Учебно-педагогический комплекс
детский сад – средняя школа №42 города Могилёва

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
по программированию
на языке Турбо-Паскаль

учитель информатики
Трашков Олег Леонидович

МОГИЛЕВ
2012

Введение

Центральным понятием программирования является алгоритм. С него начинается работа над программой, а от качества алгоритма зависит ее успешное завершение. Поэтому учится программировать, прежде всего, означает учиться разрабатывать хорошие алгоритмы и применять те, что уже известны.

Алгоритм необходимо записать. Это можно сделать на русском языке, на языке блок-схем, наконец, на алгоритмическом языке. Последний способ, становится программой для компьютера.

Существует множество языков программирования, они обладают разными достоинствами и недостатками, некоторые имеют специфическую направленность. Мы начнем знакомство с пригодным для профессиональной работы языком Паскаль (Pascal). Он был создан Н. Виртом в начале 70-х годов XX века специально для обучения программированию и был назван в честь французского математика и физика Блеза Паскаля, впервые создавшего механическое вычислительное устройство.

Урок 1

Начнем знакомство с Паскалем с программы, которая считывает два числа с клавиатуры, складывает их и выводит сумму на экран.

Текст программы	Комментарий
`Program SUMMA;`	*заголовок программы*
`Var  X,Y,Z: integer;`	*описание переменных*
`Begin`	*начало программы*
`  Write('Введите два числа ');`	*вывод на экран*
`  Readln(X,Y);`	*ввод значений X и Y*
`  Z:=X+Y;`	*присваивание суммы*
`  Write(Z);`	*вывод результата*
`End.`	*Конец программы*

1.1 Создание программы.

Каталог с библиотеками **Turbo Pascal** называется TP (если у вас Borland Pascal то BP). В этом каталоге надо найти файл с именем **turbo.exe**, подвести к нему курсор и нажать **Enter**. При запуске системы появляется окно редактора текстов программ (его можно использовать и как текстовый редактор). Для входа в меню служит клавиша **F10** (или "мышь"). Строка меню расположена вверху экрана, передвижение по ней производится клавишами управления курсором. Если экран пуст, то можно сразу набирать текст программы, делая такие же отступы, какие имеются в примере. Эти отступы облегчают чтение текста и поиск ошибок. Если на экране после запуска системы находится ненужная программа, то следует войти в пункт меню **File** и выполнить команду New. Набор каждой строки программы завершается нажатием клавиши **Enter**.

1.2 Запуск программы.

Для выполнения программы надо выйти в меню и в пункте **Run** выполнить команду **Run** (или одновременно нажать клавиши **Ctrl** и **F9**). Система сначала запускает транслятор, который переводит программу с Паскаля на язык машинных кодов и ищет синтаксические ошибки в программе. Если они найдены, то программа не будет выполняться, произойдет возврат в редактор. Поверх текста программы появляется красная строка с сообщением об ошибке. После нажатия **Esc** окно исчезает, курсор устанавливается в строку с ошибкой. Для получения информации об ошибке надо нажать **Ctrl+F1**. Когда все ошибки исправлены, программа начинает выполняться.

Задание.

1. Запустите Паскаль и наберите текст приведенной программы. Для перехода на русский нажмите правые **Shift + Ctrl**, на английский левые **Shift + Ctrl**. Запустите программу на выполнение. Если нет ошибок, то на черном экране появится текст *"Введите два числа"*. Наберите на клавиатуре через пробел два целых числа и нажмите **Enter**. После выполнения программы на экране появится окно редактора. Чтобы посмотреть полученный результат нажмите **Alt+F5**.

2. Сохраните программу. Нажмите **F2**, в появившемся окне введите имя файла (например, **PRIM1_1**).

3. Составьте программу для нахождения суммы трех чисел. Сохраните ее.

Урок 2

Сегодня, пожалуй, самый теоретически насыщенный урок. Но без основных понятий и правил обойтись невозможно. Вашу программу будет выполнять машина, а она все делает буквально. Поэтому правила написания программы очень жестки, но их немного. Одно утешает, что правила русского языка намного сложнее.

Для начала рассмотрим программу, которая должна была у вас получиться при выполнении 3 задания предыдущего урока.

```
Program SUMMA;
Var   X,Y,K,Z: integer;
Begin
  Write('Введите три числа ');
  Readln(X,Y,K);
  Z:=X+Y+K;
  Write(Z);
End.
```

На этом примере видны **основные особенности записи программы:**
1)всякая программа начинается с заголовка - слова PROGRAM и следующего за ним названия программы; (в последних версиях Паскаля это предложение писать не обязательно)
2)после заголовка располагается описательная часть программы;
3)между служебными словами BEGIN и END записывается алгоритм решения задачи;
4)команды (операторы) разделяются точкой с запятой;
5)Русский текст заключается в апострофы ' ... ';
6)перечисляемые объекты разделяются запятой.

2.1 Имена и зарезервированные слова.

Текст программы записывается при помощи латинских букв, цифр и знаков. Буквы допускаются прописные и строчные, причем для компилятора записи **VAR**, **vAR**, **VaR** идентичны.

Особую роль в тексте программы играют имена и зарезервированные слова. Имена применяются для обозначения программы и ее объектов. Имя может состоять из любого количества букв или цифр, но должно начинаться с буквы. В имя можно включать знак подчеркивания, например **Prim_1**. В программе **SUMMA** пять имен: **SUMMA**, **X**, **Y**, **K**, **Z**. Программисты часто используют осмысленные имена в своих программах. Это всегда полезно, а в сложных программах совершенно необходимо.

Зарезервированные слова применяют для обозначения операторов (команд) и других элементов языка Паскаль. Их нельзя использовать в качестве имен и во всех программах они имеют одинаковый смысл. Зарезервированными словами в нашем примере являются слова: **PROGRAM**(программа), **VAR**(переменные), **BEGIN**(начало), **READ**(читать), **WRITELN**(писать), **END**(конец).

2.2. Константы и переменные.

Данные, которыми оперирует программа, могут быть определены в ней как неизменные, либо как способные изменять свое значение в ходе выполнения программы. Первые называются константами, а вторые переменными. И переменные и константы размещаются в памяти компьютера, в так называемой *«ячейке памяти».* В программе *переменные величины* должны быть описаны в предложении **VAR** (от слова **VARIABLE** - переменная), а *константы* (постоянные величины) - в предложении **CONST**. У всякой величины имеется три основных свойства: *имя, значение, тип*.

2.3. Тип INTEGER (целый).

И в жизни и в программировании очень полезно использовать понятие типа. Предположим, вам сказали, что вы должны приобрести Джой. Больше ничего не известно: сколько у него ног, предмет это или

животное и т.д. В то же время очень многое можно сообщить, добавив лишь одно слово, определяющее тип объекта по имени Джой, например, Джой - собака или Джой - компьютер.
Любая константа или переменная, использованная в программе, принадлежит к определенному типу. Тип задает множество допустимых значений переменных, внешний вид констант, возможные операции над значениями.
Значения величины *типа INTEGER* в Паскале не может быть меньше **-32768** или больше **32767**. Константы целого типа записываются в виде последовательности цифр со знаком или без него. Переменные должны быть перечислены в описательной части программы в предложении

VAR *имя_переменной*: INTEGER;

Над величинами целого типа допустимы арифметические операции: **+** (сложение), **-** (вычитание), *****(умножение), **DIV**(деление нацело), **MOD**(нахождение остатка от целочисленного деления).
Все операции вырабатывают результат целого типа. Например,

15 DIV 4 = 3, 25 MOD 4 = 1.

Над целыми разрешено и обычное деление, оно обозначается косой чертой «**/**» и дает результат вещественного типа.

2.4. Тип REAL(вещественный).

Константы вещественного типа (числа с дробной частью) изображаются с десятичной точкой:
12.3, -1.5, -0.75 или в показательной форме: **-0.45E5, 6.7E-10, 0.355E6** (для получения числа в обычном виде надо перенести запятую на число разрядов указанных после **E** вправо, если число положительное, влево, если отрицательное). Например, **6.7E-10=0.00000000067**.
Вещественные переменные требуют описания предложением

VAR имя: REAL;

Над величинами вещественного типа допустимы арифметические операции: **+**(сложение), **-** (вычитание), *****(умножение), **/**(деление).

2.5. Оператор присваивания

Оператор присваивания придает переменной конкретное значение, например:
X:=2; Y:=5,
одновременно уничтожая старое. Редкая программа обходится без оператора присваивания.
Присваивать можно значение другой переменной или результат вычисления арифметического выражения: **A:=B; A:+B+C; X:=Y+2-Z**.

Формат команды:
<имя переменной>:=<выражение>

Исполнение команды присваивания происходит в таком порядке: сначала вычисляется *выражение*, затем полученное значение присваивается *переменной*.

Пример 1. Пусть переменная **A** имела значение **6**. какое значение получит переменная **A** после выполнения команды: **A:=2·A - 1**.
Решение. Вычисление выражения **2·A - 1** при **A=6** даст число **11**. Значит новое значение переменной **A=11**.

Пример 2.
Поменяйте между собой значения двух переменных **A** и **B**, воспользовавшись третьей переменной **R** для временного хранения значения.

Решение.

```
Program prim2_2;
Var a, b, r: real;
begin
  Write('Введите два числа ');
  Readln(a,b);
  R:=a;
  A:=b;
  B:=r;
  Write(' a=' , a, ' b=' , b);
End.
```

Задачи.

1. Определить конечное значение переменных **X** и **Y** в результате выполнения следующих алгоритмов:

а) X:=2
X:=X x X
X:=X x X x X
X:=X x X x X x X

б) X:=1.5
X:=2 x X + 1
Y:=X/2
Y:=X + Y
X:=X - Y

2. Поменяйте между собой значения трех переменных **X**, **Y** и **Z** по схеме тройного квартирного обмена: **X®Y®Z®X**.

3. Присвойте переменной **N** ее собственное значение, увеличенное в **N** раз.

4. Чему равно **X** в результате выполнения программы:

X:=2;
X:=X+X;
X:=X-X

5. Указать значения величин **a** и **b** после выполнения следующих операторов присваивания:

а) a:=5.8
b:=-7.9
b:= a
a:=b

б) a:=0
b:=-9.99
b:=a
a:=b

Урок 3

Для начала проверьте правильность выполнения предыдущих заданий. И если не все получилось, не огорчайтесь, не ошибается тот, кто ничего не делает.

1. а) X=16777216
б) X=-2 Y=6

2.
```
Program prim2_2;
Var X,Y,Z,R: real;
Begin
  Write('Введите три числа ');
  Readln(X,Y,Z);
  R:=X;
  X:=Z;
  Z:=Y;
  Y:=R;
  Write(' X=' , X, ' Y=' , Y, ' Z=', Z);
End.
```

3. N:=N*N
4. X=0
5. а) b=5.8; a=5.8
б) b=0; a=0

На предыдущем уроке вы познакомились с представителями вещественного и целого типов. На самом деле и тот и другой имеет несколько видов отличающихся диапазоном допустимых значений.
В следующей таблице приведены пять стандартных целых типов:

тип	значение	формат
SHORTINT	-128..127	Знаковый
INTEGER	-32768..32767	Знаковый
LONGINT	-2147483648.. -2147483647	Знаковый
BYTE	0..255	Беззнаковый
WORD	0..65535	Беззнаковый

и пять стандартных вещественных типов:

тип	значение	число значащих чисел
REAL	2.9*10-39..1.7*1038	11..12
SINGLE	1.5*10-45..3.4*1038	7..8
DOUBLE	5.0*10-324..1.7*10308	15..16
EXTENDED	3.4*10-4932..1.1*104932	19..20
COMP	-2*1063+1..+2*1063-1	19..20

Арифметические выражения

Арифметические выражения строятся из имен переменных, констант, знаков операций и круглых скобок так, как это принято в математике. При вычислении их значений операции выполняются в порядке приоритета: *****, **/**, **DIV**, **MOD**, а затем **+** и **-**. Операции одинакового старшинства выполняются слева направо.

Наряду с переменными и константами в арифметические выражения можно включать функции. При определении значения выражения, прежде всего, вычисляются значения входящих в него функций. В Паскале имеются следующие стандартные функции:

функция	назначение	тип результата
ABS(X)	Абсолютное значение X	Вещественный
ARCTAN(X)	Арктангенс X	Вещественный
COS(X)	Косинус X	Вещественный
EXP(X)	ex	Вещественный
FRAC(X)	Дробная часть X	Вещественный
INT(X)	Целая часть X, обнуление дробной части	Вещественный
LN(X)	Натуральный логарифм	Вещественный
PI	Значение PI=3.1415926535897932385	Вещественный
ROUND(X)	Округление до ближайшего целого	Целый
SIN(X)	Синус X	Вещественный
SQR(X)	Квадрат X	Тип аргумента
SQRT(X)	Квадратный корень X	Вещественный
TRUNC(X)	Отбрасывание дробной части	Целый

Аргумент функции обязательно заключается в скобки.
Выражение на Паскале, как впрочем, и на других языках программирования, записывается в одну строчку, а для сохранения порядка действий используются скобки. Все действия должны быть указаны.
Например, **2·X + X·Y** надо записать как **2*X + X*Y**.

