## Практическое занятие

## Тема: Среда программирования.

**Цель работы:** Изучить различные типы данных языка Pascal и оператор присваивания, научится создавать простейшие программы для реализации линейного алгоритма.

***Теоретическая часть:***

### Операторы языка Паскаль

*Оператором* называется предложение языка программирования, задающее полное описание некоторого действия, которое необходимо выполнить. Основная часть программы на языке Турбо Паскаль представляет собой последовательность операторов. Разделителем операторов служит точка с запятой. Все операторы языка Турбо Паскаль можно разделить на две группы: простые и структурные.

Операторы, не содержащие никаких других операторов, называются *простыми*. К ним относятся операторы присваивания, безусловного перехода, вызова процедуры и пустой оператор.

### Оператор присваивания

Оператор присваивания **(:=)** предписывает выполнить выражение, заданное в его правой части, и присвоить результат переменной, идентификатор которой расположен в левой части. Переменная и выражение должны быть совместимы по типу.

Оператор присваивания выполняется следующим образом: сначала вычисляется выражение в правой части присваивания, а затем его значение присваивается переменной, указанной в левой части оператора.

Например, для оператора

*Rezult:=A div B;*

сначала выполняется целочисленное деление значения переменной *A* на значение переменной *B*, а затем результат присваивается переменной *Rezult*.

Примеры применения оператора присваивания:

*A:= 8;*
*S:= A * B;*
*Ostatok := A mod B;*
*Ratio := A / B;*

### Оператор безусловного перехода (go to)

Оператор безусловного перехода *(go to)* означает "перейти к" и применяется в случаях, когда после выполнения некоторого оператора надо выполнить не следующий по порядку, а какой-либо другой, отмеченный меткой оператор.

Напомним, что метка объявляется в разделе описания меток и может содержать как цифровые, так и буквенные символы.

При использовании оператора *go to* необходимо помнить, что *областью действия метки* является только тот блок, в котором она описана. Передача управления в другой блок запрещена.

*Правила использования оператора безусловного перехода.* Использование безусловных передач управления в программе считается теоретически избыточным и подвергается серьезной критике, так как способствует созданию малопонятных и трудно модифицируемых программ, которые вызывают большие сложности при отладке и сопровождении. Поэтому рекомендуется минимальное использование оператора *go to* с соблюдением следующих правил:

• следует стремиться применять операторы перехода (если кажется невозможным обойтись без них) для передачи управления только вниз (вперед) по тексту программы; при необходимости передачи управления назад следует использовать операторы цикла;

• расстояние между меткой и оператором перехода на нее не должно превышать одной страницы текста (или высоты экрана дисплея).

Пример применения оператора безусловного перехода:

```
…
label Metka;      {в раздале описания меток описали метку с именем Metka}
…
begin             {основная программа}
{операторы основной программы}
…
go to Metka;
Metka:
{операторы основной программы помеченные меткой}
end.
```

**Условные операторы**

Условные операторы предназначены для выбора к исполнению одного из возможных действий (операторов) в зависимости от некоторого условия (при этом одно из действий может быть пустым, т. е. отсутствовать). В качестве условий выбора используется значение логического выражения.

В Турбо Паскале имеются два условных оператора: *if и case.*

**Оператор условия if**

Оператор условия *if* является одним из самых популярных средств, изменяющих естественный порядок выполнения операторов программы.

Он может принимать одну из следующих форм:

- *if <условие> then <оператор1>*
  *else <оператор2>;*
- *if <условие> then <оператор>;*

В переводе с английского языка данные форматы можно определить как:

- ЕСЛИ<*условие*>ТО<*оператор1*>ИНАЧЕ<*оператор2*>
- ЕСЛИ<*условие*>ТО<*оператор*>

Оператор условия *if* выполняется следующим образом. Сначала вычисляется выражение, записанное в условии. В результате его вычисления получается значение булевского типа.

