Гомельский государственный университет
имени Франциска Скорины

Символы и строки

Составил: Ассистент кафедры общей физики
Соколов С.И.

Гомель, 2015

Символы

Для хранения и обработки символов используются переменные типа Ansichar и wideChar. Тип Ansichar представляет собой набор ANSI-символов, в котором каждый символ кодируется восьмиразрядным двоичным числом (байтом). Тип wideChar представляет собой набор символов в кодировке Unicode, в которой каждый символ кодируется двумя байтами.

Значением переменной символьного типа может быть любой отображаемый символ:

- буква русского или латинского алфавитов;
- цифра;
- знак препинания;

И специальный символ, например, "новая строка".

Инструкция объявления символьной переменной в общем виде выглядит так:

Имя: char;
где:

.имя — имя переменной символьного типа;
.char — ключевое слово обозначения символьного типа.

Примеры:

otv: char; ch: char;

Действия с символами: операции

Результатом унарной операции

```
#<положительная неименованная константа
   целого типа>
```

является *символ*, номер которого в таблице ASCII соответствует заданному числу.

Например,

```
#100  = 'd'
#39   = ''''  {апостроф}
#232  = 'ш'
#1000 = 'ш'  {потому что (1000 mod 256)= 232}
```

Действия с символами: операции

Кроме того, к символьным переменным, как и к значениям всех порядковых типов данных, применимы операции сравнения

<, **<>**, >, **=**,

результат которых также опирается на номера символов из таблицы ASCII.

Переменную типа char можно сравнить с другой переменной типа char или с символьной константой. Сравнение основано на том, что каждому символу поставлено в соответствие число (см. приложение 2), причем символу '0' соответствует число меньшее, чем символу ‘Y’, символу 'A' — меньшее, чем 'b', символу V — меньшее, чем a. Таким образом, можно записать:

'0'<'1'<..<'9'<..<'A'<'B'<..<'Z'<'a'<'b'<..<'z‘

Символам русского алфавита соответствуют числа большие, чем символам латинского алфавита, при этом справедливо следующее:

'А'<'Б'<'В'<..<'Ю'<'Я'<'а'<'б'<'в'<...<'э'<'ю'<'я‘

В тексте программы вместо символа можно указать его код, поставив перед числом оператор #. Например, вместо константы 'Б' можно записать #193.

Действия с символами: стандартные функции

Функция

chr(k:byte):char

"превращает"; *номер символа* в *символ*.
Действие этой функции аналогично действию операции #.

Например:

```
c:= chr(48);    { c: char }
          { c = '0' }
```

Действия с символами: стандартные функции

Обратной к функции ***chr()*** является уже изученная нами функция ***ord()***.

Таким образом, для любого числа ***k*** и для любого символа ***c***

```
ord(chr(k)) = k;
chr(ord(c)) = c;
```

Строки

Строки могут быть представлены следующими типами: shortstring, Longstring и widestring. Различаются эти типы предельно допустимой длиной строки, способом выделения памяти для переменных и методом кодировки символов.

Переменной типа shortstring память выделяется статически, т. е. до начала выполнения программы, и количество символов такой строки не может превышать 255. Переменным типа Longstring и widestring память выделяется динамически — во время работы программы, поэтому длина таких строк практически не

Инструкция объявления в общем виде выглядит так:

Имя: String;
или

Имя: String [длина]

где:
- имя — имя переменной;
- string — ключевое слово обозначения строкового типа;
- длина — константа целого типа, которая задает максимально допустимую длину строки.

В тексте программы последовательность символов, являющаяся строкой (строковой константой), заключается в одинарные кавычки. Например, чтобы присвоить строковой переменной parol значение, нужно записать:

```
parol:= 'Большой секрет';
```

Или

```
parol:= '2001';
```

Обращение к строковым переменным

Особенностью строковых переменных является то, что к ним можно обращаться

- как к скалярным переменным,
- так и к массивам.

Во втором случае применяется конструкция "переменная с индексом", что обеспечивает доступ к отдельным символам строки. При этом нижняя граница индекса равна 1.

Отдельный символ строки совместим с типом ***char***.

Например,

```
S:= St[20];  { обращение к 20 эл-ту строки St }
Po:='Компьютер';    { инициализация строки }
```

Обращение к строковым переменным

Например, если в программе определены

Var S: string; C: char;

и задано

S:='Москва',

то

S[1]='M', ***S[2]='o'*** и т. д.

Обращение к строковым переменным

Элементы массива, составляющие строку можно переставлять местами и получать новые слова.

