第15卷 第4期 郑州工学院学报 Vol.15 No.4
1994年 12 月 Journal of Zhengzhou Institute of Technology Dec. 1994

TP编译中串类数据功能的局限性*

王文义 孙立贤

(郑州工学院计算机与自动化系)

摘 要: 本文从研究系统应用软件的角度出发，详细探讨了*TURBO PASCAL*编译器串处理功能的局限性及其补救办法，从而为进一步的解决一类符号处理与识别问题找到了一条途径。

关键词: 串处理；二义性；*ASCII*码

中图分类号: TP312

在系统应用软件设计中，有关符号识别和处理等非数值问题的内容通常要占很大的比重，而其中使用最多的又当属串处理功能。实现这个功能，常用的系统程序设计语言有两种，即C语言和PASCAL语言。

C语言，风靡当今世界，是一种非常灵活且功能又很强的语言，因它具有许多能对硬件进行操作的类似于汇编程序设计的功能，如位操作和寄存器操作等，故有人称它是"低级中的高级语言"。另外，随着C语言所配的运行库函数越来越丰富，所以除了unix外，近几年来，也波及到了大量的图形软件、工作站系统软件及窗口软件等。C语言，由于其随意性很强，因此使它的可读性受到了一定影响。同时，其结构性和串处理效率也存在有缺憾之处。

PASCAL语言是一种适合于结构化程序设计的经典性语言。数据类型丰富，结构严谨、可读性强是它突出的特点。如Borland Corp.的TURBO PASCAL V6.0就在其较低版本基础上增加了许多字符处理功能，尤其是串处理，首字节存贮的设置使得它的处理效率高于C语言。所以，从某种角度来看，PASCAL语言较之C语言更适宜于用来实现一类系统应用软件对字符的处理和识别工作。当然，不存在十全十美的计算机语言，PASCAL语言也不例外。笔者通过对Borland TURBO PASCAL V6.0（下简称TP）大量的实验测试及应用，发现它在串处理功能应用上也存在着某些不尽人意之处。针对这些不足，也都找到了相应的解决办法。现加以总结整理，以供读者参考。

通常我们在研制某些系统应用软件，如词法分析器，专用编译器或高性能的编辑工具时，串处理功能的使用频度是相当高的，因此用户很希望有一种串型数据，它既能象字符数组那样，以下标形式进行访问，也可以对其串名直接进行I／O。TP的使用说明中，

* 收稿日期: 1994-05-23

STRING 类型数据似可以满足这些要求，但在实际应用中却暴露出了一些问题，感到颇为不便。不面简要介绍这些问题产生的原因及对策。

1 关于串变量首字节 S[0]的问题

在 TP 中，提供有 STRING 类型数据。设串变量名为 S，则其说明方式有两种:

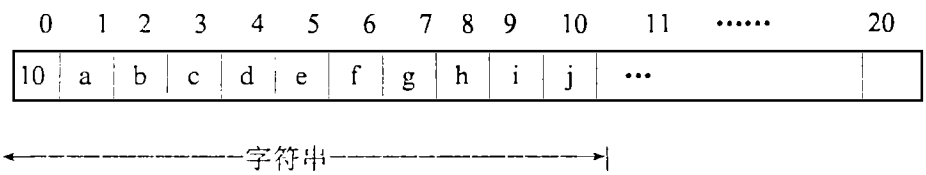
VAR S: STRING[20];

或 VAR S: STRING[ ];

其中后者隐含说明串 S 中最多可有 255 个字符。若以第一种说明为例，让

S: ='abcdefghij'

则 TP 将串 S 开辟 11 个字节，其中第一个字节 S[0]存放串长度，第 2 个字节到第 11 个字节存放串的实际内容且其下标依次为 1..10。串 S 的存贮形式如下:



