Тема: **Символьные и строковые переменные. Функции и процедуры для обработки символьных и строковых переменных.**

Цель: *Ознакомить с форматом описания символьных и строковых переменных.*

Сформировать знания об операциях над символьными и строковыми переменными.

Ознакомить с функциями и процедурами для обработки символов и строк.

СИМВОЛЬНЫЕ переменные

Символьный тип (char) – это тип данных для работы с отдельными символами в языке Pascal.

Значениями переменных этого типа являются отдельные символы: **буквы, цифры, знаки.**

В оперативной памяти переменная типа **char** занимает **1 байт=8 бит**.

Формат описания:

```
Var
    имя_переменной: char;
```

Например:

```
Var
    u, v, X1, X2: char;
```

Операции над символьными величинами

Над символьными переменными можно выполнять следующие действия:

присваивание, *например*, **u:=’a’; X1:=’D’;**

v:=’!’; X2:=#33; {нельзя написать u:=’abc’}

ввод, *например*, **read (u, v, X1,X2); readln (u, v, X1,X2);**

вывод, *например*, **write (u, v, X1,X2); writeln (u, v, X1,X2);**

сцепление *(***конкатенация***)* **–** операция соединения нескольких символов и обозначается символом + (плюс), *например*, **write (u+v+X1+X2);**

сравнение символов применяя операции сравнения: =, <=, >=, <, >, <>

- при проверке сравниваются коды символов
- символы упорядочены в соответствии с их кодами

(латинский алфавит и символы кириллицы: заглавные, строчные)

Например,

операция сравнения	**результат**
’7’<’c’	**true**
’b’=’B’	**false**

Пример 1: программа, которая выполняет операции **присваивание**, **ввод, вывод** и **сцепление** четырёх символов.

```
Program Primer1;
    var X1, X2, X3, X4: char;
Begin
    X4:='!';                    {присваивание значения}
    write ('Введите три символа:  ' );     {вывод сообщения на экран}
    read(X1, X2, X3);           {ввод трёх символов без
                                 пробелов}
    writeln (X1+X2+X3+X4);      {вывод результата операции
                                 сцепления}
end.
```

Результат выполнения программы может выглядеть:

```
Введите три символа:  УРА
УРА!
```

Критерии отметки:

- набор текста программы (**готовой**) 1 балл
- программа откомпилирована (без ошибок) 1 балл
- программа выдает результат 0,5 баллов
- программа выдает правильный результат 0,5 баллов
- программа правильно сохранена в своей папке 1 балл
- программа с комментариями 1 балл

- выполнено задание, предполагающее перенос опыта в нестандартную ситуацию 5 баллов

Пример 2: напечатать на экране строчные буквы латинского алфавита от **z** по **a**.

```
Program Lat_bukv;

 Var

   c:char;

begin

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

End.
```

Задание: **Составить программу, которая выведет на экран:**

а) **прописные буквы латинского алфавита от A по K;**

б) **символы с кодами от 60 до 71.**

Пример 3: программа для подсчёта количества ! среди символов s_1, …, s_n до 1-ой *

```
Program pr3;
Var
      S:char;
      k: integer;
Begin
      k:=0;
      Readln(s);          {ввод 1, a, b, !, !, c, !, *, !, *,  – до 1-ой звездочки}
      While s<>'*' do
            begin
                  If s='!' then   k:=k+1;
                  Readln(s);
            end;
      Writeln(k);         {результат  3}
End.
```

*Если используется оператор **read**, то символы вводятся без пробелов и без всяких знаков.*

ФУНКЦИИ ДЛЯ ОБРАБОТКИ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ

Succ(x1) **– возвращает следующий символ;**

Pred(x1) **– возвращает предыдущий символ;**

Ord(x1) **– возвращает значение кода символа;**

Chr(n) **– возвращает символ с кодом *n*.**

Например,

```
x1:= Succ('0');      {символ, следующий за символом 0, равен символу 1}

x2:= Pred('3');      {символ, предшествующий символу 3, равен 2 }

x3:= Chr(65);        {символ, соответствующий коду 65, равен A }

n:= Ord('A');        {код символа A равен 65 }
```

СТРОКОВЫЕ переменные

Строка *– это последовательность символов.*

Строковый тип (string) – это тип данных для работы со строками на языке Pascal.

Значениями переменных этого типа являются строки, содержащие не более 255 любых символов.

Формат описания:

```
var   имя: string;
```

или

```
var   имя: string [n];
```

string *– служебное слово*

n *– максимальное количество символов в строке, если не указано, то значение 255.*

Над строковыми переменными можно выполнять основные действия: **присваивание**, **ввод**, **вывод**, **сцепление** *(***соединение, конкатенация***)* и **сравнение строк**.

