Символы и строки

1

Символьный (литерный) тип (1)

Стандартное имя типа **char** (character).

Значениями этого типа являются элементы набора литер, определяемого реализацией языка. В Турбо-Паскале символы составляют множество **ASCII** (***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*) – 256 символов. Множество ASCII содержит заглавные и строчные латинские буквы, цифры; может содержать русские буквы. Некоторые символы не имеют изображения (например, «конец строки»).

- Размер памяти, занятый символом – 1 байт.
- Множество символов ***упорядочено***.
- Функция **ord(c)** (0<= ord(c) <= 255) дает порядковый номер литеры c.
- Функция **chr(i)** дает литеру (значение типа char) с порядковым номером 0<=i<=255.

ord(chr(i))=i
chr(ord(c))=c

2

Символьный (литерный) тип (2)

- Функция **pred(c)** (ord(c)>0) дает предыдущую литеру.
- Функция **succ(c)** (ord(c)<255) дает последующую литеру.

pred(c)=chr(ord(c)-1)
succ(c)=chr(ord(c)+1)

- Графическое представление символа (имеющего его) – соответствующий знак в кавычках (апострофах) '+', 'y', '.', 'л', "". Символ, не имеющий графического представления, можно задать с помощью записи #<порядковый номер символа>, например, #7 - гудок.
- *Цифры '0','1',...,'9' имеют подряд идущие порядковые номера. Малые латинские буквы 'a','b',...,'z' имеют подряд идущие номера. Большие латинские буквы 'A','B',...,'Z' имеют подряд идущие порядковые номера.*
- Операции сравнения над символами такие же, как для чисел. Результат – булевский (логический) тип. Эти операции над символами эквивалентны сравнениям порядковых номеров символов.

3

Символьный (литерный) тип (3)

Задача 1. Напечатать все символы таблицы ASCII.

```
var i:integer;
begin
  for i:=0 to 255 do
    write(chr(i))
end.
```

Задача 2. Напечатать все малые латинские буквы в обратном алфавитном порядке.

```
var c:char;
begin
  for c:='z' downto 'a' do
    write(c)
end.
```

4

Символьный (литерный) тип (4)

Задача 3. Перевести однозначное натуральное число в символ-цифру, изображающее это число.

```
var n:integer;
   c:char;
...
 c:=chr(ord('0')+n)
```

Задача 4. Перевести символ-цифру в число, которое данная цифра изображает.

```
var c:char;
   n:integer;
...
 n:=ord(c)-ord('0')
```

5

Ограниченные типы (1)

Стандартные дискретные типы (целый, символьный, булевский) – предопределены, но пользователь может создать (определить) собственный дискретный тип.

Ограниченный тип определяется сужением (ограничением) допустимого диапазона значений некоторого стандартного дискретного типа. Задается минимальное и максимальное значение диапазона.

Примеры:

1..10	ограничение некоторого
-100..100	целого типа
'a'..'z'	ограничение символьного типа

Описание переменных:

```
var
   s:1..10;
```

6

Ограниченные типы (2)

Можно ввести идентификатор для нового типа. Для этого в программе вводится раздел описаний, начинающийся со служебного слова **type**:

```
type
   digits=0..9;
   sto=-100..100;
   letter='a'..'z';
var
  n:digits;
```

Ограниченный тип *наследует* все свойства базового типа (в частности, набор допустимых операций).

Активно используете ограниченные типы:

- **наглядность программы,**
- **контроль ошибочных выходов значений за пределы заданного диапазона (как при трансляции, так и при выполнении).**

7

Символьные массивы

***Символьные массивы* – одномерные массивы, состоящие из элементов символьного типа и индекс принадлежит ограниченному целому типу.**

Например:

```
var s:array[1..26] of char;
```

***Дополнительные средства для работы*: изображение конкретного значения символьного массива в виде строки**

```
s:='Пример символьного массива';
```

Это изображение может использоваться в операторах присваивания, в описаниях констант и в процедуре write.

8

Определение строкового типа (1)

Строковый тип обобщает понятие символьных массивов, позволяя динамически менять длину строки.

Строковый тип данных определяет множество символьных цепочек произвольной длины от нуля символов до заданного их числа.

В описании строкового типа указывается максимальная длина строки для этого типа.

Максимальная длина строки, если не указана, то подразумевается равной 255, иначе может быть любое целое от 0 до 255.

Пример:

```
type
    line=string[80];
var
    myLine:line;              {максимум 80 символов}
    myLineShort:string[10]; {максимум 10 символов}
    s: string;                {максимум 255 символов}
```

9

Определение строкового типа (2)

Описание

Var St: string[80];

резервирует для **St** 81 байт памяти.

Если учесть, что на каждый символ отводится один байт, возникает естественный вопрос, а откуда взялся еще один байт, 81-й, а точнее нулевой байт?

Дело в том, что после того, как переменной **St** присвоено какое-то значение, нулевой байт содержит фактическую длину строки **St**. Длина строки в байтах при этом равна значению кода символа, находящегося в **St[0]**.