Задания

1. Найдите значения переменных, если это возможно. Учтите, что число **7.0** является вещественным, т.к. оно имеет дробную часть, хотя и равную нулю. Операции **MOD** и **DIV** можно выполнять только над целыми числами:

a) A:=21 DIV 5
B:= 20 MOD 5
C:= 14 DIV 6.0
D:= 14 MOD 0
E:= 5 MOD 13

b) A:= 2 MOD 3
B:= 36.0 MOD 6
C:= 81 DIV 0
D:= 38 DIV 6
E:= 3 DIV 2

2. Найдите значения переменных, если это возможно:

a) A:=SQR(100)
B:=sqrt(100)
C:=SQR(-10)
D:=SQRT(-10)
E:=SQR(0.9)
F:=SQRT(0)

b) A:=sqrt(9)
B:=SQR(9)
C:=SQRT(-9)
D:=SQR(-9)
E:=SQRT(0.0)
F:=SQR(0.1)

3. Найдите значения переменных, если это возможно:

a) A:=ROUND(6.9)
B:=ROUND(6.48)
C:=TRUNC(9.5)
D:=FRAC(9.5)
E:=INT(9.5)
F:=TRUNC(-17)
G:=FRAC(17)
H:=INT(-17)

b) A:=ROUND(15.39)
B:=ROUND(15.8)
C:=TRUNC(-39)
D:=FRAC(39)
E:=INT(39)
F:=TRUNC(5.6)
G:=FRAC(-0.3)
H:=INT(1.25)

Урок 4

Наконец-то мы вплотную приступим к программированию. Для усвоения правил написания основных операторов мы будем решать небольшие задачки, которые быстрей было бы сделать в тетрадке, чем писать программу. Но для сложной программы у нас не хватает знаний.
Мало программ обходится без ввода данных, и совсем нет таких, которые не выводят полученные результаты. Написать такую программу можно, но кому она понадобится?

4.1.Ввод.

Для сообщения данных компьютеру служит оператор ввода. Он помещает вводимое значение переменной в отведенную для него ячейку.
Оператор ввода:
READ (*список переменных*), где *список переменных* - последовательность имен переменных, разделенных запятыми.
Например, ***READ (X,Y,Z); READ (BETA);***
Оператор **READ** останавливает работу программы и ждет, пока пользователь наберет на клавиатуре число и нажмет **Enter**. Введенное число помещается в оперативную память, в отведенную ячейку, имеющую имя указанное в операторе. Если список ввода содержит несколько имен, то для каждого надо ввести свое значение. Вводимые числа разделяют пробелами или нажатием клавиши **Enter**.
Заканчивается ввод всегда клавишей **Enter**. После работы этого оператора курсор располагается за последним введенным символом, но не переводится на новую строку.
Для перевода курсора на новую строку экрана дисплея после ввода данных, используется оператор **READLN** (*список переменных*).
Оператор **READLN** отличается от **READ** еще и тем, что, введя необходимое количество данных, пропускает все остальные, набранные до нажатия клавиши **Enter**.

4.2. Вывод.

Для вывода результатов работы программы служит оператор
WRITE(*список вывода*).
Список вывода может содержать имена переменных, числовые и текстовые константы, выражения. Элементы в списке разделяются запятыми. Если указана переменная, то на экран выводится ее значение, константа выводится без изменения, значения выражений вначале вычисляются, а затем высвечиваются на экране.
Вслед за выражением после двоеточия можно указать ширину поля экрана, в котором разместится выводимое значение.
Например, оператор **WRITE(10:3, 55:6)** высветит на экране **.10.....55** (точка означает пробел, пустую позицию экрана). Вывод происходит в том месте экрана, где находится курсор.
При выводе вещественных значений можно указать, сколько десятичных цифр следует сохранить в дробной части числа. Количество цифр указывается вслед за шириной поля после двоеточия.
Например, если **X=3.14159**, а **Y=2.71468**, то оператор **WRITE(X:6:2,Y:8:3)** высветит на экране **..3.14....2.715**.
Чтобы прокомментировать выводимые значения, в список вывода можно помещать строки любых символов, заключенные в апострофы (одинарные кавычки).
Например,
WRITE('Ответ:', X:4, 'км/сек.').
Эти строки появятся на экране без кавычек. Так при **X=3.5** этот оператор выведет: ***Ответ: 3.5 км/сек.***
Перевод курсора на новую строку осуществляется оператором пустого вывода **WRITELN;**
Если надо перевести курсор после вывода, то применяется оператор
WRITELN(*список вывода*).

Пример программы. Пусть требуется найти сумму, произведение и разность двух данных чисел. Для каждого из чисел надо придумать имя переменной и указать ее тип. Затем ввести эти числа в отведенные ячейки и, используя возможности оператора вывода напечатать результаты.
При решении задач имена присваиваются не только исходным данным, но и результатам, а также получаемым промежуточным значениям. Поскольку в рассматриваемом примере надо получить три результата, введем для них переменные **X**,**Y**,**Z**. В программе этим переменным будут присвоены значения суммы, произведения и разности двух вводимых чисел.

```
Program prim_4;
Var a,b,x,y,z:real;
Begin
  Write('введите два числа через пробел, затем нажмите Enter');
  Readln(a,b);
  X:=a+b;
  Y:=a*b;
  Z:=a-b;
  Writeln('a+b=',x);
  Writeln('a*b=',y);
  Writeln('a-b=',z);
End.
```

Задания:

1. Напишите программу, которая запрашивает два числа, находит остаток от деления первого на второе и выводит результат.
2. Составьте программу нахождения периметра квадрата, если задана его площадь.
3. Даны два числа. Найти их среднее арифметическое.
4. Найти площадь кольца по заданным внешнему и внутреннему радиусам.
5. Даны катеты прямоугольного треугольника. Найти его периметр.
6.* Поменять местами значения переменных **X** и **Y**, не используя дополнительной переменной.

Правильные ответы для тех, кто выполнил задания предыдущего урока.

1. Найдите значения переменных, если это возможно. Учтите, что число 7.0 является вещественным, т.к. оно имеет дробную часть, хотя и равную нулю:

a) A:=21 DIV 5=4
B:= 20 MOD 5=0
C:= 14 DIV 6.0 *(нельзя делить веществ)*
D:= 14 MOD 0 *(деление на 0)*
E:= 5 MOD 13=5

b) A:= 2 MOD 3=2
B:= 36.0 MOD 6 *(нельзя делить веществ)*
C:= 81 DIV 0 *(деление на 0)*
D:= 38 DIV 6=6
E:= 3 DIV 2=1

2. Найдите значения переменных, если это возможно:

a) A:=SQR(100)=10000
B:=sqrt(100)=10.0
C:=SQR(-10)=100
D:=SQRT(-10) *(выражение <0)*
E:=SQR(0.9)=0.81
F:=SQRT(0)=0.0

b) A:=sqrt(9)=3.0
B:=SQR(9)=81
C:=SQRT(-9) *(выражение < 0)*
D:=SQR(-9)=81
E:=SQRT(0.0)=0.0
F:=SQR(0.1)=0.01

3. Найдите значения переменных, если это возможно:

a) A:=ROUND(6.9)=7
B:=ROUND(6.48)=6
C:=TRUNC(9.5)=9
D:=FRAC(9.5)=0.5
E:=INT(9.5)=9.0
F:=TRUNC(-17)=-17
G:=FRAC(17)=0.0
H:=INT(-17)=-17.0

b) A:=ROUND(15.39)=15
B:=ROUND(15.8)=16
C:=TRUNC(-39)=-39
D:=FRAC(39)=0.0
E:=INT(39)=0.0
F:=TRUNC(5.6)=5
G:=FRAC(-0.3)=-0.3
H:=INT(1.25)=1.0

Урок 5

Сегодня не будет новых операторов, сегодня мы будем учиться составлять алгоритмы, используя те знания, которые уже имеем.

5.1. Обмен значениями.

Начнем с разбора 6-го задания предыдущего урока. Первая мысль, приходящая в голову, это написать программу, похожую на эту:

A:= B;
B:= A;

Но эта программа работать не будет (в обеих переменных будет значение **B**).

Теперь поищем правильное решение. Обозначим начальное значение **A** за A_1, **B** за B_1. Тогда необходимо, чтобы по окончании работы программы **$A=B_1$**, а **$B=A_1$**.

0) $A=A_1$; $B=B_1$;

1) Занесем в переменную A результат суммирования A и B (A := A + B):

$A = A_1 + B_1$; $B = B_1$;

2) Занесем в переменную B разность A и B (B := A - B):

т.к. $A = A_1 + B_1$; то $B=(A_1+B_1)-B = A_1$;

3) Занесем в переменную A разность A и B (A := A - B):

$A = B_1$; $B = A_1$;

Код программы:

```
Program prim_4;
Var a,b:integer;
Begin
  Write('введите два числа ');
  Readln(A,B);
  A:=A+B;
  B:=A-B;
  A:=A-B;
  Writeln('A=',A);
  Writeln('B=',B);
End.
```

5.2. Эффективные алгоритмы.

Вы уже знаете, что в Паскале отсутствует возможность возведения в степень, не считая квадрата. Поэтому для получения a^{20} нужно **a*a*a*a…*a 19 раз**. Но если учесть, что результат умножения можно сохранить в промежуточной переменной, ответ можно найти за *5 действий*.

```
Program prim_4;
Var a,b:real;
Begin
  Write('введите число');
  Readln(A);
  B:=A*A;    {получаем A во 2}
  B:=B*B;    {получаем A в 4}
  B:=A*B;    {получаем A в 5}
  B:=B*B;    {получаем A в 10}
  B:=B*B;    {получаем A в 20}
  Writeln('A в 20 степени=',B:0:2);
  {Вместо указания ширины поля для вывода числа лучше ставить 0, тогда
компилятор отведет столько позиций под целую часть, сколько получилось в
ответе, а для дробной части я посчитал возможным оставить 2 позиции}
End.
```

В фигурные скобки {...} в программе можно заключать комментарий, компилятор пропускает текст, заключенный в такие скобки, а комментарий позволяет вспомнить о чем программа, если вы позднее возвращаетесь к ее тексту.

Задание для тренировки

1. Дано вещественное число **A**. Не пользуясь никакими арифметическими операциями, кроме умножения, получить:

1) A^4 за две операции;
2) A^6 за три операции;
3) A^7 за четыре операции;
4) A^8 за три операции;
5) A^9 за четыре операции;
6) A^{13} за пять операции;
7) A^{15} за пять операции;
8) A^{19} за пять операции;
9) A^{21} за шесть операции;
10) A^{28} за шесть операции;

5.3. Целочисленная арифметика. Задачи на целочисленное деление.

В программировании существует целый класс задач, где действия производятся только с целыми числами. При решении подобных задач обычно используются операции над целыми числами **MOD** и **DIV**.
Сегодня мы познакомимся с задачами на целочисленное деление.

Задача. Дано расстояние в сантиметрах. Найти число полных метров в нем.
Код программы:

```
Program prim_4;
Var a,b:integer;
Begin
  Write('введите расстояние в сантиметрах');
  Readln(a);
  b:=a mod 100;
  Writeln(b,' метров ');
End.
```

Задания для тренировки

2. Дана масса в килограммах. Найти число полных центнеров в ней.
3. Дана масса в килограммах. Найти число полных тонн в ней.
4. Дано расстояние в метрах. Найти число полных километров в нем.
5. Дан прямоугольник с размерами 543x130 мм. Сколько квадратов со стороной 130 мм можно отрезать от него?
6.* Дано целое число **k** (*1 ≤k ≤ 365*). Присвоить целочисленной величине **n** значение **1, 2,...,6** или **0** в зависимости от того, на какой день недели (*понедельник, вторник, ..., суббота или воскресенье*) приходится **k-й** день года, в котором *1 января - понедельник*.

Правильные ответы для тех, кто выполнил задания предыдущего урока.