Пример 4: составить программу, которая выполняет **присваивание**, **ввод**, **соединение** и **вывод** трёх строк. Первая строка известна – «Информатика».

```
Program Primer4;
    var Y1, Y2, Y3: string;
Begin
    Y1:='Информатика';                  {присваивание значения}
    writeln ('Введите две строки:␣' );
    readln (Y2);                        {ввод строки Y2}
    readln (Y3);                        {ввод строки Y3}
    writeln (Y1, ',␣', Y2, ',␣', Y3);   {вывод}
    writeln (Y1 + ':␣' + Y2 + '␣и␣' + Y3);   {соединение и вывод}
end.
```

Результат выполнения программы может выглядеть:

```
Введите две строки:
компьютер                {Enter}
программы                {Enter}
Информатика,␣компьютер,␣программы
Информатика:␣компьютер␣и␣программы
```

Строковый тип данных

Тип данных *string* обладает свойствами и **простых** и **составных** типов:

- при вводе и выводе строк используют имя строки – свойство **простой** переменной;
- строка как совокупность из ***n*** символов, т.е. как массив символов – **составной** тип данных. К любому символу в строке можно обратиться так же, как к элементу массива из ***n*** символов. Для этого после имени строки надо указать порядковый номер символа в строке.

Например, **St:='ИНФОРМАТИКА';**

тогда **St[2]** *соответствует* 'Н';

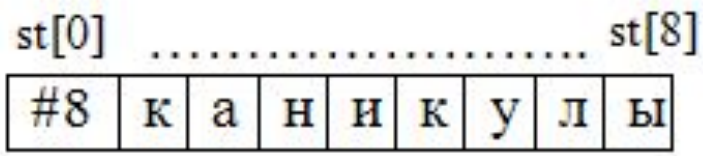
St[3] *соответствует* 'Ф';

```
St:='ИНФОРМАТИКА';
X2:=St[2];    {переменная X2 имеет значение  Н }
X3:=St[3];    {переменная X3 имеет значение  Ф }
```

В памяти ЭВМ под строку отводится максимальная длина + 1 байт. Байт располагается в начале строки и указывает длину строки.

Пример:

```
var st:string[8];
...
st:='каникулы';
```




```
Var
st1,st2:string[10];     {st1 и st2 отводится по 11байт}
rt1:string[213];        { 214 байт}
rt2:string;             { 256 байт}
```

```
Rt2:='';    {пустая строка – апостроф и апостроф}
```

Функции для работы со строками

length(st) — **вычисляет** текущую длину в символах строки **st**. *Результат имеет целочисленный тип.*

```
N:=length (‘студент‘);        {N =7}
```

copy(st, poz, n) — **выделяет** из строки **st** подстроку длиной **n** символов, начиная с позиции **poz**. *Если **poz** больше длины строки, то результатом будет пустая строка.*

```
s1:='Turbo Pascal' ;
     1 2 3 4  5 6 7 8 9 10 11 12
s2:=copy(s1,1,5) ;        {Turbo}
s3:=copy(s1,7,3);     {Pas}
```

concat(str1,str2,...,strn) — **выполняет** сцепление строк **str1**, **str2**, ... **strn** в том порядке, в каком они указаны в списке параметров.

```
S:=concat('AA', 'XX', 'Y');          { AAXXY }
```

или

```
s:='AA'+'XX'+'Y';
```

Функции для работы со строками

pos (str1, str) — **определяет** (**находит**) первое появление (т.е. номер позиции) в строке **str** подстроки **str1**.

Результат имеет **целочисленный тип** и равен номеру той позиции, где находится первый символ подстроки **str1**.

Если в **str** подстроки **str1** не найдено, то результат равен 0.

Str:= *'Turbo Pascal' ;*
1 2 3 4 5 6 7 8 9 10 11 12

n1:=pos(*'Pascal'*,str); **{ 7 }**

n2:=pos(*'pascal'*,str); **{ 0 }**

n3:=pos(*'r'*,str); **{ 3 }**

Процедуры обработки строковых переменных

delete (st,poz,n) — **удаление** **n** символов в строке **st** , начиная с позиции **poz**

*Если значение **poz** больше, чем размер строки, то ничего не удаляется.*

delete ('спорт',1,1); ***{ порт }***

delete ('спорт',2,1); ***{ сорт }***

Удаление всех пробелов в начале строки **st** :

while st[1]=' ' **do** delete (st,1,1);

Удаление пробелов в конце строки **st** :

while st[length(st)]=' ' **do** delete (st,length(st),1);

Процедуры обработки строковых переменных

insert (str1,str2,poz) — **вставка** строки **str1** в строку **str2**, начиная с позиции **poz** (первый параметр — **что** вставляем, второй параметр — **куда**)

- *при вставке контролируйте длину полученной строки (сообщения об ошибке не выдается.)*

insert ('порт','с',2); ***{ спорт }***

Есть ли ошибка, то где? Исправьте.

