МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

Факультет післядипломної освіти

“Конструювання програм і мови програмування”

Програма, методичні вказівки до лабораторних робіт, курсового проектування та самостійної роботи

слухачів спеціальності 8.080403

“Програмне забезпечення автоматизованих систем”

2002

Конструювання програм і мови програмування. Програма, методичні вказівки до лабораторних робіт, курсового проектування та самостійної роботи слухачів спеціальності 8.080403 “Програмне забезпечення автоматизованих систем” /Укл.: С.К. Корнієнко, С.Г. Мілінчук, М.В. Калініна. – Запоріжжя: ЗНТУ, 2002. - 66 с.

Укладачі:

С.К. Корнієнко, доцент, к.т.н
С.Г. Мілінчук, асистент
М.В. Калініна, асистент

Рецензент: Е.І. Ткачьов, доцент, к.т.н

Відповідальний за випуск: С.К. Корнієнко

Затверджено
на засіданні кафедри КВР

Протокол № 1 від 17.09.2001

Методичні вказівки, призначені для використання на лабораторних заняттях із дисципліни "Конструювання програм і мови програмування", виконання курсової роботи, а також у процесі самостійної роботи слухачів факультету післядипломної освіти.

Методичні вказівки окрім указівок із виконання лабораторних робіт містять програму дисципліни, основні теоретичні відомості з мови Borland Pascal, вказівки до виконання курсового проекту.

ЗМІСТ

1 ПРОГРАМА КУРСУ

1.1 Введення в програмування

Коротка історична довідка.

Етапи підготовки задачі для отримання розв'язку за допомогою ЕОМ. Алгоритм і його властивості. Засоби опису алгоритмів: словесні, графічні. Схем алгоритмів. Правила оформлення схем у відповідності до вимог стандартів ЄСПД. Мови програмування та їх класифікація. Стандарти мов. Системи програмування та їх оболонки. Тенденції розвитку мов програмування.

1.2 Середовище Borland-Pascal

Можливості середовища Borland-Pascal: робота з файлами; робота з вікнами; редагування текстових файлів; компіляція, компонування, налагодити програм та їх виконання; установка опцій програм оболонки.

1.3 Основні поняття Borland-Pascal

1.3.1 Елементи мови: алфавіт, лексеми, конструкції. Ідентифікатор. Терми: константи, змінні, структури. Прості типи даних. Оператори. Властивості операторів. Операторні дужки. Оператор присвоєння. Арифметичні операції, операції порівняння; елементарні функції; вирази.

Елементарний ввід-вивід. Структура програми.

1.3.2 Внутрішнє представлення даних різних типів. Позиційні системи числення. Алгоритми перетворення чисел з однієї системи числення в іншу.

1.3.3 Формати цілих і реальних чисел. Формати співпроцесора із плаваючою точкою. Емуляція форматів.

1.3.4 Внутрішнє представлення символів. Коди ASCII. Керуючі символи і символи, які мають відображення на екран. Відповідність клавіш клавіатури і кодів ASCII. Scan-коди.

1.3.5 Алгебра висловлювань. Висловлювання. Логічні операції над висловлюваннями.

1.3.6 Внутрішнє представлення булевих даних. Логічні операції над булевими даними, над цілими числами .

1.3.7 Класифікація даних на Паскалі. Прості типи даних на Паскалі. Порядкові типи: цілі, булеві, символьні, перечислюванні, тип-діапазон. Реальні типи. Стандартні функції для роботи з простими типами даних.

1.4 Керуючі структури

Стиль написання програм на мові Borland-Pascal. Структурне програмування. Різновид структур алгоритмів. Програмування лінійних алгоритмів. Програмування розгалужених алгоритмів. Умовний оператор. Оператор варіанта. Оператор безумовного переходу. Поняття циклу. Типи алгоритмів циклічної структури. Цикл із передумовою. Цикл із післяумовою. Цикл із параметром. Вкладені цикли. Програмування ітераційних процесів.

1.5 Структури

1.5.1 Структурні типи. Масиви. Багатовимірні масиви. Ввід і вивід масивів. Запаковані масиви. Матриці. Перетворення і побудова матриць.

1.5.2 Рядки, операції над рядками. Стандартні процедури і функції для роботи з рядками.

1.5.3 Записи. Записи з варіантами. Ввід-вивід записів. Множина. Операції над множинами.

1.5.4 Сумісність і перетворення типів .

1.5.5 Типізовані константи. Типізовані константи-масиви, константи-записи, константи-множини.

1.6 Процедури і функції

Процедури і функції. Опис процедур і функцій. Локальні і глобальні змінні. Виклик процедур і функцій. Формальні і фактичні параметри, засоби завдання параметрів. Параметри-масиви, параметри-рядки. Процедурні типи: параметри-функції, параметри-процедури. Рекурсія. Граничні і рекурсивні умови. Види рекурсії: низхідна, висхідна. Рекурсія і ітерація.

1.7 Робота з динамічною пам’яттю

1.7.1 Динамічна пам`ять. Об`ява покажчиків. Виділення та звільнення динамічної пам`яті. Використання покажчиків.

1.7.2 Стандартні процедури і функції для роботи з динамічною пам’яттю.

1.7.3 Динамічні структури. Зв'язні списки: стек, черга, двозв'язані кільця.

1.7.4 Рекурсивна обробка списків. Дерева. Двійкові дерева. Робота з деревами.

1.8 Модульне програмування

Структура модуля: заголовок, інтерфейс, розділ реалізації, розділ ініціалізації. Використовування модулів. Циклічні посилання модулів. Стандартні модулі: CRT, PRINTER, SYSTEM, DOS, OVERLAY, GRAPH. Створення власних модулів.

1.9 Файли

1.9.1 Розміщення інформації у файлах: записи фіксованої довжини, записи змінної довжини, записи невизначеної довжини. Методи доступу до файлів: послідовний, прямий.

1.9.2 Поняття файлової змінної, її призначення. Типи файлів у Паскалі: типізовані, текстові, нетипізовані. Стандартні процедури та функції спільного призначення: assign, reset, rewrite, close, eof, erase, rename, ioresult, read, realn, write, writeln.

1.9.3 Стандартні процедури та функції для роботи з текстовими файлами: eoln, eof, append, readln, writeln. Стандартні процедури та функції для роботи з типізованими файлами: seek, filesize, filepos, truncate, read, write.

1.9.4 Робота з файлами прямого доступу: корегування, доповнення та вилучення компонент. Форматування файлів. Нетипізовані файли. Процедури blockread, blockwrite.

1.9.5 Робота з клавіатурою. Модуль CRT. Функції keypressed, readkey.

1.10 Графіка

1.10.1 Адаптери, види адаптерів. Керування екраном в графічному режимі. Модуль Graph. Ініціація графічного режиму.

1.10.2 Процедури та функції.

1.11 Сучасні методи проектування програм

1.10.1 Розробка проекту програмування із використовуванням структурного підходу. Передумови виникнення структурного підходу. Постановка задачі. Вимоги та специфікації. Низхідна розробка проекту задачі.

1.10.2 Модульне програмування. Покрокова деталізація. Блок-схеми. Сегментування модулів. Оверлейні структури.

1.10.3 Реалізація програм: кодування, налагодження модуля, комплексне налагодження. Тестування модулів, комплексне тестування. Якість програм: ефективність, економія пам'яті. Документування готового програмного продукту.

1.10.4 Основні принципи побудови програми об’єктно-орієнтовним методом. Об’єкти, їх властивості, поведінка. Інкапсуляція. Класифікація об’єктів. Механізм успадкування властивостей та поведінки. Поліморфізм.

2 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Лінійний обчислювальний процес

Найпростіша програма на Паскалі виводить на екран вітання: «Hello» . Текст цієї програми складається усього з трьох рядків

```
begin
  WriteLn(‘Hello!’);
end.
```

Текст програми містить усього один оператор, що виконується, WriteLn ('Hello!'). Оператор розташований між словами begin і end, що відносяться до зарезервованих слів мови. Значення зарезервованих слів у мові програмування строго фіксовано, і вони можуть застосовуватися тільки по прямому призначенню. У середовищі програмування слова автоматично виділяються кольором. У даному випадку зарезервованими словами begin (початок) і end (кінець) обмежується тіло програми. Слово end із крапкою на кінці – ознака кінця програми, записи після неї не сприймаються компілятором. Тіло програми – послідовність операторів, що виконуються. У нашім прикладі оператор, що виконується, поки один: WriteLn('Hello!'). Цей оператор викликає стандартну процедуру виводу приведеного в апострофах тексту на екран (самі апострофи на екран не виводяться).

Оператори розділяються крапкою з комою. Варто звернути увагу, що після слова begin і перед словом end не ставиться крапка з комою, тому що ці слова не є виконуваними.

Розглянемо програму яка б розраховувала площу трикутника з основою a та висотою H.

```
Program S_treugolnika;
  Var  h,a,s:real;
       begin
      Writeln('Введіть основу трикутника A:');
      Read(a);
      Writeln('Введіть висоту трикутника H:');
      Read(h);
      S:=1/2*a*h;
      Writeln('Площа дорівнює:',s:3:2);
      end.
```

Результат виконання програми:
Введіть основу трикутника A:
6
Введіть висоту трикутника H:
3
Площа дорівнює:
9

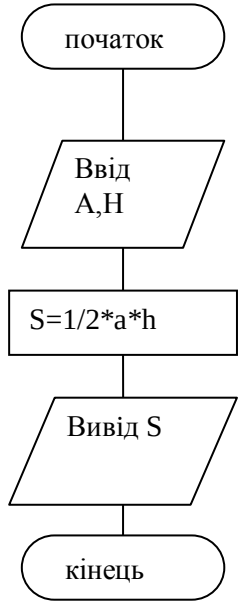


Рисунок 2.1 – Алгоритм програми

Перший рядок – заголовок програми. За правилами Паскаля (у реалізації фірми Borland) заголовок може бути відсутнім. Заголовок складається з зарезервованого слова Program і імені програми S_treugolnika. Ім'я програми – це приклад імені, визначеного користувачем. Для імен у Паскалі діють наступні правила:

- Імена можуть складатися з букв латинського алфавіту, цифр і символу _;
- Першим символом в імені не повинна бути цифра;

- Пробіли в імені повинні бути відсутніми;
- Прописні і малі літери сприймаються як синоніми.

Заголовок закінчується крапкою з комою.

У програмі є три змінні (h, a, s) величини, тобто величини які не можуть бути визначені заздалегідь. Усі константи і змінні, котрі використовуються в програмі, повинні бути описані до початку тіла програми. Опис змінної включає ім'я змінної і її тип. У загальному вигляді розділ опису змінних виглядає так:

Var <ім'я змінної>,...,< ім'я змінної > : <ім'я типу>;

Узагалі, будь-яка програма на Паскалі поділяється на описову частину і тіло програми.