注意: ① 首字节 S[0]中并不是存放的十进制数'10'，而是二进制数'00001010'。

② 若以下标访问首字节，即显示或输出 S[0]，则结果为一个 ASCII 码，并非 S[0]=10。

③ 若说明的串长度大于实际串长度，则多出字节中以随机数充之。

这里，S[0]中的'0'是否指下标? 按照 TP 编译器处理的实际结果应理解为: 所有字符串都在位置'0'处有一个包含串长度的字符，只有语句 ORD (S[0]) 才能把该字符转变成等价的字符串长度。对用户而言，尽管平时极少注意到 S[0]，但用 ORD(S[0])表示 S 长度的规定就容易产生二义性。因为按照常规，ORD 在 TP 中是 ASCII 字符集的序数函数，而用户往往注意到的是 TP 中关于"S[0]中放的是串长度"这句话上，因此让 ORDS ([0]) 转意成串长度，是容易让人误解的。实际上，编程者完全可以用 LENGTH (S) 直接测出串 S 的长度来。详情可参考 Test 源程序及其输出结果（在输出结果中，有下划线者为键盘输入内容，其它为输出内容。下同)。

Test 源程序:

```
{----TEST.PAS----}
program a;
var  s:string[20];  i,j,k:integer;
begin
  writeln('INPUT STRING S !');
  readln(s);
  j:=length(s);
  writeln('j=',j);
  for i:=0 to j do
```

运行结果:

```
{----TEST.RESULT----}
INPUT STRING S !
abcde↵
j=5
s[0]=     ord(s[0])=5
s[1]=a    ord(s[1])=97
s[2]=b    ord(s[2])=98
s[3]=c    ord(s[3])=99
```

```
    writeln('s[', i, ']=', s[i],          s[4]=d    ord(s[4])=100
    '    ord(s[', i, '])', ord(s[i]));    s[5]=e    ord(s[5])=101
end.
```

2　重复变换STRING所引起的问题

在编辑工具中，用户经常需要在不同时间内给同一串变量赋予不同的内容（如执行dos命令等）。但在TP中，这样使用，在有些情况下，会产生一些令人奇怪的结果。请看下面的Testo程序及其运行结果：Testo源程序：

```
program aa;
label 1;
var
  s:string[5];ch:char;i,j:integer;
begin
  1: writeln('INPUT STRING S !');
  s:='';
  j:=length(s);
  writeln('***j=',j);
     writeln('--------------------------------');
     writeln(')))))))))ch=',ch,'(((((((((((');
     for i:=1 to 5 do
       begin
         read(ch);
         s:=s+ch;
         writeln('>>>>>>>>>>ch=',ch,'<<<<<<<<<<s=',s);
       end;
     j:=length(s);
     writeln(']]]]]]]]]]j=',j,']]]]]]]]]]s=',s);
     delete(s,1,j);
     writeln('}}}}}}}}}}s=','s,'{{{{{{{{{{');
     goto 1;
end.
```

运行结果：

```
INPUT STRING S !
***j=0
--------------------------------
))))))))))ch= ((((((((((
abcde ↵
>>>>>>>>>>ch=a<<<<<<<<<<s=a
>>>>>>>>>>ch=b<<<<<<<<<<s=ab
>>>>>>>>>>ch=c<<<<<<<<<<s=abc
>>>>>>>>>>ch=d<<<<<<<<<<s=abcd
>>>>>>>>>>ch=e<<<<<<<<<<s=abcde
]]]]]]]]]]j=5]]]]]]]]]]s=abcde
}}}}}}}}}}s={{{{{{{{{{
INPUT STRING S !
***j=0
```

```
------------------------------
))))))))))ch=e((((((((((
<<<<<<<<<<s==
>>>>>>>>>>ch=
                <<<<<<<<<<s=
```

