CHAPTER 5

指针与字符串

5-1 字符串指针

字符串是字符的集合，可利用数组（指针常量）或指针变量来表示。

```
char *str1="Apple iPod";
```

表示 str1 指向字符串的第一个字符'A'，如图 5-1 所示。

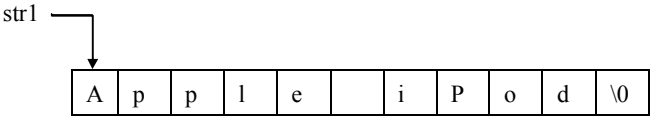

图 5-1

也可使用数组的表示法，如下所示：

```
char str2[10]=”Apple iPod”;
```

以图形表示如图 5-2 所示。

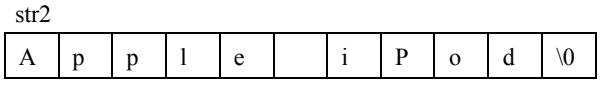

图 5-2

str2 是数组名，表示此数组第一个元素的地址（相当于&str2[0]），str2+1 表示数组第二个元素的地址（相当于&str2[1]），……，依此类推，str2+10 则是空字符（'\0'）的地址。字符串的最后都有一个空字符（'\0'），作为字符串的结束点，若缺少它，将会产生错误的答案。

还要注意的是，str1 是指针变量，而 str2 是指针常量。所以 str2 不可以使用++（自增）和--（自减）运算符。首先，利用指针变量指向一个字符串，并以字符的格式将字符串输出，请参阅范例 string3。

范例 string3

```
/* string3.c */
#include <stdio.h>
#include <conio.h>
int main()
{
      char *str = "Apple iPod";
      int i;
      printf("字符串 str 为: ");
      for(i=0; *str!='\0'; i++)
      {
            printf("%c", *str);
            str++;
      }
      getch();
      return 0;
}
```

输出结果

```
字符串 str 为: Apple iPod
```

程序利用循环输出目前指针所指向的字符，直到空字符（'\0'）为止。上例的字符串也可以使用数组的方式加以输出。请参阅范例 string4。

范例 string4

```
/* string4.c */
#include <stdio.h>
#include <conio.h>
int main()
{
      char *str = "Apple iPod";
      int i;
      printf("字符串 str 为: ");
      for(i=0; str[i]!='\0'; i++)
            printf("%c", str[i]);
      getch();
      return 0;
}
```

输出结果同范例 string3。其实最方便的方法是利用%s，直接将字符串加以输出。如范例 string5 所示。

范例 string5

```
/* string5.c */
#include <stdio.h>
#include <stdlib.h>
int main()
{
      char *str1 = "Apple iPod";
      char str2[] = "Apple iPod";

      printf("字符串 str1 为 %s\n", str1);
      printf("字符串 str2 为 %s\n", str2);

      system("PAUSE");
      return 0;
}
```

输出结果

```
字符串 str1 为 Apple iPod
字符串 str2 为 Apple iPod
```

要注意的是，此时变量给予的是数组名，如 str1 和 str2。因为数组名表示数组第一个元素的地址，所以会从此地址一直打印到字符串的结束点（空字符'\0'）。若将上一范例的 str2 数组元素的个数修改为 10，其结果为何，请参阅范例 string7。

范例 string7

```
/* string7.c */
#include <stdio.h>
#include <stdlib.h>
int main()
{
      char *str1 = "Apple iPod";
      char str2[10] = "Apple iPod";
```

```
      printf("字符串 str1 为 %s\n", str1);
      printf("字符串 str2 为 %s\n", str2);

      system("PAUSE");
      return 0;
}
```

输出结果

```
字符串 str1 为 Apple iPod
字符串 str2 为 Apple iPod+
```

在输出结果中，str2 的字符串是错误的，因为分配给 str2 的空间不足，使得它无法存放空字符（'\0'）。这也就是为什么常将数组的元素的个数省略，而由系统去决定的原因。

