вісі Y мінімальне і максимальне значення. Це дозволить виділити максимально можливу площу під виведення графіка і покращити його сприйняття. Далі вказуються три рядкових параметри - підпис графіка, підпис вісі X та підпис вісі Y, відповідно. Останній параметр - колір, яким слід виводити ці написи.
Після виведення систем координат на них слід "накласти" відповідні графіки (виклики процедур BuildChart). Для коректного виконання таких дій багато із параметрів цих процедур повинно співпадати із параметрами відповідних процедур BuidCoord. Перші чотири параметри - координати зон екрану повинні відповідати першим чотирьом параметрам відповідних попередніх процедур. Наступний параметр визначає кількість точок з яких складається графік. Чим більшим є це число тим більш плавним буде графік. Значення константи NumbOfPoint зберігається в модулі PVGr і змінити її значення можна лише в цьому модулі з наступною перекомпіляцією. Наступна четвірка параметрів має відношення до вісі абсцис і визначає значення першого елемента у масиві, значення мінімального елемента, значення останнього елемента і значення максимального елемента у масиві, який зберігає значення осі абсцис - кута alfa повороту кривошипу. Подальша четвірка параметрів вказує ті самі параметри але вже для вісі ординат - або кута fi повороту коромисла, або передаточного відношення u. Далі три параметри визначають вид лінії якою буде формуватися графік. Перший із них визначає колір лінії, другий - її тип, третій - товщину. останні два параметри - масиви по яких будуватиметься графік.

На початок сторінки Структура процедури Graphics

2.10. Процедура виведення інформації про програму

```
Procedure AboutProgram;
begin
   BWindow(5,6,75,23,1,1,$07);
   writeln(' Програма призначена для розрахунку і моделювання кривошипно-   ');
   writeln('коромислового механізму.                                        ');
   writeln(' Ведучою ланкою є кривошип, що обертається навколо точки O із   ');
```

```
   writeln('сталою кутовою швидкістю. При цьому коромисло, з''єднане із     ');
   writeln('кривошипом здійснює гойдальний рух навколо точки С.             ');
   writeln(' Програма дозволяє ввести вихідні дані (довжини ланок механізму)');
   writeln('- команда "Розрахунки/Вихідні дані" і розрахувати значення кута ');
   writeln('повороту коромисла і передаточного відношення - команда "Розра- ');
   writeln('хунки/Результати". Тут саме визначаються граничні значення усіх ');
   writeln('параметрів. Передбачена можливість запису результатів у файл.   ');
   writeln(' Команда "Рисунки" дозволяє переглянути спрощену і повну кінема-');
   writeln('тичні схеми механізму, а команда "Рух" дозволяє імітувати роботу');
   writeln('механизму із різною швидкістю і у різних напрямках обертання.   ');
   writeln(' Команда "Графіки" виводить залежності кута повороту коромисла і');
   write  ('передаточного відношення від кута повороту кривошипу.           ');
   repeat until KeyPressed;
   window(5,6,77,24);
   TextAttr:=$07;
   clrscr;
end;
```

Коментарі до коду програми

Процедура виведення інформації про програму є однією із найпростіших у програмі. Вона вимагає коректного визначення розмірів вікна для виведення (процедура `BWindow`), виведення короткої інформації про можливості програми (на одну екранну сторінку) та очищення виведеної інформації після підтвердження користувача.

На початок сторінки Структура процедури AboutProgram

2.11. Процедура виведення інформації про автора

```
Procedure AboutAuthors;
begin
   BWindow(12,6,68,23,1,1,$07);
   writeln('    Національний технічний університет України    ');
   writeln('       "Київський політехнічний інститут"         ');
   writeln('       Кафедра технології машинобудування         ');
   writeln('                                                  ');
   writeln('      Д И А Л О Г О В А     С И С Т Е М А         ');
   writeln('           розрахунку і моделювання               ');
   writeln('       кривошипно-коромислового механізму         ');
   writeln('                  (Варіант __)                    ');
   writeln('                                                  ');
   writeln('                                                  ');
   writeln('                        Виконав :студент гр.МТ-21с');
   writeln('                                 Коваленко С.В.   ');
   writeln('                                                  ');
   writeln('                        Керівник:доц.Пасічник В.А.');
   writeln('                                                  ');
   write  ('                  Київ-2003                       ');
   repeat until KeyPressed;
   window(12,6,70,24);
   TextAttr:=$07;
   clrscr;
end;
```

Коментарі до коду програми

Структура процедури виведення інформації про автора схожа на структуру попередньої процедури і повинна містити таку інформацію: назва організації у якій виконана робота, назву

системи і номер варіанту, прізвище і групу студента-автора програми, посаду і прізвище керівника, місце і рік виконання роботи.

