

О некоторых расширенных возможностях языка Паскаль при программировании в среде Турбо-Паскаль.

(подготовлено И.В.Горячей и В.Н.Пильщиковым)

Данные учебно-методические материалы предназначены для студентов первого курса факультета ВМК МГУ имени М.В.Ломоносова и могут быть для них полезны при выполнении практических заданий на ЭВМ [1] в среде Турбо-Паскаль [2], поддерживающей расширенный вариант языка Паскаль (по сравнению со стандартом на этот язык, изучаемым в рамках курса «Алгоритмы и алгоритмические языки»). Представленная в материалах вспомогательная информация является, прежде всего, полезным дополнением к основному тексту из пособия [1, с.15-24], где даётся описание практического задания №4 («Язык Паскаль. Интерфейс программы сортировки»).

1. Порядок разделов в описаниях

Если в стандарте языка Паскаль зафиксирован жесткий порядок следования разделов описаний в начале блоков (метки, константы, типы, переменные, процедуры и функции), то в Турбо-Паскале эти разделы могут быть расположены в любом порядке, причем каждый раздел может появляться многократно. Однако при этом сохраняется общее правило Паскаля: на любое имя можно ссылаться только после того, как оно описано.

2. Дополнительные целые типы

Стандартный тип *integer* определяет целые числа размером в 2 байта (от -2^{15} до $2^{15}-1$). Помимо этого в Турбо-Паскале имеются и другие стандартные целые типы со следующими именами:

shortint – целые размером в 1 байт (от -128 до +127)

longint – целые размером в 4 байта (от -2^{31} до $2^{31}-1$)

byte – неотрицательные целые в 1 байт (от 0 до 255)

word – неотрицательные целые в 2 байта (от 0 до 65535)

Отметим также, что если некоторый символ (типа *char*) надо задать его десятичным кодом, например кодом 83, то тогда вместо *chr(83)* можно записать #83. Например: `c := #83`

3. Типизированные константы

Так не очень удачно называются обычные переменные, но имеющие начальные значения, т.е. значения, которые они получают при входе в программу. Затем эти значения можно и изменить. Описываются эти переменные таким образом:

`const <имя переменной> : <её тип> = <константное выражение>;`

где константное выражение (выражение, операндами которого могут быть только константы и обращения к стандартным функциям с параметрами-константами) как раз и задает начальное значение переменной. Например:

```
const   c:   char = #27;
        attr: byte = 16*LightGray+Black;
```

...
с:=’f’;

Замечание. В качестве типизированных констант можно описывать и массивы, например:

```
const methods: array[1..10] of string = ('Direct Insertion', 'Binary Insertion',  
    'Bubble Sort', 'Shuttle Sort', 'Simple Choose', 'Shell Sort', 'Quick Sort (rec)',  
    'Quick Sort (non)', 'Simple Merger', 'Natural Merger');
```

Используемый в этом примере тип *string* объясняется далее в пункте 6.

4. Дополнительные процедуры и функции.

inc(x) – $x:=x+1$ (x – целочисленная переменная);

dec(x) – $x:=x-1$ (x – целочисленная переменная);

random(n) – функция, при каждом обращении к которой выдается новое случайное целое число из $[0, n-1]$;

halt – принудительный останов программы;

exit – принудительный выход из тела той процедуры/функции, где встретился этот оператор;

break – досрочный выход из цикла (не дожидаясь выполнения условия окончания цикла);

continue – досрочное завершение текущей итерации цикла и переход на новый шаг цикла (если такой есть).

5. Расширенные возможности оператора CASE.

В списке констант каждого варианта можно указывать диапазон констант. В конце оператора *case* допустима *else*-часть, на которую передается управление, если в предыдущих вариантах нет подходящей константы. Например:

```
case k of  
  1,4..7,15: x:=5; {выполняется при k=1,4,5,6,7 или 15}  
  9: x:=4; {выполняется при k=9}  
  10..12: inc(x); {выполняется при k=12,11 или 12}  
  else x:=0 {выполняется при остальных значениях k}  
end;
```

6. Строки переменной длины (тип *string*)

В Турбо-Паскале можно использовать строки в том смысле, как они определены в стандарте языка (как упакованные символьные массивы). Однако в Турбо-Паскале имеются более удобные строки переменной длины (далее – просто “строки”), размер которых может меняться в процессе выполнения программы. Эти строки считаются относящимися к стандартному (для Турбо-Паскаля) типу *string*.