2.2 Обчислювальний процес, що розгалужується

Розгалуження програм може бути реалізовано різними методами. Один з них - застосування оператора if (якщо). Оператор if у загальному випадку має наступну форму

if <умова> then <оператор1> else <оператор2>;

<Умова> у приведеному операторі – це логічний вираз. Якщо умова правдива, тобто логічний вираз має значення true, те виконується <оператор1>, у противному випадку, коли логічна умова має значення false , виконується <оператор2>.

Наприклад, потрібно розробити програму, яка обчислює значення функції:

$$Z=\begin{cases} \text{Sin X}, & x<=a; \\ \text{Cos X}, & a<x<b; \\ \text{Tg X}, & x>=b. \end{cases}$$

```
Program trig_fun;
Uses Crt;
  Var
    x, a, b: integer; z :real;
BEGIN ClrScr;
  TextColor(7);
```

```
  Writeln('A:'); Read(a);
  Writeln('B:'); Read(b);
  Writeln('X:'); Read(x);
 if (x<=a) then Begin
                  z:=sin(x);Writeln('SIN=',z:3:2);
                End;
 if (x>a) and (x<b) then Begin
                           z:=cos(x); Writeln('COS=',z:3:2);
                         End;
if (x>=B) then Begin
                  z:=sin(x)/cos(x); Writeln('TG=',z:3:2);
                End;
 ReadKey;
END.
```

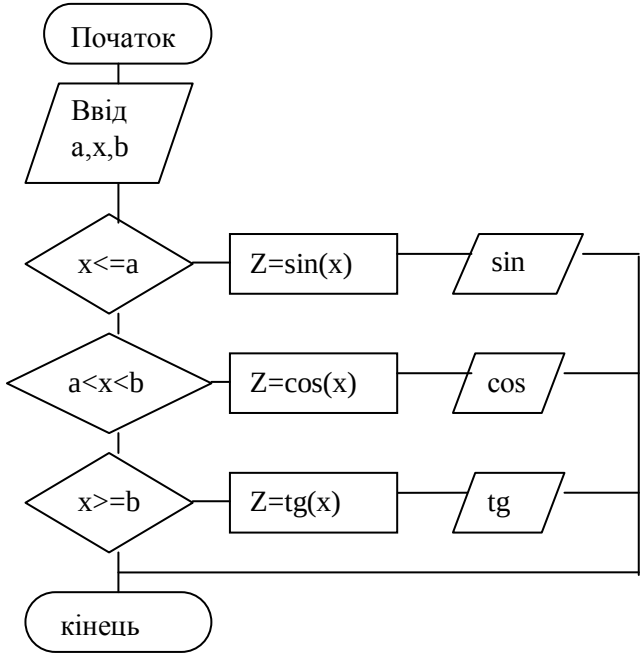


Рисунок 2.2 – Алгоритм програми

Другий приклад: програма, яка по x, y визначає, у якому квадраті знаходиться крапка.

```
Program koor;
   Uses Crt;
   Var
   Label 1;
   x,y,n:integer;
   Begin
      ClrScr;
     TextColor(7);
     Wrietln('Введіть X:'); Read(x);
Writeln('Введіть Y:'); Read(y);
   if (x<0) then Begin
                    if (y<0) then n:=3 else n:=2;
                    goto 1;
                  End;
   else Begin
          if (y<0) then n:=4 else n:=2;
         goto 1;
       End;
  1: TextColor(11);
     Writeln('Квадрант має номер N =',n); ReadKey;
   End.
```

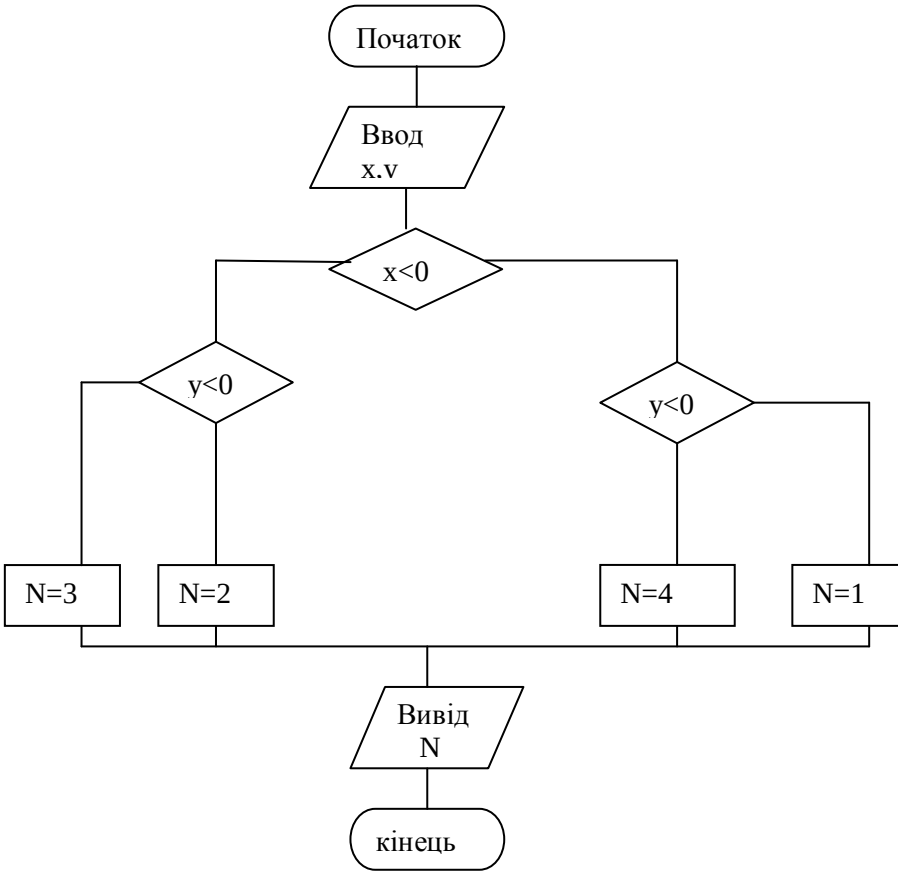


Рисунок 2.3 – Алгоритм програми

Також у Паскалі існує оператор множинного розгалуження (вибір з декількох гілок команд) case. Загальний вид цього оператора:

Case <селектор> of
<альтернатива 1> : <оператор 1>;
< альтернатива 2 > : < оператор 2>;
…....
< альтернатива N> : < оператор N>
else < оператор частини Else>
end;

Case і of – зарезервовані слова, що є відмітною ознакою оператора множинного розгалуження. Селектор – це перемінна або вираз порядкового типу. У залежності від значення даного виразу або змінної відбувається розгалуження програми.

2.3 Цикли. Обробка матриць

Багаторазові повторення однакових дій можна виконати за допомогою конструкцій, що у програмуванні називаються циклами. Повторювані дії називаються тілом циклу. У Паскалі існує кілька операторів циклу. Оператор for повторює тіло циклу задане число раз. Він має наступні синтаксичні форми:

for <лічильник циклу> := <перше значення> to
  <останнє значення> do <тіло циклу>;
for < лічильник циклу > := < перше значення > downto
  < останнє значення > do < тіло циклу >;

Лічильник циклу – це змінна одного з порядкових типів, що зберігає число повторень операторів циклу. Значення лічильника циклу змінюється автоматично від першого до останнього і збільшується на одиницю від першого до останнього значення і збільшується на одиницю для першої форми запису (з оператором to) або зменшується на одиницю для другої форми запису (з оператором downto).

Як приклад розглянемо програму, яка розраховує 20-й член послідовності An=An-1+0.001; A1=1.

```
Program posl;
   Uses Crt;
   Var
      n:integer;
      a:extended;
   Begin
    ClrScr;
    a:=1;
    For n:=2 to 20 do Begin
```

```
                    a:=a*a+0.001;
                End;
  Writeln('a=',a);
  ReadKey;
End.
```

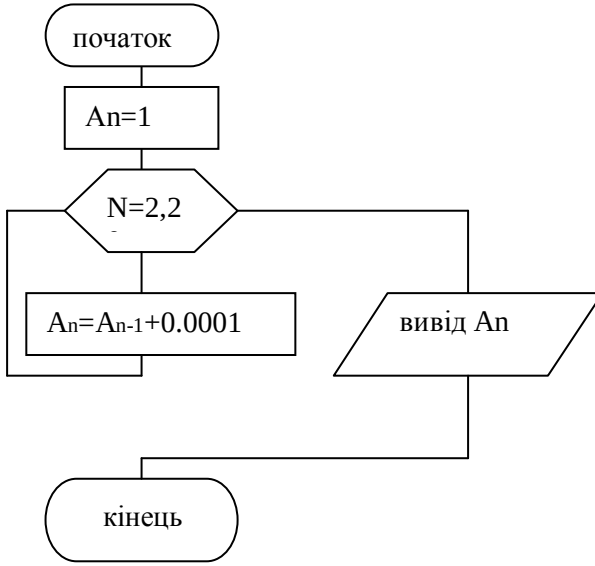


Рисунок 2.4 – Алгоритм програми

Масив - це структурований тип для резервування в пам'яті місця під велику кількість однорідних елементів. Структура масиву може бути лінійної, табличної або, у загальному випадку, n-мірної. Одне обмеження – загальний розмір цієї структури не повинен перевищувати 65 520 байт. Оперувати можна як з окремими компонентами масиву, так і з масивом у цілому. Варто чітко розрізняти значення елемента масиву й індекси елемента, тобто його n-мірний номер у структурі. Простий приклад:

Var y:array[1..10] of integer;

Array – зарезервоване слово Паскалю, що перемінна є масивом. У дужках приведений діапазон зміни індексів масиву.