В первом случае, если значение выражения есть *True* (истина), выполняется *<оператор1>,* указанный после слова *then* (в переводе –"то"). Если результат вычисления выражения в условии есть *False* (ложь), то выполняется *<оператор2>*.

Во втором случае, если результат выражения *True*, выполняется *<оператор>,* если *False* - оператор, следующий сразу за оператором *if*. Операторы *if* могут быть *вложенными*.

Пример фрагмента программы с оператором условия if:

```
…
Read(Ch) ;
if Ch='N' then Parol:= True
  else Parol:= False;
Read(X) ;
if Parol = True then
  if X = 100 then Write('Пароль и код правильные')
  else
  begin
  Writeln('Ошибка в коде');
  Halt(1)
  end;
```

…

В данном примере с клавиатуры считывается значение переменной символьного типа *Ch*. Затем проверяется условие *Ch='N'*. Если оно выполняется, то переменной *Parol* булевского типа присваивается значение *True*, если условие не выполняется, *False*. Затем с клавиатуры считывается значение кода *X*. Далее оператор if проверяет условие Parol = True. Если оно имеет значение *True*, то выполняется проверка введенного пароля оператором *if X=100*. Если условие *X=100* имеет значение *True*, то выводится сообщение "Пароль и код правильные", и управление в программе передается на оператор, следующий за словом *end*, если оно имеет значение *False*, выполняется составной оператор, стоящий после, слова *else*, который выводит на экран видеомонитора сообщение "Ошибка в коде", и вызывает стандартную процедуру *Halt(1)* для остановки программы.

*Особенность применения оператора if*. При использовании вложенных условных операторов может возникнуть синтаксическая неоднозначность, например:

*if условие1 then if условие2 then <оператор1> else <оператор2>*

Возникающая двусмысленность, к какому оператору *if* принадлежит часть *else <оператор2>,* разрешается тем, что служебное слово *else* всегда ассоциируется (связывается) с ближайшим по тексту служебным словом *if,* которое еще не связано со служебным словом *else*.

В связи с этим следует проявлять аккуратность при записи вложенных операторов условия.

***Пример 1.*** Составить программу, которая вычисляет частное двух целых чисел. В связи с тем, что делить на нуль нельзя, организовать контроль ввода данных.

Для контроля вводимых значений делителя используем оператор условного перехода *if ... then ... else.*

Текст программы может выглядеть следующим образом:

```
program Primer1;
var
A, B : integer;
Rezult: real;
Begin
Write('Введите значение делимого A: ');
        Read(A) ;
        Write('Введите значение делителя B: ');
        Read(B) ;
        if B=0          {Контроль ввода числа B}
then Writeln('На нуль делить нельзя')      {Условие выполнено}
else
{Условие не выполнено}
        begin
        Rezult := A / B;
Writeln('Частное чисел ',A,' и ' ,B, ' = ', Rezult);
        end;
      end.
```

7 Любая переменная имеет имя – идентификатор. По правилам языка Паскаль имя переменной должно начинаться с буквы и может содержать буквы (только латинские), цифры и знак подчеркивания. Длина имени – до 126 символов.

8 Арифметические действия и выражения в Паскале:

– «+» - сложение;

– «-» - вычитание;

– «*» - умножение;

– «/» - деление;

– mod - нахождение остатка от деления;

– div - деление нацело (находить остаток от деления и делить нацело можно только целые числа);

– для указания порядка действий используются только круглые скобки, их может быть несколько, главное, чтобы количество открывающихся скобок равнялось количеству закрывающихся;

– sqr(x) – возведение аргумента в квадрат;

– sqrt(x) – извлечение квадратного корня;

– abs(x) – модуль.

9 Общий вид оператора присваивания: <Имя_переменной>:=<арифметическое выражение>.

10 Для типов переменной слева и арифметического выражения справа от знака присваивания выполняются правила:

- если переменная вещественного типа, то арифметическое выражение может быть как целого, так и вещественного типа, выражение преобразуется к вещественному типу;

- если переменная слева целого типа, то арифметическое выражение только целочисленное.