На початок сторінки Структура процедури
AboutAuthor

Налагодження і верифікація програми

Зміст

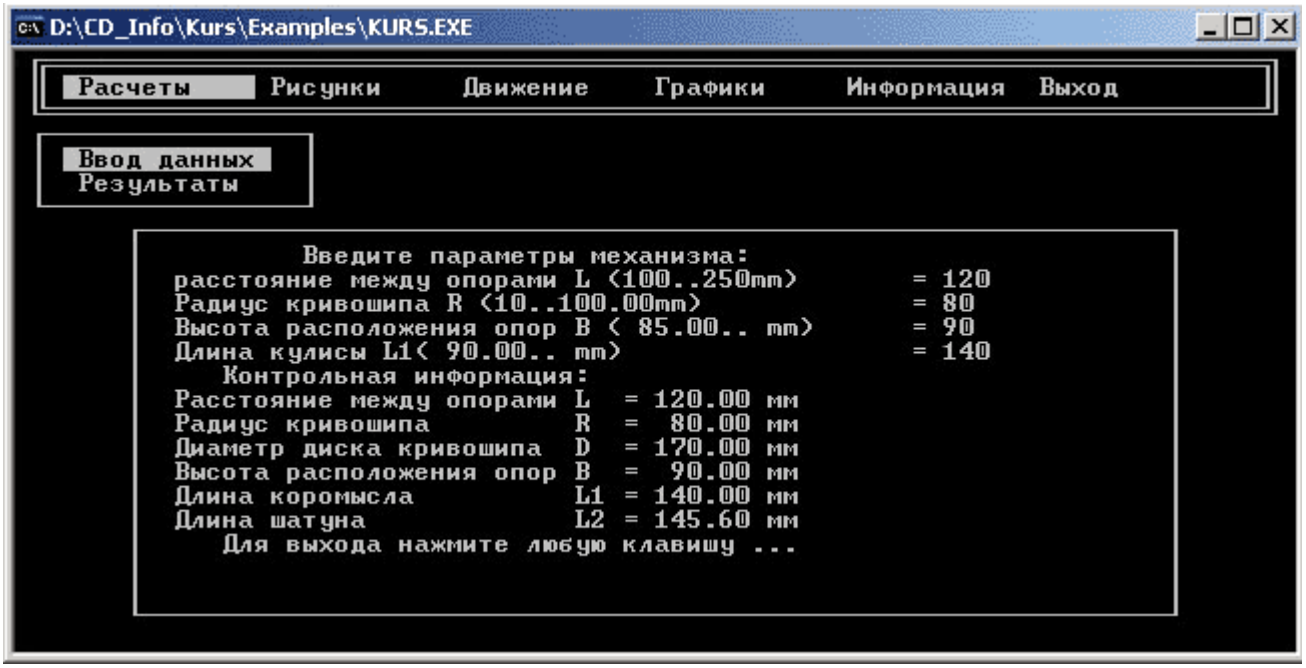
Розглянутий варіант роботи пройшов налагодження і верифікацію, які проводились як для окремих блоків так і для системи в цілому. При налагодженні аналогічної системи слід звернути увагу на такі моменти.

1. Налагодження і верифікація функцій положення механізма і передаточного відношення

Ці функції є основою програмної системи і використовуються в різних блоках. тому уникати помилок у них слід у першу чергу. При налагодженні функцій звертайте увагу на розмірності величин, які передаються у функцію і отримуються із неї. Особливу увагу приділяйте величинам, які можуть вимірюватись у градусах або радіанах. Функція, також, не повинна аварійно завершувати роботу в результаті переповнення регістра або ділення на нуль. Значення які видає функція повинні бути перевірені іншими доступними засобами, наприклад на калькуляторі, або за допомогою електронної таблиці *Microsoft Excel*. Результати обов'язково повинні співпадати.

