Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования
«МАТИ» - Российский государственный технологический университет им. К.Э. Циолковского

Кафедра «Высшая математика»

Начало программирования на Паскале. Задачи разветвляющей и линейной структуры

Методические указания к лабораторной работе по курсу «Информатика»

Составитель: Сидоров Б.Н.

Москва 2015

Методические указания предназначены для студентов, изучающих курс «Информатика»
Рассмотрены основные операторы языка Паскаль и их использование при решении задач. Даны задачи для самостоятельного решения.

1. Структура программы

Program имя программы;
uses имя стандартного модуля;
const имя=значение;
type имя типа = описание типа;
var имя переменной1, имя переменной2 : тип;
имя переменной: тип;
function имя функции(имя параметра :тип):тип функции;
раздел описаний функции
begin
операторы функции;
имя функции:=вычисленное значение;
end;
begin
операторы программы
end.

Пояснения.

5. Первая строка программы - заголовок; строка начинается словом ***Program,*** которое относится к зарезервированным словам т.е. к словам которые в языке имеют совершенно определённые значения (в данном случаи - присваивание программе имени) и которое нельзя употреблять для других целей (например, имени переменной); за словом ***Program*** следует *ИМЯ,* присваиваемое программе
6. Для разделения операторов используется точка с запятой ‘ ; ‘ т.е. ‘ ; ‘ставится после каждого оператора, за исключением ***begin*** и ***do***.
7. За заголовком программы может находится строка подсоединения модулей библиотек ***uses*** имя стандартного модуля; библиотеки используются когда программе требуются выполнить определённые в них действия, например процедуры очистки экрана CLRSCR нет в стандартном Паскале, поэтому перед её использованием необходимо подсоединить библиотеку , где она определена ***uses*** crt.
8. После следуют операторы описания; любая переменная и константа, используемая в программе должна быть описана; описание констант начинается с зарезервированного слова ***const ,*** затем идёт имя константы и её значение, например, ***const*** pi=3.14
9. Если в программе используется нестандартный тип данных (о них позже), то он досжен быть описан после зарезервированного слова ***type***
10. Описание всех переменных, используемых в программе (обязательно!) начинается зарезервированным словом ***var***

сокращением от variable ; далее перечисляются имена переменных одного и того же типа, разделенных запятыми, затем ставится двоеточие и указывается тип; после имени типа ставится точка с запятой.

Например ***var*** i: integer; - описывается переменная i целого типа

s:string; - описывается переменная s значением которой может быть строка

y,z :real - описывается переменные y и z :вещественного типа

11. После описаний переменных может идти описание процедур и функций, используемых в программе
12. После раздела описаний идёт выполняемая часть программы, которая заключается в операторные скобки ***begin end***.

2. Стандартные типы данных.

Тип данных определяет:

13.множество допустимых значений.

14.множество операций, возможных с данным типом.

Целый тип.

Значением целого типа является натуральное число со знаком в диапазоне -32768 - +32767

Описание: VAR имя : INTEGER;

Операции:

+(сложение); - (вычитание); * (умножение) div (деление нацело) mod (остаток от деления);

пример: 17 mod 5 =2; 17 div 5 = 3

Наряду с INTEGER; существуют другие целые типы, отличающиеся лишь диапазоном принимаемых значений:

Тип	Диапазон
byte	0 .. 255
shorint	-128 .. 127
word	0 .. 655353
longint	-2 миллиарда .. +2 миллиарда

Вещественный тип

Значением вещественного типа является действительное число(число с десятичной точкой). Вещественные значения могут изображаться в форме с фиксированной точкой (8.32; -4.1234 0.6435) или в показательной форме: *число***E***степень***;** где **E** заменяет 10 в математической записи числа; например:

Запись на Паскале	Математическая запись
8.32E+03	$8.32*10^{3}$
7.2E+00	$7.2*10^{0}$
1.3E-02	$1.3*10^{-3}$
2E10	$2*10^{10}$

Описание: VAR имя : REAL;

Операции:

+(сложение); - (вычитание); * (умножение); / (деление)

Функции

abs(x) возвращает модуль x

sqr(x) возвращает квадрат числа x
sqrt(x) возвращает значение корня квадратного из x
exp(x) возвращает e в степени x
ln(x) возвращает натуральный логарифм из x
sin(x) возвращает синус x
cos(x) возвращает косинус x
arctan(x) возвращает аргтангенс x
trunc(x) преобразует вещественное x в целое число путём отбрасывания дробной

Символьный тип

Функции

pred(ch)возвращает предыдущий символpred('E')='D'
succ(ch) возвращает следующий символ succ('E')='F'
chr(x) возвращает символ с кодом x chr(128)='Б'
ord(ch) возвращает код символа ch ord('A')=65

Строковый тип.