**Задание**

1 Составить и записать алгоритм решения задачи в графическом и словесно-формульном виде

2 Написать программу на языке Паскаль

3 Ввести программу и запустить её на исполнение.

4 Отладить программу (найти и исправить возможные ошибки).

5 Протестировать алгоритм с различными данными.

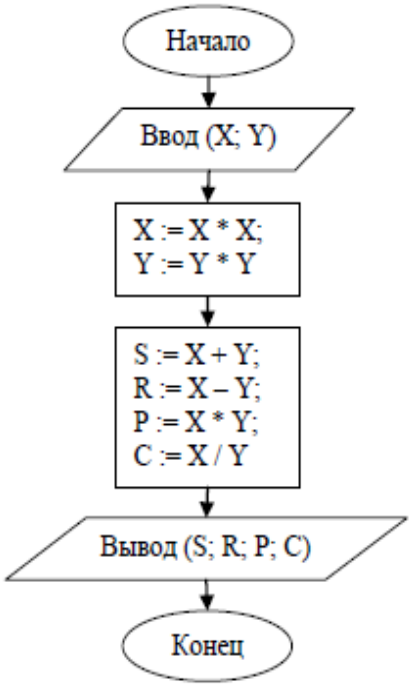
**Пример задачи.**

**Исходные данные:**

Даны два ненулевых числа. Найти сумму, разность, произведение и частное их квадратов

**Решение:**

1 Алгоритм в графическом и словесном виде:



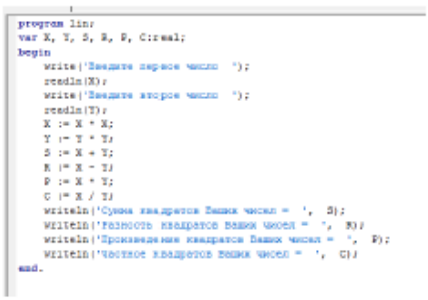

1. Начало;
2. Ввод (X; Y);
3. X := X * X;
4. Y := Y * Y;
5. S := X + Y;
6. R := X – Y;
7. P := X * Y;
8. C := X / Y;
9. Вывод (S; R; P; C);
10. Конец.

2 Программа на языке Паскаль:

```
program lin;
var X, Y, S, R, P, C:real;
begin
   write('Введите первое число  ');
   readln(X);
   write('Введите второе число  ');
   readln(Y);
   X := X * X;
   Y := Y * Y;
   S := X + Y;
   R := X – Y;
   P := X * Y;
```

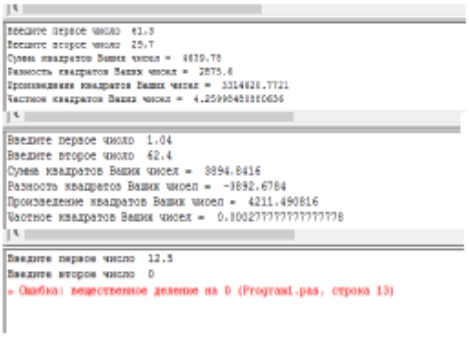
```
C := X / Y;
writeln('Сумма квадратов Ваших чисел = ', S);
writeln('Разность квадратов Ваших чисел = ', R);
writeln('Произведение квадратов Ваших чисел = ', P);
writeln('Частное квадратов Ваших чисел = ', C);
end.
```

3 Программный код для запуска




4 Ошибок в программе нет

5 Тестирование программы с различными данными:




6 Вывод: Программа, осуществляющая линейный алгоритм, работает правильно, но для вычисления частного двух чисел необходима проверка второго на равенство нулю. Следовательно, для корректной работы программы надо изменить тип алгоритмической структуры.

## Оператор выбора case

Если один оператор *if* может обеспечить выбор из двух альтернатив, то оператор выбора *case* позволяет сделать выбор из произвольного числа имеющихся вариантов. Он состоит из выражения, называемого *селектором* (*selection — выбор альтернативы*), и *списка параметров*, каждому из которых предшествует список *констант выбора* (список может состоять и из одной константы).