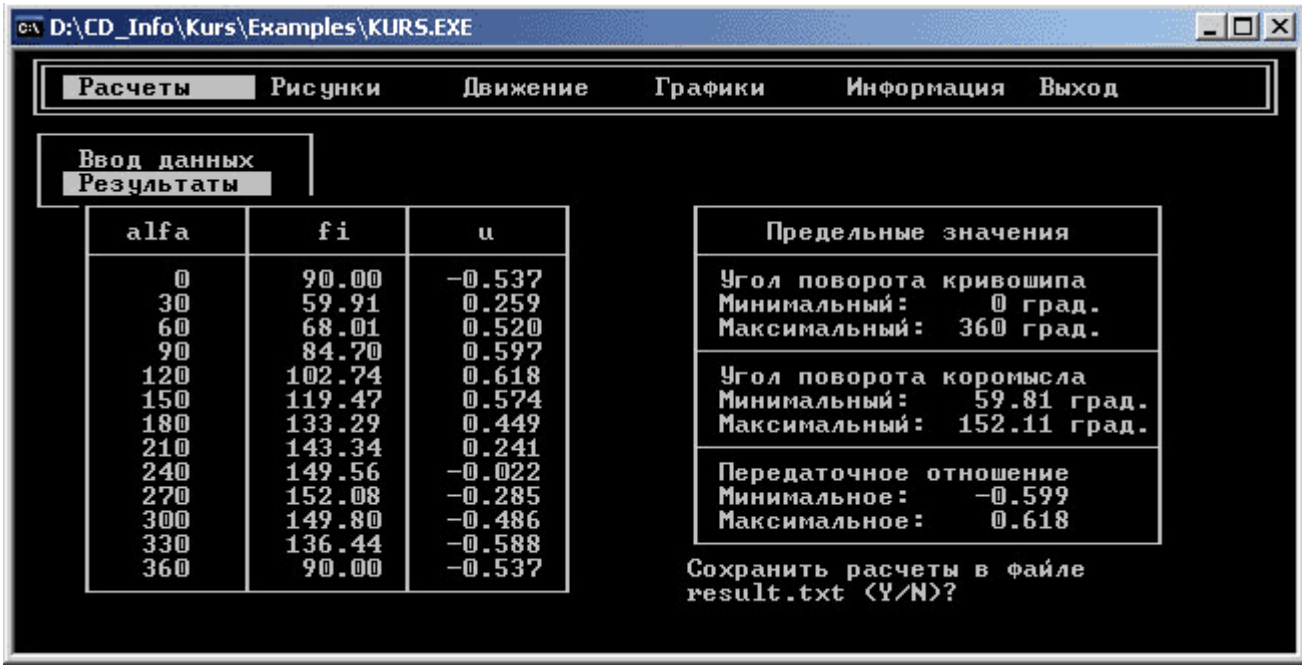
2. Налагодження і верифікація процедури введення даних

Цей процес полягає у перевірці того, яка поводить себе процедура при введення різних даних, як граничних даних діапазону, так і середніх даних. Тут слід звернути увагу на те, щоб при перерахунках на з'являлися від'ємні діапазони довжин, не видавалися занадто великі або занадто маленькі числа. При введенні некоректних даних слід проаналізувати як поводить себе процедура, чи не пропускає некоректні дані, чи вводяться вони на тому самому місці де ви планували. Після виведення контрольної інформації переконайтеся, що біля відповідних параметрів виводяться ті самі числа, які ви щойно вводили. Переконайтеся також, що значення присвоювались глобальним змінним і вони будуть доступними у інших блоках. Вікно введення вихідних даних наведено нижче.



3. Налагодження і верифікація виведення результатів

Проконтролюйте відповідність даних таблиці із розрахунками і таблиці і граничними значеннями. Якщо будуть невеликі розбіжності, треба їх проаналізувати, чи можливо це для конкретної задачі і для конкретних початкових даних. Порівняйте результати розрахунку за допомогою вашої програми і за допомогою калькулятора. Порівняйте результати, які виводяться на екран і ті, які записуються у файл `result.txt`. Порівняйте результати свого варіанту із тестовою програмою `TestResult`, яка зберігається на компакт-диску в папці `Kurs`. Вікно виведення результатів наведено нижче.