Пример 5: программа, в которой из слова "СТРОКА" будет получено слово "СЕТКА"

```
Var
   st : string [6];      { 6 – длина строки СТРОКА }
Begin
   st:='СТРОКА';         {присваивание значения}
   delete (st, 3, 2);    {удаляем с третьей позиции два символа→
                          результат – СТКА }
   insert ('Е', st, 2);  {вставим символ Е на вторую позицию в
                          СТКА }
   writeln (st);         {выводим результат на экран – СЕТКА}
End.
```

Пример 6: программа для подсчёта количества **!** в строке, введённой с клавиатуры

```
Program pr6;
Var
   s : string;
   i, k : integer;
Begin
     k:=0;
     writeln('Введите строку');
     readln(s);
     for i:=1 to length(s) do
         if s[i]='!' then k:=k+1;
    writeln('Восклицательных знаков' ,k:4);
End.
```

Задания:

Составить программу:

1. ***получить из слова***

   металлоискатель

   слова: МЕТАЛЛ, ИСКАТЕЛЬ, ЛОМ и …

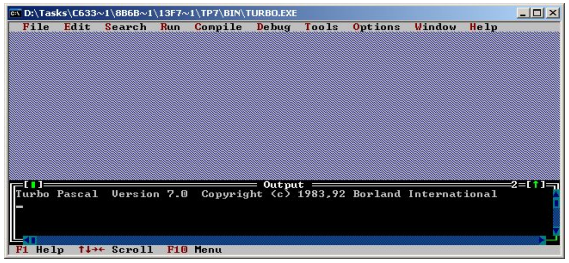
2. ***с клавиатуры вводят строку. Определите какой символ ‘а’ или ‘е’ встречается чаще.***

Пример 7: преобразовать строчные буквы в заглавные

```
Var
    s : string;
    i: integer;
Begin
    writeln ('Введите исходную строку:  ');
    readln (s);
        for i:=1 to length(s) do
        s[i]:=upcase(s[i]);   {Преобразование одного символа}
writeln(s);
End.
```

Алгоритм выполнения задачи

1. Загрузите программу Turbo Pascal (**TP7\BIN\TURBO.EXE**).



Открыть новый файл **File \ New**.

2. Набрать текст программы.
3. Выполнить **компиляцию** – проверка на наличие синтаксических ошибок (Compile\Compile или **Alt+F9**).
4. Выполнить проверку (подобрать исходные значения и посчитать результат).
5. Запустить программу на выполнение (Run\Run или **Ctrl+F9**).
6. Просмотреть полученный результат (**Alt+F5**) и сравнить его с результатом проверки.
7. Сохранить программу в своей папке:
   a). **File \ Change Dir…** – укажите и выберите каталог, который нужен;
   b). **File \ Save As…** – введите имя файла, не более 8 символов).

Процедуры обработки строковых переменных

процедура преобразования типов:

str (number, st) — *преобразование числового значения величины **number** в строку **st**.*

```
var     s1,s2,s3,s4:string;
          num1:integer;
          num2:real;
        begin
          num1:=5;
          num2:=5,78;
        str(num1,s1);          {s1=’5’}
        str(num1:3,s2);        {s2=’5’}
        str(num2,s3);          {s3=’5.78000000000E+00’}
        str(num2:3:1,s4);        {s4=’5.8’}
```

Процедуры обработки строковых переменных

процедура преобразования строки в число:

val (st, number, code) — преобразует значение ***st*** в величину целочисленного или вещественного типа и помещает результат в ***number***.

Code - целочисленная переменная.

Ошибки нет, значение **code** равно нулю, если не число **code** содержит номер позиции первого ошибочного символа, а значение **number** не определено.

```
s1:='5,78';
s2:='5,78';
   val(s1,num1,cod1);   {num1=5.78   cod1=0}
   val(s2,num2,cod2);   {cod2=2 – второй символ ошибочный}
```

Пример 8: преобразовать строки в число

```
Var
    s:string;
    n, error : integer;
Begin
    repeat
        write ('Введите число ');{сообщение ввести число,}
        readln (s);                { а вводим строку }
        val(s,n,error);      {преобразуем строку в число}
        if error>0 then writeln ('Неверный символ № ',error)
    until error=0;
    { продолжение программы }
```