Пример 1. Вывод исходного слова справа налево: "авксоМ"

```
for  i:=1  to  N div 2  do  begin
     C:=S[i];  S[i]:=S[N-i+1];  S[N-i+1]:=C
     end;
ShowMessage(S);
```

Пример 2. Поиск и замена заданного символа в строке

```
for  i:=1  to  N  do
  if  S[i]= ' ' then  ShowMessage('найден
символ                        пробел');
for  i:=1  to  N  do
  if  S[i]='/' then  S[i]:='\';
          { замена символа "/"  на  "\" }
```

Операция конкатенации

Результатом выполнения операции ***конкатенации*** "+", является строка, в которой исходные строки-операнды соединены в порядке их следования в выражении.

Например,

```
X:='Пример';   Y:='сложения';   Z:='строк';
ShowMessage(X + Y + Z);
ShowMessage(Y + '   ' + Z + '   ' + X);
```

На экран будут выведены строки:

Примерсложениястрок
сложения строк Пример

'Кро'+' 'код'+ 'ил'

позволит получить новую строку

'Крокодил'

Операция конкатенации

Тип ***String*** допускает и ***пустую строку*** – строку, не содержащую символов:

```
EmptyStr := '';   {подряд идущие кавычки}.
```

Она играет роль нуля (нейтрального элемента) операции конкатенации:

```
EmptyStr + X = X + EmptyStr = X.
```

Операция сравнения

Строки - это единственный структурированный тип данных, для элементов которого определен порядок и, следовательно, возможны операции сравнения.

Сравнение строк происходит посимвольно, начиная с первого символа.

Строки равны, если имеют одинаковую длину и посимвольно эквивалентны.

Над строками определены также отношения (операции логического типа):

=, <>, <, >, <=, >=.

Таблица 3.1. Сравнение строк

Строка 1	Строка 2	Результат сравнения
Иванов	Иванов	Строки равны
Алексеев	Петров	Строка 1 меньше строки 2
Иванова	Иванов	Строка 1 больше строки 2

Функция *Length*

Формат:

```
Length(X :string): byte;
```

Возвращает длину строки - аргумента ***X***. Причем, длина пустой строки ***Length(EmptyStr) = 0***.

Тип результата – ***Byte***.

Примеры:

Исходные данные: ***S := ‘крокодил‘;***

Оператор: ***j := length(S);***

Результат: ***j = 8***

Исходные данные: ***T := ‘Компьютерный класс‘;***

Оператор: ***j := length(T);***

Результат: ***j = 18***

Функция *Length*

Задача: ввести строку с клавиатуры и заменить все буквы «а» на буквы «б».

```
var s: string;
    i: integer;
begin
   …
   …
   for i:=1 to Length(s) do
      if s[i] = 'a' then s[i] := 'б';
   …
end.
```

длина строки

Функция *Pos*

Формат:

```
Pos(Y, X :string): byte;
```

Отыскивает первое вхождение строки ***Y*** в строке ***X*** (считая слева направо) и возвращает номер начальной позиции вхождения.

Если ***X*** не содержит ***Y***, функция вернет 0 (***Pos(Y, X) = 0***).

Тип результата – ***Byte***.

Примеры:

Исходные данные: ***S := ‘крокодил‘;***

Оператор: ***i := pos('око‘, S);***

Результат: ***i = 3***.

Оператор: ***i := pos('я', ‘крокодил');***

Результат: ***i = 0***.

Поиск в строке

Поиск в строке:

```
var n: integer;
```

```
s := 'Здесь был Вася.';
n := Pos ( 'e', s );
if n > 0 then
      ShowMessage('Буква e – это s[', n, ']')
else ShowMessage ('Не нашли');
n := Pos ( 'Вася', s );
s1 := Copy ( s, n, 4 );
```

s[3]

3

n = 11

Особенности:

- функция возвращает номер символа, с которого начинается образец в строке
- если слова нет, возвращается 0
- поиск с начала (находится **первое** слово)

Функция *Copy*

Формат:

```
Copy(X :string; Index, Count :byte): string;
```

Копирует (выделяет) подстроку строки ***X***, начиная с позиции ***Index*** и содержащую следующие ***Count*** символов.

Тип результата – ***String***.

Если ***Index*** больше длины строки, то результатом будет пустая строка.

Если же ***Count*** больше, чем длина оставшейся части строки, то результатом будет только ее "хвост":

Функция *Copy*

```
s := '123456789';

s1 := Copy ( s, 3, 6 );
s2 := Copy ( s1, 2, 3 );
```

с 3-его символа

6 штук

'345678'

'456'

Исходные данные: ***S := ‘крокодил‘;***

Оператор: ***b:= copy(S, 2, 3);***

Результат: ***b = 'рок'.***

Исходные данные: ***T := ‘Компьютерный класс‘;***

Оператор: ***c:= copy(T, 14, 53);***

Результат: ***c = 'класс'.***

copy('abc3de Xyz', 2, 4) = 'bc3d'

copy('abc3de Xyz', 12, 4) = ''

copy('abc3de Xyz', 8, 14) = 'Xyz'

Процедура *Delete*

Формат:

```
Delete(X :string; Index, Count :byte);
```

Удаляет из строки ***X*** подстроку, начиная с позиции, заданной числом ***Index***, длиной, заданной числом ***Count***.