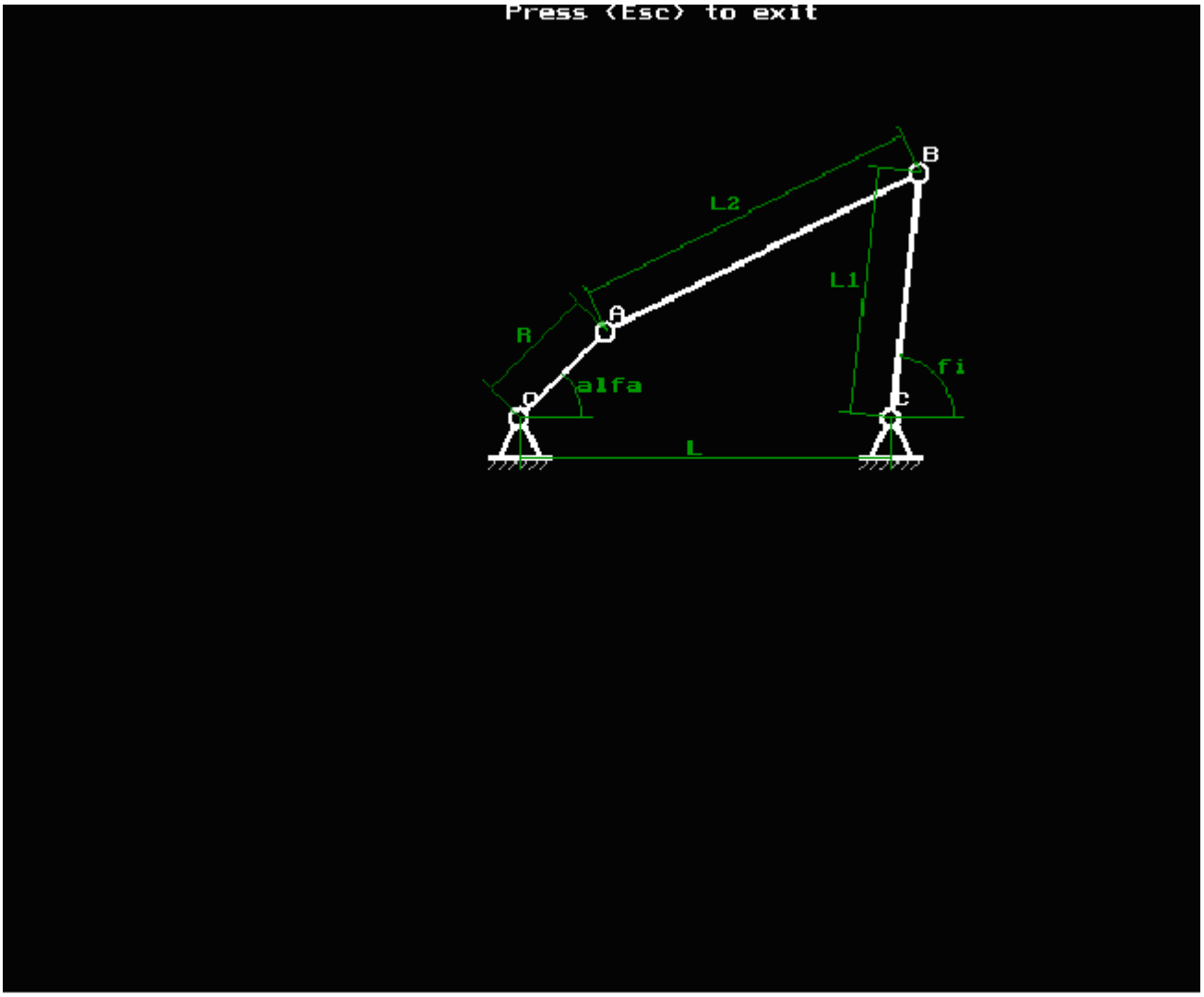


При налагодженні результатів розрахунку слід також перевірити наявність і зміст текстового файлу із результатами. Такий файл повинен містити відомості про розміри механізму, таблицю значень та граничні значення.
Приклад змісту файла "`result.txt`".

```
   Параметры механизма:
 Расстояние между опорами L  = 120.00 мм
 Радиус кривошипа          R  =  80.00 мм
 Диаметр диска кривошипа   D  = 170.00 мм
 Высота расположения опор  B  = 170.00 мм
 Длина кулисы              L1 = 150.00 мм
 Длина коромысла           L2 = 190.00 мм
   Таблица расчетных значений:
 alfa       fi        u
    0      0.00      0.53
   30     22.63     -0.90
   60     44.77      0.89
   90     65.89      0.87
  120     85.32      0.82
  150    102.31      0.71
  180    116.10      0.54
  210    126.17      0.30
  240    132.15      0.03
  270    133.27     -0.25
  300    126.56     -0.50
  330     99.15     -0.92
  360      0.00      0.53
  Предельные значения:
Угол поворота кривошипа
Минимальный:     0 град.
Максимальный:  360 град.
Угол поворота кулисы
Минимальный:     0.00 град.
Максимальный:  133.54 град.
Передаточное отношение
Минимальное:   -381.827
Максимальное:    9.708
```

