МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни «***ІНФОРМАТИКА ТА СИСТЕМОЛОГІЯ***»

«Основи програмування на мові Pascal»

Рівень вищої освіти – «Бакалавр»
Спеціальність 103 «Науки про Землю»

Одеса, 2019

Передмова

Метою дисципліни є формування у студентів теоретичних знань по алгоритмізації процесів обробки інформації та практичних навичок по основам програмування.

Мови ЕОМ виступають як засіб спілкування. Якщо людська мова - це засіб спілкування людей, то мова ЕОМ - засіб спілкування людини й машини. Мова Pascal була розроблена професором Ніклаусом Віртом (Швейцарська Вища технічна школа, м. Цюріх). Названо мову було на честь французького математика XVII століття Блеза Паскаля, що в 1640 році створив арифметичну (або рахункову) машину. Незважаючи на те, що він був розроблений як засіб для навчання техніці програмування, згодом він став популярним і поза сферою освіти.

На відміну від людського, мова ЕОМ не допускає двозначностей і невизначеностей. Кожна мова має строго певну граматику, названу синтаксисом. Якщо пропозиція програми (оператор) не відповідає синтаксису мови, то він не має змісту. З іншого боку, синтаксично правильний оператор має однозначне трактування.

Програма для ЕОМ - це просто послідовність операторів. Оператори являють собою команди ЕОМ, випливаючи яким вона вирішує завдання. Метод, використовуючи який програма вирішує завдання, називається алгоритмом; справа програміста вибрати алгоритм, щоб скласти по ньому програму. Алгоритм може являти собою просто математичне рівняння (наприклад, квадратне рівняння) або складну процедуру.

По кожній лабораторній роботі студент повинен скласти **звіт**, який містить в собі:

1. Назву роботи. Мету.
2. Алгоритм розв'язання задачі у вигляді блок-схеми.
3. Текст (лістинг) програми.
4. Відповідь на контрольні питання.

Оформлений звіт захищається студентом усно.

Варіант індивідуального завдання надається викладачем самостійно

Правила техніки безпеки та охорона праці

Згідно з «Правилами техніки безпеки в лабораторіях інформатики» студентам забороняється:

- з`являтися та знаходитись приміщенні в нетверезому стані;
- ставити поруч з клавіатурою ємності з рідиною;
- перебувати в приміщенні в верхній одежі та завалювати нею робочі столи та стільці;
- працювати в лабораторії більше 6-ти годин на день (для вагітних жінок – більше 4-х годин);
- за власною ініціативою змінювати закріплені за ними робочі місця та знаходитись в приміщенні під час роботи іншої учбової групи;
- самостійно виконувати вмикання електроживлення лабораторії та заміну складових частин ПК, що вийшли із ладу.

У випадку виявлення несправностей обчислювальної техніки студент повинен сповістити про це викладача чи будь-кого з навчально-допоміжного персоналу лабораторії.

ЗМІСТ

Лабораторна робота №1

Алфавіт, лексика, структура програми. Організація уведення/виводу. Програмування завдань лінійної структури.

Мета роботи: Вивчення елементарних понять – алфавіт, зарезервовані слова, типи, змінні й т.п. Знайомство з основними етапами процесу розробки програми, з поняттям алгоритму, методами запису алгоритмів.

Постановка завдання: розробити алгоритм рішення завдання за заданим варіантом, скласти програму мовою Pascal, відлагодити її, виконати розрахунки для кількох варіантів вхідних даних, відповісти на контрольні питання.

Теоретичні відомості:

Алфавіт мови Pascal включає букви, цифри, шістнадцятирічні цифри, спеціальні символи, пробіли й зарезервовані слова.

Букви - це букви латинського алфавіту від **a** до **z** і від **A** до **Z**, а також знак підкреслення "**_**". У Turbo Pascal немає розходження між прописними й малими літерами алфавіту, якщо тільки вони не входять у символьні й строкові вираження.

Цифри – арабські цифри від 0 до 9.

Спеціальні символи Turbo Pascal - це символи + - * / = ,' . : ; < > [ ] ( ) { } ^ @ $ # . До спеціальних символів ставляться також наступні пари символів: <> <= >= := (* *) (. .) У програмі ці пари символів не можна розділяти пробілами, якщо вони використаються як знаки операцій відносини або обмежники коментарю.

Ідентифікатори в Turbo Pascal - це імена констант, змінних, міток, типів, об'єктів, процедур, функцій, модулів, програм і полів у записах. Ідентифікатор завжди починається буквою, за якої можуть слідувати букви й цифри. Буквою вважається також символ підкреслення, тому ідентифікатор може починатися цим символом і навіть складатися тільки з одного або декількох символів підкреслення. Пробіли й спеціальні символи алфавіту не можуть входити в ідентифікатор.

Зарезервовані слова використовуються для опису операторів, даних і інших язикових конструкцій. Зарезервовані слова не можуть використовуватись як ідентифікатори. Вони надають тексту програми більш «читабельний» вигляд, наближаючи його до тексту, написаному на природній англійській мові.

Приклади правильних ідентифікаторів:
a ALPHA MyProgram _beta

Приклади неправильних ідентифікаторів:
1Program {починається з цифри}
block#l {містить спеціальний символ}
My Prog {містить пробіл}
mod {зарезервоване слово}

У кожній програмі для обчислення потрібних результатів широко використовуються змінні. Це величина, значення якої змінюється в процесі роботи програми. Коли змінної привласнюється нове значення, її старе значення втрачається. Для оголошення змінної необхідно вказати ім'я змінної і її тип. Однотипні змінні можуть перераховуватися через кому перед вказівкою їхнього типу.

Приклад оголошення змінних:
a: integer;
b: byte;
c: boolean;

У прикладі використані змінні *a* типу Integer, *b* типу Byte і *c* типу Boolean. Тип визначає безліч припустимих значень, які може мати той або інший об'єкт, а також безліч припустимих операцій, які застосовні до нього. Перші два типи призначені для цілих даних, причому Integer може містити як додатні, так і від'ємні числа в діапазоні від – 32768 до + 32767, а Byte тільки додатні в діапазоні від 0 до 255. Ці типи вважаються сумісними. Опис інших типів даних - у **Додатку 1.**

Константа – це величина, значення якої не змінюється в процесі роботи програми. Як константи в Pascal можуть використатися цілі, дійсні й шістнадцятирічні числа, логічні константи, символи, рядки символів.

Цілі числа записуються зі знаком або без нього за звичайними правилами.

Дійсні числа записуються зі знаком або без нього з використанням десяткової крапки й/або експонентної частини. Експонентна частина починається символом **e** або **E**, за яким можуть випливати знаки «+» або «-» і десятковий порядок. Символ **e (E)** означає десятковий порядок і має сенс «помножити на 1.0 у ступені». Наприклад, 3.14E5 - 3.14 помножити на 10 у ступені 5.