6.1. Описание строк и представление их в памяти

var S: string[m]; {m – максимальная длина строки S ($m \leq 255$)}

Если $m=255$, то часть [255] можно опустить:

var T: string; {эквивалентно: var T: string[255]}

Строка S представляется в памяти как символьный массив из $m+1$ байтов, которые индексируются от 0 до m . В байте $S[0]$ хранится текущая длина k ($k \leq m$) строки, а в байтах $S[1], \dots, S[k]$ – сами символы строки (элементы же $S[k+1]$ и т.д. считаются не относящимися к текущему значению строки). Доступ к элементам строки осуществляется с помощью индексных переменных $S[i]$, которые рассматриваются как переменные типа *char*. Переменная $S[0]$ обеспечивает доступ к текущей длине строки, но следует учитывать, что эта переменная имеет тип *char*, поэтому узнать длину строки можно с помощью выражения $\text{ord}(S[0])$, однако лучше воспользоваться стандартной функцией *length* (см. ниже).

6.2 Строки-константы

Явно заданные строки записываются как ' $c_1c_2\dots c_n$ ', где c_i – символы и $n \geq 0$. Допускается пустая строка (''). Строка из одного символа (например, 'a') является одновременно и величиной типа *char*.

6.3 Операции над строками

1) *Присваивание*: $S := \langle \text{строковое выражение} \rangle$

Особый случай здесь: если длина присваиваемой строки больше максимальной длины строки S , то лишние справа символы отбрасываются. Пример:

```
var S1: string[10]; S2: string[5];
```

```
...
```

```
S1 := '12345678'; {значение S1 – строка '12345678'}
```

```
S2 := S1; {значение S2 – строка '12345'}
```

2) *Конкатенация (сцепление) строк*: $S1+S2+\dots+S_k$

Результат – строка, полученная последовательным выписыванием символов указанных строк. Если получилось более 255 символов, то 256-й и последующие отбрасываются.

3) *Сравнение строк на =, <>, <, <=, >, >=*

Длины сравниваемых строк могут быть различными. Например, 'abc' > 'ab' - истина

6.4 Стандартные функции для работы над строками

$\text{length}(S)$ – текущая длина строки S ;

$\text{pos}(SS, S)$ – номер позиции в строке S , с которой начинается первое вхождение подстроки SS в S , или 0, если SS не входит в S . Например:

```
pos('*', '+-*/+') → 3
```

```
pos('ac', 'abcde') → 0
```

$\text{copy}(S, p, n)$ – равно $S[p..p+n-1]$, т.е. выдается копия части строки S из n символов, начиная с p -го.

Особые случаи: $p > 255$ – ошибка

$p > \text{length}(S)$ – выдается пустая строка

$p+n-1 > \text{length}(S)$ – выдается $S[p..\text{length}(S)]$

6.5 Некоторые стандартные процедуры для работы над строками

Delete(S,p,n) – $S := S[1..p-1]+S[p+n..length(S)]$, т.е. из строки S удаляется n символов начиная с p-го.

Особые случаи: $p > 255$ – ошибка

$p > length(S)$ – S не меняется

$p+n-1 > length(S)$ – $S := S[1..p-1]$

Insert(SS,S,p) – вставка подстроки SS в строку S начиная с p-ой позиции.

Например: $S := 'abcde'$; $insert('XX',S,3)$; $\{S := 'abXXcde'\}$

Особые случаи: $p > 255$ – ошибка

$p > length(S)$ – $S := S+SS$

Если длина результата больше максимальной длины S, то лишние справа символы отбрасываются.

Str(x,S) – аналог *write(x)* при числовом x, но значение x (как последовательность символов) не выводится, а записывается в строку S. Параметр x должен иметь тот же вид, что и в процедуре *write*: e, или e:n, или e:n:d.

6.6 Строковые функции, параметры-строки

В Турбо-Паскале разрешены функции, значениями которых являются строки. При описании таких функций их тип указывается только именем типа:

```
type T=string[80];
```

```
function F1(...):T;... function F2(...):string; ...
```

Формальные параметры-строки также описываются только именем типа:

```
procedure P(var a:T; b:string); ...
```

Если параметр-строка вызывается по значению, то в качестве фактического параметра может быть указана строка любой длины (если она слишком длинная, то лишние справа символы будут отброшены). Если же параметр-строка вызывается по ссылке, то типы формального и фактического параметров должны быть идентичными. Из этого, в частности, следует, что у этих строк должны быть одинаковые максимальные длины; однако это ограничение можно обойти (оно не будет проверяться), если в начале программы указать директиву $\{ \$V-\}$.

Литература

1. Трифонов Н. П., Пильщиков В. Н. Задания практикума на ЭВМ (первый курс): Учебное пособие для вузов. – М.: МАКС-Пресс, 2001.
2. Фаронов В.В. Turbo Pascal 7.0. Начальный курс: Учебное пособие. – ОМДГрупп, 2003