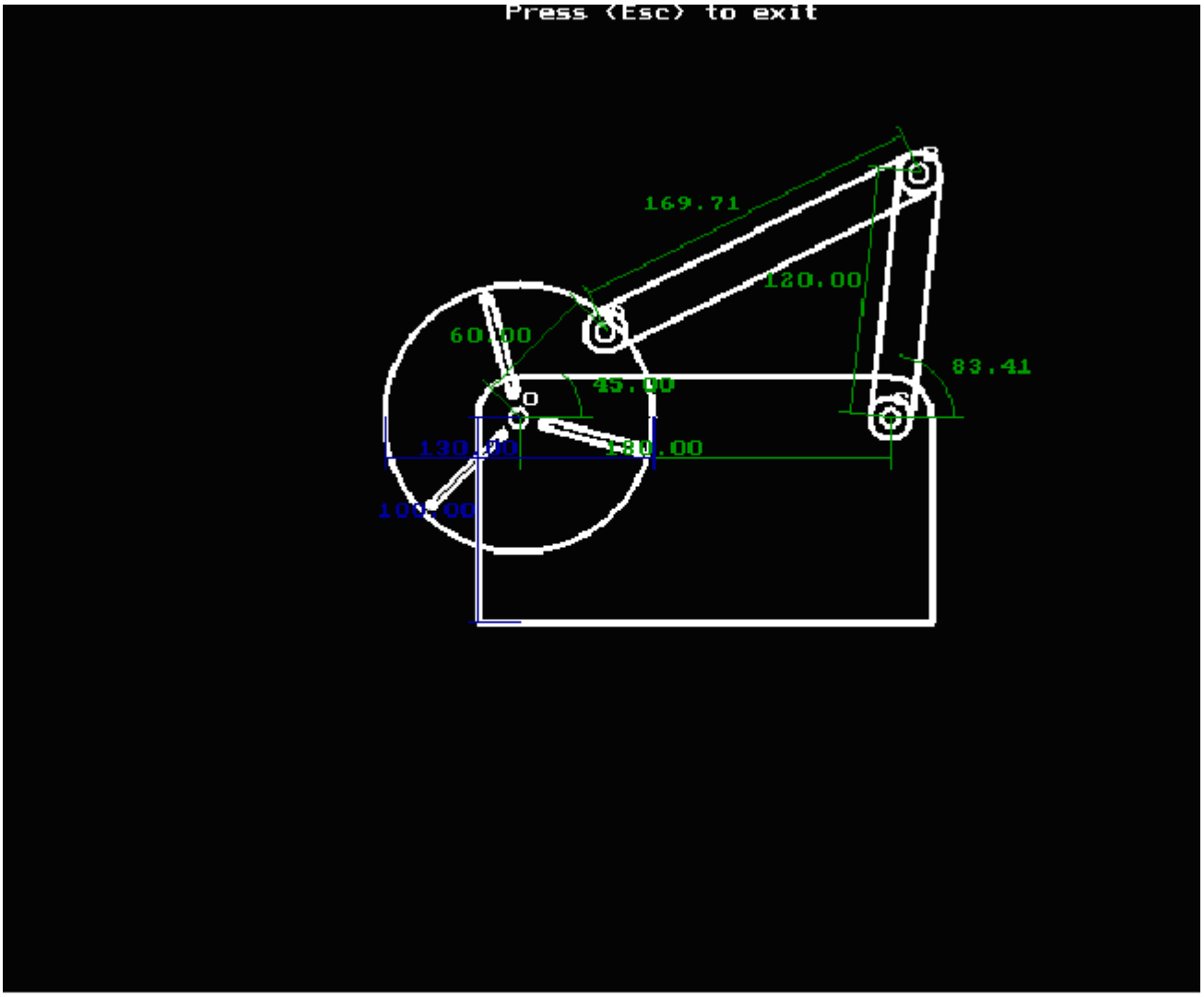
4. Налагодження процедури виведення спрощеного ескізу механізму

Цей процес полягає у візуальному контролі вигляду механізму. Звертайте увагу на те, чи змінюються розміри механізму при зміні початкових даних, чи перебудовується ескіз, якщо змінити значення кута повороту кривошипу, чи у вірному напрямку відбувається обертання кривошипу, чи пропорційними є співвідношення розмірів механізму, чи відповідають написи на розмірах початковій схемі. Вікно виведення спрощеного ескізу механізму наведено нижче.