Значением этого типа являются строки любых символов, заключённые в одинарные кавычки, например 'Pascal', 'S= ', 'Светлана ' , 'Зри в корень '

Описание: VAR имя : STRING;

Операции: + (соединение), например 'Свет' + 'лана' = 'Светлана'

3 . Основные операторы языка Паскаль.

Оператор присваивания.

A:= выражение;

Работа

15. вычислить, если это необходимо значение выражения
16. вычисленное значение присваивается переменной A

Описать переменную это значит выделить под неё ячейку памяти;
Выполнить оператор присваивания - это значит положить в ячейку памяти с именем переменной вычисленное значение Var A: INTEGER.

Var A :Integer;	A:		{ячейка памяти, связанная с переменной A}
begin			
A:=3;	A:	3	{в ячейку памяти с именем A положить число 3}
A:=A+1	A:	4	{1)взять значение из ячейки с именем A и прибавить к

нему 1

2)поместить вычисленное значение в ячейку с именем A }

Пример. Поменять местами значения переменных a и b

Фрагмент программы:.

p:=a; a:=b; b:=p; {переменная p необходима для временного хранения значения переменной, которое, иначе, при выполнении оператора a:=b, было бы потеряно }

Операторы Ввода -Вывода.

Оператор ввода:

READLN (список вводимых переменных)

где : список переменных - последовательность имен переменных, разделённых запятыми

Например READLN(x,y,betta);

При выполнении оператора READLN программа останавливается и ждёт, пока нужное количество символов не будет введено с клавиатуры. Вводимые символы разделяются пробелами или нажатием на клавишу ENTER. Заканчивается ввод всегда клавишей ENTER

Оператор вывода

WRITE(список выражений) или WRITELN (список выражений)

Выражения в списке разделяются запятыми; выражением может быть

17. имя переменной; в этом случаи выводится её значение;
18. текстовой константой : ‘ что-то ‘ - в этом случаи вс1 что заключено в апострофы отображается на экране один к одному;
19. арифметическое выражение - в этом случаи на экран выводится его значение.

Вслед за выражением после двоеточия можно указать ширину поля экрана, в котором разместится выводимое значение; при выводе вещественных значений, вслед за этим полем можно указать сколько десятичных чисел следует сохранить в десятичной части числа.

Например, если x=3.5467; y=2.123456, то WRITELN (x:6:2, y:8:4) напечатает 3.54 2.1234

WRITELN (‘x=’,x:6:3, ‘y=’y:7:4) напечатает x=3.546 y=2.1234

WRITELN (‘ Ответ x=’,x:6:1, ’ м/сек‘ ,‘y=’y:7:2) напечатает Ответ x=3.5 м/сек y=2.12

если a=32; c=6, то WRITELN (‘ax+c=0’) напечатает ax+c=0

WRITELN (a:4,’x+ ‘,c:4,’ =0’) напечатает 32x+6=0

Оператор WRITELN отличается от оператора WRITE тем, что после вывода всех значений переводит курсор в начало следующей строки.

Составной оператор.

Составной оператор объединяет несколько операторов в один и представляет собой группу из произвольного числа операторов, отделённых друг от друга точкой с запятой, и ограниченную операторными скобками begin end; его вид:

begin оператор; оператор; . . . оператор; end

Составной оператор применяется, когда несколько операторов должны выполнятся подряд при некотором условии (в операторе IF) или внутри цикла.

Условный оператор.

Для изменения последовательности выполнения программ, то есть выполнения операторов в том порядке, в котором они расположены, применяют условный оператор.

Условный оператор позволяет выполнять операторы программ в зависимости от некоторого условия.