Наступний приклад: існує масив X(10). Визначити найбільший елемент масиву X(10) та його порядковий номер.

```
Program Mass;
  Var    i, nmax, xmax: integer;
  x: array[1..10] of integer;
  Begin
  For i:=1 to 10 Do Read(x[i]);
  xmax:=x[1]; nmax:=1;
  For i:=2 to 10 Do
              If x[i]>xmax Then Begin
     xmax:=x[i];
     nmax:=i;
    End;
  Writeln('xmax=', xmax,'nmax=',nmax);
  End.
```

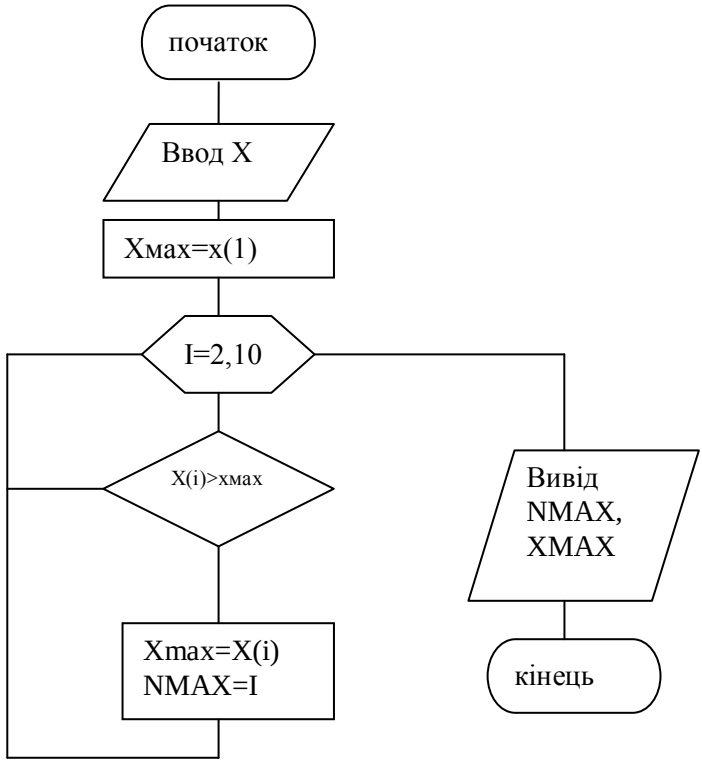


Рисунок 2.5 – Алгоритм програми

Матриця - приклад одномірного масиву з лінійною організацією, однак масиви можуть мати будь-яке кінцеве число вимірів. У наступному прикладі приведене використання матриць.

Підрахувати суму позитивних елементів кожного рядка матриці A(5,5).

```
Program Matr;
  Uses Crt;
  Var
    i, s, j:integer;
    a: array[1..5,1..5] of integer;
Begin
```

```
ClrScr; Writeln(':');
  For i:=1 to 5 Do
  For j:=1 to 5 Do Read(a[i,j]);
  For i:=1 to 5 Do
  Begin
      For j:= 1 to 5 Do
        Writeln(a[i,j]:3,' ');
        Writeln;
  End;
  For i:=1 to 5 Do
  Begin
      s:=0;
      For j:=1 to5 do
  If a[i,j]>0 Then s:=s+a[i,j];
      Writeln('S=',s);
  End;
  ReadKey;
End.
```

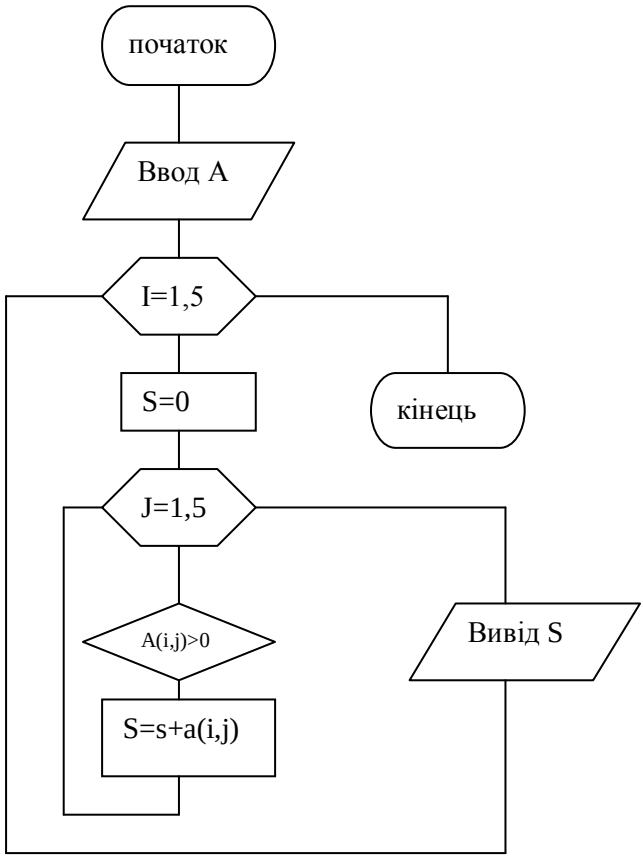


Рисунок 2.6 – Алгоритм програми

Оператор for, що був розглянутий вище, забезпечує виконання циклу задана кількість разів, однак часто циклічні дії закінчуються за умовою, тобто виконуються до досягнення певного результату. У Паскалі є два таких оператори циклу, що відрізняються тим, що в одному з них умова перевіряється на початку циклу (while...do), а в іншому - наприкінці циклу (repeat...until).

Оператор циклу while (поки, у той час як) має вигляд:

While <логічний вираз> do <тіло циклу>;

Цикл while забезпечує виконання тіла циклу, наступного за словом do

доти, поки умова має значення true (істина). В якості тіла циклу може використовуватися простий або складний оператор. Умова перевіряється перед початком кожного виконання тіла циклу, тому, якщо до першого виконання циклу умова має значення false (неправда), оператор не виконається жодного разу.

Наведемо приклад з використанням циклу while. Нехай необхідно знайти $\sum_{i=1}^{n} 1/i^2$, обчислення зробити з точністю до E.

```
var
 e,s:real;
  i:integer;
begin
writeln('Введіть E');
   readln(e);
   i:=1;
   s:=1;
 while 1/sqr(i)>e do
   begin
     i:=i+1;
    s:=s+1/(i*i);
 end;
  writeln('Сума=',s:4:4 ,'  ','номер= ',i);
  readln;
 end.
```

Результат роботи програми:

Введіть E
0.0001
Сума=1.6351 номер= 101

Оператор циклу repeat...until...(повторювати доти, поки) має вигляд:

Repeat <тіло циклу> until <логічний вираз>;

Принципова відмінність оператора repeat...until від оператора while...do у тому, що перевірка умови провадиться не перед початком виконання оператора, а в його кінці, коли вирішується питання, повторювати ще раз дію.

Як приклад використання циклу repeat...until можна навести приклад, що реалізує метод підбора цілочислового рішення Діофантова рівняння 5x-3y=1. Як першу пробу беруться одиничні значення. Далі, якщо 5x-3y>1, то у збільшується на 1, якщо 5x-3y<1, то на 1 збільшується .

```
program diophant;
var x,y : byte;
begin
  x;=1; y:=1;
  repeat
     if (5*x-3*y)>1 then y:=y+1;
     if (5*x-3*y)<1 then x:=x+1
  until (5*x – 3*y) = 1;
  writeln(‘x= ’,x,’ y= ’,y);
end.
```

Результат роботи програми:
x=2 y=3

2.4 Обробка послідовності символів

Символьний тип даних – це тип для проведення операцій із символами – літерами, цифрами і різними спеціальними значками. Ідентифікатор типу – char. Перемінна типу char займає в пам'яті 1 байт, що дозволяє закодувати 256 символів.

Наприклад, символьну змінну можна оголосити так:

Var first_char : char;

Із символьними перемінними можна робити операції присвоєння, логічні операції перевірки на рівність і нерівність. Порівняння символів по величині провадиться по номеру у таблиці ASCII.

Строковий тип даних дозволяє представити послідовності символів максимальною довжиною 255. Якщо рядок значно коротше, має сенс для економії пам'яті вказати довжину рядка в квадратних дужках при оголошенні типу. Наприклад, у DOS довжина імені файлу без розширення і шляху не повинна перевищувати 8 символів, тому

для таких рядків може бути оголошена перемінна:

```
Var fileName : string[8]  ;
```

Наступний простий приклад показує принцип використання типу string:

```
Uses CRT;
Var Name : string;
BEGIN
 ClrScr;
 Write('Як Вас звуть? ');
 ReadLn(Name);
 WriteLn('Довжина вашого імені ', Length(Name), 'символів');
 WriteLn('Ваше ім’я починається з символу '", Name[1]);
 ReadKey;
END.
```

Результат роботи програми:

Як Вас звуть? Валерій
Довжина вашого імені 7 символів
Ваше ім’я починається з символу "В"

Крім максимальної довжини рядка, визначеної в оголошенні типу або змінної, при операціях з рядками використовується поняття динамічної або поточної довжини рядка. Динамічна довжина рядка показує число реально використовуваних символів рядка і не може перевищувати максимальну довжину, що відповідає оголошенню і відведену в пам'яті.

Наступний приклад показує як підрахувати кількість букв 'a' в останньому слові введеного користувачем реченні:

```
var
   S : string;
   i,N,j:byte;
 begin
  WriteLn('Введіть рядок ');
  ReadLn(S);
  for i:=Length(S) downto 1 do
   begin
     if (S[i]<>' ') and (S[i]='a') then N:=N+1;
       if S[i]=' ' then break;
```

```
    end;
  if N=0 then WriteLn('В останньому слові немає літер "a" ') else
  WriteLn('В останньому слові ',N,' букв "a"');
end.
```

З рядками можна робити наступні дії : об'єднання, присвоєння і порівняння. Основні функції по роботі з рядками приведені в додатку Д.

2.5 Процедури і функції

Паскаль дозволяє розбити програму, відокремивши її функціонально незалежні частини. Окремі, функціонально незалежні частини програми називають підпрограмами (процедурами і функціями). Сам термін підпрограма говорить про те, що вони подібні і підлеглі програмам. Підпрограми вирішують три важливі задачі, що значно полегшують програмування :

- Рятують від необхідності багаторазово повторювати в тексті програми аналогічні фрагменти;
- Поліпшують структуру програми, полегшуючи її розуміння при розборі;
- Підвищують стійкість до помилок програмування і непередбачених наслідків при модифікації.

Заголовок процедури і функції – перший рядок підпрограми – мають вигляд:

Procedure <ім'я процедури> (<список параметрів>);

Function <ім'я функції>(<список параметрів>):тип результату ;

Якщо змінна або константа описана усередині підпрограми, вона називається локальної і доступна тільки усередині даної підпрограми. змінні із співпадаючими іменами, що можуть бути описані в основній програмі або інших підпрограмах, не мають з локальними перемінними нічого спільного. Якщо змінна описана в основній програмі і не перевизначена в підпрограмі, вона може використовуватися в підпрограмі.