Pascal замислювався автором як навчальна мова структурного програмування. Як наслідок цього програми, на ньому написані, мають досить просту, але тверду структуру. Вона така:

{ частина оголошення ім’я програми: }
Program NameOfProgram;
{ частина оголошення списку модулів, що підключають:}
Uses { список модулів через кому };
{ частина оголошення констант:}
Const { список констант і їхніх значень через ; };
{ частина оголошення типів користувача:}
Type { список типів користувача через ; };
{ частина оголошення змінних:}
Var { список змінних і їхніх типів через ;}
{ частина основної програми}
Begin { крапка входу}
{ оператори основної програми}
End. { основна крапка виходу й кінець програми}

Частини Program і Uses не є обов'язковими, однак повинні бути першими і єдиними в програмі. Const, Type і Var частин у програмі може існувати безліч, їхній порядок не нормується й визначається вимогами програмування. Частина основної програми завжди є останньої, ознакою її завершення є ключове слово End із крапкою після нього. Будь-яка інформація після завершення основної частини ігнорується компілятором.

Для знайомства з мовою Pascal спробуємо скласти нескладну програму, що здійснює вивід якого-небудь повідомлення на екран ПК. Нехай це буде фраза «Я програмую на Турбо Паскалі» (фраза може бути іншою). Можливий варіант такої програми:

Приклад:

```
Program My_First_Program;
begin
  WriteLn('Я програмую на Турбо Паскалі');
end.
```

Тепер спробуйте виконати програму. Для цього послу набору її тексту натисніть **Ctrl-F9**. Якщо Ви не помилилися при уведенні тексту, то через кілька секунд побачите швидку зміну зображень на екрані: відразу після завантаження програми Pascal очищає екран, надаючи його в розпорядження працюючої програми користувача. Такий екран називається вікном програми. Якщо в тексті були помилки, виправте їх (див. **Додаток 4**) і запустите програму заново. Після завершення прогону (робота програми часто називається її прогоном або компіляцією) на екрані знову з'явиться вікно редактора з текстом програми. Якщо Ви не

встигли розглянути зображення вікна програми, натисніть **Alt-F5**. Після натискання на будь-яку клавішу середовище поверне екран у режим відтворення вікна редактору.

Обговоримо її єдиний оператор

WriteLn();

Цікаво, що в Pascal взагалі й Turbo Pascal, зокрема, немає спеціальних операторів вводу-виводу. Для обміну інформацією з навколишнім світом у програмах, написаних мовою Pascal, використовуються спеціальні стандартні процедури. Таким чином, по своїй суті оператор *WriteLn*() є оператором звертання до вбудованої процедури виводу даних (своя назву вона одержала від *WriteLine* - записати рядок).

Поняття процедури - одне із центральних понять Pascal. Процедура - це деяка послідовність операторів, до якої можна звернутися по імені. Щоразу, коли ми називаємо в операторі ім'я процедури, ініціюється послідовність запрограмованих у ній дій.

Процедура (або оператор) *Write* здійснює вивід на екран або друкувальний пристрій. Оператор може виводити повідомлення або значення змінної. Повідомлення записуються в апострофах. Для виводу значення змінної вказується ім'я змінної. Повідомлення й змінні можна чергувати в одному списку, розділяючи комами. Курсор залишається за останнім виведеним даним. Оператор *writeln* виконує аналогічні дії й переводить курсор на наступний рядок. Оператори *write* і *writeln* допускають вивід даних із форматуванням.

Write(a:5:2);

Перше число вказує, скільки екранних знаків приділяється під вивід. Друге - вказує кількість знаків після коми в числі й може бути відсутнє.

Оператор *read* здійснює введення даних з клавіатури. Дані, що вводять, розміщуються як значення змінних, імена яких перераховані в круглих дужках за оператором read.

Read (a, b, c);

Уводяться дані теж списком, у якому вони розділяються пробілом. Уведення закінчується натисканням Enter. Курсор, що відзначає позицію наступного уведення/виводу, залишається за останнім уведеним даним. Оператор *readln* виконує аналогічні дії й переводить курсор на наступний рядок.

Частина програми, що виконується, складається з одного й більше операторів. Оператор описує деяку дію, що повинна виконати програма. Переважна більшість операторів містить зарезервовані слова, що пояснюють відповідну дію.

Оператор присвоєння - один з найбільш часто використовуваних операторів обробки даних. Його вигляд:

Змінна := вираз;
Y:= sin(x);

І змінна, і вираз повинні бути того самого типу або належати до сумісних типів. Знак операції складається із двох символів, які не можна розривати пробілом.

Складений оператор. Якщо необхідно при деякій умові виконати відразу групу операторів, їх поєднують в одному складеному операторі за допомогою операторних дужок begin – end.

Те, із чим ми мали справу вище, лише умовно можна назвати програмою. Спробуємо створити щось більш складне й корисне.

Приклад: розробити програму, яка обчислить значення У:

$Y = \frac{\ln(1 + a\sqrt{1 + a^2})}{b + \sqrt{1 + \sin(c)}}$ при цьому, в = 3,18; а й с необхідно вводити із клавіатури

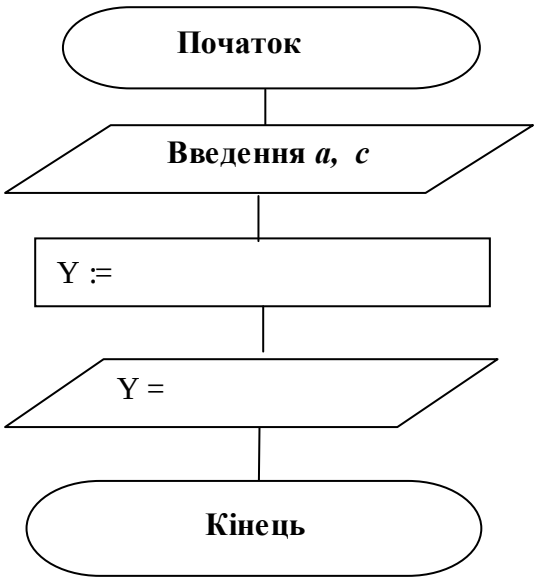
Program Primer1; {Оголошення імені програми, це - Primer1}
//не забуваємо про те, що кожний рядок програми повинен закінчуватися крапкою з комою, крім того, текст програми може супроводжуватися коментарями, укладеними у фігурні дужки або відділеними // .
Var a, c, Y: real; {перелік із трьох змінних, використовуваних у програмі з оголошенням типу}
Const b=3.18; {оголошення постійної, використовуваної в програмі}
Begin {початок програми, після нього крапка з комою не ставиться!}
Write('Введіть значення А:'); {Вивід на екран текстового повідомлення, пропозиція ввести А}
Read (A); {Оператор читання, у цьому випадку служить для уведення значення змінної А с клавіатури. На практиці - програма припиняє роботу й від користувача програми потрібно ввести значення відповідних змінних.}
Write('Введіть значення С: ');
Read (c); {Введення змінної С}