2.1 在testo中，当控制（goto）返回重新执行循环语句时，本应由键盘输入第二个字符串，但实际情况是，还未待输入，就出现了一些输出内容，且输出格式被破坏。

2.2 分析

由于在循环中使用了read输入格式，该格式不会回车换行。而键盘输入却需要在头5个字符输入结束时用敲击INTER键以示送入内存。这个动作的效果一直滞留到第二轮循环中表现出来：即把回车码（序号13）和换行码（序号10）分别置入ch中，它们是控制码，故在输出ch时，就相继执行了回车和换行两个动作，从而破坏了编程者希望的输出结果。为了证明上述分析，可在Testo中的read（ch）和S:=s+ch之间加上测试句：writeln('!!!!!!!!!! ord(ch)=', ord(ch),'!!!!!!!!!!');输出结果如下：

```
INPUT STRING S !
***j=0
--------------------------------
))))))))))ch=((((((((((
abcde
!!!!!!!!!!ord(ch)=97!!!!!!!!!!
>>>>>>>>>>ch=a<<<<<<<<<<s=a
!!!!!!!!!!ord(ch)=98!!!!!!!!!!
>>>>>>>>>>ch=b<<<<<<<<<<s=ab
!!!!!!!!!!ord(ch)=99!!!!!!!!!!
>>>>>>>>>>ch=c<<<<<<<<<<s=abc
!!!!!!!!!!ord(ch)=100!!!!!!!!!!
>>>>>>>>>>ch=d<<<<<<<<<<s=abcd
!!!!!!!!!!ord(ch)=101!!!!!!!!!!
>>>>>>>>>>ch=e<<<<<<<<<<s=abcde
]]]]]]]]]]j=5]]]]]]]]]]s=abcde
}}}}}}}}}}s={{{{{{{{{{
INPUT STRING S !
***j=0
--------------------------------
))))))))))ch=e((((((((((
!!!!!!!!!!ord(ch)=13!!!!!!!!!!
<<<<<<<<<<s==
!!!!!!!!!!ord(ch)=10!!!!!!!!!!
>>>>>>>>>>ch=
                <<<<<<<<<<s=
```

由此可见测试结果与分析是一致的。

2.3 解决办法

① 改read（ch）为readln（ch）。

② 改变循环结构，滤掉ch中的控制字符。

上述两种方法彼此独立，前者虽然简捷但却增加了用户输入的麻烦。后者则仅需加上一句过滤句并改变一下循环结构即可。其相应源程序及运行结果如LOCK1(源程序)和LOCK1(运行结果)所示。LOCK1（源程序）

```
program aaa;
label 1;
var
  s:string[5];ch:char;i,j:integer;
begin
  1: writeln('INPUT STRING S !');
  s:='';
  j:=length(s);
  writeln('***j=',j);
```

```
    writeln('--------------------------------');
    writeln('))))))))))ch=', ch,' ((((((((((((');
        i:=0;
        repeat
          read(ch);
          if(ord(ch)>=32) and (ord(ch)<=126) then
            begin
              i:=i+1;
              s:=s+ch;
              writeln('>>>>>>>>>>ch=', ch,' <<<<<<<<<<s=', s);
            end
          until i=5;
    j:=length(s);
    writeln(']]]]]]]]]]j=', j,']]]]]]]]]]s=', s);
    delete(s, 1, j);
    writeln('}}}}}}}}}}s=', s,' {{{{{{{{{{');
    goto 1;
end.
```

LOCK1 (运行结果)

```
{----LOCK1.RESULT----}
INPUT STRING S !
***j=0
--------------------------------
))))))))))ch=((((((((((
abcde
>>>>>>>>>>ch=a<<<<<<<<<<s=a
>>>>>>>>>>ch=b<<<<<<<<<<s=ab
>>>>>>>>>>ch=c<<<<<<<<<<s=abc
>>>>>>>>>>ch=d<<<<<<<<<<s=abcd
>>>>>>>>>>ch=e<<<<<<<<<<s=abcde
]]]]]]]]]]j=5]]]]]]]]]]s=abcde
}}}}}}}}}}s={{{{{{{{{{
INPUT STRING S !
***j=0
--------------------------------
))))))))))ch=e((((((((((
```

```
#$%^&↓
>>>>>>>>>>ch=#<<<<<<<<<<s=#
>>>>>>>>>>ch=$<<<<<<<<<<s=#$
>>>>>>>>>>ch=%<<<<<<<<<<s=#$%
>>>>>>>>>>ch=^<<<<<<<<<<s=#$%^
>>>>>>>>>>ch=&<<<<<<<<<<s=#$%^&
]]]]]]]]]]j=5]]]]]]]]]]s=#$%^&
}}}}}}}}}}s={{{{{{{{{{
INPUT STRING S !
***j=0
--------------------------------
))))))))))ch=&((((((((((
```

