11. Символы (Char) и строки (Strings).

Для представления символьной, т. е. текстовой информации в Турбо-Паскале используются два типа: Символьный *(Char* — от Character) и Строковой *(String)*. С помощью этих типов можно манипулировать текстом: упорядочить, контролировать и обрабатывать..

Рассмотрим представление этих типов и определенные для них операции.

11.1. Обработка символов.

Как уже было отмечено, каждый символ занимает один байт памяти. Внутреннее представление символа использует кодировку ASCII-таблицы, т.е. каждому символу соответствует номер-число от 0 до 255. Так, например, символ «A» хранится под номером 64, символ «B» – под номером 65, и т.д., т.е. соблюдается упорядоченность по алфавиту. Именно с помощью внутренних кодов можно обращаться к символам и манипулировать ими.

Рассмотрим на примерах действие стандартных функций, определенных для символьных величин.

Функция Chr(i) возвращает символ с ASCII-кодом *i*, например, процедура

```
Writeln(Chr(65));
```

выведет на экран символ «A».

Функция Ord(*ch*) возвращает ASCII-код символа *ch*. Следующая программа детектирует, является ли символ латинской буквой:

```
Uses Crt;
Var i:Integer;
    ch:Char;
Begin Write(' Введите символ:');
    ch:=ReadKey;
      i:=Ord(ch);
        Writeln;
      If (i>=65) And (i<=90) And (i>=97) And (i<=122) Then
        Writeln(' Интересующий Вас символ является буквой!');
```

```
                    Writeln(' Этот символ не является буквой!');
    End.
```

Функция Pred(ch) возвращает предыдущий символ от *ch*, т.е. символ, расположенный в ASCII-таблице перед *ch*. Аналогично, функция Succ(*ch*) возвращает последующий за *ch* символ, т.е. символ, расположенный в ASCII-таблице после *ch*. Следующий оператор

```
Writeln(Pred('B'), Succ('A'), Pred('D'));
```

выведет на экран:

ABC

Символы можно сравнивать друг с другом. Эта возможность реализуется с помощью обработки внутренних кодов каждого символа. Например,

```
Var ll,12:Char;
...
11:='A';
12:='B';
If ll>12
    Then Write ln (' Символ', 11,' больше, чем символ', 12)
    Else Write ln(' Символ', 12,' меньше/равен символу ',11);
    ...
```

Функция Upcase (*ch*) преобразовывает символьное выражение в тот же символ в верхнем регистре. Например, оператор

```
Writeln(Upcase('a'), Upcase('2'), Upcase('D'));
```

выведет на экран:

A2D

Заметим, что символ 2 не претерпел изменения, так как функция Upcase воздействует только на символы-буквы.

11.2. Представление строк.

Последовательность символов (длиной больше, чем один символ), заключенная в апострофы, воспринимается как один элемент данных – строковая переменная или константа. Максимально допустимая длина строки – 256 символов, т.е. 256 байтов, пронумерованных от 0 до 255. Но для непосредственного хранения данных используется 255 байтов. Первый байт имеет специальное назначение и зарезервирован для хранения текущей, фактической длины строки, т.е. ASCII-код первого символа соответствует количеству фактически заполненных байтов строки.

Строковая переменная объявляется либо простым назначением типа String:

```
Var st: String;
```

либо с ограничением максимальной длины:

```
Var st 80:String[80];
    st 1:String[1];
    st 10:String[10];
```

В первом случае для переменной допускается максимальная длина в 255 байтов. Отметим переменную *st1*, которая, хотя и используется для представления одного символа, но в оперативной памяти для нее отведется два байта.

Инициализация строковой переменной осуществляется точно так же, как и остальных переменных — либо с помощью оператора присвоения, либо оператором ввода с клавиатуры. Например,

```
st:='';
st 1:='A';
st 10:='example';
```

Заполнение строки начинается слева, т.е. с позиции с низшими номерами. Таким образом, распределение элементов строки по пронумерованным ячейкам будет иметь вид:

```
st0: |7|e|x|a|m|p|l|e|_|_|_|
     0 1 2 3 4 5 6 7 8 9 10
```

Здесь символ подчеркивания означает, что соответствующие ячейки не заполнены. В первой ячейке будет храниться символ с ASCII-кодом 7, так как слово *example* состоит из семи букв, а сама строка будет занимать в памяти 11 байтов.

Значение строки можно считать с клавиатуры, например:

```
Readln(st);
```

При запросе надо набрать строку без апострофов.

Вывод строки на экран осуществляется с помощью операторов Write(st) и Writeln(st).