Список параметрів, що задається в заголовку процедур і

функцій, забезпечує зв'язок підпрограми з зухвалою програмою. Через нього в підпрограму передаються вихідні дані і повертається результат. При цьому передбачено два принципово різних механізму передачі параметрів – за значенням і по посиланню. Синтаксично ці два способи відрізняються вживанням слова var перед відповідний перемінною в заголовку підпрограми. Якщо це слово є, перемінна передається по посиланню, якщо немає - за значенням. Наступний приклад яскраво ілюструє принцип передачі аргументів по посиланню і/або за значенням:

```
var a,b : integer;
procedure first(var a1 : integer; b1 : integer);
 begin
   a1:=a1+1;
   WriteLn('A в процедурі = ',a);
   b1:=b1+1;
   WriteLn('B в процедурі = ',b);
 end;

BEGIN
  a:=5;
  WriteLn('B до процедури = ', b);
  b:=10;
  WriteLn('A до процедури = ', a);
  first(a,b);
  WriteLn('A після процедури = ', a);
  WriteLn('B після процедури = ', b);
END.
```

Результат роботи програми:

A до процедури = 5
B до процедури = 10
A в процедурі = 6
B в процедурі = 10
A після процедури = 6
B після процедури = 10

Якщо змінна або константа описана в основній підпрограмі, вона вважається глобальною, і неї можуть використовувати будь-які

процедури і функції даної програми. Змінні, описані усередині підпрограми, називаються локальними і можуть бути використані тільки усередині даної підпрограми. Локальні змінні можуть бути описані як у заголовку підпрограми, так і в розділі опису змінних.

2.6 Зовнішні файли

Файл - послідовність елементів одного типу. Довжина файлу, тобто кількість елементів у цій послідовності – величина довільна, змінювана в процесі роботи. Робота з файлами полягає в запису і зчитуванні елементів цієї послідовності. Для того, щоб указати, з яким елементом файлу провадяться операції, існує поняття покажчика на доступний елемент файлу. Допустимо, ми маємо справу з файлом, у якому записуються перемінні типу Word, тоді перемінна файлового типу може бути введена так:

Var MyFile : file of word;

Для демонстрації роботи з файлами розглянемо два наступних приклади: Дано натуральне n. записати в файл G цілі числа послідовності b1, ..., bn, де при i=1, 2, 3 . . .,n значення bi дорівнює i.

```
Program Natur;
Uses CRT;
Var
    G:file of integer;
     N, i:integer;
     TextBack:byte;
     Begin
        TextBack:=TextAttr;
        TextAttr:=$0b;
        ClrScr;
       Assign(G,'g.out');
        Rewrite(G);
        Writeln(' Введіть кількість чисел');
        Readln(n);
        Writeln('Відкриття файлу g.out');
        Writeln(' Запис файлу g.out');
        For i:=1 to n  Do Write(G,i);
```

```
    Close(G);
    TextAttr:=$0b;
    Writeln(' Файл g.out успішно записано');
    TextAttr:=$0a;
    Writeln('Для повернення у DOS натиснути ENTER');
    Readln;
    TextAttr:=TextBack;
  End.
```

Результат виконання програми:

Введіть кількість чисел
120
Відкриття файлу g.out
Запис файлу g.out
Файл g.out успішно записано
Для повернення у DOS натиснути ENTER

Відомості про учня складаються з його імені, прізвища, назви класу (рік навчання та літера), у якому він вчиться. Існує файл F, у якому відомості про учнів школи. З'ясувати, у яких класах вчаться більш ніж два учня.

```
Program Loda;
Uses CRT;
  Type
    linnet=record
    Name: String ;
    Famly_Name: String;
    Year: Integer;
    Letter: Char;
   End;
Var
  FileOfSchool:File of  Linnet;
  n,i:integer;
  Inp:Linnet;
  Buffer, Count:Array [1..10] of  Integer;
  Buffer:ArrayNew [1..10] of  char;
  Flag:Boolean;
```

```
 Begin
Assign(FileofSchool,'laba. io');
Rewrite(FileofSchool);
Writeln('Введіть кількість записів');
Readln(n);
For i:=1 to n do
     Begin
        ClrScr; Gotoxy(10,1);
          Write(' Ім’я учня: ');
          Readln(Inp.Name);
          Gotoxy(10,2);
          Write(' Прізвище учня: ');
          Readln(Inp.Famly_Name);
          Gotoxy(10,3);
          Write(' Рік навчання: ');
          Readln(Inp.Year);
          Gotoxy(10,4);
          Write(' Літера: ');
          Readln(Inp.Letter);
          Repeat Until KeyPressed;
          ClrScr;
          Write(FileOfSchool,Inp);
      End;
Close(FileOfSchool);
Reset(FileOfSchool);     i:=1;
Read(FileOfSchool,Inp);
Buffer[i]:=Inp.Year;
BufferNew[i]:=Inp.Letter;
Repeat
Read    (FileOfSchool,Inp);
For n:=1 to n do
      Begin
          Flag:=True;
          If ((Inp. Year=Buffer[n]) and Inp.letter=BufferNew[n]))
          Then begin
                    Inc(Count[n]);
                    Flag:=False;
                    Break
```

```
                    end
        End;
  If Flag then Begin
                    Inc(i);
                Buffer[i]:=Inp.Year;
                    BufferNew[i]:=Inp.Letter;
               End;
  Until Eof (FileOfSchool);
  For n:=1 to i do
  If Count[n]>1 then Writeln(Buffer[n],BufferNew[n]);
  End.
```

Результат виконання програми:

Ім’я учня: Василь
Прізвище учня: Кобзів
Рік навчання: 11
Літера: А

Інформація в текстових файлах представлена в символьному вигляді, однак вони мають деякі особливості, що відрізняють їх від файлів file of char. Тип текстових файлів має в Паскалі ім'я text. Оголошення файлової змінної текстового типу

Var MyTextFile : text;

До текстових файлів можуть бути застосовані процедури Assign, Reset, ReWrite і функція EOF. Процедури Seek, FilePos і FileSize до них не застосовні, а процедури запису і читання мають істотні відмінності. У текстовому файлі інформація розділена на окремі рядки довільної довжини. Ознака кінця рядка – спеціальний маркер.

Різниця між текстовими і типізованими файлами:

- Текстові файли зручніше для сприйняття людиною, а типізовані відповідають машинному уявленню об'єктів;
- Текстові файли, як правило, довше типізованих (у випадку спроби запису однакових даних);

У наступному прикладі явно ілюструються основні принципи роботи з текстовими файлами.

Існує символьний файл F. Записати в файл G зі збереженням порядку слідування ті символи файлу F за якими у цьому файлі йде

літера "a".

```
Program fayl;
Uses CRT;
Var
   F,G:text;
   Buffer:string:
   I:integer;
     Begin
       Assign(F,'fayl.in');

 Assign(G,'fayl.out');
       Reset(F);
       Rewrite(G);
       While not Eof(F)  Do
                          Begin
                            Readln(F,Buffer);
                            For i:=1   to  254  Do
                                  If  Buffer[i+1]='a'
                                  Then write(G,Buffer[i]) ;
                                   Buffer[0]:=#0;
                          End;
       Close(F);
       Close(G);
     End.
```

Початковий файл
Fayl.in
DatartaerAghay
Файл, отриманий після виконання програми
Fayl.out
Dttrh

2.7 Графічні засоби мови Turbo Pascal

Множина графічних процедур і функцій середовища програмування Borland Pascal зібрані в модулі Graph. Для роботи в графічному режимі, принаймні, для початку, досить навчитися правильно їх використовувати. Підключення бібліотеки графічних функцій у вашій програмі здійснюється рядком

Uses Graph;

приведеної після заголовка програми.

Важливе поняття в машинній графіці - графічний примітив - сукупність пікселей, що визначають деяку геометричну фігуру. Найбільше часто – це просто лінії, прямокутники, полігони, окружності й еліпси, і природно точка.

Взаємодія програми й відеосистеми в графічних режимах забезпечують драйвери, у яких застосовуються графічний інтерфейс фірми Borland - Borland Graphics Interface (BGI).

У Паскалі передбачена процедура InitGraph(var GraphDriver, GraphMode : integer; Path : String), що переключає комп'ютер у графічний режим. Аргументи GraphDriver і GraphMode визначають графічний режим.

Принцип роботи в графічному режимі добре відбитий у наступних двох прикладах.

Отримати на екрані фігуру (обличчя), зробити так щоб воно підкліпувало.

```
Uses Crt, Graph ;
var d,m:integer;
    l,i,j,s:integer;
Procedure
tre(a,x,b,y,c,z:integer);
begin
line(a,x,b,y);
line(a,x,c,z);
line(b,y,c,z);
end;
Procedure eye(k,v,n:integer);
begin
```

```
    for l:=1 to 25 do
      Ellipse(v,n,0,36025,12-l);
       delay(k);
    for l:=25 downto 1 do
    begin
       setcolor(black);
       Ellipse (v,n,0,360,25,12-l);
          setcolor(white);
  Ellipse(v,n,0,360,5,5);
end;          setcolor(white);
   circle(v,n,25);
   circle(v,n,10);
  end;
Begin D:=Detect;
   Initgraph(d,m,'\TP'); Setcolor(white);
   Circle(201,101,200); i:=-1; j:=-1;
   while i<=20 do begin
                      inc(i);inc(j,2);
                      line(201+i,19,201+j,41);
                      line(201-i,19,201-j,41);
                    end;
tre(110,140,205,180,290,140);
tre(180,120,195,70,210,120);
circle(120,90,25);   circle(280,90,25);
circle(120,90,10);   circle(280,90,10);
for s:=1 to 5 do
begin
  eye(50,120,90);
  eye(50,280,90);
end;
CloseGraph;
End.
```

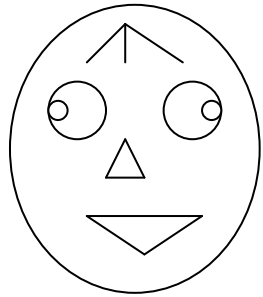


Рисунок 2.7 – Результат виконання програми

Отримати на екрані зображення курча яке літає.

```
Program Cipx;
  Uses Crt, Graph;
  Var
    d,b,i,j:integer;
    a:boolen;
Procedure Tre (x,xx,xxx,z,zz,zzz:word);
     Begin
          line(xxx,zzz,xx,zz);
          line(xx,zz,x,z);
          line(x,z,xxx,zzz);
       End;
  Procedure Sh(var i,j:integer);
   Begin
          circle(20+i,20+j,10);
          tre(1+i,10+i,10+i,20+j,18+j,22+j);
          ellipse(50+i,30+j,0,360,30,10);
          tre(40+i,60+i,60+i,30+j,40+j,30+j);
          tre(80+i,95+i,80+i,27+j,30+j,32+j);
          line(45+i,40+j,40+i,50+j);
          line(55+i,40+j,60+i,50+j);
   End;
Begin
  d:=detect;
```

```
Initgraph(d,b,'c:\tp\');
i:=0;
    j:=0;
   Sh(i,j);
   a:=true;
   While a:=true Do
                  Begin
                   Case Readkey of
                              #0: Begin Case
                                  Readkey of
                                  #72: Begin Dec(j,10);
                            Cleardevice;
                      Sh(i,j);
                                        End;
                                  #80: Begin Inc (j,10);
                           Cleardevice;
                                  Sh(i,j);
                                        End;
                                  #75: Begin Dec (i,10);
                                  Cleardevice;
                                  Sh(i,j);
                                        End;
                                  #77: Begin Inc (i,10);
                                  Cleardevice;
                                  Sh(i,j);
                                        End;
                                        End;
                  else a:=false;
                  End;
End.
```

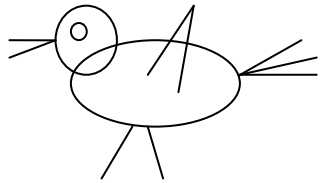


Рисунок 2.8 – Результат виконання програми

3 ЛАБОРАТОРНІ РОБОТИ

Лабораторні роботи містить кілька задач, номери яких вказані в таблицях у додатку А індивідуальних завдань для кожної лабораторної роботи. Номери задач вказані за збірником задач С.Н. Абрамова та ін. Задачі з програмування - М.:Наука.,1988 г. – 224с.