除了在定义字符串变量时，直接给予一个字符串外，也可利用其他的方式得到。请参阅范例 string10。

范例 string10

```
/* string10.c */
#include <stdio.h>
#include <stdlib.h>
int main()
{
      char *str1 = "Learning pointer now!";
      char str2[] = "Go ahead";
      char *str3;
      char str4[20];

      str3 = "I want to buy a iPod\n";
      printf("C or C++: ");
      scanf("%s", str4);

      printf("\n%s\n", str1);
      printf("%s\n", str2);
      printf("%s\n", str3);
      printf("Learning %s now\n", str4);
      system("PAUSE");
      return 0;
}
```

输出结果

```
C or C++: C
Learning pointer now
Go ahead
I want to buy a iPod

Learning C now
```

str1 和 str2 都是在定义时指定初值，str4 数组是以键盘输入的方式得到一个字符串，而 str3 指针变量则是以赋值方式得到一个字符串。

注意，若将此范例的 str3 改为数组，而 str4 以指针变量表示，则会有错误信息产生，如下所示。

```
char str3[20];
char *str4;
```

为什么呢？因为 str3 是一个数组，它是指针常量，所以 str3 已在固定的地址，不可以再指定一个字符串地址给它。str4 是一个指针变量，不可以由输入的方式来完成指定的操作。这些都是大家很容易出错的地方。

5-2 常用的库函数

通常谈到字符串的操作，不外乎计算**字符串长度（string length）**，**进行字符串复制（string copy）**、**字符串连接（string concatenate）**、**字符串比较（string compare）**。要完成这些动作，最方便的是调用字符串的库函数，但要包含 string.h 头文件，因为这些字符串库函数的原型都在这个头文件中。

以下的范例除了调用字符串的库函数外，也利用字符串指针自行编写程序，以模拟这些库函数的功能，希望大家能学到更多有关字符串指针的操作方式。让我们先从计算字符串的长度开始。

5-2-1 计算字符串的长度

字符串的长度表示此字符串有多少个字符，但不包括空字符（'\0'）。计算字符串长度的库函数为 strlen，其语法如下：

```
strlen(str1);
```

表示计算 str1 字符串共有多少个字符，但不包括空字符。请参阅范例 stringLength。

范例 stringLength

```
/* stringLength.c */
#include <stdio.h>
#include <string.h>
int stringLength(char *);

int main()
{
      int length=0;
      char *str = "Apple iPod";
      printf("The length of \"%s\" is %d\n\n", str, strlen(str));

      /* using my method */
      length = stringLength(str);
      printf("The length of \"%s\" is %d\n", str, length);
      system("PAUSE");
      return 0;
}

int stringLength(char *p)
{
      int t=0;
```

```
      while(*p !='\0') {
            t++;
            p++;
      }
      return t;
}
```

输出结果

```
The length of "Apple iPod" is 10

The length of "Apple iPod" is 10
```

在 stringLength()函数中，若*p 不为空字符（'\0'），则将变量 t 加 1，之后再把 p 指针移到下一个字符的地址。上述的操作将持续进行，直到*p 为空字符才结束，因为空字符是字符串的结束字符。

5-2-2 字符串的复制

字符串复制的库函数为 strcpy，其语法如下：

```
strcpy(str1, str2);
```

表示从 str2 字符串的第一个字符，逐一地复制到 str1 字符串。注意，第一个参数 str1 是接收者，第二个参数 str2 是给予者。请看范例 stringCopy。

范例 stringCopy

```
/*  stringCopy.c  */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
      char *s = "iPhone";
      char t[80] = " ";
      printf("Before strcpy(t, s)\n");
      printf("s = %s\n", s);
      printf("t = %s\n", t);
      printf("After strcpy(t, s)\n");
      strcpy(t, s);
      printf("s = %s\n", s);
      printf("t = %s\n", t);
      system("PAUSE");
      return 0;
}
```

输出结果

```
Before strcpy(t, s)
s=iPhone
t=
After strcpy(t, s)
s=iPhone
t=iPhone
```