Y:= (ln(1+a*sqrt(1+sqr(a))))/(b+sqrt(1+sin(c))) { := оператор присвоєння. В правій частині стоїть вираз, якій необхідно обчислити. Потім отримане значення привласнюється змінній, що стоїть в лівій частині оператора}

Writeln('Y=', Y); {Виводимо на екран значення Y; спочатку йде повідомлення в апострофах, символи між ними будуть виведені на екран у тім виді, у якому вони представлені в тексті програми, після йде вивід знайденого значення}

End. {Кінець програми, після нього завжди ставиться крапка}

Схема алгоритму рішення такого завдання буде мати такий вигляд:



Варіанти завдань для самостійної роботи:

1. Дано два числа a і b. Одержати їхню суму, різницю й добуток.

2. Дані x, y, z. Обчислити a, b, якщо:

$$a = \frac{\sqrt{|x-1|} - \sqrt{|y|}}{1 + x^2/2 + x^2/4}, b = x(arctg(z) + e^{-(x+3)})$$

3. Дано дійсні числа x і y. Одержати (|x| - |y|) / (1 + |x*y|).

4. Дані x, y, z. Обчислити a, b, якщо:

$$a = \frac{3 + e^{y-1}}{1 + x^2|y - tg(z)|}, b = 1 + |y - x| + \frac{(y-x)^2}{2} + \frac{(y-x)^3}{3}$$

5. Дано довжину ребра куба. Знайти площу грані, площа повної поверхні й об'єм цього куба.

6. Дано x, y, z. Обчислити a, b, якщо:

$$a = (1+y)\frac{x + y/(x^2) + 4}{e^{-x-2} + 1/(x^2+4)}, b = \frac{1+\cos(y-2)}{x^4/2 + \sin^2 z}$$

7. Дано два дійсних додатних числа. Знайти середнє арифметичне й середнє геометричне цих чисел.

8. Дано x, y, z. Обчислити a, b, якщо:

$$a = y + \frac{x}{y^2 + \left|\frac{x^2}{y + x^3/3}\right|}, b = (1 + tg^2\frac{z}{2})$$

9. Дано катети прямокутного трикутника. Знайти його гіпотенузу й площу.

10. Дано x, y, z. Обчислити a, b, якщо:

$$a = \frac{2\cos(x - \pi/6)}{1/2 + \sin^2 y}, b = 1 + \frac{z^2}{3 + z^2/5}$$

11. Визначити периметр правильного n-кутник, описаного біля окружності радіуса r.

12. Дані x, y, z. Обчислити a, b, якщо:

$$a = \frac{1+\sin^2(x+y)}{2 + \left|x - 2x/(1+x^2y^2)\right|} + x, b = \cos^2(arctg\frac{1}{z})$$

13. Дано сторону рівностороннього трикутника. Знайти площу цього трикутника.

14. Дано x, y, z. Обчислити a, b, якщо

$$a = \ln\left|(y - \sqrt{|x|})(x - \frac{y}{z + x^2/4})\right|, b = x - \frac{x^2}{3} + \frac{x^3}{4}$$

Контрольні питання:

1. Яка структура програми в мові Pascal ?
2. Що таке ідентифікатор?
3. Чим відрізняються константи й змінні? Як вони оголошуються?
4. Що таке зарезервоване слово?
5. Що таке тип даних? Назвіть, які типи даних ви знаєте?
6. Укажіть, як працюють оператори уведення/виводу?
7. Чим відрізняється *write* від *writeln* і *read* від *readln*?
8. Які дії реалізує оператор присвоєння?
9. Для чого використовується складений оператор?
10. Як записується коментар в програмі?

Лабораторна робота №2

Алгоритмічна структура - розгалуження.
Умовний оператор, оператор вибору.

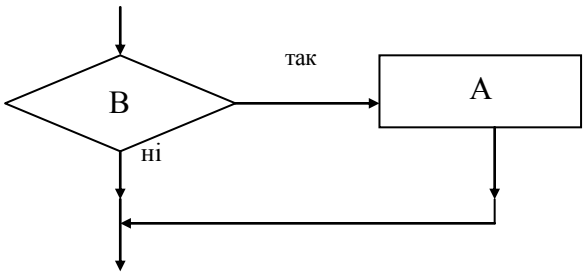
Мета роботи: Знайомство з однієї з основних алгоритмічних структур – розгалуженням. Вивчення особливостей умовного оператора й оператора вибору.

Постановка задачі: розробити алгоритм рішення завдання за заданим варіантом, скласти програму мовою Pascal, відлагодити її, виконати розрахунки для кількох варіантів вхідних даних, відповісти на контрольні питання.

Теоретичні відомості:

На практиці рідко зустрічаються задачі лінійної структури. Набагато частіше, залежно від яких-небудь проміжних результатів, необхідно організувати обчислення логічної структури. У мові Pascal є дві реалізації однієї з основних алгоритмічних структур - розгалуження – умовний оператор і оператор вибору. Ці оператори дозволяють на основі аналізу деякої умови вибрати потрібний варіант продовження обчислювального процесу. До операторів розгалуження відносять умовний оператор if та оператор вибору case.

Умовний оператор може мати дві форми: коротку й повну. Алгоритмічна схема короткої форми:



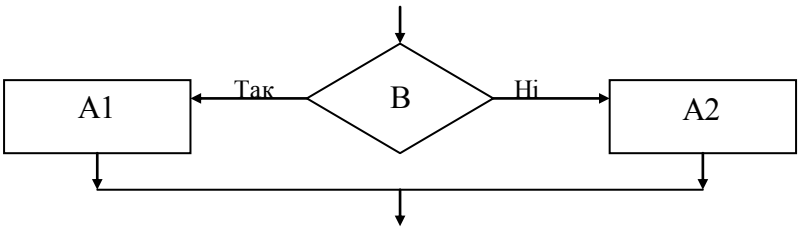
Цієї схемі відповідає такий програмний код:

If B then A;

B – умовний вираз, A - будь-який оператор Турбо Паскаля.

При виконанні оператора аналізується вираз B. Якщо він має значення True, виконується оператор A, у противному випадку нічого не відбувається й умовний оператор завершує свою роботу.

Алгоритмічна схема повної форми умовного оператора:



Цієї схемі відповідає такий програмний код:

```
If  B  then  A1
             else A2;
```

B - умовний вираз, A1, A2 - будь-які оператори Pascal.