Номер варіанту відповідає номеру студента в журналі.

3.1 Лабораторна робота №1 “Програмування лінійних і розгалужених процесів”

Мета роботи: отримати знання і навички, необхідні для програмування лінійних та розгалужених процесів, набути і закріпити на прикладах складання програм, знання алгоритмічної мови Турбо-Паскаль.

Виконання лабораторної роботи містить наступні етапи:

- Скласти алгоритми вирішення задач лінійних та розгалужених процесів.
- Написати програми алгоритмічною мовою Паскаль;
- Підготувати контрольні приклади для перевірки працездатності програм;
- Надрукувати і проаналізувати отримані результати.

Контрольні питання.

1. Яка загальна структура Паскаль-програми?
2. Що називається ідентифікатором?
3. Як записується та виконується оператор присвоювання?
4. Коментарії та їх призначення?
5. Які типи даних вам відомі?
6. Відмінність в описанні констант та змінних?
7. Стандартні процедури для вводу та виводу даних?
8. Вкажіть операції по спаду їх пріоритету.
9. Які існують у Паскалі стандартні математичні функції.

10. Оператор вибору.
11. Оператор переходу.
12. Умовний оператор.

3.2 Лабораторна робота №2 “Програмування задач циклічної структури”

Мета роботи: отримати знання і навички, необхідні для програмування циклічних процесів, набути і закріпити на прикладах складання програм, знання Алгоритмічної мови Турбо-Паскаль.

Виконання лабораторної роботи містить наступні етапи:

1. Скласти алгоритми вирішення задач циклічних процесів.
2. Написати програми алгоритмічною мовою Паскаль;
3. Підготувати контрольні приклади для перевірки працездатності програм;
4. Надрукувати і проаналізувати отримані результати.

Контрольні питання.

1. Для чого служать оператори циклу?
2. Напишіть структурну форму та блок-схему оператора циклу з параметром.
3. Напишіть структурну форму та блок-схему оператора циклу з передумовою.
4. Напишіть структурну форму та блок-схему оператора циклу з постумовою.
5. Які обмеження накладаються на параметр циклу?
6. Напишіть оператор циклу з шагом -1.

3.3 Лабораторна робота №3 “Перетворення і побудова матриць”

Мета роботи: отримати знання і навички, необхідні для роботи з матрицями.

Виконання лабораторної роботи містить наступні етапи:
1. Скласти алгоритми вирішення задач вкладених циклів.
2. Написати програми алгоритмічною мовою Паскаль;
3. Підготувати контрольні приклади для перевірки працездатності програм;
4. Надрукувати і проаналізувати отримані результати.

Контрольні питання.

1. Що називається масивом?
2. Як описується масив?
3. Що називається індексом масиву?
4. Які обмеження накладаються на індекси?
5. Які типи використовуються для вказування меж змін індексів?
6. Що називається матрицею?
7. Як здійснюється ввід матриці по рядкам?

3.4 Лабораторна робота №4 "Програмування задач з обробки послідовності символів”

Мета завдання: отримати знання і навички, необхідні для роботи з символьною інформацією.

Виконання лабораторної роботи містить наступні етапи:
1. Скласти алгоритми вирішення задач з використанням символьної інформації.
2. Написати програми алгоритмічною мовою Паскаль;
3. Підготувати контрольні приклади для перевірки працездатності програм;

4. Надрукувати і проаналізувати отримані результати.

Контрольні запитання.

1. Для чого служить тип CHAR?
2. Як описується рядковий тип?
3. Яка допустима довжина (в байтах) рядкової змінної?
4. В чому відмінність рядкової змінної від символьного масиву?
5. Які операції виконуються над зміною типу STRING?
6. Як здійснюється доступ до окремих елементів рядка ?
7. Які існують стандартні функції для роботи з рядковими змінними?
8. Що таке блок? Як описується блок?

3.5 Лабораторна робота № 5 “Програмування задач з використанням функцій та процедур”

Мета завдання: навчитися програмувати задачі з використанням процедур та функцій.

Виконання лабораторної роботи містить наступні етапи:

1. Скласти алгоритми вирішення задач з застосуванням процедур та функцій.
2. Написати програми алгоритмічною мовою Паскаль;
3. Підготувати контрольні приклади для перевірки працездатності програм;
4. Надрукувати і проаналізувати отримані результати.

Контрольні запитання.

1. Що включає в себе заголовок процедури та функції?
2. Відмінність процедури від функції?
3. Як здійснюється повернення процедури та функції?
4. Які параметри називаються формальними?
5. Які параметри називаються фактичними?
6. Як здійснюється передача параметрів з програми в блок?
7. Поняття про глобальні зміни?

8. Які зміни є локальними?
9. Що таке рекурсія?
10. Стандартні процедури для виходу з блока і програми?

3.6 Лабораторна робота № 6 “Програмування задач з використанням зовнішніх типізованих файлів”

Мета роботи: Отримати практичні навички організації роботи з зовнішніми і типізованими файлами в програмах на Турбо-Паскалі.

Виконання лабораторної роботи містить наступні етапи:
1. Утворити зовнішній файл, необхідний для п.2.
2. Написати і налагодити програму обробки зовнішніх файлів.
3. Написати і налагодити програму обробки текстових файлів.

Контрольні питання.

1. Що таке файл?
2. Що зветься компонентами файлу?
3. Як визначається довжина файлу?
4. Як задається файловий тип?
5. Як здійснюється доступ до файлу?
6. Як здійснюється ініціалізація файлу?
7. Процедури та функції для роботи з текстовими файлами.
8. Процедури та функції для роботи з типізованими файлами.

3.7 Лабораторна робота №7 “Вивчення графічних засобів мови Турбо-Паскаль”

Мета роботи: Отримати практичні навички при програмуванні графічних задач.

Виконання лабораторної роботи містить наступні етапи:

1. Скласти малюнок (креслення) зображуваного об'єкта з позначенням координат зображуваних точок.
2. Написати і налагодити програму отримання графічного зображення. Програма повинна містити коментарії до кожного елемента зображуваного об'єкта.

Контрольні питання.

1. Що таке графічний примітив?
2. Що таке графічний драйвер?
3. Як перейти у графічний режим?
4. Що таке палітра?
5. Які існують відеорежими?
6. 6.Як здійснити виведення тексту у графічному режимі?

4 КУРСОВЕ ПРОЕКТУВАННЯ

4.1 Мета і задачі курсової роботи

Мета КР – придбання практичних навичок з алгоритмізації інженерних задач, програмування на алгоритмічній мові Турбо-Паскаль, відлагодження програм та розробці програмної документації.

В результаті виконання КР студенти повинні вміти:

- розробляти алгоритм і схеми програми;
- програмувати на алгоритмічній мові Турбо-Паскаль;
- підготувати завдання для рішення задачі на ЕОМ;
- викопувати відлагодження програм;
- розробляти програмні документи відповідно до вимог стандартів ЄСПД.

Виконання КР дозволяє студентам одержати знання з основ програмування па алгоритмічній мові Турбо-Паскаль, з мовою керування завданнями, методами та засобами відлагодження програм.

Знання та навички, отримані студентами під час виконання курсових проектів та робіт, використовуються в наступних курсах та в дипломному проектуванні.

4.2 Проектування програми

> ✍ ***"Година, витрачена на вибір алгоритму, коштує 5 годин програмування".***

Етап проектування програми завдає впливу на стиль програмування, надійність, відлагодження, тестування та експлуатаційні властивості програми. Таким чином це найважливіша частина будь–якої програмної розробки. Простота при проектуванні програми – це перший крок, що веде до отримання програми яка легко читається. Крім автора програму повинен читати ще хто–небудь, тому її забезпечують коментарями, ділять на параграфи, вона стає ясною і точною.

Найважливішим кроком для отримання ефективної та правильної програми є складання алгоритму. Хороший алгоритм –

необхідна умова хорошої задачі. Формулювання задачі в вигляді чіткого алгоритму суттєво полегшує процес проектування.

Існує немало літературних джерел, де розглядають обчислювальні алгоритми. В трьохтомнику Д. Кнута "Мистецтво програмування" міститься велика кількість загальних обчислювальних алгоритмів.

Універсальністю будемо називати незалежність програми від конкретного набору даних. Хороша універсальна програма повинна оброблювати вироджені випадки та друкувати повідомлення про помилку. Тоді програма буде не лише універсальною, а й захищеною від помилок. Програма може стати універсальною, якщо в якості параметрів замість констант використовувати змінні.

Формати вхідних та вихідних даних являються частиною етапу проектування. Вхідні формати повинні бути розроблені з урахуванням максимальної зручності і мінімальної можливості помилок. Порядок змінних і формати даних, звичні для користувача, допоможуть запобігти помилок та полегшать використання програми. Постійність вхідних форматів, як правило, також сприяє зменшенню помилок. Вихідна інформація повинна ідентифікуватися без застосування других джерел, вихідні дані повинні містити:

- ідентифікацію вихідного запису;
- опис та функцію запису;
- дату;
- нумерацію сторінок.

Крім того, кожна надрукована група елементів повинна бути помічена. При табличній формі видачі повинні бути позначені рядки та колонки. Керування складністю – дуже важливий фактор в програмуванні. Розбиття на модулі – це потужний засіб керування параметром складності. Наступний крок в керуванні загальною складністю – керування складністю в межах кожного модуля; цей процес має назву структурне програмування. Структурне програмування зосереджується на однім з найбільш схильним до помилок факторі програм – логіці програми – і містить три головних складових частини:

а) проектування зверху вниз;

б) модульне програмування;

в) структурне програмування.