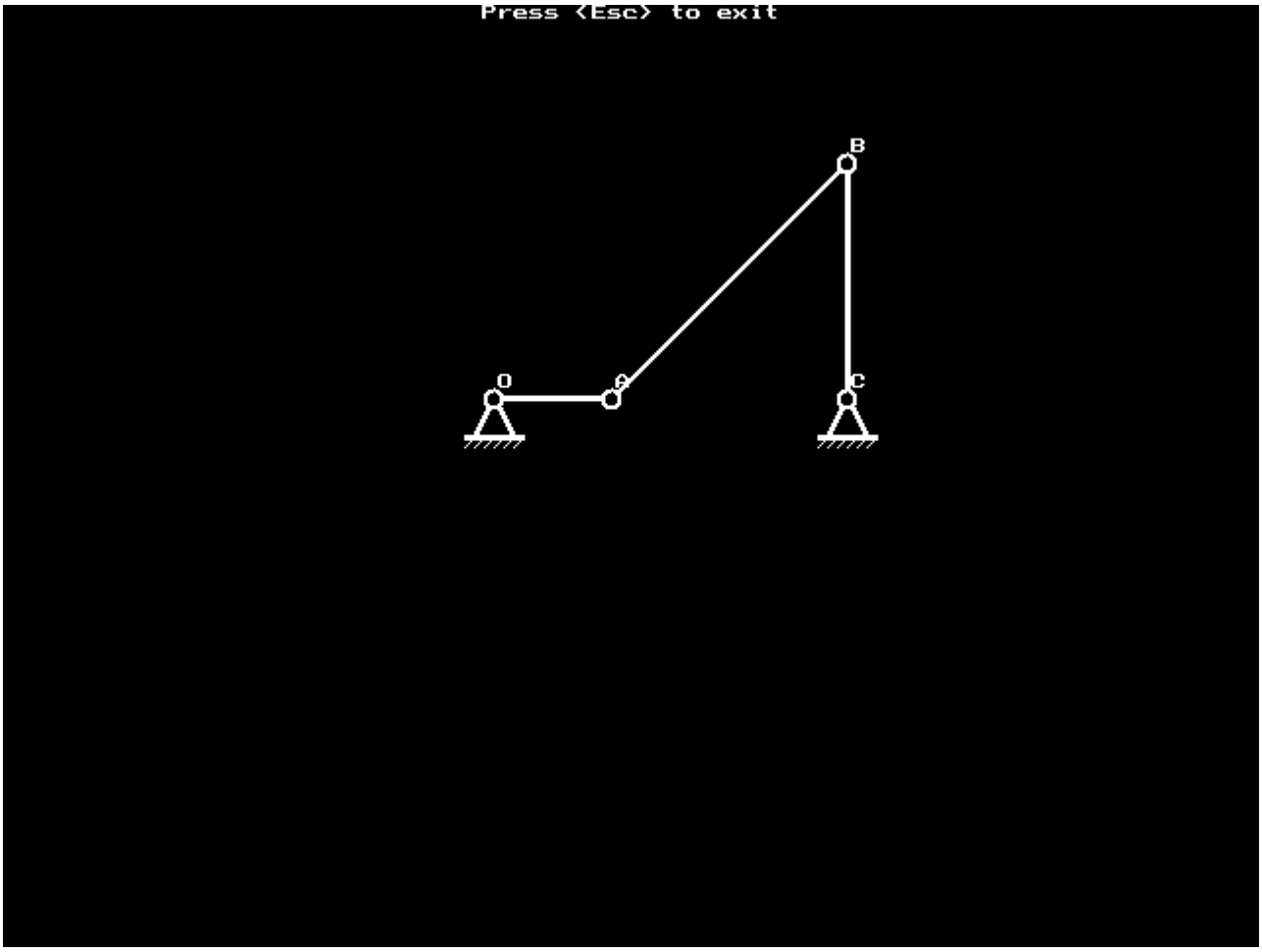
5. Налагодження повної схеми механізму

Цей процес схожий на налагодження попередньої процедури. Звертайте також увагу на те, як розташовується ескіз механізму на екрані, і особливо на те, чи коректно він перебудовується після зміни параметрів. Вікно виведення повної схеми механізму наведено нижче.



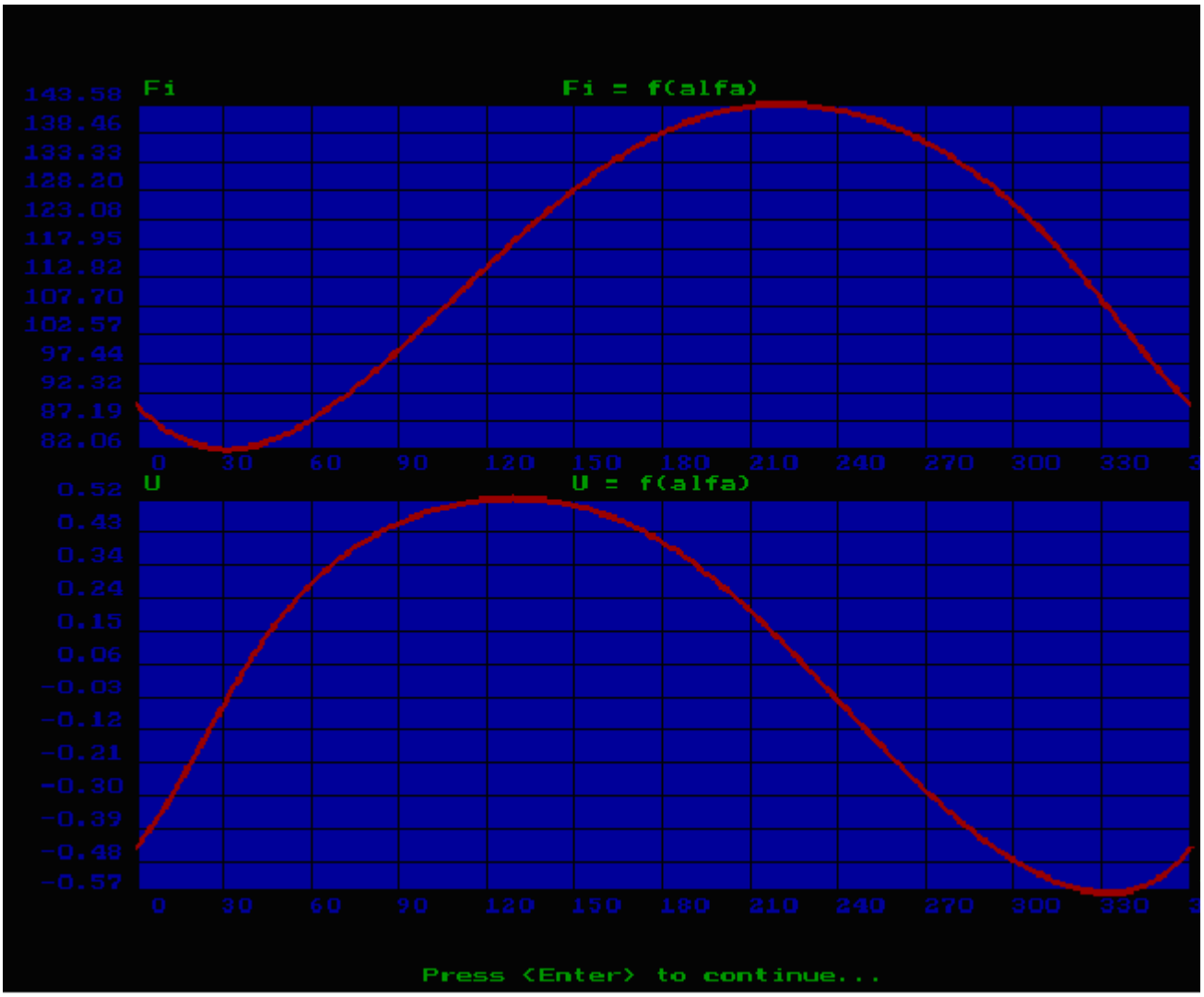
6. Налагодження процедури імітації роботи механізму

