УВАЖАЕМЫЕ СТУДЕНТЫ!

ВАМ НЕОБХОДИМО ВЫПОЛНИТЬ СЛЕДУЮЩЕЕ:

1. Ознакомиться с теорией и законспектировать не менее трех страниц.
2. Запишите все задачи из лекции.
3. Составить и ответить на вопросы.
4. Предоставит фото отчет в течении трех дней .
5. Отправить преподавателю на почту **v.vika2014@mail.ru** и указать свою Ф.И.О, группу, и название дисциплины тел 0721744922

Тема: Работа с разветвляющимися алгоритмами.

Цель: Ознакомится с разветвляющимися алгоритмами

Разветвляющиеся алгоритмы

До сих пор Вы использовали *линейные* алгоритмы, т.е. алгоритмы, в которых все этапы решения задачи выполняются строго последовательно. Сегодня Вы познакомитесь с разветвляющимися алгоритмами.

Определение. *Разветвляющимся* называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса. Каждый подобный путь называется *ветвью алгоритма.*

Признаком разветвляющегося алгоритма является наличие операций проверки условия. Различают два вида условий – *простые* и *составные*.

Простым условием (отношением) называется выражение, составленное из двух арифметических выражений или двух текстовых величин (иначе их еще называют *операндами*), связанных одним из знаков:

< - меньше, чем...

> - больше, чем...

<= - меньше, чем... или равно

>= - больше, чем... или равно

<> - не равно

= - равно

Например, простыми отношениями являются следующие:

x-y>10; k<=sqr(c)+abs(a+b); 9<>11; ‘мама’<>‘папа’.

В приведенных примерах первые два отношения включают в себя переменные, поэтому о верности этих отношений можно судить только при подстановке некоторых значений:

если x=25, y=3, то отношение x-y>10 будет *верным*, т.к. 25-3>10

если x=5, y=30, то отношение x-y>10 будет неверным, т.к. 5-30<10

Проверьте верность второго отношения при подстановке следующих значений:

a) k=5, a=1, b=-3, c=-8

b) k=65, a=10, b=-3, c=2

Определение. Выражения, при подстановке в которые некоторых значений переменных, о нем можно сказать истинно (верно) оно или ложно (неверно) называются *булевыми* (логическими) выражениями.

Примечание. Название “булевые” произошло от имени математика Джорджа Буля, разработавшего в XIX веке булевую логику и алгебру логики.

Определение. Переменная, которая может принимать одно из двух значений: True (правда) или False (ложь), называется булевой (логической) переменной. Например,

K:=True;

Flag:=False;

Second:=a+sqr(x)>t

Рассмотрим пример.

Задача. Вычислить значение модуля и квадратного корня из выражения (x-y).

Для решения этой задачи нужны уже знакомые нам стандартные функции нахождения квадратного корня - Sqr и модуля - Abs. Поэтому Вы уже можете записать следующие операторы присваивания:

```
Koren:=Sqrt(x-y);
Modul:=Abs(x-y).
```

В этом случае программа будет иметь вид:

```
Program Znachenia;
Uses
  Crt;
Var
  x, y : integer;
  Koren, Modul : real;
Begin
  ClrScr;
  write (‘Введите значения переменных x и y через пробел ‘);
  read (x, y);
  Koren:=Sqrt(x-y);
  Modul:=Abs(x-y).
  write (‘Значение квадратного корня из выражения (x-y) равно ‘);
  write (‘Значение модуля выражения (x-y) равно ‘);
  readln;
End.
```

Казалось бы задача решена. Но мы не учли области допустимых значений для нахождения квадратного корня и модуля. Из курса математики Вы должны знать, что можно найти модуль любого числа, а вот значение подкоренного выражения должно быть *неотрицательно* (больше или равно нулю).

Поэтому наша программа имеет свою допустимую область исходных данных. Найдем эту область. Для этого запишем неравенство x-y>=0 и решив его получим x>=y. Значит, если пользователем нашей программы будут

введены такие числа, что при подстановке значение этого неравенства будет равно True, то квадратный корень из выражения (x-y) извлечь можно. А если значение неравенства будет равно False, то выполнение программы закончится аварийно.

Задание. Наберите текст программы. Протестируйте программу со следующими значениями переменных и сделайте вывод.

а) x=23, y=5; б) x=-5, y=15; в) x=8, y=8.

Каждая программа, насколько это возможно, должна осуществлять контроль за допустимостью величин, участвующих в вычислениях. Здесь мы сталкиваемся с разветвлением нашего алгоритма в зависимости от условия. Для реализации таких условных переходов в языке Паскаль используют операторы If и Else, а также оператор безусловного перехода Goto.

Рассмотрим оператор If.

Для нашей задачи нужно выполить следующий алгоритм:

если x>=y,

то вычислить значение квадратного корня,

иначе выдать на экран сообщение об ошибочном введении данных.

Запишем его с помощью оператора If. Это будет выглядеть так.

```
if x>=y
  then
     Koren:=Sqr(x-y)
  else
    write (‘Введены недопустимые значения переменных‘);
```

Теперь в зависимости от введенных значений переменных x и y, условия могут выполняться или не выполняться.

В общем случае полная форма конструкции условного оператора имеет вид:

```
if  <логическое выражение>
 then
    <оператор 1>
 else
    <оператор 2>
```

Условный оператор работает по следующему алгоритму.

Сначала вычисляется значение логического выражения, расположенного за служебным словом IF. Если его результат ***истина***, выполняется <оператор 1>, расположенный после слова THEN, а действия после ELSE пропускаются; если результат ***ложь***, то, наоборот, действия после слова THEN пропускаются, а после ELSE выполняется <оператор 2>.