Метод програмування зверху вниз передбачає спочатку визначення задачі в загальних рисах, а потім поступового уточнення структури шляхом внесення більш дрібних деталей. Модульне програмування – це процес розділення програми на логічні частини, що називаються модулями, і послідовне програмування кожної частини. Структурне програмування – це метод написання добре структурованих програм. які дозволяють отримати програми, більш зручні для тестування, модифікації та використання.

4.3 Відлагодження програми

Мистецтво локалізації помилок, коли факт їх існування встановлений, носить назву відлагодження. Таким чином, відлагодження програми передбачає обов'язкову наявність помилки: в протилежному випадку ми маємо діло з тестуванням. Від загального часу розробки програми, відлагодження займає за оцінками спеціалістів, від 50 до 90 процентів.

Процес відлагодження залежить від умов функціонування програми, тобто від машини, яка використовується, мови програмування, операційної системи, специфіки задачі, а також від особливостей конкретної програми.

Відомі два підходи до відлагодження програми: при використанні одного з них значна частина часу роботи програміста витрачається на те, щоб спробувати запобігти помилкам, а ті помилки, що мають місце в програмі, знайти вручну; другий підхід передбачає максимальне використання ЕОМ для виявлення помилок. Існує ще й третій підхід до відлагодження, під час якого воно поєднується з процесом написання програми. Деякі програмісти надають перевагу написанню програм частинами, зразу намагаючись перевірити, чи функціонує кожна написана частина належним чином.

Як тільки задача визначена до кінця, програміст починає пошук можливого алгоритму або метод її рішення. Але він може вибрати невідповідний або неефективний алгоритм, і тоді весь процес вибору потрібно буде повторити спочатку.

Найкращий спосіб організації відлагодження – це зведення до

мінімуму необхідності в ній. Вдумлива розробка функціональної структури програми, яку супроводжує достатньо докладна блок–схема або опис, забезпечує умови для кращого кодування програми. Блок–схема допомагає запобігти багатьох помилок і, крім того, їх можливо використовувати в якості допоміжного засобу виявлення помилок під час відлагодження. Можливо назвати 6 типів помилок, які викликані невірними діями програміста:

1) Пропуск деяких програмних рядків.
2) Перестановка рядків програми.
3) Поява зайвих рядків.
4) Відсутність необхідних даних.
5) Непередбачені дані.
6) Невірний формат даних.

Велике значення для успішного відлагодження програми мають простота і раціональність її кодування. Коли програма написана акуратно і логічно, легше запобігти помилок або виявити їх у випадку виникнення.

Щоб програму можливо було застосувати, передусім вона повинна бути правильною, а порушення правильності може проявитися двома способами: або невірна синтаксична конструкція програми, або програма видає невірні результати. Виявлення синтаксичних помилок важливо з тієї причини, що вони неодмінно приводять до труднощів при виконанні програми.

Відлагодження починається з того моменту, коли перестають видаватися повідомлення про помилки. На початку процесу відлагодження необхідно використовувати прості тестові дані. Якщо при цьому отримуються вірні результати, слід переходити до тестування програми шляхом більш складних даних.

У випадку коли результати невірні, можливі такі ситуації:

1. Синтаксичних помилок немає, але програма не скомпонована.
2. Програма скомпонована, працює, але не видає результатів.
3. Програма скомпонована, працює, але відбувається передчасна зупинка.
4. Програма скомпонована, працює, але видає невірні результати.
5. Програма зациклилась.

Безліч програмних помилок пов'язані з невірним зчитуванням вхідних даних. Вбудування відлагоджувальних засобів в програму є не чим іншим, як захищеним програмуванням. Засоби відлагодження,

передбачені в програмі, називають зупинувачами помилок.

Існує декілька типів відлагоджувальних засобів, що застосовуються при програмуванні:

1) Друкування утримуваного в пам'яті.
2) Відслідкування шляху виконання алгоритму.
3) Відслідкування звернення до змінних.
4) Відслідкування звернення до підпрограм.
5) Перевірка індексів.
6) Відтворення значення змінних.

Взагалі з відлагодженням пов'язана найбільша частина витрат з розробки програми, тому необхідно прагнути до запобігання програмних помилок.

4.4 Об'єм і зміст КР

КР містить в собі пояснювальну записку обсягом 20-25 сторінок, яка складається з таких розділів:

- вступ
- опис програми
- керівництво програміста
- закінчення

Додатки повинні містити тексти програм, надруковані вхідні дані контрольного прикладу, надруковані результати роботи програми.

4.5 Опис окремих розділів КР

Розділ "Вступ" пояснювальної записки повинен містити відомості про програму, встановлені ГОСТ 19.502–78 "ЄСПД. Опис застосування":

- найменування програми;
- призначення програми;
- обмеження на область застосування;
- необхідні технічні засоби;
- загальні відомості про вхідні та вихідні дані.

Розділ "Опис програми" повинен мати відомості відповідно до ГОСТ 19.402–78 "ЄСПД. Опис програми":

1) Загальні відомості:
   - позначення та найменування програми;

- програмне забезпечення, необхідне для функціонування програми;
- мова програмування, на яких на яких написана програма.

2) Функціональне призначення:
- призначення програми;
- відомості про функціональні обмеження на застосування.

3) Опис логічної структури:
- алгоритм програми;
- застосовані методи;
- структура програми з описом функцій складових частин та зв'язку між ними;
- зв’язок програми з другими програмами.

Повинна бути приведена таблиця ідентифікаторів, яка вміщує в собі символічні імена всіх змінних і масивів, їх типи та призначення. Опис логічної структури програми виконують з урахуванням тексту програми на вихідній мові.

1) Використані технічні засоби:
- тип електронної обчислювальної машини;
- обладнання, яке застосовується при роботі програми.

5) Виклик та завантаження:
- спосіб виклику програми з відповідного носія даних;
- вхідні точки в програму.

Допускається вказувати адреса завантаження, відомості про використання оперативної пам'яті та об'єм програми.

6) Вхідні дані:
- характер, обмеження та попередня підготовка вхідних даних;
- формат, опис та спосіб кодування вхідних даних.

7) Вихідні дані:
- характер та організація вихідних даних;
- формат та спосіб кодування вихідних даних.

В описі повинно бути посилання на відповідний додаток, який містить в собі текст програми.

Розділ "Керівництво програміста" повинен мати відомості, необхідні для роботи з програмою. Його зміст повинен відповідати ГОСТ 29.504–79 "ЄСПД. Керівництво програміста":

1) Призначення та умови використання програми:
   - призначення і функції, що виконує програма;
   - умови, необхідні для виконання програми (об'єм оперативної пам'яті, вимоги до складу периферійного обладнання, вимоги до програмного забезпечення).

2) Характеристики програми:
   - опис основних характеристик та особливостей програми (часові характеристики, режим роботи, засоби контролю правильності виконання програми).

3) Звернення до програми:
   - опис процедури виклику програми (засоби передачі керування та параметрів даних).

4) Вхідні і вихідні дані:

опис організації вхідної та вихідної інформації, при необхідності, способи її кодування.

5) Повідомлення:
   - тексти повідомлень програмісту або оператору, що видаються програмою під час її виконання;
   - опис їх змісту і дії, які необхідно ужити за цими повідомленнями.

Розділ "Закінчення" повинен вміщувати:
   - опис засобів перевірки працездатності програми;
   - опис вхідних даних, що використовуються при тестуванні програми та результатів виконання з різними наборами вхідних даних, загальні відомості про програму, та висновок про працездатність програми;
   - можливі галузі використання результатів роботи.

4.6 Організація керівництва курсовою роботою

Кожний студент одержує індивідуальне технічне завдання, яке містить в собі вимоги до розроблюваної програми. При виконанні курсової роботи студенти повинні додержуватися графіку (табл. 1) зі звітом по кожному пункту.

Таблиця 4.1 – Графік виконання курсової роботи

Номер етапу	Номер тижня		Зміст звіту	Форма звіту
	початок етапу	закінчення етапу		
1	2	3	4	5
1	1	1	Видача завдання	
2	2	3	Кодування вхідних даних	Текст вхідних даних
3	3	4	Аналіз можливих помилок у вхідних даних	Підрозділ "Повідомлення програми" розділу "Керівництво програміста"
4	4	7	Розробка алгоритму функціонування програми	Текст програми
5	7	12	Відлагодження програми. Розрахунок контрольного прикладу.	Надруковані результати виконання контрольного прикладу
6	8	10	Розробка розділів пояснювальної записки "Вступ" та "Керівництво програміста"	Розділ ПЗ "Вступ" та "Керівництво програміста"
7	10	12	Розробка розділів ПЗ "Опис програми" та "Закінчення"	Розділи ПЗ "Опис програми" та "Закінчення"

Продовження таблиці 4.1

1	2	3	4	5
8	13	14	Оформлення ПЗ та перевірка викладачем	ПЗ
9	15	16	Захист курсової роботи	

4.7 Вказівки до оформлення та захисту КР

ПЗ повинна бути оформлена відповідно до діючих стандартів ЄСПД-90. Захист КР здійснюється в комісії з 2-3 чоловік за участю керівника КР. Захист складається з невеликої доповіді (5-7 хвилин) студента про виконану роботу та запитання членів комісії. В доповіді необхідно доповісти про поставлену задачу, алгоритми її рішення та отримані результати.

При оцінюванні КР враховуються якість роботи та результати її захисту студентом. Особлива увага звертається на повноту, якість та самостійність виконання завдання, якість оформлення ПЗ, дотримання графіку виконання КР.

4.8 Теми курсових робіт

1. Програма формування електричних принципових схем (для двох чоловіків).
2. Текстовий редактор.
3. Програма формування тексту.
4. Програма контролю знань.
5. Програма для демонстрації графіків функцій.
6. Програма для пошуку путі у лабіринті.
7. Програма аналізу функцій двох перемінних.
8. Гра "Морський бій".
9. Електронний журнал.
10. Гра "П'ятнашки".

11. Гра "Калах".
12. Програма формування зображення друкованої плати.
13. Програма розташування елементів на платі.
14. Програма формування тексту Паскаль-програм.
15. Гра "Рендзю".
16. Програма побудови лабіринту.
17. Гра "Реверси".
18. Програма для автоматизованого підготування схем програм (для двох чоловік).
19. Аналізатор Паскаль-програм.
20. Програма для навчання роботі з клавіатурою.
21. Пошук найкоротшого шляху.
22. Машинні узори.
23. Система для створення поверхонь.
24. Криптографічна система.
25. Головоломки.