1. Напишите программу, которая запрашивает два числа, находит остаток от деления первого на второе и выводит результат.

```
Program prim_3;
Var a,b,x:integer;
Begin
  Write('введите два числа ');
  Readln(a,b);
  X:=a mod b;
  Writeln('остаток от деления-',x);
End.
```

2. Составьте программу нахождения периметра квадрата, если задана его площадь.

```
Program prim_4;
Var a,b,x:real;
Begin
  Write('введите площадь квадрата ');
  Readln(a);
  X:=sqrt(a);
  B:=x*4;
  Writeln('периметр квадрата =',x:0:3);
End.
```

3. Найти площадь кольца по заданным внешнему и внутреннему радиусам.

```
Program prim_4;
Var r1,r2,s:real;
Begin
  Write('введите радиусы кольца ');
  Readln(r1,r2);
  S:=abs(2*3.14*r1-2*3.14*r2);    {разность площадей берем по модулю, так как
не знаем какое из колец является внутренним, а какое внешним}
  Writeln('площадь кольца =',s:0:2);
End.
```

Урок 6

Продолжим знакомство с целочисленной арифметикой. Очень часто необходимо чтобы программа определила, из каких цифр состоит число, или определила разряд заданной цифры или наоборот цифру в заданном разряде. Если вы попросите человека решить эту задачу, проблем не возникнет, а как быть с компьютером. Для него любое число это набор нулей и единиц, т.е. двоичный код. В отличие от нас компьютер все действия выполняет в двоичной системе. Например, число 27 хранится в его памяти как 11011. И где здесь 2 и 7?

Для решения этих задач надо вспомнить, что собой представляет любое число десятичной системы счисления (т.е. той, к которой мы с вами привыкли). В позиционной системе счисления (к которой относится десятичная система) величина, обозначаемая цифрой в записи числа, зависит от ее позиции. Например, в числе 333 первая цифра обозначает три сотни, вторая - три десятка, третья - три единицы. Любое число можно записать в виде:

$32478=3\cdot10000+2\cdot1000+4\cdot100+7\cdot10+8=3\cdot10^4+2\cdot10^3+4\cdot10^2+7\cdot10^1+8\cdot10^0$

Поэтому для обработки десятичных чисел используется **10** в соответствующей степени.
Например, надо получить число, образованное при перестановке цифр заданного числа.

```
Program prim_6;
Var n,x1,x2,m:integer;
Begin
  Write('введите двузначное число');
  Readln(n);
  X1:=n mod 10;    {выделяем из числа единицы}
  X2:=n div 10;    {получает число десятков в числе}
  M:=x1*10+x2;     {число единиц умножаем на 10, получаем десятки}
  Writeln(m);
End.
```

Пусть дано n=27.
X1:= 27 mod 10 = 7
X2:= 27 div 10 = 2
M:=7·10+2 = 72, что и требовалось получить.

К сожалению, таким образом, мы можем определить лишь крайние цифры числа. А если цифра, которая нам нужна, стоит не с краю? Не беда, сделаем ее крайней.
Например, дано трехзначное число, надо определить среднюю цифру числа.
Пусть a=246
B:=a div 10 = 24
B:=b mod 10 = 4
или по другому:
B:=a mod 100 = 46
B:=b div 10 = 4

Задания для тренировки

1. Дано двузначное число. Найти:

a. число десятков в нем;
b. число единиц в нем;
c. сумму его цифр;
d. произведение его цифр.

2. Дано трехзначное число. Найти:

a. Число единиц в нем;
b. Число десятков в нем;
c. Сумму его цифр;
d. Произведение его цифр.

3. Дано трехзначное число. Найти число, полученное при прочтении его цифр справа налево.

4. Дано трехзначное число. В нем зачеркнули первую слева цифру и приписали ее в конце. Найти полученное число.

5. Дано трехзначное число. В нем зачеркнули последнюю справа цифру и приписали ее в начале. Найти полученное число.

6. Дано трехзначное число. Найти число, полученное при перестановке первой и второй цифр заданного числа.

7. Дано трехзначное число. Найти число, полученное при перестановке второй и третьей цифр заданного числа.

8. * Дано вещественное число **A**, содержащее два знака до запятой и два после. Получить новое число, поменяв в числе **A** целую и дробную части.

9. * В кассе имеются купюры достоинством в **K** *рублей* и в **1** *рубль*. Выдать **N** *рублей* минимальным набором купюр заданного достоинства.

Разбор тренировочного задания урока 5.

6.* Дано целое число **k** (*1 ≤k ≤ 365*). Присвоить целочисленной величине **n** значение **1, 2,...,6** или **0** в зависимости от того, на какой день недели (*понедельник, вторник, ..., суббота или воскресенье*) приходится **k-й** день года, в котором *1 января - понедельник*.

Так как *1 января* приходится на *понедельник*, то легко определить на какой день приходится **k-й** день года. Для этого достаточно разделить нацело на **7**, а полученный остаток и будет днем недели.

```
Program prim_5;
Var n,k:integer;
Begin
  Write('введите день года');
  Readln(k);
  n:=k mod 7;
  Writeln(k,' день года - это ', n,' день недели ');
End.
```

Урок 7

В рассмотренных до сих пор алгоритмах и программах все команды (операторы) выполнялись последовательно одна за другой в том порядке, в каком они были записаны (линейные алгоритмы). Однако таким образом может быть построен алгоритм для решения далеко не всякой задачи.
В практике хорошо известны задачи, дальнейший ход решения которых зависит от выполнения какого-либо условия. В жизни часто приходится действовать в зависимости от обстоятельств, от каких-то условий. Но если в жизни мы часто ищем выход из ситуации только тогда, когда попали в неё, в программе необходимо предусмотреть все действия которые необходимо выполнить после проверки условия, как в случае его выполнения, так и в случае невыполнения. Чтобы изменять последовательность выполнения различных частей программы, применяют условный оператор.

7.1. Условный оператор

Условный оператор позволяет выполнять или пропускать операторы программы в зависимости от некоторого условия. Условный оператор может иметь две формы:
Полный оператор
IF условие **THEN** оператор_1 **ELSE** оператор_2;
Неполный оператор
IF условие **THEN** оператор;

Если перевести на русский язык английские слова то получим:
ЕСЛИ *условие* ***ТО*** *оператор_1* ***ИНАЧЕ*** *оператор_2*; или ***ЕСЛИ*** *условие* ***ТО*** *оператор*;

В качестве условия применяют операции сравнения: **=**, **<>**, **<=**, **>=**, **<**, **>**. Слева и справа от знака сравнения записывают арифметические выражения.
Например, оператор

```
If x<>0 then z:=y/x  else write('Ошибка!');
```

присвоит переменной **Z** значение частного **y/x**, если **x<>0**, в противном случае высветит на экране слово **"Ошибка!"**.

7.2. Составной оператор

В некоторых случаях после слов **THEN** и **ELSE** надо выполнить не один оператор, а несколько. Тогда эти операторы заключаются в так называемые операторные скобки, где **BEGIN** - открывающая скобка, **END** - закрывающая скобка. Все операторы находящиеся внутри операторных скобок называются составным оператором. Перед словом **ELSE** точка с запятой никогда не ставится.

Формат команды: **BEGIN** оператор; оператор; ... оператор **END**;

Например:

```
If a<b then
        begin
        R:=a; A:=b;  B:=r
        end;
```

После выполнения такого оператора в переменной **A** будет большее, а в переменной **B** - меньшее из двух значений, находившихся там ранее.

В качестве выполняемого в условном операторе действия может быть другой условный оператор.
Например:

```
If sqr(x)+sqr(y)>1 then
        If x>y then z:=0  else z:=1;
```

При такой форме записи со сдвигом вправо для каждого внутреннего действия, легко понять, к какому из двух слов **IF** относится слово **ELSE**.

Рассмотрим пример программы с использованием условного оператора.
Пусть для двух целых чисел надо определить, являются они четными или нет. Для проверки четности используем условие: *остаток от деления на 2 четного числа равен 0.*

```
Program prim_7;
Var a,b:integer;
Begin
  Write('введите два целых числа');
  Readln(a,b);
  If a mod 2 = 0 then Writeln (' a - четное ')
  Else Writeln (' a - нечетное ');
  If b mod 2 = 0 then Writeln (' b - четное ')
  Else Writeln (' b -нечетное ');
End.
```

Задания для тренировки

2. Ввести два числа. Напечатать сначала меньшее, затем большее из них.
3. Даны числа **x** и **y**. Вычислите число **z**, равное **x+y**, если **x<=y**, и **1 - x + y** в противном случае
4. Даны два числа. Выведите первое из них, если оно больше второго, и оба числа, если это не так.
5. Если данное число **x** меньше нуля, то **z** присвойте значение большего из двух чисел **x** и **y**, иначе **z** присвойте значение полусуммы этих чисел.
6.* Даны два числа. Меньшее из них замените полусуммой этих чисел, а большее - их произведением.
7. Даны радиус круга и сторона квадрата. У какой фигуры площадь больше?
8. Дано целое число. Определить:

a) Является ли оно четным;
b) Оканчивается ли оно цифрой 7;
c) Делится ли оно на 13.

Разбор тренировочного задания урока 6.

8.* Дано вещественное число **A**, содержащее два знака до запятой и два после. Получить новое число, поменяв в числе **A** целую и дробную части.

Данная задача не относится к целочисленной арифметике, ведь дано вещественное, а не целое число. Попробуем найти целую и дробную части, а потом просто соберем новое число увеличив дробную часть в 100 раз и уменьшив целую часть тоже в 100 раз.

```
Program prim_8;
Var a,b,x1,x2:real;
Begin
  Write('введите число');
  Readln(a);
  X1:=int(a) ;    {целая часть}
  X2:=frac(a);    {дробная часть}
  b:=x1/100+x2*100;    {новое число}
  Writeln(b);
End.
```

9.* В кассе имеются купюры достоинством в **K** *рублей* и в **1** *рубль*. Выдать **N** рублей минимальным набором купюр заданного достоинства.

Определим сколько купюр достоинством **K** необходимо для выдачи суммы наиболее близкой к данной, но меньше ее. А затем какую сумму осталось выдать.

```
Program prim_6;
Var k,n,m,p:integer;
Begin
  Write('введите число рублей');
  Readln(n);
  Write('введите достоинство купюры');
  Readln(k);
  m:=n div k;
  p:=n mod k;
  Writeln(p,' купюр по 1 рублю, ', m,' купюр по ',k,'руб. Итого=',p+m);
End.
```

Урок 8

Сегодняшний урок мы начнем с разбора 6-го задания предыдущего урока, так как её решение содержит некоторые подводные камни, на которые, как правило, натыкаются начинающие программисты.

6.* Даны два числа. Меньшее из них замените полусуммой этих чисел, а большее - их произведением.

На первый взгляд задача решается просто:

```
If a>b then begin b:=(a+b)/2; a:=a*b end
      else begin a:=(a+b)/2; b:=a*b end;
```

Но если выполнить эту команду, то ответ получится неверным, почему?
Предположим a=10, b=20. Что же получится в результате выполнения оператора. Так как a<b, то условие, указанное в операторе не выполнено, следовательно, выполнится

```
else begin a:=(a+b)/2; b:=a*b end;
```

подставим значения и получим:
a:=(a+b)/2=(10+20)/2=15
b:=a·b =15·20=300 вместо 10·20=200 потому, что к этому моменту первоначальное значение a=10 изменилось и стало равно 15.
Следовательно, надо как-то сохранить первоначальные значения. Можно запомнить их в дополнительных переменных:

```
x:=a;
y:=b;
If a>b then begin b:=(x+y)/2; a:=x*y end
      else begin a:=(x+y)/2; b:=x*y end;
```

а можно запомнить в дополнительных переменных полусумму и произведение:

```
Program prim_6;
Var a,b,x,y:real;
Begin
  Write('введите два числа');
  Readln(a,b);
  X:=a*b;         {запомним значение произведения}
  Y:=(x+y)/2;     {запомним значение полусуммы}
  If a>b then begin b:=y; a:=x end
    else begin a:=y; b:=x end;
  Writeln (' a= ',a:0:2,' b=',b:0:2);
End.
```

Логические операции и выражения

Если условие выполняется, то говорят, что соответствующее выражение истинно, если не выполняется - выражение ложно.

Для построения сложных условий в Паскале имеются четыре логических операции:
NOT - отрицание (НЕТ),
AND - логическое умножение (И),
OR - логическое сложение (ИЛИ),
XOR - исключающее "или".

Результаты логических операций для различных значений операндов приведены в таблице, где использованы обозначения: **T - true** (*истина*), **F - false** (*ложь*).

A	B	Not A	A and B	A or B	A xor B
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

где **A** и **B** результат операции отношения.

Приоритеты логических операций: 1) not; 2) and; 3) or; 4) xor.