Например, присваивание

St := ‘abcdefgh’;

дает такой результат:

St[0] = Chr(8).

St[1] = ‘a’.

…

St[8] = ‘h’.

Фактическая длина строковой переменной St равна Ord(St[0]).

10

Определение строкового типа (3)

Значение 255 для верхнего предела длины строки объясняется тем, что одиночный байт может принимать 256 различных значений, от 0 до 255.

Изображение строки такое же, как для символьных массивов, может использоваться

- в операторах присваивания,
- как фактические параметры подпрограмм,
- в описании констант.

Символьный массив в отличие от строки может быть очень большим!

11

Строковые операции (1)

Операция конкатенация (сцепление) применяется для соединения (сцепления) нескольких строк в одну результирующую строку. Эта операция является бинарной ассоциативной операцией и обозначается знаком +.

Например, выражение
'Т ' + 'у' + 'р' + 'бо' + 'паскаль' дает 'Турбо паскаль'.

Длина результирующей строки не должна превышать 255.

Примеры:

```
myLine:='Короткая строка';          {длина 15}
myLine:=MyLine+' стала длинее';     {длина 28}
myLine:='';                         {длина 0}
```

12

Строковые операции (2)

Операции **сравнения** =,<>,>,<,>=,<=

При сравнении строк используется ***лексикографический*** порядок:

а) сравнение строк производится с первых символов (слева направо) до первого несовпадающего символа, и та строка считается больше, в которой первый несовпадающий символ больше;

б) если строки имеют различную длину и одна строка полностью входит во вторую, то более короткая строка меньше чем более длинная.

Примеры:

'abcd'>='abc'

'abcde'>'aacde'>'aacdd'

13

Строковые операции (3)

Элементы строки нумеруются целыми числами, начиная с 1.

Доступ к отдельным элементам строки

myLine[<выражения типа 1..255>]

Элемент строки принадлежит типу char.

Выражения строкового типа - выражения, в которых операндами служат строковые данные:

- строковые константы,
- строковые переменные,
- функции со строковым значением,
- литерные выражения.

Строковые выражения могут использоваться там, где используется строковый тип. *В операторах присваивания, если значение переменной после выполнения присваивания превышает по длине максимально допустимую величину, то все лишние символы отбрасываются!*

		значение
a:string[6]	a:='группа 1';	'группа'
a:string[8]	a:='группа 1';	'группа 1'
a:string[2]	a:='группа 1';	'гр'

14

Строковые операции (4)

Стандартная функция **Length(<выражение строкового типа>).**
Значение функции есть текущая длина строки – фактического параметра.

Пример:

```
var myLine:string;
   i:integer;
  .....
for i:=1 to Length(myLine) do
   myLine[i]:=chr(ord(myline[i])+1);
```

15

Строковые операции (5)

Нет полной идентичности между строковыми типами и символьными массивами, поэтому возможна ошибка при работе с элементами строки без учета ее текущей длины.

Пример:

```
var
  str:string[26];
  i:integer;
begin
str:='A';
for i:=1 to 26 do
  str[i]:=chr(ord('A')+i-1);
writeln(str)
end.
```

Предполагается, что данная программа сформирует строку из 26 букв латинского алфавита, но печать writeln(str) дает просто A!

Объяснение: присваивание значений элементам строки не влияет на ее текущую длину (здесь текущая длина = 1).

16

Строковые операции (6)

Правильная программа:

```
var
     str:string[26];
     i:integer;
begin
   str:='';
   for i:=1 to 26 do
     str:=str+chr(ord('A')+i-1);
   writeln(str)
end.
```

17

Стандартные процедуры и функции для работы со строками (1)

Процедура delete(var St:string;Poz:integer;N:integer)

Производит удаление N символов строки St, начиная с позиции Poz. Если Poz>255, то осуществляется программное прерывание.

```
St:='abcdef';
delete(St,4,2);
writeln(St);              {abcf}
St:='река Волга';
delete(St,1,5);
writeln(St);              {Волга}
```

Процедура insert(Source:string;var S:string;Index:integer)

Осуществляет вставку строки Source в строку S, начиная с позиции Index.

```
S:='Золотойключик';
insert(' ',S,8);
writeln(S);           {Золотой ключик}
```

18

Стандартные процедуры и функции для работы со строками (2)

Функция copy(S:string;Index:integer;Count:integer):string

Функция выделяет из строки S подстроку длиной Count символов, начиная с позиции Index.

Если Index > length(S), то возвращается пустая строка.

Если Count+Index > length(S), то возвращается конец строки.

Если Index > 255, то диагностируется ошибка при выполнении.

copy('abcdefg',2,3)='bcd'

copy('abcdefg',4,10)='defg‘

Функция pos(Substr:string;S:string):integer

Функция обнаруживает первое появление в строке S подстроки Substr. Результат равен номеру той позиции, где начинается подстрока Substr. Если подстроки нет, то результат равен 0.

pos('de','abcdef')=4

pos('z','abcdef')=0

19