3 对几种特殊情况的处理

TP 中的 STRING 型变量，并不是在任何情况下都可以访问其名的。下面分几种情况讨论:

3.1 当以下标形式逐个输入字符形成串 S 时，既不能访问其名，也不能用 LENGTH 检测出其长度，但以下标形式却可以任意访问。请参看 Test1 源程序及其输出结果:

Test1 源程序:

```
program aaaa;
var      s:string[5];  ch:char;i,j:integer;
begin
  writeln('INPUT STRING S !');
  s:='';
for i:=1 to 5 do
 begin
   read(ch);
   s[i]:=ch;
 end;
writeln('^^^^^s=',s,'^^^^^');
 writeln('#####l=',length(s),'#####');
 for i:=1 to 5 do write(s[i]);
 writeln;
end.
```

运行结果:

```
INPUT STRING S !
abcde↓
^^^^^s=^^^^^
#####l=0######
abcde
```

3.2 若以追加形式形成串 S，则既可访问串名，也可以用下标进行访问。请看下述 Test2 程序及其结果。

Testl2 源程序:

```
{----TEST2.PAS----}
program aaaaa;
var     s:string[5]; ch:char; i,j:integer;
begin
  writeln('INPUT STRING S !');
  s:='';
for i:=1 to 5 do
 begin
   read(ch);
   s:s+ch;
 end;
writeln('^^^^^s=',s,'^^^^^');
 writeln('#####l=',length(s),'#####');
 for i:=1 to 5 do write(s[i]);
 writeln;
end.
```

运行结果:

```
INPUT STRING S !
abcde↵
^^^^^s=abcde^^^^^
#####l=5#####
abcde
```

3.3　若以函数 CONCAT 逐个字符连接成串 S，则结果同 3.2。参看程序 Test3 及其结果。

Test3 源程序:

```
program aaaaaa;
var     s:string[5]; ch:char; i,j:integer;
begin
  writeln('INPUT STRING S !'),
  s:='';
for i:=1 to 5 do
 begin
   read(ch);
   s:=concat(s,ch);
 end;
writeln('^^^^^s=',s,'^^^^^');
 writeln('#####l=',length(s),'#####');
 for i:=1 to 5 do write(s[i]);
 writeln;
end.
```

运行结果:

```
INPUT STRING S !
abcde↵
^^^^^s=abcde^^^^^
#####l=5#####
abcde
```

4 结束语

TP 中的串处理功能非常丰富，使用也十分灵活，只要应用得当，完全可以胜任各种不同的复杂要求。只是由于在 TP 中没有给出各种使用方法的详细边界条件说明，如在什么样的环境中可以使用什么样的语句以及语句间如何有效的搭配等，给编程者带来了一定的困难。作者通过上述实践和探讨，提出了几种合理使用串处理功能的方法，建议对此有兴趣的读者不妨也试一试，总结出一些更好的经验来，从而在研制某些应用软件时，少走一些弯路。

参 考 文 献

1 陈世录.张华凯.谭军安等编译.TURBO PASCAL 技术参考大全.中科院希望电脑技术公司.1991

2 刘乃琦编著.IBM PC 混合语言编程技术.电子工业出版社.1992

3 萧黎.尤晓东等编.TURBO C V2.0 运行库函数源程序与参考大全.中科院希望电脑技术公司.1990

The Functional Limitation of TP Compiler in Dealing with the String Type Datta

Wang Wenyi Sun Li-xian
(Zhengzhou Institute of Technology)

Abstract: From the viewpoint of the application of the system software, this paper presents a thorough study of the functional Limitation of the Turbo Pascal compiler in dealing with the string type data, and provides the remedial measures so as to find a new way of treating and distinguishing a Kind of signals.

Keywords: String manipulation; ambiguity; ASCII code