Примеры логических выражений:
a) (0<x) AND (x<=1)
b) (a=0) OR (abs(x)<5)
c) NOT (x=y)

Операции отношений имеют более низкий приоритет, чем логические операции, поэтому их следует заключать в скобки при использовании с логическими операциями.
Из переменных, констант, сравнений, логических операций и скобок можно строить логические выражения.

Рассмотрим следующую задачу: Имеется прямоугольное отверстие со сторонами **a** и **b** и кирпич с ребрами **x**, **y**, **z**. Требуется определить пройдет ли кирпич в отверстие.

Решение
Кирпич имеет три грани, каждую из которых мы можем повернуть на 90 градусов, т.е. для каждой грани надо проверит два случая. Итого шесть. Получаем условие:

(a>x)and(b>y) or (b>x) and (a>y) or (a>x) and(z>y) or (z>x) and(a>y) or (b>x) and(z>y) or (z>x) and(b>y)

Код программы:

```
Program prim_8;
Var a,b,x,y,z:integer;
    F:boolean;
Begin
  Write('введите размеры отверстия');
  Readln(a,b);
  Write('введите размеры кирпича');
  Readln(x,y,z);
  If (a>x)and(b>y) or (b>x)and(a>y) or (a>x)and(z>y) or
     (z>x)and(a>y) or (b>x)and(z>y) or (z>x)and(b>y)
  then Writeln ('Кирпич пролезет в отверстие')
    Else Writeln ('Кирпич не пролезет в отверстие');
End.
```

Тренировочные задания

1. Установить, истинны или ложны следующие условия: (A=0) and not (B=0) or not (a=0) and (B=0) при **a)** A=0, B=0
b) A=0, B=1

2. Ввести три числа. Выбрать и напечатать наибольшее из них.

3. Написать программу, которая требует ввода времени дня и, в зависимости от введенного значения, желает *доброго утра*, *доброго дня*, *доброго вечера* или *спокойной ночи*.

4. Даны три числа. Найдите наибольшее значение их суммы и произведения.

5. Даны три числа **a**, **b**, **c**. Удвойте эти числа, если они являются упорядоченными по возрастанию.

6. Проверьте, есть ли среди трех заданных чисел равные.

7. Дано двузначное число. Определить:

a) какая из его цифр больше, первая или вторая;
b) одинаковы ли его цифры.

8. Известны площади круга и квадрата. Определить:

a) Уместится ли круг в квадрате;
b) Уместится ли квадрат в круге.

9. Дано трехзначное число. Выяснить, является ли оно палиндромом (*«перевертышем»*), т.е. таким числом, десятичная запись которого читается одинаково слева направо и справа налево.

Разбор тренировочного задания урока 7.

8. * Дано целое число. Определить:

a) Является ли оно четным;
b) Оканчивается ли оно цифрой 7;
c) Делится ли оно на 13.

```
Program prim_6;
Var a:integer;
Begin
  Write('введите целое число');
  Readln(a);
  If a mod 2 = 0 then Writeln (a,' - четное ')
    Else Writeln (a,' -нечетное ');
  If a mod 10 = 7 then Writeln (a,'- оканчивается на 7 ')
    Else Writeln (a,' -не оканчивается на 7 ');
  If a mod 13 = 0 then Writeln (a,'- делится на 13 ')
    Else Writeln (a,' -не делится на 13');
End.
```

Урок 9

Мало знать оператор, надо уметь его использовать.
Сегодня мы займемся разбором типичных задач на условный оператор.

Целочисленная арифметика и условный оператор

Задача

Дано трехзначное число. Определить:

a) Является ли сумма его цифр двузначным числом;
b) Является ли произведение его цифр трехзначным числом;
c) Больше ли числа A произведение его цифр;
d) Кратна ли пяти сумма его цифр;
e) Кратна ли сумма его цифр числу A.

Код программы:

```
Program prim;
Var a,x1,x2,x3:integer;
    x,s,p:longint;
Begin
    Write('введите трехзначное число');
    Readln(a);
    X1:=x div 100;
    X2:=(x div 10) mod 10;
    X3:= x mod 10;
    S:=x1+x2+x3;
    If (s>9) and (s<100) then writeln (' Сумма двузначное число ')
      Else Writeln('Сумма не двузначное число ');
    P:=x1*x2*x3;
    If (p>99) and (p<1000) then writeln ('Произведение трехзначное число')
      Else Writeln('Произведение не трехзначное число ');
    Write('Введите число A');
    Readln(a);
    If (p<a) then writeln (' произведение больше ', a)
      Else Writeln('произведение не больше ', a);
    If s mod 5 = 0 then writeln (' сумма цифр кратна 5')
      Else Writeln('сумма цифр не кратна 5 ');
    If s mod a = 0 then writeln (' сумма цифр кратна ',a)
      Else Writeln('сумма цифр не кратна ', a);
End.
```

Тренировочные упражнения

1. Дано трехзначное число.
  a) Верно ли, что все его цифры одинаковы?
  b) Определить, есть ли среди его цифр одинаковые.
2. Дано четырехзначное число. Определить:
  a) Равна ли сумма двух первых его цифр сумме двух его последних цифр;
  b) Кратна ли трем сумма его цифр;
  c) Кратно ли четырем произведение его цифр;
  d) Кратно ли произведение его цифр числу **A**.
3. Дано натуральное число.
  a) Верно ли, что оно заканчивается нечетной цифрой?
  b) Верно ли, что оно заканчивается четной цифрой?
4. Является ли число **A** делителем числа **B**? А наоборот?
5. Дано четырехзначное число **N**. Выяснить:
  a) Является ли число палиндромом?
  b) Верно ли, что все четыре цифры этого числа различны.

6. Трамвайный билет имеет шестизначный номер. Выяснить, является ли билет «*счастливым*». Билет назовем «*счастливым*», если сумма первых трех цифр равна сумме последних трех цифр.
Примечание. Так как шестизначное число больше 32767 (тип **Integer**), необходимо номер билета определит как тип **Longint** (до 10 знаков).
7. Выяснить пройдет ли кирпич в круглое отверстие.

Разбор заданий урока 8.

1. Установить, истинны или ложны следующие условия: (A=0) and not (B=0) or not (a=0) and (B=0) при
a) A=0, B=0 выражение ложно.
Выполним действия по порядку:
1) B=0 истина
2) Not(истина) =ложь
3) (A=0) истина
4) истина and ложь =ложь
5) not(a=0) =ложь
6) ложь and истина = ложь
7) ложь or ложь = ложь
b) При A=0, B=1 выражение истинно

2. Ввести три числа. Выбрать и напечатать наибольшее из них.

```
Program prim_2;
Var a,b,c:real;
Begin
  Write('введите три числа');
  Readln(a,b,c);
  If (a>b) and (a>c) then writeln (' max= ',a:0:2);
  If (b>a) and (b>c) then writeln (' max= ',b:0:2);
  If (c>a) and (c>b) then writeln (' max= ',c:0:2);
End.
```

8. Известны площади круга и квадрата. Определить:
a) Уместится ли круг в квадрате;
b) Уместится ли квадрат в круге.

```
Program prim_8;
Var a,r,s1,s2:real;
Begin
  Write('введите площадь круга и квадрата');
  Readln(s1, s2);
  A:=sqrt(s2);
  R:=sqrt(s/3.14);
  If (r<a/2) then writeln (' круг уместится в квадрате')
    Else writeln (' круг неуместится в квадрате');
  If (a*sqrt(2))>(4*r) then writeln ('квадрат уместится в круге')
    Else Writeln('квадрат неуместится в круге');
End.
```

9. Дано трехзначное число. Выяснить, является ли оно палиндромом («*перевертышем*»), т.е. таким числом, десятичная запись которого читается одинаково слева направо и справа налево.

```
Program prim_9;
Var a,b,c:integer;
Begin
  Write('введите трехзначное число');
  Readln(a);
  If (a div 100)=(a mod 10) then writeln (' это палиндром ')
    Else Writeln(' это не палиндром');
End.
```

Урок 10

В своей практической деятельности человек постоянно сталкивается с задачами, при решении которых требуется многократно повторять одни и те же действия. Для составления алгоритмов решения таких задач используются команды повторения (циклы).

Цикл - это замечательное изобретение, которое, в сущности, и делает компьютеры такими ценными. Он позволяет многократно повторить любую часть программы. Цикл не может выполняться вечно, он заканчивается по какому-либо условию. Проверка этого условия может производиться в начале каждого повторяющегося шага, в этом случае цикл называется **ПОКА**. При проверке условия в конце каждого шага цикл называется **ДО**. Разновидностью цикла **ДО** является цикл **ПЕРЕСЧЕТ**.

10.1.Оператор цикла WHILE (цикл ПОКА).

Формат оператора:

WHILE *логическое выражение* **DO**;

Оператор будет повторятся пока истинно логическое выражение. Перед каждым повторением оператора значение логического выражения вычисляется заново. Если необходимо повторить несколько операторов, их следует объединить в составной оператор, т.е. заключить в операторные скобки **begin ... end**.

Этот цикл может не выполниться ни разу, если условие при входе в него оказалось ложным.

Таким образом, цикл **ПОКА** содержит условие повторения цикла.

Пример 1. Программа подсчета суммы **S** первых 1000 членов гармонического ряда 1+1/2+1/3+1/4+...1/N.

```
Program Summa;
Var S:real;N:integer;
Begin
  S:=0;N:=0;
  While n<1000 do
  Begin
    N:=n+1;
    S:=s+1/n
  End;
  Writeln(s);
End.
```

Пример 2. Вычислить наибольший общий делитель двух натуральных чисел **A** и **B**.

Воспользуемся для этого алгоритмом Евклида: будем уменьшать каждый раз большее из чисел на величину меньшего до тех пор, пока оба числа не станут равны.

```
Program NOD;
Var  a,b:integer;
Begin
  Write ('введите два натуральных числа')
  Readln(a,b)
  While a<>b do If a>b then a:=a-b else b:=b-a;
  Writeln('НОД=',a);
End.
```

Пример 3 . Начав тренировки, лыжник в первый день пробегал 10 км. Каждый следующий день он увеличивал длину пробега на 10% от предыдущего дня. Определить в какой день он пробежит больше 20км, в какой день суммарный пробег за все дни превысит 100км.

```
Program prim_10;
Var S:real; N:integer;
Begin
  S:=10;N:=1;
  While s<20 do
```

```
  Begin
    N:=n+1;
    S:=s*0.1
  End;
  Writeln(' дневной пробег больше 20 км на ',n,' день');
  S:=10;N:=1;
  While s<100 do
  Begin
    N:=n+1;
    S:=s+s*0.1 {накапливаем суммарный пробег}
  End;
  Writeln('за ',n,' пробежит больше 100 км');
End.
```

Тренировочные задания

1. Даны целые числа **a** и **b** (a>b). Определить:
a) Результат целочисленного деления **a** на **b**, не используя стандартную операцию целочисленного деления;
b) Остаток от деления **a** на **b**, не используя стандартную операцию вычисления остатка.

2. Известны оценки по информатике каждого из 20 учеников класса. В начале списка перечислены все десятки, затем все остальные оценки. Сколько учеников имеют по информатике оценку «10»?. Условный оператор не использовать.

3. Напечатать минимальное число, большее 200, которое нацело делится на 17.

4. Гражданин *1 марта* открыл счет в банке, вложив *1000000 рублей*. Через каждый месяц размер вклада увеличивается на 2% от имеющейся суммы. Определить: за какой месяц величина ежемесячного увеличения вклада превысит *30000 рублей*. Через сколько месяцев размер вклада превысит *1200000 рублей*.

5.* В некоторой стране используются денежные купюры достоинством в **1**, **2**, **4**, **8**, **16**, **32** и **64**. Дано натуральное число **N**. Как наименьшим количеством таких денежных купюр можно выплатить сумму **N** (указать количество каждой из используемых для выплаты купюр)?
Предполагается, что имеется достаточно большое количество купюр всех достоинств.

Разбор заданий урока 9.

5. Дано четырехзначное число **N**. Выяснить, является ли число палиндромом?