При виконанні оператора аналізується вираз B. Якщо він має значення True, виконується оператор A1, у противному випадку виконується оператор A2.

Умовні оператори можуть бути вкладеними, тобто на місці оператора A в короткій формі й A1(A2) у повній формі можуть стояти умовні оператори. При вкладенні умовних операторів різних форм може виникнути ситуація, коли частин else менше частин then.

Наприклад:

```
If  X >Y  then
    If A = 0 then B:= 0
                 else B:= 1;
```

у цьому випадку вважається, що частина else зв'язується з найближчою по ходу написання програми частиною then, що не має частини else. Для наведеного приклада при виконанні умови X >Y змінна B одержить значення 0 або 1 залежно від виконання умови A=0. Однак якщо умова X >Y не виконується, другий умовний оператор не буде працювати й значення B не зміниться.

Оператор вибору дозволяє вибрати одне з декількох можливих продовжень програми. Параметром, по якому здійснюється вибір, служить

ключ вибору - вираження будь-якого порядкового типу (цілого, символьного, логічного).

Структура оператора вибору така:

Case <ключ вибору > of <список вибору > [else <оператори >] end

case, of, else, end - зарезервовані слова (вибрати, з, інакше, кінець).

Оператор вибору працює в такий спосіб. Спочатку обчислюється значення вираження <ключ вибору>, а потім у послідовності операторів <список вибору> відшукується такий, перед яким йшла константа, рівна обчисленому значенню. Знайдений оператор виконується, після чого оператор вибору завершує свою роботу. Якщо в списку вибору не буде знайдена константа, що відповідає обчисленому значенню ключа вибору, керування передається операторам, що йдуть за словом else. Частина else <оператори> може бути відсутня. Тоді при відсутності в списку вибору потрібної константи нічого не відбудеться, і оператор вибору просто завершить свою роботу.

Приклад. Необхідно розробити програму, що буде визначати, який сьогодні день тижня:

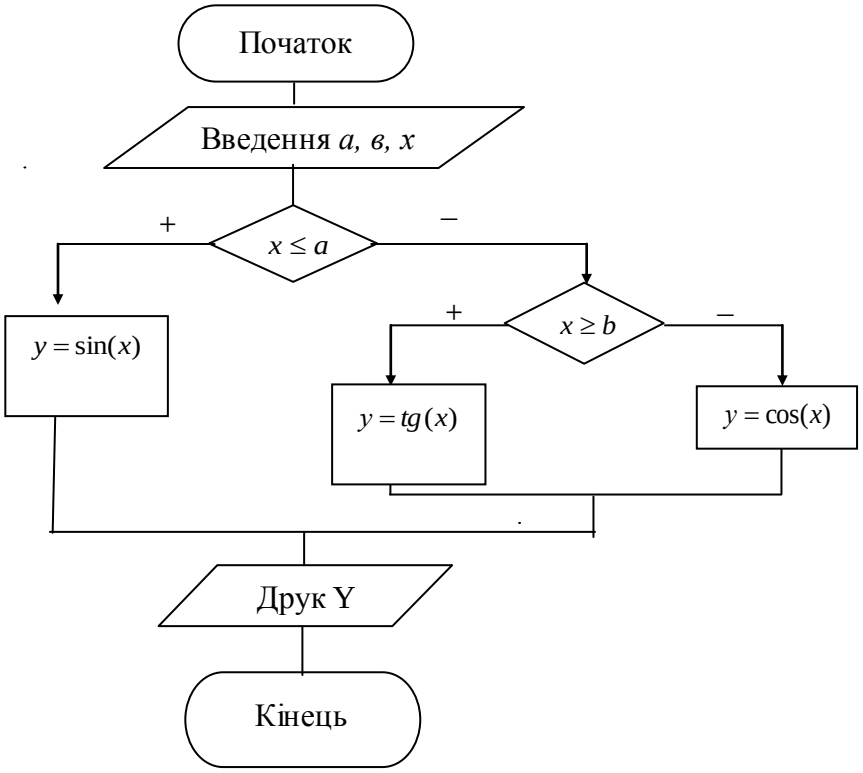
```
Program Nedelya;
Var  x: integer;
Begin
 Writeln(‘Введіть номер дня тижня ’);
 Read(x);
    Case  x  of
      1:  writeln (‘Сьогодні понеділок’);
      2:  writeln (‘Сьогодні вівторок’);
      3:  writeln (‘Сьогодні середа’);
      4:  writeln (‘Сьогодні четвер’);
      5:  writeln (‘Сьогодні п’ятниця’);
      6:  writeln (‘Сьогодні субота’);
      7:  writeln (‘Сьогодні неділя’);
    end;
end.
```

На закінчення розглянемо приклад, що знаходить значення функції У, використовуя умовний оператор if:

$$Y = \begin{cases} \sin x, & \text{якщо} \quad x \le a, \\ \cos x, & \text{якщо} \quad a < x < b, \\ \operatorname{tg} x, & \text{якщо} \quad x \ge b. \end{cases}$$

```
Program Primer2;
var   x ,y: real ;   { об'явлення початкових даних}
      a, b: integer ;
begin
 writeln(' Введіть значення x, a i b ') ;{ друк підказки для введення даних}
 readln ( x, a, b ) ;                          {введення початкових даних }
    if  x <= a       {перевірка першої умови задачі, якщо умова істинна –
обчислюється значення функції}
       then  y := sin(x)
{ якщо умова помилкова - перевірка другої умови }
          else  if  x <= b
             then  y:= cos(x)              {обчислення значення функції Y }
               else  y:= sin(x)/cos(x) ; {обчислення іншого значення функції У}
 writeln ('Y=', y);       { друк результату }
End.
```

Алгоритмічна схема рішення буде мати такий вигляд:



Варіанти завдань для самостійної роботи:

1. Дано дійсні числа x, y. Отримати: max(x,y); min(x,y);

2. Дано дійсні числа x, y, z. Отримати: max(x,y,z);

3. Дано дійсні числа x, y, z. Отримати: min(x,y,z);

4. Дано дійсні числа a, b, c. Подвоїти ці числа, якщо a > b > c, і замінити їхніми абсолютними значеннями, якщо це не так.

5. Дано дійсні числа x, y. Обчислити z:

$$z = \begin{cases} x - y, \text{ коли } x > y \\ y - x + 1, \text{ в іншому випадку} \end{cases}$$

6. Дано два дійсних числа. Вивести перше число, якщо воно більше другого, і два числа, якщо це не так.

7. Дано два дійсних числа. Замінити перше число нулем, якщо воно менше або дорівнює другому, і залишити без зміни в противному випадку.

8. Дано дійсні числа x, y. Менше із цих двох чисел замінити їхньою напівсумою, а більше - їхнім подвоєним добутком.