若 t 原来就有字符串，则此字符串会被 s 字符串所覆盖，用户要特别小心使用，以免数据遗失。字符串的复制动作，如图 5-3 所示。

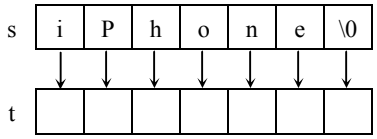

图 5-3

注意！t 字符串要有足够的空间来容纳 s 字符串。

若将范例 stringCopy 中的

```
char t[80] = " ";
```

改为

```
char *t;
```

将会有错误产生。因为 t 是指针变量，且没有一个指向的内存空间，这是我们常犯错的地方。解决的方法是要先分配内存给 t 指针变量。如下所示：

```
t =(char *) malloc(80*sizeof(char));
```

接下来，我们参照字符串复制的原理，自行编写一个用户自定义函数，以模拟 strcpy 函数的做法，如范例 myStringCopy 所示。

范例 myStringCopy

```
/* myStringCopy.c */
#include <stdio.h>
#include <string.h>
void myStringCopy(char *, char *);

int main()
{
      int length=0;
      char str2[80];
      char *str1 = "Apple iPod";

      printf("调用字符串库函数: strcpy()...\n");
      strcpy(str2, str1);
      printf("str2=%s\n\n", str2);

      /* using my method */
      myStringCopy(str2, str1);
      printf("调用用户自定义函数: myStringCopy()...\n");
      printf("str2=%s\n", str2);
      system("PAUSE");
      return 0;
}

void myStringCopy(char *dest, char *source)
{
      while((*dest = *source) != '\0') {
            source++;
            dest++;
      }
}
```

输出结果

```
调用字符串库函数: strcpy()...
Str2=Apple iPod
```

```
调用用户自定义函数: myStringCopy()...
Str2=Apple iPod
```

在 myStringCopy() 函数中，利用 while 循环，将*source 的内容指定给*dest，若字符不是'\0'，则将 source 和 dest 的指针移到下一个地址，继续执行复制的动作。

若只需复制 n 个字符时，则调用 strncpy 库函数，其语法如下：

```
strncpy(str1, str2, n);
```

将 str2 的前 n 个字符复制到 str1。请参阅范例 myStringCopy_n。

范例 myStringCopy_n

```
/* myStringCopy_n.c */
#include <stdio.h>
#include <string.h>
void myStringCopy_n(char *, char *, int n);

int main()
{
      int length=0;
      char str2[80];
      char *str1 = "Apple iPod";
      char str3[80];

      printf("调用字符串库函数: strncpy()...\n");
      strncpy(str2, str1, 5);
      str2[5]='\0';
      printf("str2=%s\n\n", str2);

      /* using my method */
      myStringCopy_n(str3, str1, 5);
      str3[5]='\0';
      printf("调用用户自定义函数: myStringCopy_n()...\n");
      printf("str3=%s\n", str3);
      system("PAUSE");
      return 0;
}

void myStringCopy_n(char *dest, char *source, int n)
{
      int i=1;
      while( (i <= n) && (*dest = *source) != '\0' ) {
            source++;
            dest++;
            i++;
      }
}
```

输出结果

```
调用字符串库函数: strncpy()...
Str2=Apple
调用用户自定义函数: myStringCopy_n()...
Str3=Apple
```

myStringCopy_n() 函数大致上与 myStringCopy() 函数相同，但由于它只复制字符串前 n 个字符，所以多了 i 变量，用以判断它是否小于 n。接下来，讨论