ЛІТЕРАТУРА

1. Абрамов С.А., Зима Е.В. Начало программирования на языке Паскаль. –М.: Экономика, 1987. – 112с.
2. Абрамов С.А., Зима Е.В. Задачи по программирования. –М.: Наука, 1988.
3. Андерсон Р. Доказательство правильности программы. –М.: Мир, 1982. – 164 с.
4. Безбородов Ю.М. Индивидуальная отладка программ. – М.: Наука. 1982.– 164 с.
5. Бородич Ю.С., Вальвачев А.Н., А.И. Кузмич. Паскаль для персональних комп'ютерів. Довідковий посібник.-М.: Вища школа, 1991р.-230 с.
6. Боэм Б. и др. Характеристики качества программного обеспечения.–М.: Мир. 1981.–208 с.
7. Ван Тассел Д. Стиль, разработка, зффективность, отладка и испытание программ: Пер. с англ. –М.: Мир. 1981. –320 с.
8. Гласс Р. Руководство по належному программированию. –М.: Финансы и статистика. 1982.
9. Грогон Н. Программирование на языке Паскаль: Перев. с анг. –М.: Мир, 1982–368 с.
10. Зелковец М. и др. Принцип разработки программного обеспечения. Перев. с анг. – М.: Мир, 1982 – 368 с.
11. Йодан Э. Структурное программирование и конструирование программ.–М.: Мир, 1989. – 416 с.
12. Довгаль С.И., Литвинов Б.Ю., Сбитнев А.И. Турбо Паскаль 7.0 Объектно-ориентированное программирования. Локальные сети.-К.: Информсистемасервис, 1993г, – 462 с.
13. Епанешников А.М., Епанешников В.А. Turbo Vision. Основы практического использования.-М.:ДиалогМифи,1995г. – 234 с.
14. Кнут. Искусство программирования для ЕОМ. Том 3. М:Наука, 1976-1978г. – 736 с.
15. Любимский Э.З., Мартынюк В.В., Трифонов Н.П. Программирование. - М.: Наука, 1980г,-607с.
16. Марченко А.И. Программирование в среде ТР 7.0.-К.:Век, 1998г. – 510 с.

17. Д. Прайс Программирование на языке Паскаль: Практическое руководство. М.: Мир, 1987 г.-232 с.
18. Практическое руководство по программированию. Под редакцией Н. Рашби, П. Хита, Б. Мика, - М: Радио и Связь, 1986г.-166с.
19. Скэнлон Л. Персональные ЭВМ IBM PC и XT. Программирование на языке ассемблера.-М.: Радио и связь, 1989 г.-336с.
20. Смирнов Курс высшей математики. 3 том,- М: Наука, 1974 г,-323с.
21. Фаронов В.В. Основы Турбо-Паскаля.- М.:УИЦ МВТУ, 1992 г. – 304с.
22. Холстед М.Х. Начала науки о программах /Пер. с англ. В.М. Юфы.- М.: Финансы и статистика, 1981г.-128 с., ил.
23. Хьюз Дж., Мичтом Дж.. Структурный подход к программированию.- М.: Мир, 1980г.-278 с.
24. Стандартизация языков программирования. Под ред. чл. корр. АН УССР Е.Л. Ющенко, -К.: Техніка, 1989г.-155с.
25. Вирт Н. Язык программирования Паскаль (пересмотренное сообщение).–М: Статистика. 1977. – Вып.9
26. Вирт Н. Алгоритмы + Структуры данных=программы: Пер. с англ. – М.: Мир, 1985. – 392с.

Додаток А
Завдання до лабораторних робіт

Таблиця А.1 - Завдання до лабораторної № 1

Номер варіанту	Номери задач	Номер варіанту	Номери задач
1	53, 31к	15	60г, 569
2	29, 554	16	60в, 68а
3	555, 32д	17	60б, 67д
4	57а, 32е	18	60а, 68б
5	558, 32г	19	57а, 72а
6	559, 32в	20	51, 72б
7	560, 32б	21	50, 73
8	561, 31л	22	46, 76а
9	562, 33а	23	45, 76б
10	563а, 57г	24	44, 76в
11	563б, 57в	25	43, 76г
12	580, 57б	26	41, 76д
13	60е, 67д	27	11е, 59и
14	60д, 578	28	35б, 59к

Таблиця А.2 - Завдання до лабораторної № 2

Номер варіанту	Номери задач	Номер варіанту	Номери задач
1	2	3	4
1	143, 106, 755д	15	140, 113а, 119в
2	146, 107, 755г	16	150, 113б, 119г
3	144а, 143з, 755е	17	149, 116а, 119е
4	144б, 116в, 755ж	18	148, 116б, 119б
5	146, 116г, 755з	19	147, 102, 119а
6	152, 116д, 758а	20	56б, 103, 758д
7	153, 116е, 758б	21	157, 104, 758е
8	155, 115е, 758в	22	158, 114а, 758ж
9	156а, 115ж, 758г	23	166, 114б, 758з
10	145в, 108, 755в	24	161, 114в, 759а
11	145б, 109, 755б	25	163, 114г, 759б

Продовження таблиці А.2

1	2	3	4
12	145а, 110, 755а	26	137в, 114д, 759в
13	142, 111, 119е	27	137г, 114е, 759г
14	141, 112, 119д	28	36о, 114ж, 760б

Таблиця А.3 - Завдання до лабораторної № 3

Номер варіанту	Номери задач	Номер варіанту	Номери задач
1	395, 692а, 674	15	408, 692з, 694б
2	400, 767б, 413а	16	369, 393б, 383
3	397а, 692б, 675	17	412б, 692и, 694в
4	397б, 412а, 678	18	370а, 393а, 383
5	412б, 692в, 679а	19	412в, 692к, 694г
6	396, 405, 688	20	412г, 693а, 694д
7	412в, 692г, 679б	21	409, 693б, 694е
8	372, 394а, 376а	22	370б, 394г, 385
9	412г, 692д, 376б	23	378а, 410е, 694ж
10	373б, 401а, 676а	24	378б, 691а, 694з
11	371, 394б, 387	25	410д, 691б, 694е
12	403а, 692е, 687	26	370б, 394в, 386
13	373а, 402б, 376б	27	368, 393в, 382
14	403б, 692ж, 694а	28	367, 393г, 381

Таблиця А.4 - Завдання до лабораторної № 4

Номер варіанту	Номери задач	Номер варіанту	Номери задач
1	2	3	4
1	251, 265, 361	15	257г, 270г, 421б
2	252а, 266, 362	16	257д, 70д, 421в
3	252б, 267, 363	17	257е, 312а, 421г
4	253а, 268, 364	18	258, 312б, 546а
5	253б, 269а, 365а	19	259, 313, 546б
6	253в, 269б, 365б	20	260а, 314, 546в
7	253г, 269в, 365в	21	260б, 315а, 547а

Продовження таблиці А.4

1	2	3	4
8	254, 269г, 365г	22	260в, 15б, 548
9	255, 69д, 366(1)	23	261а, 315а, 549
10	256а, 69е, 366(2)	24	261б, 316б, 550
11	256б, 269ж, 366(3)	25	262а, 316в, 551а
12	257а, 270а, 419	26	262б, 316г, 551б
13	257б, 70б, 420	27	263, 16д, 551в
14	257в, 270в, 421а	28	264, 360, 552а

Таблиця А.5 - Завдання до лабораторної № 5

Номер варіанту	Номери задач	Номер варіанту	Номери задач
1	424, 470, 317	15	440в, 342, 331а
2	425, 469, 318	16	441, 344, 31б
3	426, 468, 319	17	442, 345, 332
4	427, 467, 320	18	443, 346, 333
5	428, 466, 321	19	444, 347, 334а
6	429, 465, 322	20	445, 348, 334б
7	430, 464, 323	21	446, 349, 334в
8	431, 463, 324	22	447, 350а, 334г
9	423, 462, 325	23	445, 350б, 335а
10	433, 461, 326	24	456, 350в, 335б
11	434, 453, 327	25	457, 350г, 335в
12	435, 336в, 328	26	458, 350д, 335г
13	440а, 336г, 329	27	460а, 351а, 36а
14	440б, 340, 330	28	460б, 351б, 36б

Таблиця А.6 - Завдання до лабораторної № 6

Номер варіанту	Номери задач	Номер варіанту	Номери задач
1	2	3	4
1	527, 488а	15	493а, 472д
2	526, 488б	16	493б, 473а
3	525, 488в	17	493в, 473б

Продовження таблиці А.6

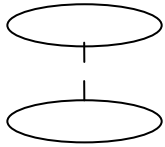
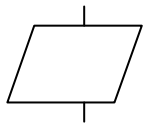
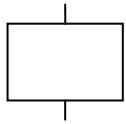
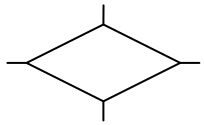
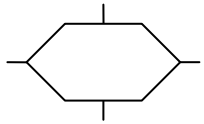
1	2	3	4
4	523а, 489а	18	494, 437в
5	523б, 489б	19	495, 484
6	522, 490	20	496а, 474а
7	521, 491	21	496б, 474а
8	520, 480а	22	497, 474б
9	479, 480б	23	498а, 474в
10	482, 480в	24	498б, 474г
11	485, 472а	25	499а, 474д
12	486, 472б	26	499в, 474е
13	492, 472в	27	519а, 490
14	519б, 472г		

Таблиця А.7 - Завдання до лабораторної № 7

Номер варіанту	Номери задач	Номер варіанту	Номери задач
1	509а, 964 (мал. 11а)	15	506д, 965 б(мал. 12а)
2	509б, 964 (мал. 11б)	16	506е, 965б(мал. 12б)
3	510а, 964 (мал. 11в)	17	507а, 965б(мал. 12в)
4	510б, 964 (мал. 11г)	18	507б, 965б(мал. 12г)
5	510в, 964 (мал. 11д)	19	507в, 965б(мал. 12д)
6	511а, 964 (мал. 11е)	20	507г, 965б(мал. 12е)
7	511б, 964 (мал. 11ж)	21	507д, 965б(мал. 12ж)
8	512, 964 (мал. 11з)	22	507е, 965б(мал. 12з)
9	513а, 964 (мал. 11и)	23	507ж, 965б(мал. 12и)
10	513б, 964 (мал. 11к)	24	507з, 965б(мал. 12к)
11	506а, 964 (мал. 11л)	25	508а,965б(мал12м)
12	506б, 964 (мал. 11м)	26	508б,965б(мал. 12н)
13	506в, 964 (мал. 11н)	27	506е, 965б(мал. 12л)
14	506г, 964 (мал. 11о)		

Додаток Б
Основні символи, які застосовуються у схемах алгоритмів

Таблиця Б.1 – Основні символи

Найменування	Позначення	Функція, що виконується
Пуск – зупинка Початок – кінець		Початок - кінець (вхід та вихід у підпрограмах)
Увід – вивід		Перетворення даних у форми придатку для обробки (увід), чи відображення результатів
Процес		Обчислювальна дія чи послідовність обчислювальних дій
Рішення		Перевірка умов
Модифікація		Керування циклом
		Обчислювання по алгоритму описаному окремо (підпрограма)

Додаток В
Завантаження середовища програмування Borland Pascal

Для початку роботи в середовищі Borland Pascal потрібно запустити файл bp.exe (або turbo.exe, якщо Ви вирішили обмежити себе програмуванням у для реального режиму). Завантаження програми може провадитися з параметрами в рядку запуску або без них. З використовуваних при запуску параметрів найбільше значення має файл конфігурації середовища програмування, у якому зберігаються параметри її настроювання. Коли цей файл не зазначений, система використовує файл конфігурації bp.tp (або, відповідно, turbo.tp). Якщо такий файл не буде знайдений, він може бути автоматично створений системою.