9. Визначити, яка із двох фігур - коло або квадрат має більшу площу. Відомо, що сторона квадрата дорівнює a, радіус кола r. Вивести на печатку назву й значення площі більшої фігури.

10. Дано дійсне число a. Обчислити f(x), якщо

a) $$f(x) = \begin{cases} x^2, \text{коли } -2 \le x \pi 2 \\ 4, \text{ в іншому випадку} \end{cases}$$

b) $$f(x) = \begin{cases} x^2 + 4x + 5, \text{коли } x \le 2 \\ \dfrac{1}{x^2 + 4x + 5}, \text{ в іншому випадку} \end{cases}$$

c) $$f(x)=\begin{cases}0, \text{ коли } x \le 0\\ x, \text{ коли } 0 \le \text{x } \pi 1\\ \text{x}^2, \text{ в іншому випадку}\end{cases}$$

d) $$f(x)=\begin{cases}0, \text{якщо } x \le -5\\ x^2 - x, \text{ якщо } -5 \le \text{x } \pi 5\\ \text{x}^2 - \sin \text{x}^2, \text{ в іншому випадку}\end{cases}$$

Контрольні питання:

1. Які види розгалужень реалізовані в Turbo Pascal?
2. Для чого використовуються розгалуження? Наведіть приклади.
3. Які форми умовних операторів вам відомі?
4. Як працює умовний оператор?
5. Чи можуть умовні оператори бути вкладеними?
6. Укажіть структуру оператора вибору.

Лабораторна робота №3

Організація циклів.

Мета роботи: ознайомитись з поняттям “циклу”. Вивчити основні типи циклів.

Постановка задачі: розробити алгоритм рішення завдання за заданим варіантом, скласти програму мовою Pascal, відлагодити її, виконати розрахунки для кількох варіантів вхідних даних, відповісти на контрольні питання.

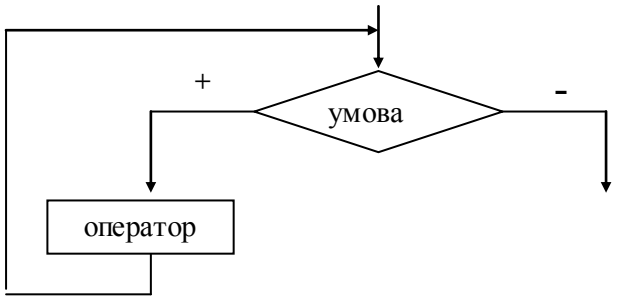
Теоретичні відомості:

Цикл – це послідовність операторів, що може виконуватися більше одного разу. У мові Pascal є кілька механізмів для конструювання циклів.

Є три основних типи циклів. Це цикл While (Eng. - поки), цикл For (Eng. - для) і цикл Repeat (Eng. - повторювати). Найпростіший з них – оператор while. При виконанні оператора while повторюється певна група операторів доти, поки в операторі while булева умова істинна.

Оператор циклу while **з попередньою перевіркою умови** записується таким чином:

While < умова> do <оператор>



While, do – зарезервовані слова (Eng.- поки [виконується умова], робити); <умову> - вираження логічного типу; <оператор> - довільний оператор Pascal.

Якщо вираз <умова> має значення True, то виконується <оператор>, після чого обчислення виразу <умова> і його перевірка повторюються. Якщо <умова> має значення False, оператор while припиняє свою роботу.

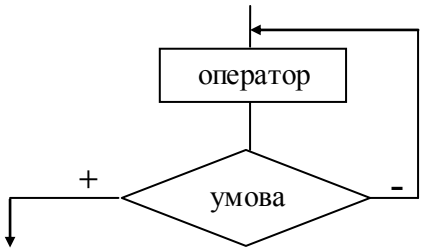
Розглянемо приклад використання оператора while на фрагменті програми, що обчислює ступінь числа 3 у діапазоні між 1 і 300.

```
a:=1;
  while  a<300  do
     begin
       writeln(a);
        a:=a*3;
      end;
```

Даний цикл буде виконуватися до тих пір, коли умова (a<300) не прийме значення False.

Оператор циклу repeat, відомий як **оператор циклу з постумовою**, має такий вид:

repeat <тіло_циклу> until <умова>



Тут repeat, until – зарезервовані слова (Eng.- повторювати доти поки не буде виконана умова); <тіло_циклу> - довільна послідовність операторів Pascal; <умова> - вираження логічного типу.

Оператори <тіло_циклу> виконуються хоча б один раз, після чого обчислюється вираження <умова>: якщо його значення є False, оператори <тіло_циклу> повторюються, у противному випадку оператор repeat... until завершує свою роботу. Той же фрагмент програми розглянемо в контексті даного оператора:

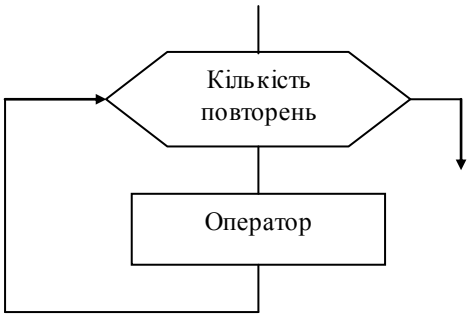
```
a:=1;
   repeat
      writeln(a);
      a:=a*3;
   while a>=300;
```

На відміну від циклу while, тут умова буде виконуватися, поки умова (a<300) не прийме значення True. Крім того, тіло циклу repeat не потрібно містити в операторні дужки begin…end; між ключовими словами repeat і until можна помістити будь-яку кількість операторів.

Цикл for є одним з основних видів циклів, які є у всіх універсальних мовах програмування. Основна ідея, закладена в його функціонування, полягає в тім, що оператори, що перебувають усередині циклу, виконуються фіксоване число раз, у той час як змінна циклу (відома ще як індексна змінна) пробігає певний ряд значень.

Оператор циклу for має таку структуру:

for <параметр циклу> := <початкове значення> to <кінцеве значення> do <оператор>



Тут for, to, do, down to – зарезервовані слова (Eng.- для, до, виконати, вниз);

Параметр циклу – змінна типу integer (точніше, будь-якого порядкового типу); початкове значення – вираз того ж типу; кінцеве значення – вираз того ж типу; <оператор> – довільний оператор Pascal. При виконанні оператора For спочатку обчислюється вираз <початкове значення> і здійснюється присвоювання <параметр циклу>:=<початкове значення>. Після цього циклічно повторюється:

- перевірка умови, якщо умова не виконана, оператор For завершує свою роботу;
- виконання оператора <оператор>;
- нарощування змінної <параметр циклу > на одиницю.

