Обработка символьной информации

(TurboPascal)

Обработка символов

Символьные данные включают в себя символьные константы и символьные переменные.

Обработка символов

Символьная константа (строка символов) – это последовательность буквенно-цифровых и / или специальных символов, заключенная в апострофы.
Примеры, ‘дом’ , ‘name’ , ‘123’ , ‘’ , ‘ ‘ .
Если необходимо записать символ апострофа, он удваивается.

Обработка символов

Имя **символьной переменной** образуется так же, как имя числовой переменной.

В языке Паскаль для работы с символьными данными используются два основных типа: CHAR и STRING.

Тип **CHAR** предназначен для хранения 1 символа.

Тип **STRING** (строка) предназначен для хранения последовательности символов.

Операции над символьными данными

Над данными типа **char** можно выполнять две операции: операцию **присваивания и сравнения**:

- Переменной типа char можно присваивать значение константы типа char либо значение другой переменной типа char.
- Из двух символов больше тот, порядковый номер в таблице которого больше.

Существует таблица символов ПК, в которой каждый символ имеет свой порядковый номер.

Для получения этого номера используется функция ***ORD(C)***.

Для получения символа соответствующего определённому номеру используется функция ***CHR(N)***, где 0<=N<=255.

Над данными типа **string** выполняются:

Операция присваивания:

```
var
  s1, s2 : string;
begin
  s2 := ‘abc’;
  s1 := s2;
end.
```

Операция конкатенации или склеивания:

```
s1 := ‘строка1’;
s2 := ‘строка2’;
s3 := s1 + s2;
s3 := s1 + ‘строка’;
s3 := s1 + ‘a’;
s3 := ‘Мама мыла’ +’ ’ + ‘раму.’
```

Операции отношения =, <>, >, <, >=, <=. Данные операции выполняются над двумя строками посимвольно, слева направо с учетом внутренней кодировки символов. Если одна строка меньше другой по длине, недостающие символы короткой строки заменяются значением *CHR(0).*

Примеры:

‘’ < ‘.’

‘A’ > ‘1’

‘Turbo’ < ‘Turbo Pascal’

‘Паскаль’ > ‘Turbo Pascal’

Стандартные процедуры и функции

LENGTH (S) — функция типа INTEGER; вычисляет длину строки S.

```
var
  s : string[50]; n : integer;
begin
  s := 'Hello World!';
  n := Length(s);
 WriteLn (‘длина строки -’, n );
end.
```

COPY(s, n, x) — функция типа *string;*
копирует из строки **s** **x** символов, начиная с символа с номером **n**.

```
var
  s1, s2 : string;
begin
  s1 := 'программа';
  s2 := copy(s1, 4, 5);
  WriteLn(s2);
end.
```

CONCAT(S1 [,S2, ... ,SN]) — функция типа *string;* выдает строку, представляющую собой сцепление строк—параметров *S1, S2, ..., SN.* Данная функция несколько медленнее операции ‘+’.

```
const
  s1 = 'Мама';
var
  s2, s3, s : string;
begin
  s2 := 'мыла';
  s3 := 'раму.';
  s := concat(s1, ' ', s2, ' ', s3);
  WriteLn(s);
end.
```

POS(S1, S) — функция типа *INTEGER;* отыскивает в строке S первое вхождение подстроки *S1* и выдает номер позиции, с которой она начинается; если подстрока не найдена, возвращается ноль.

```
var
  s : string;
begin
  s := ‘Мама мыла раму.’;
  WriteLn(pos(‘ ‘, s));
end.
```

UPCASE (C) — функция типа *CHAR;* переводит строчную латинскую букву C, соответствующую заглавную букву;

если значением *C* является любой другой символ (в том числе строчная буква русского алфавита), функция выдает его без преобразования.

DELETE (S, N, C) — процедура; удаляет *C* символов из строки *S*, начиная с символа с номером *N*.

```
var
  s : string;
begin
  s := 'котелок';
  delete(s, 3, 4);
  WriteLn(s);
end.
```

INSERT (S, ST, N) — процедура; вставляет подстроку *S* в строку *ST*, начиная с символа с номером *N.*

```
var
  s1, s2 : string;
begin
  s1 := 'кок';
  s2 := 'решо';
  insert(s2, s1, 3);
  WriteLn(s1);
end.
```

Очистка экрана

```
Program pr;
    Uses  crt;
    Var   …;
    begin
          clrscr;
          …
    end.
```