Для решения этой задачи надо записать число в обратном порядке. Для этого единицы умножаем на 1000, прибавляем число десятков умноженное на 100, число тысяч умножаем на 10 и прибавляем число десяти тысяч. Если полученное число будет равно исходному, значит это палиндром.

```
Program prim9_5a;
Var  x,y:integer;
Begin
  Write('введите четырехзначное число');
  Readln(x);
  y:=(x mod 10)*1000+((x div 10) mod 10)*100+((x div 100) mod 10)*10+x div
1000;
  If x=y then writeln ('число палиндром ')
    Else Writeln('это не палиндром ');
End.
```

6. Трамвайный билет имеет шестизначный номер. Выяснить, является ли билет «*счастливым*». Билет назовем «*счастливым*», если сумма первых трех цифр равна сумме последних трех цифр. Примечание. Так как шестизначное число больше 32767 (тип **Integer**), необходимо номер билета определит как тип **Longint** (до 10 знаков).

```
Program prim9_6;
Var  x1,x2:integer;
     x,:longint;
Begin
  Write('введите номер билета');
  Readln(x);
  X1:=x div 100000 + (x div 10000) mod 10 + (x div 1000) mod 10; {сумма
первых трех цифр}
  X2:=(x div 100) mod 10 + (x div 10) mod 10 + x mod 10; {сумма последних
трех цифр}
  If x1=x2 then writeln (' билет счастливый ')
    Else Writeln('билет не счастливый ');
End.
```

Урок 11

Продолжим знакомство с операторами цикла, имеющимися в Паскале. Повторение группы операторов (тела цикла) можно организовать и с помощью оператора, где проверка условия осуществляется после выполнения тела цикла.

11.1. Оператор цикла REPEAT (цикл ДО)

Формат оператора:

REPEAT *оператор; оператор; ... оператор* ***UNTIL*** *логическое условие;*

Часть программы, заключенная между служебными словами **REPEAT** и **UNTIL**, повторяется до тех пор, пока не станет истинным логическое выражение, стоящее после слова **UNTIL**.
Между словами **REPEAT** (*повторить*) и **UNTIL** (*до тех пор, пока*) можно записать любое количество операторов без использования операторных скобок.
В отличие от оператора **WHILE** вычисление логического выражения происходит не до, а после очередного повторения цикла. Из-за этого цикл **REPEAT** обязательно выполнится хотя бы раз, а цикл **WHILE** может не выполнится ни разу.
Если условие в **цикле ПОКА** является условием продолжения повторений, то условие в **цикле ДО** - условием выхода из цикла, его завершения. Поэтому для одной и той же задачи эти условия противоположны.

Пример 1. Составить программу подсчета суммы **S** первых 1000 членов гармонического ряда 1+1/2+1/3+1/4+...1/N, используя оператор цикла **REPEAT**.

```
Program Summa;
Var   S:real;N:integer;
Begin
  S:=0;N:=0;
  repeat
    N:=n+1;
    S:=s+1/n
  Until n>1000;
  Writeln(s);
End.
```

11.2.Поиск наибольшего числа

Предположим, нам необходимо ввести с клавиатуры **N** чисел, найти из них наибольшее и вывести его.
Для решения этой задачи предлагается следующий алгоритм:
1. Ввести первое число в переменную Max.
2. Ввести следующее число в переменную Next.
3. Если Next>Max, то Max:=Next.
   Пункты 2 и 3 повторять, пока не будут введены все числа.
4. Вывести значение переменной Max.

Действительно ли будет напечатано наибольшее из **N** чисел?
Докажем это.
После выполнения первого пункта в переменной **Max** находится наибольшее из уже введенных чисел. Это справедливо, т.к. введено лишь одно число.
Повторение пунктов 2 и 3, в сущности, представляет собой цикл, который выполняется, пока не будут введены все числа. Если перед очередным повторением цикла в **Max** находится наибольшее из введенных чисел, то после выполнения пунктов 2 и 3 там снова будет наибольшее из введенных чисел.
В последнем пункте значение **Max** будет выведено.
Этот пример показывает, что алгоритм можно доказать, как доказывают математическую теорему. Программируя доказательный алгоритм, можно не опасаться ошибок в алгоритме, конечно, если нет ошибок в доказательстве.

```
Program maximum;
Var    N, max, next, k: integer;
Begin
  Write('Введите количество чисел'); Readln(n);
  Write('Введите число'); Readln(max);
  k:=1;
  repeat
    Write('Введите число'); Readln(next);
    K:=k+1;
    If next>max then max:=next
  Until k=n;
  Writeln(max);
End.
```

Тренировочные задания

1. Введите с клавиатуры 6 чисел и определите их среднее арифметическое.

2. Напишите программу, которая вводит целые числа с клавиатуры и складывает их, пока не будет введено число 0.

3. Напечатайте 20 первых степеней числа 2.

4. Найдите минимальное число из **N** чисел.

5. Дано натуральное число. Выяснить, является ли оно *простым*, т.е. делится только на 1 и на само себя.

Разбор заданий урока 10.

1. Даны целые числа **a** и **b** (a>b). Определить:

a) Результат целочисленного деления a на b, не используя стандартную операцию целочисленного деления;

```
Program prim_10_1a;
Var   a, b, n: integer;
Begin
  Write('Введите два числа, a>b ');
  Readln(a,b);
  N=0;
  While a>b do
  Begin
    N:=n+1;
    A:=a-b
  End;
  Writeln(' результат целочисленного деления ',n);
End.
```

2. Известны оценки по информатике каждого из 20 учеников класса. В начале списка перечислены все десятки, затем все остальные оценки. Сколько учеников имеют по информатике оценку «10»?. Условный оператор не использовать.

```
Program prim_10_2;
Var   x ,n: word;
Begin
  Write('Введите оценку');
```

```
  Readln(x);
  N:=0;
  While x=5 do
  Begin
    N:=n+1;
    Write('Введите оценку');
    Readln(x);
  End;
  Writeln(' имеют отлично ',n, ' учеников');
End.
```

5.* В некоторой стране используются денежные купюры достоинством в **1**, **2**, **4**, **8**, **16**, **32** и **64**. Дано натуральное число **N**. Как наименьшим количеством таких денежных купюр можно выплатить сумму **N** (указать количество каждой из используемых для выплаты купюр)? Предполагается, что имеется достаточно большое количество купюр всех достоинств.

```
Program prim_10_5;
Var   n,k,x: word;
Begin
  Write('Введите число'); Readln(n);
  K:=64;
  While n>0 do
  Begin
    If n>=k then begin Writeln(n div k, 'купюр по ', k);n:=n mod k end;
    K:=k div 2
  End;
End.
```

Урок 12

Компьютеры имеют дело не только с числами. Едва ли не больше времени они бывают заняты обработкой текста. В Паскале для этого есть специальный тип данных, который называется **CHAR** (от слова *character - символ*).

12.1. Тип CHAR (символьный или строковый или литерный)

Его значениями являются отдельные символы: буквы, цифры, знаки.
Символьные константы заключаются в апострофы, например, 'A', 'B', 'C', '4', '7', ' '(*пробел*).
Символьные переменные описываются предложением:
Var *имя переменной*: **char**;

Символьные значения можно вводить и выводить, присваивать, сравнивать.
Ниже приведен пример, где выполняются все эти действия.

```
Var  x,y:char;
Begin
  Write('Введите символ');
  Readln(x);
  Y:='A';
  If x<y then write ('X') else write ('y');
  {на экран буде выдан символ хранящийся в переменной
  X или Y в зависимости от проверки условия}
End.
```

Сравнивать символы можно благодаря тому, что в машинной памяти они хранятся в виде целых чисел (кодов символов). Из двух символов большим считается тот, код которого больше.
Символы упорядочены следующим образом:
'A'<'B'<...<'Z'
'a'<'b'<...<'z'
'0'<'1'<...<'9'
'а'<'б'<...<'я'
'А'<'Б'<...<'Я'
Для символов допустимы все шесть операций сравнения: =, <=,>=,<,>,<>.

12.2. Стандартные символьные функции

В Паскале имеются стандартные символьные функции:
CHR(N) - возвращает в программу символ с кодом N,
ORD(S) - возвращает код символа S,
PRED(S) -возвращает предыдущий символ
SUCC(S) - возвращает следующий символ

Примеры:
CHR(128) = Б
ORD(':') = 58
PRED('Б') = А
SUCC('Г') = Д

Каждый символ имеет свой уникальный двоичный код. Коды всех символов сведены в таблицу. Первая половина таблицы стала международным стандартом, который называется **ASCII - American Standard Code Information Interchange** (*читается «аски код»*), в ней кроме прочего содержится латинский алфавит, вторая имеет разные варианты для разных языков. Кириллица (*русский алфавит*) имеет несколько стандартов. В Паскале используется стандарт **КОИ-8**.

Пример использования переменной символьного типа: Составить программу, по которой компьютер многократно вычисляет сумму **A+B** при различных значениях **A** и **B**. В конце каждого этапа появляется запрос о продолжении или прекращении вычислений: *«Завершить программу? (Д/Н)»*.

```
Var A,B:real; { PROGRAM имя; не обязательное предложение}
    C : char;
Begin
  repeat
    Write('Введите два числа'); Readln(a,b);
    Writeln(a+b:0:2);
    Writeln('Завершить программу?(Д/Н)');
    Readln(c);
  Until c='Д'; {программа завершит работу если будет введено Д}
End.
```

Тренировочные задания

1. Что вернет функция **CHR(ORD(X))**?

2. Определить значения следующих функций (как правило, таблица аски кодов есть в любом справочнике по программированию):

CHR(68)
ORD('d')
PRED(1)
SUCC('Я')

3. С клавиатуры вводится два числа. Составить программу, сравнивающую эти числа и в зависимости от результата сравнения, выводящую на экран нужный знак. Например: 3<5 или 3=3 или 3>2. Процедуру *WRITE* для вывода результата разрешается использовать только один раз.

4. Составить программу, по которой компьютер находит произведение нечетных чисел, начиная с единицы, и до тех пор, пока на вопрос, задаваемый после каждого шага вычислений: *«Продолжить вычисления? (Д/Н)»,* отвечают '**Д**'.

Разбор заданий урока 11.

2. Напишите программу, которая вводит целые числа с клавиатуры и складывает их, пока не будет введено число 0.

```
Program sum;
Var N,s: integer;
Begin
  S:=0;
  repeat
    Write('Введите число'); Readln(n);
    S:=s+n;
  Until n=0;
  Writeln('S=',s);
End.
```

3. Напечатайте 20 первых степеней числа 2.

```
Var N,s:longint;
Begin
  S:=1; n:=1; {Начальное значение S присваиваем 1, при нулевом
       начальном значении умножение всегда даст в результате 0}
  repeat
    s:=s*2;
    Write(s,' ');
    N:=n+1;
  Until n>20;
End.
```

5. Дано натуральное число. Выяснить, является ли оно простым, т.е. делится только на 1 и на само себя.

Для решения этой задачи надо исходное число делить последовательно на 2, 3, 4 и т.д. до тех пор, пока исходное число не разделится без остатка. После выхода из цикла мы проверим, чему равен делитель, на который разделилось число.
Если исходное число простое, то оно разделится без остатка только само на себя (единицу исключаем сразу).

```
Program pr_11_5;
Var N, k: word;
Begin
  Write('Введите число');
  Readln(n);
  k:=1;
  repeat
    K:=k+1;
  Until n mod k = 0;
  If k=n then Writeln('число простое') else write('число не простое');
End.
```

Урок 13

Сегодня мы познакомимся с оператором, который обеспечивает повторение цикла, управляемое переменной.

13.1. Цикл ПЕРЕСЧЕТ (прямой)

Формат оператора:

FOR *переменная* **:=** *выражение* 1 **TO** *выражение* 2 **DO** *оператор*;

Переменная должна быть порядкового типа. Порядковыми называются все простые типы, значения которых можно расположить в возрастающем порядке. Из известных нам это: **INTEGER**, **WORD**, **LONGINT**, **BYTE**, **CHAR**. ***Выражение 1*** и ***Выражение 2*** должны быть того же типа, что и переменная. Чтобы цикл выполнился хотя бы раз ***выражение 1*** должно быть не больше ***выражения 2***.

Выполнение начинается с вычисления значений ***выражения 1*** и ***выражения 2***, затем переменная получает значение ***выражения 1*** и делается проверка, не превышает ли значение переменной ***выражения 2***. Если не превышает, выполняется оператор стоящий после служебного слова **DO**. После завершения оператора переменная получает следующее по порядку значение, и все повторяется, начиная с проверки.
Когда значение переменной становится равным ***выражению 2***, оператор выполняется последний раз.