Наступний простий приклад ілюструє використання циклу з параметром.

```
For i:=1 to 10 do
  M[i]:=i*2;
```

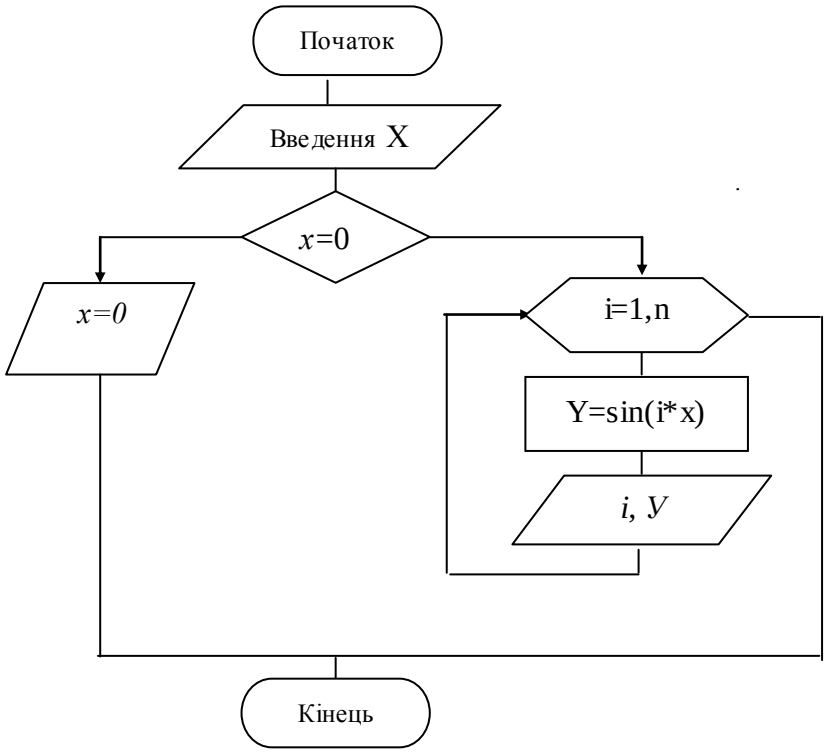
У цьому прикладі дано повторення оператору присвоєння, при цьому управляюча змінна і послідовно приймає значення 1, 2, ……9, 10.

Розглянемо *приклад*, в якому знайдемо значення функції У, використовуя оператор циклу:

Y= sin(nx), якщо n змінюється від 1 до 10

```
Program Primer3;
const n = 10 ;
var     x, y : real ;          { об'явлення початкових даних}
        i : integer ;
begin
  writeln(' input x ') ;    { друк підказки для введення даних}
  readln ( x ) ;            { введення початкових даних }
    if  x = 0               { перевірка умови задачі }
     then  writeln('x=0'){ умова істина - друк об'яви "x=0"}
        else                { умова помилкова }
for  i:=1  to  n  do
  begin
    y:=sin(x*i);            { обчислювання функції y }
    writeln('i=', i:3, '  y=',  y:3:2);  { друк результату }
  end;
  readln;
end.
```

Алгоритмічна схема рішення буде мати такий вигляд:



Варіанти завдань для самостійної роботи:

Скласти блок-схему й програму табулювання заданої функції.

№	Вигляд функції	a	b	$X_{поч}$	$X_{кін}$	ΔX
1	$Y = \frac{arctgbx}{1+2\sin x}$	-	0,75	1,35	6,5	0,8
2	$Y = \sqrt[5]{\frac{a+bx}{\ln x}}$	19,6	7,8	14,6	-	6
3	$y = \frac{a\ln x}{b+\sqrt{x}}$	1,38	1,26	60	-	10
4	$y = \frac{\sin x+1}{\sqrt{x+b}}$	-	1,68	1,2	2,4	0,2
5	$y = \frac{\ln(a-b)}{a\sqrt{x}}$	0,36	5,5	10	-	6
6	$y = \frac{e^{ax}+b}{1+2\cos x}$	0,9	-	0	1,2	0,15
7	$y = \frac{a+\sqrt[3]{x}}{\sin bx-2}$	1,24	0,67	-	12,4	0,45
8	$y = \frac{a\sqrt{x-bx}}{2a+x}$	-	0,45	40	60	4,5
9	$y = \frac{b-\sqrt{ax}}{b+\sqrt{x}}$	-	7,67	3,5	4	0,1

10	$y = e^{-x}\frac{a+b}{2x}$	4,6	-	0,75	1,8	0,3
11	$y = \frac{tgax - b}{e^{ax}}$	0,9	-	2,3	8,9	1,3
12	$y = \frac{arctgbx}{1+\sqrt{ax}}$	-	0,85	17,2	24,6	2

Контрольні питання:

1. Які види розгалужень реалізовані в Turbo Pascal?
2. Для чого використовується принцип розгалуження? Наведіть приклади.
3. Як працює оператор циклу For?
4. Чи може оператор циклу For змінюватися з довільним кроком?
5. Якими зарезервованими словами позначається оператор циклу while?
6. Як працює оператор циклу while?
7. Якими зарезервованими словами позначається оператор циклу repeat?
8. Як працює оператор циклу repeat?
9. Чим відрізняється цикл repeat від інших ?
10. Скільки операторів може бути після ключового слова repeat?

Лабораторна робота №4

Структурні типи даних.
Одномірні масиви. Сортування масивів.

Мета роботи: знайомство із структурним типом даних. Отримання практичних навиків при роботі з одномірними масивами.

Постановка задачі: розробити алгоритм рішення завдання за заданим варіантом, скласти програму мовою Pascal, відлагодити її, виконати розрахунки для кількох варіантів вхідних даних, відповісти на контрольні питання.

Теоретичні відомості:

Одним із структурованих типів даних є регулярний тип або **масив**. Масив являє собою сукупність зв'язаних даних, що складає з фіксованого числа елементів одного типу, що, як уже згадувалося, називається базовим. Для визначення масиву досить указати його базовий тип, а також число елементів у масиві й метод їхньої нумерації. Можливий доступ до будь-яких окремих елементів масиву - для цього досить указати ім'я масиву й номер (індекс) потрібного елемента.