Можно обращаться к каждому элементу-символу строки по отдельности с помощью индекса – номера позиции в строке, например,

```
Var c,d:Char;
    st:String[20];
BEGIN
    st:='This is the string';
    c:=st[1];
    d:=st[9];
    Writeln(c,d);
END.
```

Результат;

```
Tt
```

Строку можно сравнить с символьным одномерным массивом (см. в следующей главе), обращение к элементам которого формально осуществляется подобным же образом. Различие состоит в том, что, во-первых, длина строки ограничена сверху числом 255, а кроме того, в нулевом элементе-символе строкового выражения хранится информация об его активной, текущей длине, т.е. о количестве заполненных элементов. Восстановить длину можно с помощью функции "Ord".

```
Var st:String[40];
BEGIN
    st:='This is the string';
    Writeln('The length of the string:');
    Writeln(st);
    Writeln('Is:', Ord(st[0]));
END.
```

Результат:

```
The length of the string:
This is the string
Is: 18
```

Т.е. текущая длина строки *st* равняется 18, хотя переменная описана как строка с максимально допустимой длиной в 40 символов.

11.3. Обработка строк.

Для обработки строк в Турбо-Паскале определены операции:

+ — сцепления, объединения;
<, >, <=, >=, <> — сравнения;
специальные процедуры и функции.

Рассмотрим действие этих операторов на примерах.

11.3.1. Объединение строк.

Операция + объединяет, сцепляет две или больше строк вместе, например,

```
Vars,sl,s2,s3: String;
BEGIN sl:='This';
    s2:='is';
    s3 :='the string';
    s:=sl+s2+s3;
    Writeln(s +'!');
END.
```

На экране дисплея появится результат:

This is the string!

11.3.2. Логические операции над строками.

Строковые выражения можно сравнивать друг с другом с помощью операций отношений. Они имеют такую же смысловую нагрузку, что и для численных выражений.

Сравнение строковых выражений осуществляется слева направо до первого несовпадающего символа, причем «большей» принимается та строка, в которой ASCII-код текущего символа имеет большее значение. Например,

```
sl:='abcd l';
s2:='abcd 2';
If s1 >s2
    Then Writeln('The string "', s1, '" is larger than the string "', s2,'" !')
    Else Writeln('The string "', s2,'" is larger than the string "', s1, '" !');
Writeln(sl, '<',s2,'=',sl>s2);
Writeln(sl,'>', s2,'=', sl>s2);
```

Результат:

The string "abcd2" is larger than the string "abcd 1"!
abcd 1<abcd2= TRUE
abcd2 > abcdl = FALSE

Приоритет логических отношений ниже, чем приоритет операции сцепления строк. Это означает, что если в сложном выражении встречаются подряд операции сцепления и логических отношений, то сперва будет выполняться операция объединения строк, и уже потом будет произведено сравнение. Т.е. логические выражения

```
'ab'+'cd'='abcd'
```

и

```
('ab'+'cd')='abcd'
```

идентичны и обе имеют значение *True:*

```
If 'ab'+'cd'=abcd'
    Then Writeln(' The strings are the same ')
    Else Writeln(' The strings are different');

    Результат:

The strings are the same
```

11.3.3. Процедуры и функции для обработки строк.

Для обработки строковых выражений в Турбо-Паскале поддерживаются стандартные процедуры и функции.

Процедуры:

Delete(Var st:String, index, count: Integer) – из строки *st* удаляет *count* количество символов, начиная с позиции под номером *index*,
Insert(source:String; Var st:String; index:Integer) вставляет строковое выражение *source* в строку *st*, начиная с позиции под номером *index*.
Str(x[:width:decimals]; Var st:String) преобразовывает числовую Переменную в *st* строковое выражение. Переменная *x* может иметь тип Integer или Real. Формат записи числа в строковом выражении такой же, что и для процедуры вывода.

Val(st:String; Var x; Var code:Integer) преобразовывает *st* строковое выражение в *x*-числовую переменную, которая может иметь тип Integer или Real, в зависимости от формата задания строкового выражения. Переменная *code* идентифицирует, насколько успешно прошло преобразование. Если *code=0*, преобразование завершилось без ошибок, иначе переменная выдаст код ошибки.

Функции:

Concat(sl [,s2,...] : String):String сцепляет, соединяет строковые выражения (аргументы) и выдает одно строковое выражение. Результат получается таким же, как при использовании операции объединения строк +.

Copy(st:String; index, count:Integer):String выдает подстроку строки *st*, состоящую из *count* количества символов, начиная с позиции (элемента) под номером *index*.

Length(st:String):Integer выдает текущую длину строки.

Pos(substr, st:String):Byte выдает номер первого элемента, позицию, начиная с которой в строке *st* встречается выражение подстроки *substr*.