Вид оператора

IF (условие) **then** оператор когда - истина
Else оператор когда – ложь

Если условие истинно, то выполнится оператор когда истина, иначе – оператор когда ложь.

Условия бывают простыми и сложными. Простые используют операции сравнения: =, < >, >, <, >=, <=; сложные – это несколько простых соединенных логическими операциями(см. далее)

Пример 1. Определить четность/ нечетность целого числа x

```
If (x mod 2=0) then writeln ('число x четно')
               else  writeln ('число x нечетно')
```

mod – операция нахождения остатка при целочисленном делении x на 2, если остаток 0 – то число четно, иначе нечетно.

Замечание:

1. перед else ; не ставится

2.Условный оператор может быть записан и в сокращенном виде:

IF ( ) then оператор когда истина в этом случае оператор выполняется только когда условие истинно, иначе пропускается.

Пример 2. Заменить переменную x её модулем

If x<0 **then** x:=-x

Если же x положительно, то никакого действия выполнять не надо, поэтому else отсутствует.

3. Если по исполнению или не исполнению, условие должно выполнить более одного оператора, то использую составной оператор.

```
Begin
    опер – 1
    опер – 2
    …….
    опер – n
end;
```

Пример 3. вычислить x = min (x,y)
То есть так перераспределить переменные, что бы x оказалось меньше из двух значений x или y

```
If x>y then
        Begin
            p:=x;
            x:=y;
            y:=p;
end;
```

Задача 1:

Определить стоимость телефонного разговора. Если время разговора превышает 10 минут, то к каждой последующей минуте причислить скидку 25% от тарифа.

```
Const tariff = 4; skidka = 25;
Var t, st : real;
Begin
        Writeln('программа определяет стоимость телефонного разговора')ж
        Write('Введите время разговора ->');
  Readln(t);
     If t <=10 then st:=t* tariff else
        St:=10* tariff + (t – 10) * tariff * skidka/100
  Writeln('стоимость = ', st : 6 : 2 );
  Readln
End.
```

Логический тип

Результатом операции сравнения является одно из двух значений TRUE (истина) или FALSE (ложь)/
TRUE и FALSE – это константы логического типа
Переменная, которая может принимать значение этих констант, TRUE и FALSE, называется логической.

Логическая переменная описывается предложением

Var имя : boolean

Значение логической переменной можно присваивать

A:= TRUE ; A:=2<3; A:=x<3,

но нельзя вводить.

Из переменных, констант, сравнений и логических операций строятся сложные условия, которые являются логическими выражениями.

Логические операции

2. **Логические операции not.**

Операция определяется одной переменной, инвертирую её значения (значение истину на ложь или наоборот)

Not истина = ложь
Not ложь = истина

3. **Логические умножения** – конъюнкции – определяются над двумя переменными. Возвращает истину только в том случае если обе переменные истины.

(ложь) and (ложь) = ложь
(ложь) and (истина) = ложь
(истина) and (ложь) = ложь
(истина) and (истина) = истина

Иначе можно представить в виде таблицы истинности – показав значение логической функции, на всех упорядоченных значениях логической переменной, если таких переменных 2 – то значений 4, если n – то 2^n

A	B	A and B	A or B	A xor n
ложь	ложь	ложь	ложь	ложь
ложь	истина	ложь	истина	истина
истина	ложь	ложь	истина	истина
истина	истина	истина	истина	ложь

3. **Логическое сложение OR** – дизъюнкция – определяется над двумя переменными. Возвращает истину, если хотя бы одна из них является истиной.

4. **Исключающий или XOR** – определяется над двумя переменными. Возвращает истину, тогда, когда только одна переменная истина.

Вложенные условия.

Вложенные условия образуются тогда, когда в случае сложения (или несложения) первого условие проверяется второе, то есть внутри оператора if используется еще один if .
Глубина вложенности неограниченна, но использование более двух if затрудняет чтение программы и поэтому его следует избегать. Альтернативой вложенным if является использование сложных условий.

Пример 4.