字符串的连接。

5-2-3 字符串的连接

字符串的连接的库函数为 strcat，其语法如下：

```
strcat(str1, str2);
```

表示将 str2 字符串连接到 str1 字符串的尾端。注意，str1 必须要分配足够的空间。请参阅范例 stringConcate。

范例 stringConcate

```
/* stringConcate.c */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void stringConcate(char *, char *);

int main()
{
      char str2[80]="I want to buy a ";
      char *str1 = "Apple iPod";
      char str3[80]="I want to buy a ";

      printf("使用库函数 strcat()....\n");
      strcat(str2, str1);
      printf("str2 = %s\n\n", str2);

      /* using my method */
      stringConcate(str3, str1);
      printf("调用用户自定义函数 stringConcate()....\n");
      printf("str3 = %s\n", str3);
      system("PAUSE");
      return 0;
}

void stringConcate(char *dest, char *source)
{
      while(*dest != '\0')
         dest++;
      while((*dest=*source) != '\0')
      {
            source++;
            dest++;
      }
}
```

输出结果

```
使用库函数 strcat()....
Str2= I want to buy a Apple iPod
调用用户自定义函数 stringConcate()....
Str3= I want to buy a Apple iPod
```

在 stringConcate() 函数中，利用循环先追踪 dest 字符串的结束点，接下来，将 source 指针指向的字符（*source）指定给（*dest），直到*source 是空字符为止。

若只要连接 n 个字符时，则可调用 strncat 库函数，其语法如下：

```
strncat(str1, str2, n);
```

意思是将 str2 的前 n 个字符连接到 str1。请参阅范例 stringConcate_n。

范例 stringConcate_n

```
/* stringConcate_n.c */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void stringConcate_n(char *, char *, int);

int main()
{
      char str2[80]="I want to buy a ";
      char *str1 = "Apple iPod";
      char str3[80]="I want to buy a ";

      printf("使用库函数 strncat()....\n");
      strncat(str2, str1, 5);
      printf("str2 = %s\n\n", str2);

      /* using my method */
      stringConcate_n(str3, str1, 5);
      printf("调用用户自定义函数 stringConcate_n()....\n");
      printf("str3 = %s\n", str3);
      system("PAUSE");
      return 0;
}

void stringConcate_n(char *dest, char *source, int n)
{
      int i=1;
      while(*dest != '\0')
            dest++;
      while( i<=n && (*dest=*source) != '\0' ) {
            source++;
            dest++;
            i++;
      }
}
```

输出结果

```
使用库函数 strncat()....
Str2= I want to buy a Apple
调用用户自定义函数 stringConcate_n()....
str3= I want to buy a Apple
```

stringConcate_n() 函数大致上和 stringConcate 函数相同，只是多了一个 i 变量用以判断是否大于 n。

5-2-4 字符串的比较

字符串的比较也可以说是最常用的。若要比较两个字符串是否相等，有些人可能会这样编写程序，如范例 stringCompare10 所示。

范例 stringCompare10

```
/* stringCompare10.c */
#include <stdio.h>
#include <stdlib.h>

int main()
{
      char str1[] = "Honda Civic";
      char str2[] = "Honda Civic";
      printf("str1 指向的地址: %x\n", str1);
      printf("str2 指向的地址: %x\n", str2);

      if (str1 == str2)
            printf("str1 和 str2 是相等的\n");
      else
            printf("str1 和 str2 是不相等的\n");
      system("PAUSE");
      return 0;
}
```

输出结果

```
str1 指向的地址: 22ff60
str1 指向的地址: 22ff50
str1 和 str2 是不相等的
```

此程序是在比较 st1 和 str2 的地址是否相等，而不是比较内容是否相等。因为 str1 和 str2 是数组名，表示的是数组第一个元素的地址。所以此范例并不是在比较字符串的内容。

若改用指针变量的话，则情况又不一样了，请参阅范例 stringCompare20。

范例 stringCompare20

```
/* stringCompare20.c */
#include <stdio.h>
#include <stdlib.h>
int main()
{
      char *str1 = "Honda Civic";
      char *str2 = "Honda Civic";
      char *str3 = "Honda Accord";

      printf("str1 指向的地址: %x\n", str1);
      printf("str2 指向的地址: %x\n", str2);
      printf("str3 指向的地址: %x\n", str3);

      if (str1 == str2)
            printf("str1 和 str2 是相等的\n");
      else
            printf("str1 和 str2 是不相等的\n");

      if (str1 == str3)
            printf("str1 和 str3 是相等的\n");
      else
            printf("str1 和 str3 是不相等的\n");
      system("PAUSE");
      return 0;
}
```