ПРИМЕР 1. Напечатать ряд из повторяющихся чисел 20 в виде:
20 20 20 20 20 20 20 20 20 20

Код программы:

```
var  i: byte;
begin
  for I:=1 to 10 do write(20,' ');
end.
```

ПРИМЕР 2. Напечатать числа следующим образом:
10 10.4
11 11.4
...
25 25.4

Код программы (переменная используется не только для управления циклом но и для вывода на экран в качестве результата):

```
var  i: byte;
begin
  for I:=10 to 25 do write(I,' ',I+0.4:0:1);
  {при сложении целого I и вещественного 0.4 получаем
  вещественный результат, значит надо выполнить его
  форматирование при выводе на экран}
end.
```

13.2. Цикл ПЕРЕСЧЕТ (обратный)

Возможен вариант оператора, когда переменная принимает последовательно убывающие значения.
Формат оператора:

FOR *переменная* **:=** *выражение* 1 **DOWNTO** *выражение* 2 **DO** *оператор*;

В этом случае, чтобы цикл выполнился хотя бы раз, ***выражение 1*** должно быть не меньше ***выражения 2***.
Например:

For c:='z' **downto** 'a' **do** writeln(c);

Тренировочные задания

1. Напечатать столбиком:

а) все целые числа от **20** до **35**;

б) квадраты всех целых чисел от **10** до **b** (значение **b** вводится с клавиатуры; b>=10);

в) третьи степени всех целых чисел от **a** до **50** (значение **a** вводится с клавиатуры; a<=50);

г) все целые числа от **a** до **b** (значения **a** и **b** вводятся с клавиатуры; b>=a).

2. Напечатать числа следующим образом

25 25.5 24.8

26 26.5 25.8

. . .

35 35.5 34.8

3. Распечатать в столбик таблицу умножения на 7.

4. Вывести столбиком следующие числа: 2,1 2,2 2,3 . . . , 2,8

5. Вывести столбиком следующие числа: 2,2 2,4 2,6 . . . , 4,0 4,2

6. Вывести столбиком следующие числа: 4,4 4,6 4,8 . . . , 6,2 6,4

Разбор заданий урока 12.

1. CHR(ORD(X))=X

2. Определить значения следующих функций (как правило, таблица аски кодов есть в любом справочнике по программированию):

CHR(68)=D

ORD('d')=100

PRED(1) =0

SUCC('Я')=a

3. С клавиатуры вводится два числа. Составить программу сравнивающую эти числа и в зависимости от результата сравнения выводящую на экран нужный знак. Например: 3<5 или 3=3 или 3>2. Процедуру WRITE для вывода результата разрешается использовать только один раз.

```
Var x,y:integer;
    C:char;
Begin
  Write('Введите два числа');
  Readln(x,y);
  If x=y then c:='=';
  If x>y then c:='>';
  If x<y then c:='<';
  Writeln(x,c,y);
End.
```

Урок 14

На предыдущих уроках вы познакомились с операторами служащими для организации циклов. Сегодняшний урок мы посвятим решению задач связанных с проверкой различных свойств натуральных чисел. К натуральным числам, относятся целые положительные числа.

Задача 1

Дано натуральное число **N**. Определить, является ли оно простым, т.е. делится нацело только на **1** и на само себя. (Мы решали эту задачу в 12 уроке, но я хочу предложить вам другой алгоритм решения.)

Для решения необходимо проверить, делится ли исходное число на числа от **2** до **N-1**. Если число делится хотя бы на одно из чисел без остатка, то число **N** не будет простым. Мы можем уменьшить интервал проверяемых делителей так, как наибольший из возможных, это **N/2**. Введем так же дополнительную переменную **F**, она будет принимать значение **1**, если обнаружится хотя бы один из делителей для исходного числа. Первоначально присвоим **F** значение **0**, т.е. предположим, что число простое.
Переменные:
N - исследуемое число;
i - переменная цикла;
F - вспомогательная переменная.

Код программы.

```
Var  i,n,f:word;
Begin
  Write('Введите натуральное число');
  Readln(n);
  F:=0;
  For i:=2 to n div 2 do if n mod i=0 then f:=1;
  if f=1 then writeln('число',n:6,' не простое')
  Else writeln('число',n:6,' простое');
End.
```

Количество выполнения циклов можно еще уменьшить. Ведь на самом деле если число **N** делится на какое-то число **A** без остатка (кроме **1** и самого себя), то оно имеет и второй делитель **B=N/A**, т.е. если число не простое, то его всегда можно записать как **N=A·B**. При возрастании первого делителя, значение второго будет уменьшаться. Получается, что достаточно проверить все числа от **2** до **SQRT(n)** (корень квадратный из n), но так как результат вычисления корня вещественный, а параметр цикла должен иметь целое значение, применим функцию определения целой части числа **TRUNC**, и получим:

For i:=2 **to** trunc(sqrt(n)) **do if** n mod i=0 **then** f:=1;

Задача 2

Даны натуральные числа **M** и **N**. Определить, являются ли они взаимно простыми. Взаимно простые числа не имеют общих делителей, кроме **1**.

Для решения задачи:
- вводим натуральные числа **M** и **N**;
- в цикле от **2** до наименьшего числа порождаем **i** и проверяем, является ли оно одновременно делителем **M** и **N**;
- в зависимости от значения **F** выводим результат.

```
Var  k,m,n,f:word;
Begin
  Write('Введите 2 натуральных числа');
  Readln(n,m);
  F:=0;
  if n>m then k:=m else k:=n; {k - наименьшее из 2-х чисел}
  for i:=2 to k do if (n mod I=0) and (m mod i = 0) then f:=1;
```

```
  if f=1 then writeln('числа не взаимно простые')
    Else writeln('числа взаимно простые');
End.
```

Тренировочные задания

1. Дан интервал натуральных чисел от **N** до **M**. Определить все простые числа в этом интервале.

2. Дано натуральное число **N**. Определить все простые числа не превосходящие **N**.

3. Дано натуральное число **N**. Разложить его на простые множители.

4. Дано натуральное число **N**. Определить, является ли оно совершенным. Совершенное число **N** равно сумме всех своих делителей, не превосходящих само **N**.

5. Дано натуральное число **N**. Определить, является ли оно автоаморфным. Автоаморфное число **N** равно последним разрядам квадрата этого числа: 5<->25, 6<->36, 25<->625.

Разбор заданий урока 13

3. Распечатать в столбик таблицу умножения на 7.

```
var  i: byte;
begin
  for i:=1 to 10 do writeln(7,'*',i,'=',7*i);
end.
```

4. Вывести столбиком следующие числа: 2,1 2,2 2,3 . . . , 2,8

```
var  i: byte;
begin
  for i:=21 to 28 do writeln(i/10:0:1);
end.
```

5. Вывести столбиком следующие числа: 2,2 2,4 2,6 . . . , 4,0 4,2

```
var  i: byte;
begin
  for i:=11 to 21 do writeln(i*2/10:0:1);
end.
```

Урок 15

Сегодняшний урок мы посвятим обработке числовых последовательностей. Для этого обычно используется оператор цикла.
Алгоритмы для обработки последовательностей чаще относятся к одному из двух типов: поиск, проверка условий.
Для последовательностей характерно, что в каждый момент времени нам доступен только один элемент последовательности. Поэтому все алгоритмы строятся с учетом однократного последовательного просмотра.
Рассмотрим несколько программ. В каждой из них одновременно рассматривается только очередной член последовательности. Алгоритмы для решения таких задач называются *алгоритмами с линейным поиском*.

Задача 1

Вводится последовательность из **N** целых чисел. Найти сумму всех отрицательных чисел.

```
Var  i,n,x,sum:integer;
Begin
  Write('Введите длину последовательности N=');
  Readln(n);
  Sum:=0;
  For i:=1 to n do
  Begin
    Write('Введите число');
    Readln(x);
    if x<0 then sum:=sum+x
  end;
  if sum=0 then writeln('отрицательных чисел нет')
    else writeln('сумма отрицательных чисел =', sum);
End.
```

Задача 2

Вводится последовательность ненулевых чисел, **0** – конец последовательности. Определить, сколько раз последовательность меняет знак.

```
Var  old, new: real;
     K: integer;
Begin
  Write('введите число');
  Readln(old);
  Write('введите число');
  Readln(new);
  K:=0;
  Repeat
    if new*old<0 then k:=k+1;
    Old:=new;
    Write('введите число');
    Readln(new);
  Until new=0;
  if k>0 then writeln ('Последовательность меняет знак ',k,' раз')
    else writeln ('Последовательность не меняет знак ');
end.
```

Тренировочные задания

1. Вводится последовательность из **n** произвольных чисел. Определить, сколько раз последовательность меняет знак.
2. Вводится последовательность чисел, **0** – конец последовательности. Определить, содержит ли последовательность хотя бы два равных соседних числа.
3. Вводится последовательность чисел, **0** – конец последовательности. Найти два наименьших

числа.
4. Вводится последовательность из **N** целых чисел. Найти наибольшее из всех отрицательных чисел.
5. Вводится последовательность из **N** целых чисел. Найти, сколько в ней нулей.

Разбор заданий урока 14

3. Дано натуральное число **N**. Разложить его на простые множители.

```
Var  n,i,j:word; F:Byte;
Begin
  Write('Введите натуральное число');
  Readln(n);
  Write(n:6,'=1');     {любое число имеет множитель 1}
  F:=0;
  j:=n;
  for i:=2 to n div 2 do
  begin
    if j mod i=0 then
    begin f:= 1; {найден множитель больше 1}
      while j mod i = 0 do
     {цикл определяет, сколько множителей i в исходном числе N}
      begin
        write('*',i);
        j:=j div i
      end;
    end;
  end;
  if f=0 then writeln('*',n);
End.
```

4. Дано натуральное число **N**. Определить, является ли оно совершенным. Совершенное число **N** равно сумме всех своих делителей, не превосходящих само **N**.

```
Var  n,i,sum:word;
Begin
  Write('Введите натуральное число');
  Readln(n);
  Sum:=0;
  for i:=1 to n div 2 do if n mod i=0 then sum:=sum+i;
  if sum=n then writeln('число ',n,' совершенное')
    else writeln(('число ',n,' не совершенное' );
End.
```

5. Дано натуральное число **N**. Определить, является ли оно автоморфным. Автоморфное число **N** равно последним разрядам квадрата этого числа: 5<->25, 6<->36, 25<->625.

```
Var  n,r,m:word;
Begin
  Write('Введите натуральное число');
  Readln(n);
  M:=n; r:=1;
  While m>0 do    {в цикле определяем разрядность введенного числа (r)}
  Begin
    M:=m div 10;
    R:=r*10;
  End;
  if (n*n mod r)=n then writeln('число ',N,' автоморфно')
    else writeln(('число ',N,' не автоморфно');
End.
```

Урок 16

Оставим пока числовые последовательности, их существует большое многообразие, и позднее мы к ним вернемся. А сейчас попробуем решить следующую задачу:

Составить программу, которая в зависимости от порядкового номера дня недели (1, 2, ..., 7) выводит на экран его название (понедельник, вторник, ..., воскресенье).
Для этого воспользуемся условным оператором.

```
Var    x: byte;
Begin
  Write(' введите число от 1 до 7');
  Readln(x);
  if x=1 then writeln('понедельник');
  if x=2 then writeln('вторник');
  if x=3 then writeln('среда');
  if x=4 then writeln('четверг');
  if x=5 then writeln('пятница');
  if x=6 then writeln('суббота');
  if x=7 then writeln('воскресенье');
End.
```

А если бы речь шла о названия месяца, то операторов пришлось бы использовать еще больше, так как условный оператор позволяет осуществлять ветвление программы только по двум направлениям, одно из которых соответствует выполнению проверяемого условия, а другое – невыполнению этого же условия. А если для переменной необходимо выполнить в зависимости от условий ряд действий придется использовать вложенные операторы или несколько операторов подряд.
В таком случае лучше подойдет ***оператор ВАРИАНТА*** (или ***ВЫБОРА***).

Если условный оператор напоминает дорожную развилку, то оператор выбора – это разделение пути на множество дорог, по одной из которых пойдет выполнение программы.
Формат оператора:

```
  CASE выражение OF
    P1:<оператор 1>;
    P2:<оператор 2>;
    . . .
    PN:<оператор N>;
      ELSE <оператор N+1>
  END;
```

Выражение порядкового типа вычисляется, и его значение отыскивается в одном из списков констант. После этого выполняется соответствующий оператор. Если значение выражения не совпало ни с одной из меток, то выполняется оператор из строки ELSE. Сокращенная форма оператора не содержит ELSE. Метки оператора варианта могут быть константами любого типа, но их тип должен совпадать с типом выражения.
Тогда код нашей программы будет выглядеть иначе:

```
Var  x: byte;
Begin
  Write(' введите число от 1 до 7'); Readln(x);
  Case x of
    1: writeln('понедельник');
    2: writeln('вторник');
    3: writeln('среда');
    4: writeln('четверг');
    5: writeln('пятница');
    6: writeln('суббота');
    7: writeln('воскресенье');
  end;
End.
```

Если при выборе альтернативы необходимо выполнение нескольких операторов, то нужно заключить их в операторные скобки **BEGIN . . . END;**, т.е. использовать составной оператор.
При использовании оператора **CASE** можно использовать диапазон допустимых значений параметра. Например, чтобы определить, введена ли как символьная переменная цифра, можно написать:

```
  CASE i OF
    '0'..'9':writeln('цифра');
    ELSE writeln ('не цифра')
  END;
```

Задача. Для целого числа **K** от **1** до **99** напечатать фразу «*Мне k лет*», учитывая при этом, что при некоторых значениях **K** слово «*лет*» надо заменить на слово «*год*» или «*года*». Например, *11 лет*, *22 года*, *51 год*. (Данное решение можно улучшить, используя дополнительно условный оператор. Попробуйте.)

```
var k:byte;
begin
  write('Введите число лет'); readln(k);
  case k of
    1,21,31,41,51,61,71,81,91:writeln('Мне ',k,' год');
    2..4,22..24,32..34,42..44,52..54,62..64,72..74,82..84,92..94:writeln('Мне
',k,' года');
    else writeln('Мне ',k,' лет');
  end;
end.
```

Тренировочные задания

1. Составить программу, которая в зависимости от порядкового номера месяца (1, 2, ..., 12) выводит на экран его название (январь, февраль, ..., декабрь).