Записать условие, что x принадлежит множеству x э [-2;3)

4. используем сложное условие:
   if (x >= -2) and (x<3) then writeln (‘принадлежит’)
   else writeln(‘не принадлежит’)ж

б) используем вложенный оператор:
   if (x >=-2) then
   if x<3 then writeln(‘ принадлежит’) else writeln(‘не принадлежит’)
   else writeln(‘не принадлежит’)

первое else относится к вложенному if то есть выполнится тогда когда x>=3, а второе else относится к первому условно, то есть выполняется тогда, когда x<-2.

Пример 5:

Записать условие, что x принадлежит (-∞; -2] U (3; ∞)

5. используя сложное условие if (x<=-2) or (x>3) then
   writeln(‘принадлежит’)
   else writeln (‘не принадлежит’)

6. используя вложенный оператор if (x<=-2) then
   writeln(‘принадлежит’)
   else if (x>3) then writeln(‘принадлежит’)
   else writeln (‘не принадлежит’);

   В отличии от первого примера здесь второй оператор выполняет когда первое условие ложь, то есть по else. Из примеров может показаться, что использование сложных условий предпочтительнее, однако это справедливо когда у нас есть 2

альтернативы(принадлежит или нет) или более трех. Когда же есть только 3 то использование вложений if оправдывает себя.

Пример 6:

Записать решение уравнения ах+в=0

Здесь как раз случай с тремя возможными альтернативами

7. уравнение имеет единственное решение а≠0
8. уравнение не имеет решения а=0 в≠0
9. уравнение имеет бесконечно много решений а=0 в=0

а) используем сложное условие

```
IF (a < > 0) then writeln('x=', -в/a);
IF (a=0) and (в< > 0) then writeln('уравнение не имеет решения');
IF (a=0) and (в= 0) then writeln('уравнение имеет бесконечно много решений');
```

б) используя вложение IF получим несколько короче

```
IF (a < > 0) then writeln('x=', -в/a)
  Else
    If (в< > 0) then writeln('уравнение не имеет решения')
          Else writeln('уравнение имеет бесконечно много решений');
```

Пример 7:

Вычислить значение функции.

$$У = \begin{cases} X+5, & \text{если } x<=-5 \\ -X-5, & \text{если } -5< x \le 0 \\ 5-X^2, & \text{если } x>0 \end{cases}$$

Задача похожа на предыдущую, так как в ней есть только 3 альтернативы, то для её решения используем вложение IF

```
……
IF (x<=-5) then y:=x+5
```

Else

If ($x <= 0$) then $y := -x - 5$

Else $y := 5 - x^2$

Задача 2: Написать программу определяющую по двум введенным углам треугольника, заданных в градусах, существует ли он, и если существует, то его вид.

Здесь если треугольник существует, то возникает 3 возможности; которые мы последовательно проверим используя 3 оператора IF со сложным условием

```
   Var A, B : real;
     Begin
          Writeln('программа определяет вид треугольника по двум
введенным углам')
          Writeln('введите углы треугольника');
             Write('A='');       Readln(A);
             Write('B='');        Readln(B);
          If (A<0) or (B<0) or (A+B >=180) then
                   Writeln(' не существует')
             Else
                 Begin
                   If(a +b =90) then writeln('прямоугольный');
                   If(a<90) and(b<90)  and (a+b >90)
writeln('остроугольный');
                   If(a>90) and(b>90)  and (a+b <90)
writeln('тупоугольный');
          End ;{else}
     Readln;
End.
```

Задачи для самостоятельного решения:

1. По двум углам треугольника определить, существует ли он и является ли он прямоугольным.
2. По двум углам треугольника определить, существует ли он и является ли он остроугольным.
3. По двум углам треугольника определить, существует ли он и является ли он тупоугольным.
4. По трем сторонам треугольника определить, существует ли он и является ли он прямоугольным.
5. По трем сторонам треугольника определить, существует ли он и является ли он остроугольным.
6. По трем сторонам треугольника определить, существует ли он и является ли он тупоугольным.
7. Написать программу, решающую уравнение x*x=-a
8. Написать программу, решающую уравнение x*x=a
9. Написать программу, решающую уравнение |x|=-a
10. Написать программу, решающую уравнение |x|=-a
11. Написать программу, решающую неравенство a*x>0
12. Написать программу, решающую неравенство a*x<0