输出结果

```
str1 指向的地址: 4156dc
str2 指向的地址: 4156dc
str3 指向的地址: 4156e0
str1 和 str2 是相等的
str1 和 str3 是不相等的
```

此范例也是在比较地址，因为指针变量也是地址。从输出结果得知，str1 与 str2 的地址是相等的，因为 str1 与 str2 指针变量都指向相同的字符串"Honda Civic"。而 str3 "Honda Accord"是不一样的字符串，所以 str3 指向的地址和 str1 与 str2 不一样。

也可以利用 malloc 函数先分配内存，再指定一个字符串给它，如范例 stringCompare30 所示。

范例 stringCompare30

```
/* stringCompare30.c */
#include <stdio.h>
#include <stdlib.h>

int main()
{
      char *str1 = (char *)malloc(20);
      str1 =  "Honda Civic";
      char *str2 = (char *)malloc(20);
      str2 = "Honda Civic";
      char *str3 = (char *)malloc(20);
      str3 = "Honda Accord";

      printf("str1 指向的地址: %x\n", str1);
      printf("str2 指向的地址: %x\n", str2);
      printf("str3 指向的地址: %x\n", str3);

      if (str1 == str2)
            printf("str1 和 str2 是相等的\n");
      else
            printf("str1 和 str2 是不相等的\n");

      if (str1 == str3)
            printf("str1 和 str3 是相等的\n");
      else
            printf("str1 和 str3 是不相等的\n");
      system("PAUSE");
      return 0;
}
```

一般我们不会采用前两种方式来比较两个字符串是否相等，这里的用意是让您了解这样的比较方法是错的。当然利用字符串的库函数 strcmp 来比较两个字符串是否相等是最正统也是最安全的。strcmp 函数的语法如下：

```
strcmp(str1, str2);
```

此语句表示比较 str1 与 str2 这两个字符串是否相等。请参阅范例 stringCompare 所示。

范例 stringCompare

```
/* stringCompare.c */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int stringCompare(char *, char *);

int main()
{
      char *str1 = "Honda Civic";
      char *str2 = "Honda Accord";
      int value;

      printf("str1: %s\n", str1);
      printf("str2: %s\n", str2);
      printf("\n 使用库函数 strcmp(str1, str2)....\n");
      value=strcmp(str1, str2);
      switch(value) {
            case 0:
                  printf("str1 与 str2 相等\n");
                  break;
            case 1:
                  printf("str1 大于 str2\n");
                  break;
            case -1:
                  printf("str1 小于 str2\n");
                  break;
      }

      /* using my method */
      value=stringCompare(str1, str2);
      printf("\n 调用用户自定义函数 stringCompare(str1, str2)....\n");
      if(value==0)
            printf("str1 与 str2 相等\n");
      else if(value>0)
             printf("str1 大于 str2\n");
      else
             printf("str1 小于 str2\n");
      system("PAUSE");
      return 0;
}

int stringCompare(char *x, char *y)
{
      for( ; *x == *y; x++, y++)
            if(*x=='\0')
      return 0;
      return *x-*y;
}
```

输出结果

```
str1: Honda Civic
str2: Honda Accord

使用库函数 strcmp(str1,str2)....
str1 大于 str2

调用用户自定义函数 stringCompare(str1,str2)....
str1 大于 str2
```

在 stringCompare()函数中，比较*x 和*y 的内容是否相等，假如相等，则再判断*x 是否为空字符，若不是，则将*x 和*y 的内容相减后返回。以流程图表示，如图 5-4 所示。