У рядку запуску середовища програмування після опцій може бути зазначене ім'я файлу, що містить програму на Паскалі. Якщо це зробити, то файл автоматично завантажиться в середовище програмування.

Звичайно, Ви можете використовувати командний рядок DOS, але, скоріше Ви зволієте використовувати оболонки. Якщо Ви використовуєте Norton Commander (або аналогічний файловий менеджер), то можна знайти потрібний файл у системі дисків і каталогів і, помістивши на цей файл курсор, натиснути Enter.

Додаток Д
Гарячі клавіші середовища програмування Borland Pascal

F1 – Виклик довідки
F2 – Зберегти файл
F3 – Відкрити файл
F4 – Виконати програму до поточної позиції курсору
F5 – Розширити вікно редактора програми
F6 – Наступне вікно
F7 – Трасування програми
F8 – Покрокове виконання програми
F9 – Компонування програми
Alt – BkSp – Відмінити останню дію
Shift – Del – Вирізувати виділений текст
Ctrl – Ins – Скопіювати виділений текст
Shift – Ins – Уставити з буфера текст
Ctrl – Del – Видалити виділений текст
Ctrl – F9 – Виконати програму
Ctrl – F2 – Перервати програму
Alt – F9 – Компілювати програму
Alt – F9 – Показати екран DOS
Ctrl – F5 – Змінити розміри або пересунути вікно редактора програми
Shift – F6 – Попереднє вікно
Alt –F3 – Закрити файл
Alt – 0 – Список відкритих файлів
Shift – F1 – Показати зміст довідки
Ctrl – F1 – Пошук довідки по виділеному
Alt – F11 – Попереднє вікно довідки
Alt – X – Вихід

Додаток Е
Основні процедури і функції Borland Pascal

Таблиця Е.1 – Основні процедури і функції

Інтерфейс	Призначення
Арифметичні операції	
Function Abs (x : real) : real;	Абсолютне значення
Function ArcTan (x : real) : real;	Арктангенс кута
Function Cos (x : real) : real;	Косинус кута в радіанах
Function Exp(x : real) : real;	Показова функція (експонента)
Function Frac(x : real) : real;	Дробова частина числа
Function Int(x : real) : real;	Ціла частина числа
Function Ln (x : real) : real;	Логарифм
Function Pi : real;	Число π
Function Random : real; Function Random(Range : word) : word; Якщо задано діапазон Range – значення типу word від 0 до (Range-1), якщо не заданий – речовинне число від 0 до 1.	Випадкове число
Procedure Randomize;	Ініціалізація генератора випадкових чисел випадковим значенням
Function Sin(x : real) : real;	Синус кута в радіанах
Function Sqr(x : real) : real;	Квадрат аргументу
Function Sqrt(x : real) : real;	Квадратний корінь з аргументу
Перетворення типів	
Function Chr(i : byte) : char;	Повернення символу по його коду в таблиці ASCII
Function Ord(i : <порядковий тип>) : longint;	Перетворення типу char і boolean до цілого типу; для цілих аргументів повторює значення аргументу

Продовження таблиці Е.1

Інтерфейс	Призначення
Перетворення типів	
Function Round(x : <речовинний тип>) : longint;	Округлення значення речовинного типу до цілого
Function Trunc(x : < речовинний тип >) : longint;	Усікає значення речовинного типу до цілого
Введення і висновок	
Procedure Assign (FileVar : <файловий тип >, FileName); FileVar – перемінна файлового типу; FileName – рядок, що містить ім'я файлу	Присвоєння імені зовнішнього файлу змінній
Procedure ChDir(NewDir : string); NewDir – новий директорій	Зміна поточного директорія
Procedure Close (FileVar : <файловий тип>); FileVar – перемінна файлового типу;	Закриття відкритого файлу
Function Eof(FileVar : <файловий тип>); FileVar – перемінна файлового типу;	Визначає, чи досягнуть кінець файлу
Procedure Erase(FileVar : <файловий тип>); FileVar – перемінна файлового типу;	Видалення зовнішнього файлу
Function FilePos(FileVar : <файловий тип>) : longint; FileVar – перемінна файлового типу;	Повертає поточну позицію у файлі
Function FileSize(FileVar : <файловий тип>) : longint; FileVar – перемінна файлового типу;	Повертає поточний розмір файлу
Procedure GetDir (Driveer : byte; var CurrenDir : string); Driver – числовий код диска (0 – поточний, 1 -А, 2 – У, тощо) CurrentDir – поточних директорій;	Повертає поточних директорій на заданому диску

Продовження таблиці Е.1

Інтерфейс	Призначення
Введення і висновок	
Function IOResult : integer;	Повертає стан останньої виконаної операції введення/виводу
Procedure MkDir (NewDir : string); NewDir – новий директорій;	Відкриває новий директорій
Procedure Read (FileVar : <файловий тип>, var v1,v2,…); ProcedureRead(FileVar:Text,varv1,v2,…); ProcedureRead(v1,v2,…) FileVar–перемінна файлового типу, для читання при введенні з клавіатури може не вказуватися; v1,v2,…-перемінні	Зчитує одне або більш значень з файлу в одну або більш перемінних
Procedure ReName(Read (FileVar : <файловий тип>, NewName); FileVar – перемінна файлового типу;	Перейменовує зовнішній файл
Procedure ReSet(Read (FileVar : <файловий тип>); FileVar – перемінна файлового типу;	Відкриває існуючий файл
Procedure ReWrite(Read (FileVar : <файловий тип>); FileVar – перемінна файлового типу;	Створює і відкриває новий файл
Procedure RmDir (DelDir : string); DelDir – каталог, який вилучається	Вилучання порожнього каталогу
Procedure Seek(Read (FileVar : <файловий тип>; Pos : позиція покажчика); FileVar – перемінна файлового типу;	Пересуває покажчик на заданий компонент файлу
Procedure Truncate (Read (FileVar : <файловий тип>); FileVar – перемінна файлового типу;	Усікає розмір файлу до поточної позиції у файлі

Продовження таблиці Е.1

Інтерфейс	Призначення
Введення і висновок	
Procedure Write (FileVar : <файловий тип>, var v1,v2,…); ProcedureWrite(FileVar:Text,varv1,v2,…); ProcedureWrite(v1,v2,…) FileVar–перемінна файлового типу, для читання при введенні з клавіатури може не вказуватися; v1,v2,…- перемінні	Записує одне або більш значень перемінних у файл
Робота з рядками	
Function Concat (s1,s2,…:string):string; S1,s2,…- послідовність поєднуваних рядків;	Об'єднання (конкатенація) рядків
Function Copy (Source : string; Pos, Count : integer) : string; Source – вихідний рядок; Pos - номер позиції, з яким починається копіювання; Count – число символів, які потрібно скопіювати	Копіює підстроку з рядка
Procedure Delete (var Source : string; Pos, Count : integer) : string; Source – вихідний рядок; Pos - номер позиції, з яким починається видалення; Count – число символів, що видаляються	Видаляє підстроку з рядка
Procedure Insert (Source : string; var Dest : string; Pos : integer) : string; Source – рядок, що вставляється; Dest - рядок з яким проводиться операція; Pos - номер позиції, з яким починається вставка	Уставляє підстроку в рядок

Продовження таблиці Е.1

Інтерфейс	Призначення
Робота з рядками	
Function Length (S : string) : integer;	Визначає динамічну довжину рядка
Function Pos(SubStr , S : string) : byte; S - рядок, у якій відбувається пошук; SubStr - шукана підстрока, результат - позиція в S, починаючи з якою іде шукана підстрока. Якщо підстрока не утримується в рядку, то повертається 0	Пошук підстроки в рядку
Procedure Str (X : <речовинний тип>; s : string); X –число, яке перетворюється; S – результуючий рядок	Перетворює чисельне значення в рядок
Function UpCase(Symbol : char) : char;	Перетворює букви латинського алфавіту в прописні
Procedure Val (Valuestring : string; var X : <речовинний тип>; errCode : integer); ValueString –рядок, який перетворюється; X – результат цілого або речовинного типу; ErrCode – код помилки	Перетворить рядок у чисельне значення
Текстові файли	
Procedure Append (var F : Text);	Відкриває існуючий файл для додавання тексту в кінець файлу
Function EoLn(var : Text); Значення функції true, якщо досягнуть кінець рядка у файлі, false – якщо немає	Визначає, чи досягнутий у файлі кінець рядка
Procedure Flush (var F : Text);	Виштовхує вміст буфера у файл

Продовження таблиці Е.1

Інтерфейс	Призначення
Текстові файли	
Procedure ReadLn(var F : Text, v1,v2,…); Procedure ReadLn(var v1,v2,…); V1,V2…- перемінні	Виконує ті ж дії, що і Read, а потім робить пропуск до початку наступного рядка файлу
Function SeekEof(var F : Text) : boolean;	Визначає, чи досягнутий кінець файлу, пропускаючи пробіли і символи табуляції
Function SeekEoln(var F : Text) : boolean;	Визначає, чи досягнутий у файлі кінець рядка, пропускаючи пробіли і символи табуляції
Function SetTextBuffer(var F : Text; var Buf; Size : word); Function SetTextBuffer(var F : Text; var Buf); Buf – буфер; Size – розмір буфера;	Призначає буфер уведення, висновку для текстового файлу
Procedure WriteLn(var F : Text, v1,v2,…); Procedure WriteLn(var v1,v2,…); V1,V2…- перемінні	Виконує ті ж дії, що і Write, а потім додає до файлу маркер кінця рядка