Окрім того, що контролювалося для ескізів, слід звернути увагу на саму імітацію. Перевірте, чи у вірному напрямку обертається кривошип, чи не "розтягуються" ланцюги механізму під час руху, чи можна своєчасно завершити перегляд імітації роботи механізму тощо. Імітація роботи механізму наведена нижче.



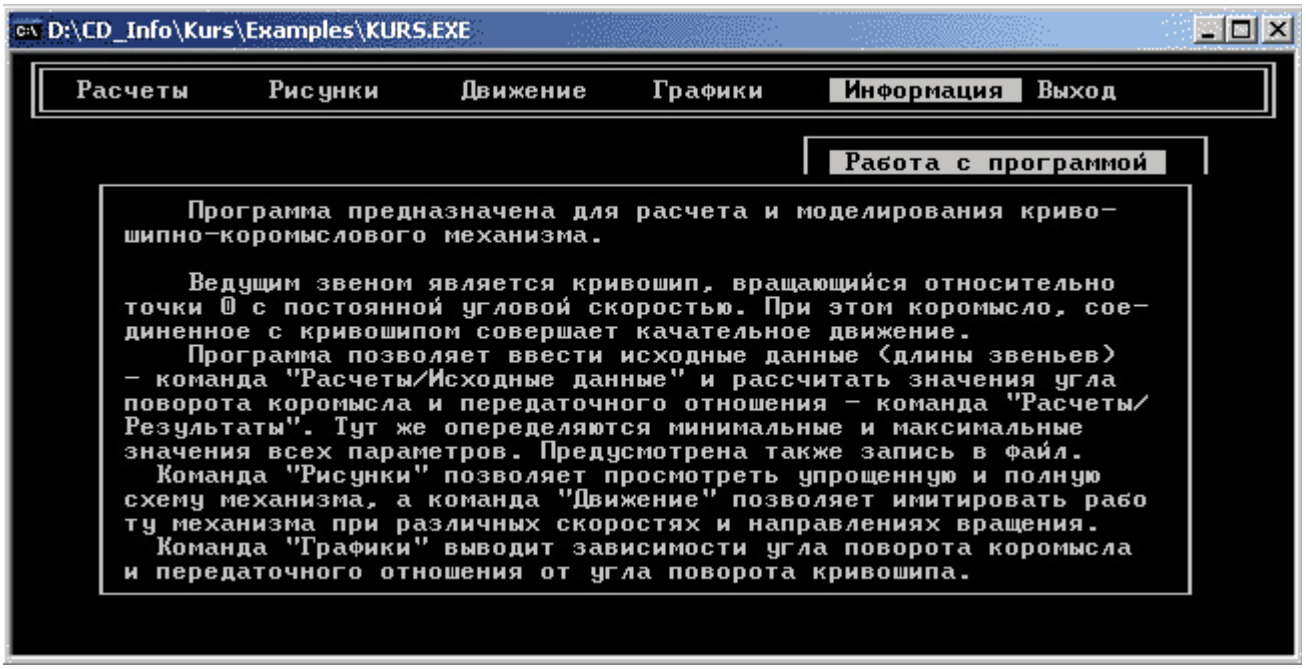
7. Налагодження і верифікація процедури виведення графіків