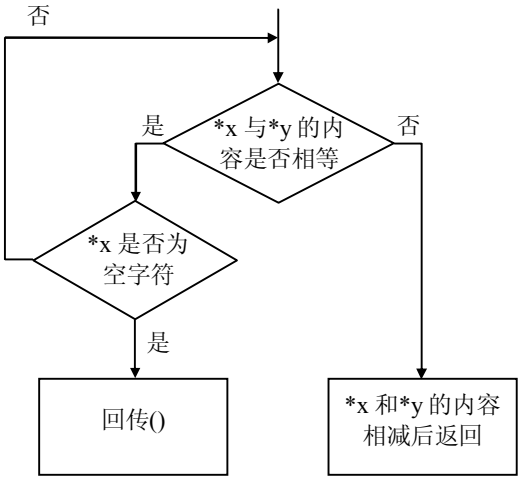

图 5-4

有时在比较两个字符串时，并不需要比较整个字符串，只要比较前面几个字符即可，如范例 stringCompare_n 所示。

范例 stringCompare_n

```
/* stringCompare_n.c */
#include <stdio.h>
#include <string.h>
int stringCompare_n(char *, char *, int);

int main()
{
      char *str1 = "Honda Civic";
      char *str2 = "Honda Accord";
      int value;

      printf("str1: %s\n", str1);
      printf("str2: %s\n", str2);
      printf("\nUsing strncmp(str1, str2, 5)....\n");
      value=strncmp(str1, str2, 5);
      switch(value) {
            case 0:
                  printf("str1 is equal to str2\n");
                  break;
            case 1:
                  printf("str1 is greater than str2\n");
                  break;
            case -1:
                  printf("str1 is less than str2\n");
                  break;
      }

      /* using my method */
```

```
      value=stringCompare_n(str1, str2, 5);
      printf("\nUsing stringCompare_n(str1, str2, 5)....\n");
      if(value==0)
            printf("str1 is equal to str2\n");
      else if(value>0)
            printf("str1 is greater than str2\n");
      else
            printf("str1 is less than str2\n");
   system("PAUSE");
      return 0;
}

int stringCompare_n(char *x, char *y, int n)
{
      int i;
      for(i=1; (*x == *y) || i<=n; x++, y++)
      if(*x == '\0')
      return 0;
      return *x-*y;
}
```

输出结果

```
str1: Honda Civic
str2: Honda Accord

Using strncmp(str1, str2, 5)....
str1 is equal to str2
Using stringCompare_n(str1, str2, 5)....
str1 is equal to str2
```

以上仅讨论有关字符串的长度、复制、连接与比较等四个主题，同时以库函数运行，并编写用户自定义函数，以模拟这些库函数的用法，期望读者能从这些主题中学到更多有关字符串指针的操作方式，这也是本书的重点之一。

5-3 调试题

请将以下的程序加以修正。

1.

```
/* stringBugs10.c */
#include <stdio.h>
#include <stdlib.h>
int main()
{
     char *str1 = "Learning pointer now!";
     char str2[] = "Go ahead";
     char str3[20];
     char *str4;

     str3 = "I want to buy an iPod\n";
     printf("c or C++: ");
     scanf("%s", str4);

     printf("\n%s\n", str1);
     printf("%s\n", str2);
     printf("%s\n", str3);
     printf("Learning %s now\n", str4);
```

```
    system("PAUSE");
    return 0;
}
```

2.

```
/*  stringCopyBugs.c  */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
    char *s = "iPhone";
    char *t;
    printf("Before strcpy(t, s)\n");
    printf("s = %s\n", s);
    printf("t = %s\n", t);
    printf("After strcpy(t, s)\n");
    strcpy(t, s);
    printf("s = %s\n", s);
    printf("t = %s\n", t);
    system("PAUSE");
    return 0;
}
```

5-4 程序实战

1. strchr(str, 'd')函数是寻找 d 字符第一次在字符串 str 中出现的地址。试编写模拟此函数的程序，并加以测试。
2. stricmp(t, s)函数是比较 t, s 两个字符串，且忽略大小写的差异。试编写模拟此函数的程序，并加以测试。