Опис типу масиву задається в такий спосіб:

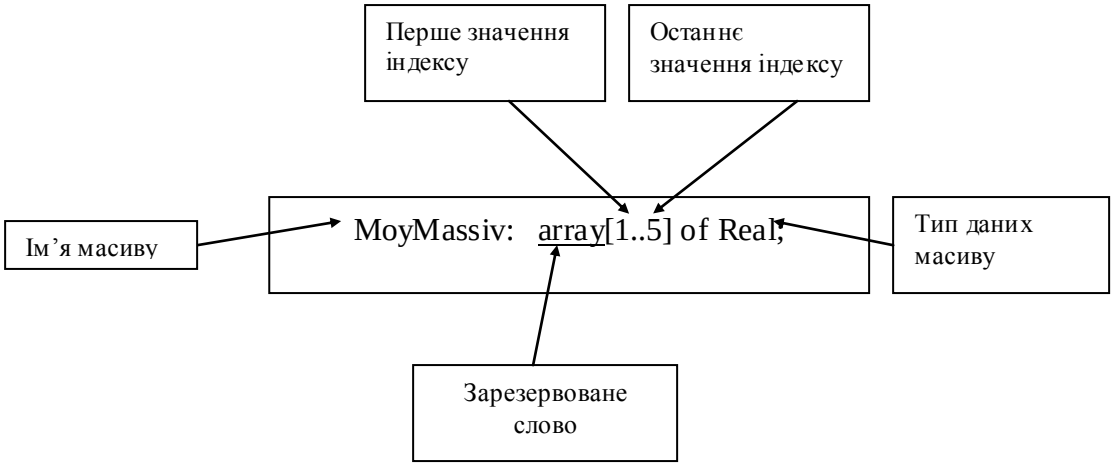
<ім'я_типу> = array [<сп. інд. типів>] of <тип>

Тут <ім'я типу> – правильний ідентифікатор; array, of – зарезервовані слова (Eng.- масив, з); <сп. інд. типів> - список з одного або декількох індексних типів, розділених комами; квадратні дужки, що обрамляють список – вимоги синтаксису; <тип> - будь-який тип Pascal.

Наприклад:

```
Massiv: array [1..100] of Real;
{одномірний масив із 100 елементів дійсного типу}
```

Розглянемо більш детально опис масиву:



Для звертання до елемента масиву в програмі вказується ім'я масиву й у квадратних дужках - індекс елемента.

Приклад:
A [2]
а) ввести значення: *read* (A[2]);
б) змінити значення: A[2]:=5;
в) повідомити значення: *write*(A[2]);

Пошук елемента в масиві: часто виникає необхідність знайти в масиві певний елемент, наприклад, знайдемо мінімальний елемент масиву і його індекс (порядковий номер).

```
Program Primer4;
Var   A: array [1..5] of integer;
       i, j, min: integer;
Begin
Writeln(‘ Введіть 5 цілих чисел);
  For i:=1 to 5 do
     Read (A[i]);
min:=a[1];
 For i:=1 to 5 do
  if min > a[i]  then
   Begin
      min:= a[i];
          j:= i;
   end;
 Writeln(‘Мінімальний елемент масиву = ’, min, ‘його індекс = ’, j );
END.
```

Сортування масиву. Вирішити цю задачу можна, наприклад, за допомогою *алгоритму сортування обміном.* Спосіб полягає в тому, щоб послідовно переглядати масив, порівнюючи черговий його елемент із попереднім. При цьому, якщо черговий елемент виявляється більше попереднього, елементи міняються місцями і так до кінця масиву. Таким чином, в результаті першого перегляду найбільший елемент масиву виявиться переміщений на останнє місце.

Приклад: Дан масив із n елементів. Необхідно упорядкувати всі елементи по зростанню.

```
{Введення масиву}
……..…
{Сортування масиву}
 for i:=1 to n-1 do
  for  j:=i+1 to n do
 Begin
     if  a[i]>a[j]  then
     begin
       temp:=a[i];
       a[i]:=a[j];
       a[j]:=temp;
     end;
  end;
{Вивід масиву}
……..…..
```

На наступному прикладі покажемо, як знайти **кількість непарних елементів масиву.** Поняття парності-непарності ставиться тільки до цілих чисел і визначається за допомогою функції mod (залишок від ділення). Так, якщо **a mod 2 = 0**, число a – парне, якщо ж **a mod 2 <> 0** – непарне.

```
{Введення масиву}
……..…
{Пошук кількості непарних елементів}
kvo:=0;
for i:=1 to n do
 if a[i] mod 2 <> 0 then
kvo:=kvo+1;
{Вивід кількості непарних елементів масиву}
……
```

Пошук **суми/добутку** всіх **елементів масиву.** Для цього задаємо змінну, яку дорівнюємо нулю, потім додаємо до неї по черзі всі значення елементів масиву.

```
{Введення масиву}
……..…
{Знаходження суми елементів}
sum:=0;              {змінна, в якій ми будемо накопичувати нашу суму}
   for i:=1 to n do   {звертаємось до всіх елементів масиву за допомогою циклу for }
       sum:=sum+a[i];
{Вивід суми елементів масиву}
……
```

Варіанти завдань для самостійної роботи:

1. У масиві X(10) знайти добуток парних елементів.
2. У масиві A(12) знайти суму від'ємних елементів.
3. У масиві B(25) знайти кількість нульових елементів.
4. У масиві Z(18) знайти суму елементів, більші числа A (A ввести з клавіатури).
5. У масиві X(8) знайти суму непарних елементів.
6. У масиві Y(20) замінити від'ємні елементи числом 1000.
7. У масиві Z(25) знайти середнє арифметичне елементів масиву.
8. У масиві K(20) знайти суму парних елементів.
9. У масиві X(20) замінити від'ємні елементи їхніми модулями.
10. У масиві B(12) знайти кількість парних елементів.
11. У масиві K(24) знайти мінімальний елемент.
12. У масиві Y(14) знайти максимальний елемент

Контрольні питання:

1. Дайте визначення масиву.
2. Як описується масив у програмі?
3. Що таке розмірність масиву?
4. Як одержати доступ до певного елемента масиву?
5. Як у програмі організувати введення й вивід масиву?
6. Що таке індекс елементу?
7. Які операції дозволено над елементами масиву?

Додаток 1

Типи даних

Цілі типи

Назва	**Довжина, байт**	**Діапазон значень**	**Потужність типу**
Byte	1	От 0 до 255	256
ShortInt	1	От -128 до +128	256
Word	2	От 0 до 65 535	65536
Integer	2	От -32 768 до + 32 767	65536
LongInt	4	От -2 147 483 648 до + 2 147 483 647	4294497296

Дійсні типи

Назва	**Довжина, байт**	**Діапазон десяткового порядку**	**Кількість значущих цифр**
Single	4	-45…+38	7..8
Real	6	-39…+38	11..12
Double	8	-324…+308	14..16
Extended	10	-4951…+4932	19..20
Comp	8	$-2*10^{63}+1\ldots+2*10^{63}-1$	19..20

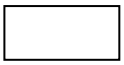
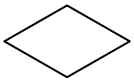
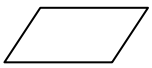
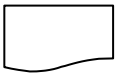
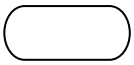
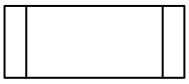
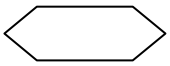
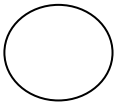
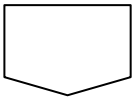
Додаток 2