Формат записи оператора case:

*case <выражение-селектор> of*
*<список1>: <оператор1; >*
*<список2>: <оператор2; >*
*...*
*<списокN>: <операторN>*
*else <оператор>*
*end;*

Оператор *case* работает следующим образом. Сначала вычисляется значение выражения-селектора, затем обеспечивается реализация того оператора, константа выбора которого равна текущему значению селектора. Если ни одна из констант не равна текущему значению селектора, выполняется оператор, стоящий за словом *else*. Если слово *else*

отсутствует, активизируется оператор, находящийся за словом *end*, т. е. первый оператор за границей *case*.

Селектор должен относиться к одному из целочисленных типов (находящихся в диапазоне — 32768..32767): булевскому, литерному или пользовательскому. Список констант выбора состоит из произвольного количества значений, или диапазонов, отделенных друг от друга запятыми. Границы диапазона записываются двумя константами через разграничитель "..". Тип констант в любом случае должен совпадать с типом селектора. В синтаксическом описании, приведенном выше, предполагается использование одного оператора для каждой альтернативы, но при необходимости можно задать несколько операторов, сгруппировав их в составной оператор. В то же время ветвь *else* допускает использование последовательности операторов, разделенных символом ";".

*Правила использования оператора case.* При использовании оператора выбора *case* должны выполняться следующие правила:

1. Значения выражения "переключателя", записанного после служебного слова *case*, должны принадлежать дискретному типу (лат. discretus — прерывистый, дробный, состоящий из отдельных частей); для целого типа они должны лежать в диапазоне *integer*.

2. Все константы, предшествующие операторам альтернатив, должны иметь тип, совместимый с типом выражения.

3. Все константы в альтернативах должны быть уникальны в пределах оператора варианта (т. е. повторения констант в альтернативах не допускаются); диапазоны не должны пересекаться и не должны содержать констант, указанных в данной или других альтернативах.

Формы записи оператора case.

Селектор интервального типа:

```
case I of
1..10 : Writeln  ('число ', I: 4, ' в диапазоне 1- 10');
11.. 20 : Writeln  ('число ', I:4, ' в диапазоне 11-20');
21.. 30 : Writein  ('число', I:4, ' в диапазоне 21-30')
else Writeln  ('число ', I:4, ' вне пределов контроля');
 end;
```

Селектор целочисленного типа:

```
case I of
  1 : Z := I + 10;
  2 : Z := I + 100;
  3 : Z := I + 1000;
end;
```

Селектор перечисляемого пользовательского типа:

```
var
    Season: (Winter, Spring, Summer, Autumn) ;
begin
case Season of
  Winter:  Writeln('Winter');
  Spring:  Writeln('Spring');
  Suauner:  Writeln  (' Summer') ;
  Autumn:  Writeln('Autumn')
      end; {конец case}
end;
```

***Пример 2.*** Составить программу, которая по введенному пользователем номеру дня недели выводит на экран его название.

```
program Day_Week;
var Day : byte;
begin
```

```
  Write ('Введите номер дня недели: ');
  Readln(Day) ;
  case Day of  {Вычисление значения селектора и выбор}
  1: Writeln('Понедельник') ;
  2: Writeln('Вторник') ;
  3: Writeln('Среда');
  4: Writeln('Четверг');
  5: Writeln<'Пятница');
  6: Writeln('Суббота' ) ;
  else
  Writeln('Воскресенье');
end;
```

end.