2. Написать программу, которая бы по введенному номеру месяца выдавала соответствующее этому месяцу время года.

3. Составить программу, которая читает натуральное число **N** в десятичном представлении (*N<=10000*), а на выходе выдает это же число в десятичном представлении и на естественном языке. Например: *7 – семь*, *204 – двести четыре*, *52 – пятьдесят два*.

Задачи на повторение

4. Дано натуральное число. Определить, является ли разность его максимальной и минимальной цифр четным числом.

5. Дано натуральное число. Определить, сколько раз в нем встречается цифра, равная старшей.

Разбор заданий урока 15

3. Вводится последовательность чисел, **0** – конец последовательности. Найти два наименьших числа.

```
var x,min1,min2:integer;
begin
  write('Введите число '); readln(x);
  min1:=x;
  write('Введите число '); readln(x);
  min2:=x;
  repeat
   if x<=min1 then
   begin
     min2:=min1;
     min1:=x
   end
   else if x<min2 then min2:=x;
   write('Введите число '); readln(x);
  until x=0;
  write('Два наименьших числа ',min1,' и ', min2);
end.
```

4. Вводится последовательность из **N** целых чисел. Найти наибольшее из всех отрицательных чисел.

```
var i,n,x,max:integer;
begin
  write('Введите длину последовательности N= '); readln(n);
  repeat
    write('Введите число '); readln(x);
    n:=n-1;
    if x<0 then max:=x;
  until (x<0) or (n=0);
  for i:=1 to n do
  begin
    write('Введите число'); readln;
    if (x<0)and(max<x) then max:=x
  end;
  if max=0 then writeln('отрицательных чисел нет ')
    else writeln('наибольшее из отрицательных ', max);
end.
```

Урок 17

Сегодняшний урок мы посвятим проблеме правильного ввода данных, точнее, контроля за вводом данных. Обычно в условии задачи указывается диапазон допустимых входных значений. И если данные введены некорректно, то даже верная программа может выдать неправильный ответ.

Например:
Составить программу, по которой компьютер печатает последовательность вида:

10, 100, 1000, ..., 10^n (n<10)

```
Код программы:
var i,n,s:longint;
begin
  write('Введите значение n<10');
  readln(n);
  s:=1;
  for i:=1 to n do
  begin
    s:=s*10;{находим степень}
    write(s,' ');
  end;
  readln;
end.
```

Если не сделать защиту, то при вводе числа больше 9 программа выполнится, но ответ будет абсурдным (так как пойдет переполнение допустимого значения переменной S). Попробуйте! А если сделать проверку входного данного, то программа не будет выполняться, пока не будет введено допустимое значение.

```
var i,n,s:longint;
begin
  repeat {Цикл закончится если n<10}
    write('Введите значение n<10');
    readln(n);
  until n<10;
  s:=1;
  for i:=1 to n do
  begin
    s:=s*10;{находим степень}
    write(s,' ');
  end;
  readln;
end.
```

Существуют и другие методы контроля, но мы пока остановимся на контроле диапазона входных данных. Контроль за типом входных данных выполняет компилятор. Например, в случае, если в переменную целого типа вы попытаетесь ввести вещественное число, он просто прервет выполнение программы.

Тренировочные задания

Постарайтесь в программах сделать контроль на допустимость вводимых данных.

1. Дано натуральное число N. Определить, является ли оно палиндромом. Число палиндром можно читать справа налево и слева направо: 4, 88, 121, 767767 и т.д.
2. Вводится последовательность из N целых чисел. Найти наибольшее число.
3. Дано натуральное число. Найти:
- **a.** Число, получаемое при прочтении его цифр справа налево;
- **b.** Число, получаемое в результате приписывания по двойке в начало и конец записи исходного числа;
- **c.** Число, получаемое удалением из исходного всех цифр А;
- **d.** Число, получаемое из исходного перестановкой его первой и последней цифр;
- **e.** Число, образованное из исходного приписыванием к нему такого же числа.

Разбор тренировочных заданий урока 16.

3. Составить программу, которая читает натуральное число N в десятичном представлении (N<=10000), а на выходе выдает это же число в десятичном представлении и на естественном языке. Например: 7 - семь, 204 - двести четыре, 52 - пятьдесят два.

```
var n,k:word;
begin
  write('Введите число '); readln(n);
  k:=n div 100;
  n:=n mod 100;
  case k of
    9:write('девятьсот ');
    8:write('восемьсот ');
    7:write('семьсот ');
    6:write('шестьсот ');
    5:write('пятьсот ');
    4:write('четыреста ');
    3:write('триста ');
    2:write('двести ');
    1:write('сто ');
  end;
  k:=n div 10;
  if k<>1 then n:=n mod 10;
  case k of
    9:write('девяносто ');
    8:write('восемьдесят ');
    7:write('семьдесят ');
    6:write('шестьдесят ');
    5:write('пятьдесят ');
    4:write('сорок ');
    3:write('тридцать ');
    2:write('двадцать ');
  end;
  case n of
    19:write('девятнадцать ');
    18:write('восемнадцать ');
    17:write('семнадцать ');
    16:write('шестнадцать ');
    15:write('пятнадцать ');
    14:write('четырнадцать ');
    13:write('тринадцать ');
    12:write('двенадцать ');
    11:write('одиннадцать ');
    10:write('десять ');
    9:write('девять ');
    8:write('восемь ');
    7:write('семь ');
    6:write('шесть ');
    5:write('пять ');
    4:write('четыре ');
    3:write('три ');
    2:write('два ');
    1:write('один ');
  end;
  readln;
end.
```

Задачи на повторение:

4. Дано натуральное число. Определить, является ли разность его максимальной и минимальной цифр четным числом.

```
var k,n,m,max,min:word;
begin
  write('Введите натуральное число '); readln(n);
  max:= n mod 10;
  min:=n mod 10;
  n:=n div 10;
  while n div 10<>0 do
  begin
    k:=n mod 10;
    if k>max then max:=k;
    if k<min then min:=k;
    n:=n div 10;
  end;
  if k>max then max:=k;
  if k<min then min:=k;
  if (max-min) mod 2=0 then writeln('разница четное число ')
    else writeln('разница не четное число ');
  readln;
end.
```

5. Дано натуральное число. Определить, сколько раз в нем встречается цифра, равная старшей.

```
var k,n,m,i:longint;
begin
  write('Введите натуральное число '); readln(n);
  m:=n;
  while m div 10<>0 do m:=m div 10; {определяем цифру старшего разряда}
  i:=1; { один раз уже встретилась в старшем разряде}
  while n div 10<>0 do
  begin
    k:=n mod 10;
    if k=m then i:=i+1;
    n:=n div 10;
  end;
  writeln('старшая цифра встречается в числе ',i,' раз');
  readln;
end.
```

Урок 18

В уроке 12 вы уже познакомились с символьным типом данных CHAR, который позволяет работать с отдельными символами текста. Для обработки более крупных текстовых единиц - строк введен тип данных, который называется STRING (строка).

Значениями этого типа являются строки любых символов длиной до 255.

Переменные строки должны быть описаны предложением:

```
VAR имя: STRING
```

Строки можно присваивать, сравнивать, вводить, выводить и соединять. Соединение обозначается знаком "+". Вот примеры некоторых операций сравнения над строками:

'стол'<= 'столик ' true
'ABC' <'ADBA' true
'12' <'2' true
'пар'+ 'о' +'воз' 'паровоз'

На основе этих примеров сформулируйте правила сравнения строк.

Среди всевозможных значений строк есть пустая строка. Она изображается двумя апострофами (одинарными кавычками), между которыми ничего нет. Чтобы ввести этот символ в состав строки, надо повторить его дважды. Например, оператор

write('об''явление')

выведет на экран: об'явление.

Программисту доступны отдельные символы строковой переменной, для этого кроме имени переменной надо указать порядковый номер символа в строке. Например, если описана переменная X:STRING, то X[1] - это первый символ строки, X[2] - второй и т.д.

У X[0] особая роль - хранить длину строки. Значением X[0] является символ, код которого равен количеству символов в строке. Но для определения длины строковой переменной обычно используется функция

```
LENGTH(строковая переменная).
```

Например, если N:=LENGTH(x); - N присвоится значение равное числу символов в строке.

При описании строковой переменной мы можем ограничить длину строки, указав ее максимально возможный размер, тогда в строке будет храниться только указанное число символов.

```
Var
  a,b:string[4];
begin
  write('введите слово');
  readln(a);
  write(a);
  readln
end.
```

Если при выполнении этой программы ввести слово КУКУРУЗА, то программа выведет КУКУ.

ЗАПОМНИТЕ. Если при выполнении программы необходимо ввести значение для нескольких строковых переменных, для каждой из них должен быть указан свой оператор ввода READLN. Например,

```
Var
  a,b,c:string;
begin
  readln(a);
  readln(b);
  readln(c);
  write(a+b+c);
  readln
end.
```

Проверьте, что произойдет, если записать READLN(a,b,c); или READ(a,b,c).

Пример 1.

Составить программу определяющую, какая из двух фамилий длиннее. Фамилии имеют разную длину.

```
Var
  a,b:string;
begin
  readln(a);
  readln(b);
  if length(a)>length(b) then write(a) else write(b);
  readln
end.
```

Пример 2.

Даны два слова. Составить программу определяющую верно ли, что первое слово начинается на ту же букву, которой оканчивается второе слово.

```
Var  x,y:byte;
     a,b:string;
begin
  readln(a);
  readln(b);
  x:=length(b); {определяем длину слова b, чтобы узнать номер последнего символа}
  if a[1]=b[x] then write('верно') else write('неверно');
  readln
end.
```

Тренировочные задания

1. Дано название города. Определить, четно или не четно количество символов в нем.

2. Дано слово. Вывести на экран его третий символ и дважды его последний символ.

3. Дано слово. Верно ли, что оно начинается и оканчивается на одну и ту же букву?

4. Дано слово. Получить и вывести на экран буквосочетание, состоящее из его третьего и последнего символа.

5. Составить программу которая запрашивает название футбольной команды и повторяет его на экране со словами: "Это чемпион!".

Разбор заданий урока 17.