Математичні функції, використовувані при складанні виражень:

Ім'я функції	**Математичне значення**	**Тип результату**
a mod b	Залишок від діления a на b	Ціле
a div b	Ціла частина ділення a на b	Ціле
abs(a)	|a|	Співпадає з типом аргументу
sqr(a)	a^2	Співпадає з типом аргументу
sqrt(a)	$\sqrt{a}$	Дійсне
sin(a), cos(a), arctan(a)	sin a, cos a, arctg a	Дійсне
ln(a)	ln a	Дійсне
exp(a)	e^x	Дійсне
ln(a)/ln(b)	$\log_b a$	Дійсне
exp(x*ln(a))	a^x	Дійсне
trunc	виділяє цілу частину дійсного аргументу шляхом відсікання дробової частини	Ціле
round	округляє аргумент до найближчого цілого	Ціле

Додаток 3

Умовні позначки, які використовуються при складанні блок-схем

Символ	Виконувана функція (пояснення)	Назва
1. Блок обчислень		Виконує обчислювальну дію або групу дій
2. Логічний блок		Вибір напрямку виконання алгоритму в залежності від умови
3. Блоки уведення/виводу		Введення або вивід даних не залежно від типу фізичного носія
		Вивід даних на друкувальний пристрій
4. Початок/кінець (вхід/вихід)		Початок або кінець програми, вхід або вихід у підпрограму
5. Визначений процес (підпрограма)		Обчислення по стандартній або користувальницькій підпрограмі
6. Блок модифікації (заголовок циклу)		Виконання дій, що змінюють пункти алгоритму
7. З'єднувач		Вказівка зв'язку між перерваними лініями у межах однієї сторінки
8. Міжсторинковий з'єднувач		Вказівка зв'язку між частинами схеми, розташованої на різних сторінках

Додаток 4

Повідомлення й коди помилок

Код помилки	Повідомлення про помилки періоду	Переклад з англійської мови	Пояснення
2	Identifier expected	не зазначений ідентифікатор	у цьому місці повинен перебуває ідентифікатор
3	Unknown identifier	невідомий ідентифікатор	цей ідентифікатор не був описаний
4	Duplicate identifier	подвійний ідентифікатор	спроба двічі описати той самий ідентифікатор.
5	Syntax error	синтаксична помилка	у вихідному тексті знайдений неприпустимий символ.
10	Unexpected end of file	не знайдений кінець файлу	причини цього повідомлення можуть бути наступні: - вихідний файл закінчився перед останнім End основного розділу операторів; імовірно в програмі неоднакова кількість операторів Begin і End; - не закінчений коментар.
11	Line too long	занадто довгий рядок	максимальна довжина рядка, яку може обробити компілятор, дорівнює 126 символам.
12	Type identifier expected	тут потрібний ідентифікатор типу	не зазначен тип ідентифікатора
16	Disk full	диск заповнений	потрібно видалити деякі файли або скористатися новим диском
20	Variable identifier expected	відсутній ідентифікатор змінної	на цьому місці повинен бути ідентифікатор змінної
21	Error in type	помилка в оголошенні типу	оголошення типу не може починатися із цього символу.

Продовження таблиці - Повідомлення й коди помилок

26	Type mismatch	невідповідність типу	це повідомлення може бути викликано наступними причинами: - несумісні типи змінної й виразу в операторі присвоєння; - тип виразу не поєднається з типом індексу при оголошенні масиву; - несумісні типи операндів у виразі.
33	Label identifier expected	потрібний ідентифікатор мітки	мітка не позначена за допомогою ідентифікатора, як це потрібно з контексту програми.
36	BEGIN expected	потрібен Begin	
37	END expected	потрібен End	
38	Integer expression expected	потрібен вираз типу Integer	
41	Operand types do not match operator	типи операндів не відповідають операції	дана операція не може бути застосована до зазначених операндів
42	Error in expression	помилка у виразі	даний символ не може брати участь у виразі зазначеним образом. Можливо, не зазначена операція між двома операндами.
50	DO expected	потрібен оператор Do	
57	THEN expected	потрібно Then	
58	TO or DOWNTO expected	потрібно To або Downto	
62	Division by zero	ділення на нуль	попередня операція намагається виконати ділення на нуль

Продовження таблиці - Повідомлення й коди помилок

64	Cannot Read or Writevariables of this type	немає можливості вважати або записати змінні даного типу	порушено наступні обмеження: -процедура Read може зчитувати змінні символьного, цілого, дійсного й строкового типів; -процедура Write може виводити змінні символьного, цілого, дійсного, логічного й строкового типів.
76	Constant out of range	константа порушує границі	можливі причини повідомлення: - спроба вказати індекс масиву, що виходить за його границі; - спроба привласнити змінній значення, що виходить за межі, припустимі для типу цієї змінної.
79	Integer or real expression expected	потрібен вираз дійсного або цілого типу	
81	Label already defined	мітка вже визначена	дана мітка вже позначає оператор.
85	«;» expected	потрібно вказати «;»	
97	Invalid FOR control variable	невірний параметр циклу оператора For	
98	Integer variable expected	потрібна змінна цілого типу	попередня змінна повинна мати цілий тип
103	Integer or real variable expected	потрібна змінна типу Integer або Real.	
113	Error in statement	помилка в операторі	даний символ не може бути першим символом в операторі
207	Invalid floating point operation	неприпустима операція із плаваючою комою	Можливі причини повідомлення: - від'ємний аргумент функції Sqrt; - аргумент функції Ln дорівнює нулю або має від'ємне значення.

СПИСОК ЛІТЕРАТУРИ

1. Боон К. Паскаль для всех: Пер. с гол. – М.: Энергоатомиздат, 1988. – 190 с.
2. Зуев Е.А. Язык программирования Turbo Pascal 6.0, 7.0 – М.:Веста, Радио и связь, 1993 – 384с.
3. Прайс Д. Программирование на языке Паскаль: Практическое руководство. Пер. с англ. – М.: Мир, 1987. – 232 с.
4. Меженный О.А. Turbo Pascal. Самоучитель.: М.: Издательский дом «Вильямс», 2006. – 336 с.
5. Фаронов В.В. Turbo Pascal: Учебное пособие. – СПб.: Питер, 2007. – 367 с.
6. Немнюгин С.А. Turbo Pascal. Практикум. 2-е изд. / - СПб.: Питер, 2007. – 268 с.
7. Йенсен К., Вирт Н. Паскаль. Руководство для пользователя. - М.: Финансы и статистика,1989.
8. Джонс Ж., Харроу К. Решение задач в системе Pascal . - М.: Финансы и статистика, 1991.