Тип результата – ***String***.

Если число ***Index*** больше размера строки, то подстрока не удаляется.

Если число ***Count*** больше имевшегося количества, то удаляются символы до конца строки.

Тип результата – ***String***.

Процедура *Delete*

```
s := '123456789';
Delete ( s, 3, 6 );
```

6 штук

'123456789'

'129'

строка меняется!

с 3-его символа

Исходные данные: ***S := 'крокодил';***

Оператор: ***delete(S, 4, 3);***

Результат: ***S = 'кроил'.***

Оператор: ***delete(S, 1, 1);***

Результат: ***S = 'роил'.***

Исходные данные: ***s = 'abc3de Xyz'***

Оператор: ***delete(S, 8, 13);***

Результат: ***S = 'abc3de '.***

Ниже приведена инструкция while, в результате выполнения которой удаляются начальные пробелы из строки st.

```
while(pos(' ',st) = 1) and(length(st) > 0) do delete (st,1,1);
```

Процедура *Insert*

Формат:

```
Insert(Y, X :string; Index :byte);
```

Вставляет строку ***Y*** в строку ***X***, начиная с позиции, заданной числом ***Index***.

Тип результата – ***String***.

Если ***Index*** выходит за конец строки, то подстрока ***Y*** припишется в конец строки ***X***.

Если результат длиннее, чем допускается для строки ***X***, произойдет его усечение справа.

Процедура *Insert*

```
s := '123456789';
Insert ( 'ABC', s, 3 );

Insert ( 'Q', s, 5 );
```

'12ABC3456789'

'12ABQC3456789'

что вставляем

куда вставляем

начиная с 3-его символа

Исходные данные: ***S := 'крокодил';***

Оператор: ***d := copy(S, 3, 3);***

Результат: ***d = 'око'.***

Оператор: ***insert('Н', d, 3);***

Результат: ***d = 'окНо'.***

Примеры

```
s := 'Вася Петя Митя';
n := Pos ( 'Петя', s );
Delete ( s, n, 4 );
Insert ( 'Лена', s, n );
```

```
6
'Вася  Митя'
'Вася Лена Митя'
```

```
s := 'Вася Петя Митя';
n := length ( s );
s1 := Copy ( s, 1, 4 );
s2 := Copy ( s, 11, 4 );
s3 := Copy ( s, 6, 4 );
s := s3 + s1 + s2;
n := length ( s );
```

```
14
'Вася'
'Митя'
'Петя'
'ПетяВасяМитя'
12
```

Пример 1. Проверить, является ли заданный символ S строчной гласной буквой русского алфавита.

Решение. `pos(S, 'аэоуыяеёюи') > 0`

Пример 2. Выделить часть строки после первого пробела.

Решение.

```
copy(s, pos(' ', s)+1, Length(s)-pos(' ', s))
```

Пример 3. Удалить последний символ строки S.

Решение. `delete(S, Length(S), 1)`

Пример 4. Выделить из строки ***S*** подстроку между ***i***-й и ***j***-й позициями, включая эти позиции.

Решение. `copy(s, i, j-i+1)`

Пример решения задачи

Задача: Ввести имя, отчество и фамилию. Преобразовать их к формату «фамилия-инициалы».

Пример:

```
Введите имя, фамилию и отчество:
Василий Алибабаевич Хрюндиков
Результат:
Хрюндиков В.А.
```

Алгоритм:

- найти первый пробел и выделить имя
- удалить имя с пробелом из основной строки
- найти первый пробел и выделить отчество
- удалить отчество с пробелом из основной строки
- «сцепить» фамилию, первые буквы имени и фамилии, точки, пробелы…

Программа

```
var s, name, otch: string;
    n: integer;
begin
   s:=InputBox('Ввод','Введите ФИО','0');

   n := Pos(' ', s);
   name := Copy(s, 1, n-1); { вырезать имя }
   Delete(s, 1, n);
   n := Pos(' ', s);
   otch := Copy(s, 1, n-1); { вырезать отчество }
   Delete(s, 1, n);          { осталась фамилия }
   s := s + ' ' + name[1] + '.' + otch[1] + '.';
   ShowMessage(s);
```