Символьный тип данных в Паскале

Символьный тип данных Char — тип данных, значениями которого являются одиночные символы. Данный тип может содержать всего один любой символ (Например: «*», «/», «.», «!» и другие).
Каждый такой символ занимает 8 бит памяти, всего существует 256 восьмибитовых символов. Все символы, используемые символьным типом Char записаны в таблице символов ASCII (American Standart Code for Information Interchange) или Американский стандарт кода для обмена информацией.

Символьные константы заключаются в апострофы, например '.', '*', '7', 's'. Также символьную константу можно записать с помощью символа — «решетки», например #185 — выведет символ под номером 185 из таблицы ASCII (это символ '№').

К символьному типу применимы 5 функций: Ord, Chr, Pred, Succ и Upcase.

Функция Ord преобразовывает символ в её числовой код из таблицы ASCII.

Функция Chr преобразует числовой код символа в сам символ. Chr обратна функции Ord.

Функция Pred возвращает значение предыдущего символа из таблицы ASCII.

Функция Succ обратная функции Pred, то есть возвращает следующий символ из вышеописанной таблицы ASCII.

Функция UpCase применима только для строчных букв английского алфавита. Данная функция преобразует строчные английские буквы в заглавные.

Пример программы на Паскаль с использованием функции Ord:

```
Var x:char; // Описание переменных (x - символьный тип)
Begin //Начало программы
readln (x); //Считывание переменной
writeln (ord (x)); //Вывод номера в таблице ASCII
end. //Конец программы
```

Пример программы на Паскаль с использованием функции Chr:

```
Var x:integer; // Описание переменных (x - целочисленный тип)
Begin //Начало программы
readln (x); //Считывание переменной
writeln (chr (x)); //Вывод символа по номеру в таблице ASCII
end. //Конец программы
```

Пример программы на Паскаль с использованием функций Pred и Succ:

```
Var x:char; // Описание переменных (x - символьный тип)
Begin //Начало программы
readln (x); //Считывание переменной
writeln (pred (x)); //Вывод предыдущего символа в таблице ASCII
writeln (succ (x)); //Вывод следующего символа в таблице ASCII
end. //Конец программы
```

Пример программы на Паскаль с использованием функции UpCase:

```
Var x:char; // Описание переменных (x - символьный тип)
Begin //Начало программы
readln (x); //Считывание переменной
writeln (upcase (x)); //Вывод английской буквы верхнего регистра
end. //Конец программы
```

Строковый тип данных в Паскале

Строки в Паскале – это данные типа **string**.

Они используются для хранения последовательностей символов. В Паскале длина стандартной строки ограничена 255 символами. Под каждый символ отводится по одному байту, в котором хранится код символа. Кроме того, каждая строка содержит еще дополнительный байт, в котором хранится длина строки (первый байт, индекс которого 0).

Если заранее известно, что длина строки будет меньше 255 символов, то программист может сам задать максимальную длину строки.

Примеры описания строк:

```
type str_type = string[12];
const n = 50;
var s1: string;
      s2, s3: str_type;
      s4: string[n];
      s5, s6, s7: string[7];
```

Объявление типизированной константы для типа **string** осуществляется так:

```
const s: string = 'FreePascal‘
```

st := '' - пустая строка.

Операции над строками

1. Строки можно присваивать друг другу. Если максимальная длина переменной слева меньше длины присваиваемой строки, то лишние символы справа отбрасываются.
s1 := 'this is text'; s2 := s1; …

2. Строки можно объединять с помощью *операции конкатенации*, которая обозначается знаком +.
… s1 := 'John'; s2 := 'Black'; s1 := s1 + ' ' + s2; …

3. Строки можно сравнивать друг с другом с помощью *операций отношения*. При сравнении строки рассматриваются посимвольно слева направо, при этом сравниваются коды соответствующих пар символов. Строки равны, если они имеют одинаковую длину и посимвольно эквивалентны. В строках разной длины существующий символ всегда больше соответствующего ему отсутствующего символа. Меньшей будет та строка, у которой меньше код первого несовпадающего символа (вне зависимости от максимальных и текущих длин сравниваемых строк).
'abc' > 'ab' (true); 'abc' = 'abc' (true) ; 'abc' < 'abc ' (false)

4. К отдельному символу строки можно обращаться как к элементу массива символов, например s1[3]. Символ строки совместим с типом **char**, их можно использовать в выражениях одновременно, например: s1[3] := 'h'; writeln (s2[3] + 'r'); …

5.Можно осуществлять коррекцию любого символа строковой переменной(например, str[3]:='j').

6. Элементы строки нумеруются с единицы.

7. Чтобы узнать текущую длину стрки, достаточно применить функцию **ord** к нулевому элементу строки. Например: writeln(ord(st[0])) …

Процедуры и функции для работы со строками

При работе со строками, как правило, возникает необходимость выполнять их копирование, вставку, удаление или поиск. Для эффективной реализации этих действий в Паскале предусмотрены стандартные процедуры и функции. Они кратко описаны ниже.

Функция **Concat (s1, s2, ..., sn)** возвращает строку, являющуюся слиянием строк s1, s2, ..., sn.
Функция **Copy (s, start, len)** возвращает подстроку длиной len, начинающуюся с позиции start строки s.
Процедура **Delete (s, start, len)** удаляет из строки s, начиная с позиции start, подстроку длиной len.
Процедура **Insert (subs, s, start)** вставляет в строку s подстроку subs, начиная с позиции start.
Функция **Length (s)** возвращает фактическую длину строки s, результат имеет тип byte.
Функция **Pos (subs, s)** ищет вхождение подстроки subs в строку s и возвращает номер первого символа subs в s или нуль, если subs не содержится в s.

Процедуры преобразования типов

Процедура **Str (x, s)** преобразует числовое значение x в строку s, при этом для x может быть задан формат, как в процедурах вывода write и writeln.
Например:
x := 123; s := str(x:6,s); - Результат: s = ' 123'.

Процедура **Val (s, x, errcode)** преобразует строку s в значение числовой переменной x, при этом строка s должна содержать символьное представление числа. В случае успешного преобразования переменная errcode равна нулю. Если же обнаружена ошибка, то errcode будет содержать номер позиции первого ошибочного символа, а значение x не определено.

Пример.
Задача: выяснить сколько пробелов в введенной строке.

```
program Probely;
var s: string;
    i , n: integer;
begin
  n := 0;
  writeln('Введите исходную непустую строку:');
  readln(s);
  for i:=1 to length(s) do
  if s[i]=' ' then n := n+1;
  writeln('В строке "',s,'    ',n,' пробелов.');
end.
```