В данном примере на экран выводится приглашение: 'Введите номер дня недели:', с клавиатуры считывается целочисленное значение дня недели и присваивается переменной *Day*. Затем в зависимости от значения селектора DAY обеспечивается реализация того оператора, константа выбора которого равна текущему значению селектора. Например, если значение *Day* равно 3, то реализуется оператор *Writeln('Среда')*. Если значение *Day* равно 7, а ни одна из констант не равна этому значению селектора, то выполняется оператор, стоящий за словом *else* (на экран выводится текст *'Воскресенье'*). Если слово *else* отсутствует, активизируется оператор, находящийся за словом *end*, т. е. первый оператор за границей *case*. Если значение *Day* не равно значению ни одной из констант выбора *(например, Day=8 или Day=0),* то активизируется оператор, находящийся за словом *end*, т. е. первый оператор за границей *case* - оператор *end*.

**Оператор повтора while**

Оператор *while (пока)* часто называют *оператором цикла с предусловием* за то, что проверка условия выполнения тела цикла производится в самом начале оператора.

Формат записи:

*while <условие продолжения повторений> do*
*<тело цикла>;*

*Условие* - булевское выражение, *тело цикла* - простой или составной оператор.

Перед каждым выполнением тела цикла вычисляется значение выражения условия. Если результат равен *True*, тело цикла выполняется и снова вычисляется выражение условия. Если результат равен *False*, происходят выход из цикла и переход к первому после *while* оператору.

**Пример программы с использованием оператора повтора while**

Программа *DemoWhile* производит суммирование 10 произвольно введенных целых чисел.

```
program DemoWhile;
const
Limit  =10;  {Ограничение на количество вводимых чисел}
var Count, Item, Sum: integer;
begin
  Count:=0;    {Счетчик чисел}
  Sum:= 0;     {Сумма чисел}
while  (Count < Limit)  do  {Условие выполнения цикла}
                  begin
                       Count:=  Count+1;
  Write('Введите ', Count, ' - е целое число: ');
  Readln(Item);{Ввод  очередного числа с клавиатуры}
  Sum:=  Sum+Item;
                  end;
```

```
Writeln('Сумма  введенных чисел равна ', Sum) ;
end.
```

В данном примере в разделе описания констант описана константа *Limit=10,* задающая ограничение на количество вводимых чисел. В разделе описания переменных описаны переменные *Count, Item, Sum* целочисленного типа. В начале выполнения программы обнуляются значения счетчика введенных чисел *Count* и их суммы *Sum*. Затем выполняются цикл ввода 10 чисел и их суммирование. Вначале оператор *условия while* проверяет условие *Count* < *Limit*. Если условие верно, то выполняется составной оператор в теле цикла:

```
begin
  Count:= Count+1;
  Write('Введите ', Count, '-е целое число: ');
  Readln(Item) ;
  Sum:= Sum+Item;
End;
```

в котором вводится значение очередного числа, и на это значение увеличивается значение суммы. После этого управление в программе вновь передается оператору цикла *while*, опять проверяется условие *Count* < *Limit*. Если условие верно, то выполняется составной оператор и т. д., пока значение переменной *Count* будет меньше 10. Как только значение *Count* станет равно 10 и условие *Count* < *Limit* не будет соблюдено, выполнение цикла завершится, а управление в программе будет передано на оператор, находящийся за словом *end*, т. е. первый оператор за границей *while.* Это вызов процедуры *Writeln,* которая выведет сообщение *'Сумма введенных чисел равна'* и напечатает значение переменной *Sum*.

### *Порядок выполнения работы*

1. Изучить теоретические сведения по теме: “ Написание программы на Паскале с использованием операторов присваивания и безусловного перехода ”.
2. Получить индивидуальное задание у преподавателя и разработать программу в соответствии с поставленной задачей.
3. Показать работающую программу преподавателю.
4. Ответить на контрольные вопросы.

### *Контрольные вопросы*

1. Основные элементы программирования.
2. Основные характеристики программы. Понятия языка, оверлеев, глобальных и локальных блоков.
3. Операторы языка программирования Паскаль. Оператор присваивания. Формат, примеры.
4. Условные операторы. Оператор if. Формат, описание. Основные правила использования. Особенности использования вложенного оператора if.
5. Примеры использования оператора if.
6. Оператор выбора case. Формат, описание.
7. Правила использования оператора case. Примеры использования.