1. Дано натуральное число N. Определить, является ли оно палиндромом. Число палиндром можно читать справа налево и слева направо: 4 88 121 767767 и т.д.

```
var k,n,m:longint;
begin
  Repeat
    write('Введите натуральное число '); readln(n);
  until n>0;
  m:=n; k:=0;
  while m div 10<>0 do {собираем число в обратном порядке}
  begin
    k:=k*10+m mod 10;
    m:=m div 10
  end;
  k:=k*10+m mod 10;
  if k=n then writeln('данное число палиндром')
    else writeln('это не палиндром');
  readln;
end.
```

3. Дано натуральное число. Найти:

a) Число, получаемое при прочтении его цифр справа налево;

b) Число, получаемое в результате приписывания по двойке в начало и конец записи исходного числа;

c) Число, получаемое удалением из исходного всех цифр А;

d) Число, получаемое из исходного перестановкой его первой и последней цифр;

e) Число, образованное из исходного приписыванием к нему такого же числа.

```
var  k,n,m,p:longint;
     a:byte;
begin
{a} Repeat {проверка ввода}
    write('Введите натуральное число '); readln(n);
```

```
  until n>0;
  m:=n; k:=0;
  while m div 10<>0 do {собираем число в обратном порядке}
  begin
   k:=k*10+m mod 10;
   m:=m div 10;
  end;
  k:=k*10+m mod 10;
  writeln('обратное число=',k);
  {б} Repeat {проверка ввода}
      write('Введите натуральное число '); readln(n);
  until n>0;
  m:=n;
  k:=10;
  while m div 10<>0 do {определяем разрядность числа} -
  begin
    k:=k*10;
    m:=m div 10;
  end;
  m:=(2*k+n)*10+2;
  writeln('число с двойками=',m);
  {в} Repeat {проверка ввода}
      write('Введите натуральное число '); readln(n);
  until n>0;
  Repeat {проверка ввода}
    write('Введите цифру ');readln(a);
  until (a>=0) and (a<10);
  m:=n; k:=1; p:=0;
  while m div 10<>0 do {собираем число исключая цифру}
  begin
    if m mod 10<>a then
    begin
      p:=p+(m mod 10)*k;
      k:=k*10
    end;
    m:=m div 10;
  end;
  if m mod 10<>a then p:=p+m mod 10*k;
  writeln('число без цифры ',a:2,'=',p);
  {г} Repeat {проверка ввода}
      write('Введите натуральное число больше 9='); readln(n);
  until n>9;
  m:=n; k:=10; p:=0;
  a:=m mod 10;
  m:=m div 10;
  while m div 10<>0 do {собираем число, исключая цифру}
  begin
    p:=p+(m mod 10)*k;
    k:=k*10;
    m:=m div 10
  end;
  p:=a*k+p+m;
  writeln('число с перестановкой первой и последней цифр ',a:2,'=',p);
  {д} Repeat {проверка ввода}
      write('Введите натуральное число '); readln(n);
  until (n>0)and(n<21474);
  m:=n; k:=10;
  while m div 10<>0 do {определяем разрядность}
  begin
    k:=k*10;
    m:=m div 10
  end;
  p:=n*k+n;
  writeln('число= ',a:2,'=',p);
end.
```

Урок 19

Из предыдущего урока вы узнали, что можно обращаться к отдельным символам строки, указав их порядковый номер. Сегодня мы разберем задачи, в которых используется это свойство строковой переменной.

Пример 1.
Дано слово, состоящее из четного числа букв. Вывести на экран его первую половину.

```
Var  i,x:byte;
     a:string;
begin
  repeat
    write('Введите слово из четного числа букв');
    readln(a);
    x:=length(a); {определяем длину слова}
  until (x mod 2 = 0);
  x:= x div 2; {применяем целочисленное деление}
  for i:=1 to x do write(a[i]);
end.
```

Пример 2.
Составить программу, которая печатает заданное слово в обратном порядке.

```
Var  i,x:byte;
     a:string;
begin
  write('Введите слово ');
  readln(a);
  x:=length(a);{определяем длину слова}
  for i:=x downto 1 do write(a[i]);
end.
```

Пример 3.
Дано предложение. Определить число пробелов в нем.

```
Var  i,x,k:byte;
     a:string;
begin
  write('Введите предложение');
  readln(a);
  x:=length(a); {определяем длину слова}
  k:=0;
  for i:=1 to x do if a[i]=' ' then k:=k+1;
  writeln(k);
end.
```

Пример 4.
Дано предложение. Определить порядковый номер первой встреченной буквы 'к'. Если такой буквы нет, сообщить об этом.

```
Var  i,x,k,f:byte;
     a:string;
begin
  write('Введите предложение');
  readln(a);
  x:=length(a);{определяем длину слова}
  k:=0; I:=0; f:=0;
  repeat
    I:=I+1;
    if a[i]<>'к' then k:=k+1 else f:=1;
  until (I=x)or( a[i]='к' );
  if f=1 then writeln(k) else writeln(' Нет такой буквы');
end.
```

Тренировочные задания.

1. Дано слово. Получить его часть, образованную идущими подряд буквами, начиная с m-й и кончая n-й (m<n).
2. Дано слово из четного числа букв. Поменять местами его половины.
3. Дано слово. Перенести первые k его букв в конец.
4. Получить строку, состоящую из пяти эвездочек, т.е. символов "*".
5. Дано предложение. Напечатать все его буквы "и".

Разбор заданий урока 18.

1. Дано название города. Определить, четно или не четно количество символов в нем.

```
Var  x:byte;
     a:string;
begin
  write('Введите слово');
  readln(a);
  x:=length(a);
  if odd(x) then write('не четно') else write('четно');
end.
```

Значение функции odd(x) истинно, если число x не четное.

2. Дано слово. Вывести на экран его третий символ и дважды его последний символ.
3. Дано слово. Верно ли, что оно начинается и оканчивается на одну и ту же букву?
4. Дано слово. Получить и вывести на экран буквосочетание, состоящее из его третьего и последнего символа.

```
Var  x:byte;
     a:string;
begin
  repeat
    write('Введите слово не менее четырех букв');
    readln(a);
    x:=length(a);          {определяем длину слова}
  until x>= 4;
  write(a[3],a[x]);
end.
```

5. Составить программу которая запрашивает название футбольной команды и повторяет его на экране со словами: "Это чемпион!".

```
Var   a:string;
begin
  write('Введите название футбольной команды');
  readln(a);
  write(a, ' Это чемпион!');
end.
```

Урок 20

Для работы со строковыми переменными в Паскале существует набор стандартных процедур и функций. Их применение упрощает решение задач. Хочу напомнить что результат выполнения функции должен быть запомнен в переменной соответствующего типа, если конечно она, функция, не является элементом выражения.

20.1 Функция копирования строки или ее части.

S:=COPY(*строка*, *позиция*, N);

Функция копирования называется также "вырезкой". Результатом выполнения функции будет часть строки начиная с указанной позиции длиной N.

Код программы для решения задачи из 19 урока будет выглядеть иначе с применением данной функции.

Пример 1.

Дано слово, состоящее из четного числа букв. Вывести на экран его первую половину.

```
Var  x:byte;
     a:string;
begin
  repeat
    write('Введите слово из четного числа букв');
    readln(a);
    x:=length(a);          {определяем длину слова}
  until (x mod 2 = 0);
  x:= x div 2;          {применяем целочисленное деление}
  write(copy(a,1,x); {вместо оператора цикла}
end.
```

20.2 Функция поиска подстроки в строке.

N:=POS(*подстрока*, *исходная строка*)

Функция определяет, содержится ли подстрока в исходной строке, и если да то определяет номер символа в исходной строке с которого начинается подстрока. Если такого символа нет, то значение функции будет равно нулю. Подстрока может состоять и из одного символа.

Рассмотрим задачу из 19 урока.

Дано предложение. Определить порядковый номер первой встреченной буквы 'к'. Если такой буквы нет, сообщить об этом.

```
Var  x:byte;
     a:string;
begin
  write('Введите предложение');
  readln(a);
  x:=pos('к',a);
  if x=0 then writeln(' Такой буквы нет') else writeln(x);
end.
```

20.3.Процедура удаления части строки

DELETE(*строка*, *начальный номер*, *количество символов*)

Удаляет из исходной строки указанное количество символов.

Пример из 19 урока.

Дано слово, состоящее из четного числа букв. Вывести на экран его первую половину.

```
Var  i,x:byte;
     a,p:string;
begin
  repeat
    write('Введите слово из четного числа букв');
    readln(a);
    x:=length(a);          {определяем длину слова}
```

```
  until (x mod 2 = 0);
  x:= x div 2;          {применяем целочисленное деление}
  delete(a,x+1,x);
  write(a);
end.
```

20.4.Процедура вставки подстроки в строку

```
  INSERT(строка1, строка2, позиция);
```

Строка1 вставляется в строку2 начиная с указанной позиции.

Тренировочные задания.

1. Дано предложение. Определить число вхождений в него некоторого символа.
2. Дано предложение. Заменить в нем все вхождения буквосочетания "ах" на "ух".
3. Дано слово. Проверить, является ли оно "перевертышем", т.е. читается одинаково как с начала, так и с конца.
4. Дано слово:
  a. удалить из него первую из букв "о", если такая буква есть;
  b. удалить из него последнюю из букв "т", если такая буква есть.
5. Дано предложение. Удалить из него все буквы "с".

Разбор заданий урока 19.

1. Дано слово. Получить его часть, образованную идущими подряд буквами, начиная с m-й и кончая n-й (m<n).

```
Var  i,m,n:byte;
     a,b:string;
begin
  write('Введите слово');
  readln(a);
  write('Введите значения m и n (m<n)');
  readln(m,n);
  b:='';
  if length(a)>=n then for i:=m to n do b:=b+a[I];
  writeln(b);
end.
```

2. Дано слово из четного числа букв. Поменять местами его половины.

```
Var  i,x,y:byte;
     a,b:string;
begin
  repeat
    write('Введите слово из четного числа букв');
   readln(a);
    x:=length(a);          {определяем длину слова}
  until (x mod 2 = 0);
  y:= x div 2;
  b:='';         {применяем целочисленное деление}
  for i:=y+1 to x do b:=b+a[I];
  for i:=1 to y do b:=b+a[i];
  write(a);
end.
```

5. Дано предложение. Напечатать все его буквы "и".

```
Var  i,m,n:byte;
     a:string;
begin
   write('Введите слово');
   readln(a);
   x:=length(a);   {определяем длину слова}
   for i:=1 to x do if a[I]='и' then write('и');
end.
```

Урок 21

21.1. Преобразование числового значения в строковое.

```
STR(K, S);
```

Процедура преобразовывает числовое значение величины K в строку S. После числа K может записываться формат, аналогично формату вывода. Если в формате указано недостаточное для вывода количество разрядов, поле вывода автоматически расширятся до нужной длины.

Значение K	Процедура	Результат
145	STR(K:6,S)	'___145'
0.23E+04	STR(K:10,S)	'______2300'
45678	STR(K:3,S)	'45678'

21.2.Преобразование строки в число.

```
VAL(S, K, Code);
```

Процедура преобразует значение S в величину целочисленного или вещественного типа и помещает результат в K. Значение S не должно содержать незначащих пробелов в начале и в конце строки. Code - целочисленная переменная, если преобразование невозможно, то содержит номер позиции первого ошибочного символа, если все нормально, то значение Code равно нулю.

21.3.Преобразование строчной буквы в прописную.

```
UpCase(ch);
```

Функция символьная. Возвращает прописную латинскую букву, если ch - строчная. В остальных случаях возвращает аргумент без изменения.

Составить программу, которая после ввода строки строчных латинских букв заменяет их на прописные.

```
var s:string;
    I:byte;
begin
  Write(' Введите слово');
  Readln(s);
  For I:=1 to Length(s) do s[I]:=UpCase(s[I]);
  Writeln(s);
end.
```

Решим следующую задачу используя стандартные функции.
Дано слово. Переставить первые три и последние три буквы, сохранив порядок их следования.

```
var  x:byte;
     a:string;
begin
  repeat
    write('Введите слово не менее шести букв');
    readln(a);
    x:=length(a);          {определяем длину слова}
  until (x>=6);
  a:=copy(a,x-2,3)+copy(a,4,x-6)+copy(a,1,3);
  writeln(a);
end.
```

Тренировочные задания.

1. Вычислите длину самого короткого слова в предложении из трех слов, разделенных пробелами.
2. Заданы фамилия, имя и отчество учащегося, разделенные пробелом. Напечатайте его фамилию и инициалы.
3. Даны два слова. Составте программу, определяющую можно или нет из букв слова A составить слово B.
4. Даны два слова. Определить, сколько начальных букв первого слова совпадает с начальными буквами второго слова.
5. Дан текст. Верно ли, что в нем есть пять идущих подряд одинаковых символа?

Разбор заданий урока 20.

1. Дано предложение. Определить число вхождений в него некоторого символа.

```
Var  i,x,k:byte;
     a:string; ch:char;
begin
  write('Введите слово ');
  readln(a);
  write('Введите символ ');
  readln(ch);
  x:=length(a);   {определяем длину слова}
  k:=0;
  for i:=1 to x do if a[I]=p then k:=k+1;
  write('таких символов ',k);
end.
```

2. Дано предложение. Заменить в нем все вхождения буквосочетания "ах" на "ух".

```
Var  x:byte;
     a:string;
begin
  write('Введите предложение');
  readln(a);
  while pos(a,'ax')<>0 do
  begin
    x:=pos (a,'ax');
    delete(a,x,2)
    insert('yx',a,x)
  end;
  write( a );
end.
```

3. Дано слово. Проверить, является ли оно "перевертышем", т.е. читается одинаково как с начала, так и с конца.

```
Var  i,x:byte;
     a,p:string;
begin
  write('Введите слово ');
  readln(a);
  x:=length(a);         {определяем длину слова}
  p:='';    {пустая строка}
  for i:=x downto 1 do p:=a[I]+p;    {собираем слово в обратном порядке}
  if a=p then write('слово перевертыш') else write('слово не перевертыш');
end.
```