Управляющая структура if может показаться негибкой, так как выполняемые действия могут быть описаны только одним оператором. Иногда может потребоваться выполнение последовательности операторов. В этом случае хотелось бы заключить всю последовательность в воображаемые скобки. В Паскале предусмотрен этот случай.

Если в качестве оператора должна выполниться серия операторов, то они заключаются в операторные скобки begin-end. Конструкция *Begin ... End* называется *составным* оператором.

```
if  <логическое выражение>
  then
     begin
       оператор 1;
       оператор 2;
       ...
     end
  else
     begin
       оператор 1;
       оператор 2;
       ...
     end;
```

<u>Определение</u>. *Составной оператор* - объединение нескольких операторов в одну группу. Группа операторов внутри составного оператора заключается в операторные скобки (begin-end).

```
begin
     оператор 1;
     оператор 2;
end;
```

С учетом полученных знаний преобразуем нашу программу.

```
        Program Znachenia;
        Uses
          Crt;
Var
  x, y : integer;
  Koren, Modul : real;
Begin
  ClrScr;
  write (‘Введите значения переменных x и y через пробел ‘);
  read (x, y);
  if x>=y
    then
      begin
        Koren:=Sqr(x-y)
        Modul:=Abs(x-y)
        write (‘Значение квадратного корня из выражения (x-y) равно ‘);
        write (‘Значение модуля выражения (x-y) равно ‘);
      end
    else
      write (‘Введены недопустимые значения переменных‘);
  readln;
End.
```

Составным оператором является и такой оператор

```
begin
  S:=0;
end.
```

Символ “;” в данном случае разделяет оператор присваивания S:=0 и *пустой* оператор.

Пустой оператор не влечет никаких действий и в записи программы никак не обозначается.

Например, составной оператор

```
begin
end.
```

включает лишь один пустой оператор.

Если Вы обратили внимание, программа на языке Паскаль всегда содержит один составной оператор – раздел операторов программы.

Внимание! Перед служебным словом *Else* разделитель (точка с запятой) не ставится.

Отметим, что большинство операторов в программах на языке Паскаль заканчиваются точкой с запятой, но после некоторых операторов точка с запятой не ставится. Сформулируем *общие правила употребления точки с запятой*:

1. Каждое описание переменной и определение константы заканчиваются точкой с запятой.

2. Каждый оператор в теле программы завершается точкой с запятой, если сразу за ним не следуют зарезервированные слова *End, Else, Until*.

3. После определенных зарезервированных слов, таких, как *Then, Else, Var, Const, Begin,* никогда не ставится точка с запятой.

Рассмотрим еще один пример.

Задача. Вывести на экран большее из двух данных чисел.

```
Program Example1;
Var
   x, y : integer; {вводимые числа}
Begin
   writeln('Введите 2 числа '); {вводим два целых числа через пробел}
   readln(x,y);
   if x>y
     then
        writeln (x)  {если x больше y, то выводим x}
     else
        writeln (y)  {иначе выводим y}
    readln;
End.
```

Можно также использовать и *сокращенную (неполную)* форму записи условного оператора. Эта форма используется тогда, когда в случае невыполнения условия ничего делать не надо.

Неполная форма условного оператора имеет следующий вид.

```
if  <логическое выражение>
  then
     <оператор>
```

Тогда если выражение, расположенное за служебным словом IF. в результате дает *истину,* выполняются действия после слова THEN, в противном случае эти действия пропускаются.

<u>Задача</u>. Составить программу, которая, если введенное число отрицательное меняет его на противоположное.

```
Program Chisla;
Var
   x : integer; {вводимое число}
Begin
   writeln('Введите число '); {вводим целое число}
   readln(x);
   if x<0
     then
         x:=-x;
    writeln (x);
    readln;
End.
```

Выберите из предложенного ниже списка задачи для самостоятельного решения.

1. Если целое число M делится нацело на целое число N, то вывести на экран частное от деления, в противном случае вывести сообщение M на N нацело не делится.

2. Запишите условный оператор, в котором значение переменной c вычисляется по формуле a+b, если a – нечетное и a*b, если a – четное.

3. Вычислить значение функции:

$$y=\begin{cases} x^2+5, x>3 \\ x-8, x\le 3 \end{cases}$$

4. Написать программу для подсчета суммы только положительных из трех данных чисел.

5. Даны три числа. Написать программу для подсчета количества чисел, равных нулю.

6. Напишите программу, упростив следующий фрагмент программы:

```
if a>b then c:=1;
if a>b then d:=2;
if a<=b then c:=3;
```

```
if a<=b then d:=4.
```

7. Каким будет значение переменной a после выполнения операторов:

```
a:=3;
if a<4
  then
     begin
       Inc(a,2);
            Inc(a,3);
     end;
```

8. Найти количество положительных (отрицательных) чисел среди четырех целых чисел A, B, C, D.

9. Составьте программу, которая уменьшает первое введенное число в пять раз, если оно больше второго введенного числа по абсолютной величине.

10. Для данного значения X вычислить значение функции, которая определяется следующим образом:

Y=sin(x), если x >=1

Y=cos(x), если x < 1

11. Определить является ли введённое число чётным.

12. Компьютер спрашивает: "Что сегодня нужно всем?" и если получает ответ ЭВМ, то пишет "Ну, конечно ЭВМ!", иначе "Это тоже нужно всем, но нужнее ЭВМ!"

13. Написать программу, по которой компьютер приветствовал бы только своего хозяина, а при попытке ввести какое-либо другое имя спрашивал бы: "А где (например) Вася?"

14. Написать программу, определяющую, есть ли в введенном числе дробная часть.

15. Написать программу, рисующую круг в случае введения пользователем числа 1 и квадрат во всех других случаях.