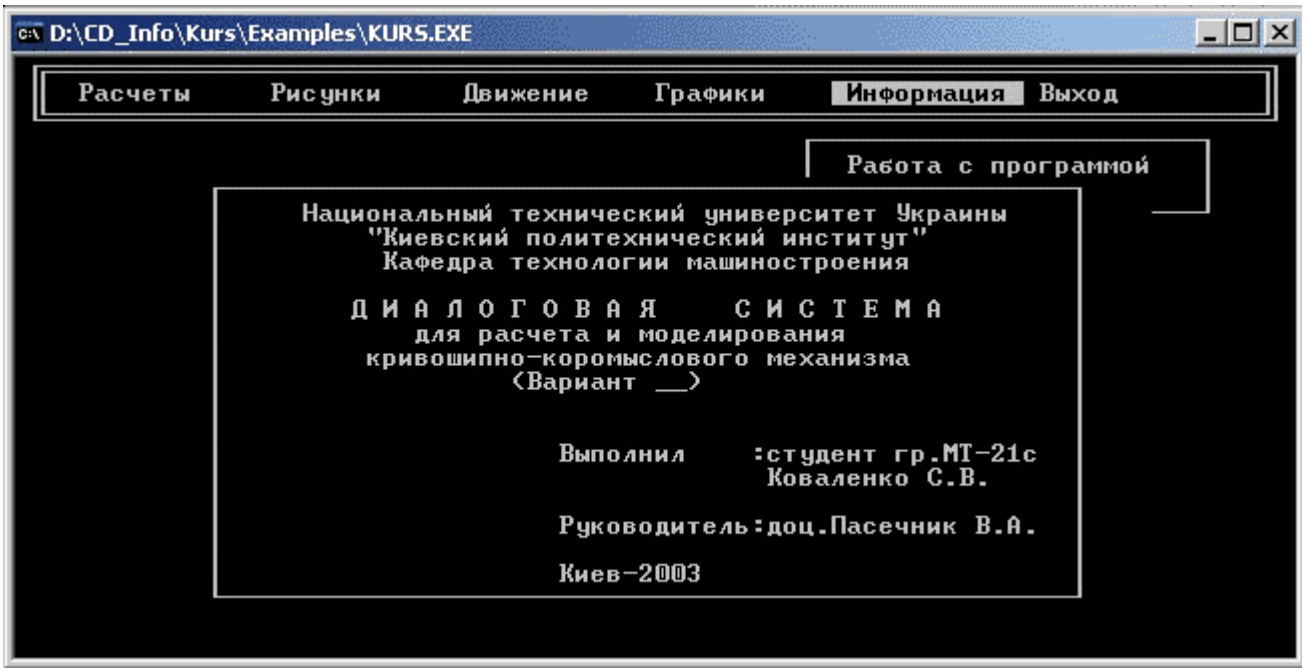
Цей процес полягає у порівнянні того, що видається у процедурі виведення результатів і відображається на відповідному графіку. Перевірте одиниці представлення інформації, переконайтеся. що не відбулося віддзеркалювання графіка, тощо. Перевірити форму графіків можна за допомогою тестової програми `TestGraphics`, яка зберігається на компакт-диску в папці `Kurs`. Вікно виведення графіків наведено нижче.



8. Налагодження процедур виведення інформації про програму і про автора

Тут потрібно лише переконатися, що всі інформація вміщується на екран. У процедурі виведення інформації про програму записуйте тільки те, що виконує ваша програма. У процедурі виведення інформації про автора не забудьте перевірити, чи є ваше прізвище серед авторів програми ;-). Вікна виведення інформації про програму і про автора наведені нижче.





9. Налагодження програми в цілому полягає у перевірці взаємодії окремих блоків програми. Тут основну увагу слід звернути на передачу основних параметрів. Усі дані, які були введені користувачем у процедурі `InputDatas`, повинні бути доступними у інших процедурах - `Result`, `Chema`, `Mechanizm`, `Graphics`. Програма повинна стабільно працювати при різних вихідних даних і виконувати ті дії, на які розраховував користувач.

Отримання результату і його інтерпретація

Отриманий результат є комп'ютерною моделлю певного простого механізму. Математична модель, що закладена в основу програми, є суттєво спрощеною та ідеалізованою. У ній, наприклад не враховуються похибки механізму, геометричні характеристики перерізу ланок і фізичні характеристики матеріалу з якого вони зроблені, не враховується тертя, опір тощо. В нашому випадку порівнювати результати роботи системи із результатами натурних випробувань реального механізму не має сенсу і не має можливості, адже виконана робота є

учбовою.

В той же час, програма дозволяє вивчати особливості даного механізму в межах діапазонів параметрів. Це може дозволити досліджувати механізм і робити певні висновки про нього (наприклад, наявних "мертвих" точок, точок екстремального навантаження тощо). Розроблена програма має також за мету ознайомити із принципом дії певного механізму.

Система може удосконалюватись. Для цього необхідно вводити нові функції і процедури. В межах завдання роботу можна вважати виконаною.

Передача замовнику результату роботи

Закінчену і налагоджену програму слід передати замовнику. Замовником учбової курсової роботи є викладач. У такому випадку слід підготувати звіт, дотримуючись вимог.

Окрім звіту передається файл курсової роботи. Усі файли, які необхідні для коректної роботи програми слід записати в окрему папку. Ця папка повинна мати ім'я, що складається із коду групи і прізвища автора (латинськими літерами), наприклад `MT-22c_Kovalenko`. У цій папці повинні знаходитись вихідні тексти і відкомпільовані файли головної програми і власного модуля. Додатково у папці повинен знаходитись драйвер графічного режиму `egavga.bgi` та файл драйвер кирилиці `keyrus.com`.

Під час захисту курсової роботи необхідно продемонструвати знання, відповідно до вимог.

Супровід системи

Для учбової системи, якою є курсова робота з інформатики, супровід не виконується.