程序设计与算法(一)

李文新 郭炜

字符串

C++中的字符串

字符串有三种形式。

1. 用双引号括起来的字符串常量，
   如"CHINA" ， "C++ program "。

C++中的字符串

字符串有三种形式。

1. 用双引号括起来的字符串常量，
   如"CHINA" ， "C++ program "。

2. 存放于字符数组中，以‘\0’字符（ASCII码为0)结尾

C++中的字符串

字符串有三种形式。

1. 用双引号括起来的字符串常量，
   如"CHINA" ， "C++ program "。

2. 存放于字符数组中，以‘\0’字符（ASCII码为0)结尾

3. string对象。string是C++标准模板库里的一个类，专门用于处理字符串(略)。

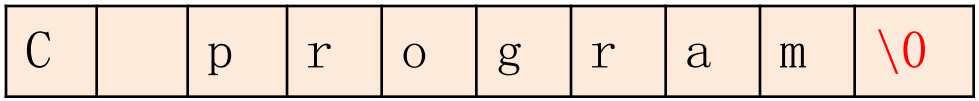
字符串常量

- 字符串常量占据内存的字节数等于字符串中字符数目加1，多出来的是结尾字符‘\0’。

字符串常量

- 字符串常量占据内存的字节数等于字符串中字符数目加1，多出来的是结尾字符‘\0’。

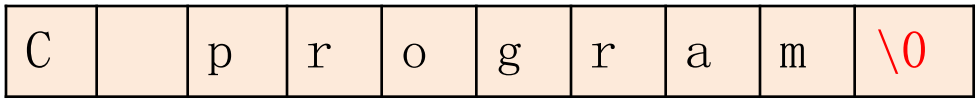
- 字符串 "C program" 在内存中的布局：



字符串常量

- 字符串常量占据内存的字节数等于字符串中字符数目加1，多出来的是结尾字符‘\0’。

- 字符串 "C program" 在内存中的布局：



- 字符串的长度不包含’\0’

字符串常量

- "" 也是合法的字符串常量，称为“空串”，
  空串仍然会占据一个字节的存储空间，存放 ‘\0’。

字符串常量

- "" 也是合法的字符串常量，称为“空串”，
  空串仍然会占据一个字节的存储空间，存放 ‘\0’。

- 如果字符串常量中包含双引号，则双引号应写为‘\"’。而‘\’字符在字符串中出现时，须连写两次，变成‘\\’。例如：

```
cout << "He said: \"I am a stu\\dent.\"";
```

=> He said: "I am a stu\dent."

用一维char数组存放字符串

- 包含‘\0’字符的一维char数组，就是一个字符串。其中存放的字符串即为 ‘\0’前面的字符组成。

用一维char数组存放字符串

- 包含‘\0’字符的一维char数组，就是一个字符串。其中存放的字符串即为 ‘\0’前面的字符组成。

- 用char数组存放字符串，数组元素个数应该至少为字符串长度+1

用一维char数组存放字符串

- 包含‘\0’字符的一维char数组，就是一个字符串。其中存放的字符串即为 ‘\0’前面的字符组成。

- 用char数组存放字符串，数组元素个数应该至少为字符串长度+1

- char数组的内容，可以在初始化时设定，也可以用C++库函数进行修改，还可以用对数组元素赋值的办法任意改变其中的某个字符。

用一维char数组存放字符串

- 包含‘\0’字符的一维char数组，就是一个字符串。其中存放的字符串即为 ‘\0’前面的字符组成。

- 用char数组存放字符串，数组元素个数应该至少为字符串长度+1

- char数组的内容，可以在初始化时设定，也可以用C++库函数进行修改，还可以用对数组元素赋值的办法任意改变其中的某个字符。

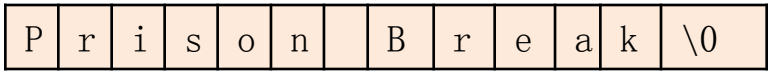
- 字符数组同样可以用cout、printf输出，用cin、scanf读入。用cin、scanf将字符串读入字符数组时，会自动在字符数组中字符串的末尾加上‘\0’

字符串程序示例

```
#include <iostream>
#include <cstring> //包含字符串库函数的声明
using namespace std;
int main()
{
   char title[] = "Prison Break"; //title最后一个元素是'\0'
   char hero[100] = "Michael Scofield";
   char prisonName[100];
   char response[100];
   cout << "What's the name of the prison in " << title << endl;
   cin >> prisonName;  //输入字符串
   if( strcmp( prisonName, "Fox-River") == 0 ) //字符串比较函数
        cout << "Yeah! Do you love " << hero << endl;
   else {
        //字符串拷贝函数
        strcpy( response, "It seems you haven't watched it!\n");
        cout << response;
   }
```

字符串程序示例

```
#include <iostream>
#include <cstring> //包含字符串库函数的声明
using namespace std;
int main()
{
   char title[] = "Prison Break"; //title最后一个元素是'\0'
   char hero[100] = "Michael Scofield";
   char prisonName[100];
   char response[100];
   cout << "What's the name of the prison in " << title << endl;
   cin >> prisonName;  //输入字符串
   if( strcmp( prisonName, "Fox-River") == 0 ) //字符串比较函数
        cout << "Yeah! Do you love " << hero << endl;
   else {
        //字符串拷贝函数
        strcpy( response, "It seems you haven't watched it!\n");
        cout << response;
   }
```



字符串程序示例

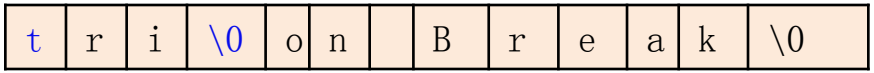
```
#include <iostream>
#include <cstring> //包含字符串库函数的声明
using namespace std;
int main()
{
    char title[] = "Prison Break"; //title最后一个元素是'\0'
    char hero[100] = "Michael Scofield";
    char prisonName[100];
    char response[100];
    cout << "What's the name of the prison in " << title << endl;
    cin >> prisonName;  //输入字符串
    if( strcmp( prisonName, "Fox-River") == 0 ) //字符串比较函数
         cout << "Yeah! Do you love " << hero << endl;
    else {
         //字符串拷贝函数
         strcpy( response, "It seems you haven't watched it!\n");
         cout << response;
    }
```

P	r	i	s	o	n		B	r	e	a	k	\0

What's the name of the prison in Prison Break

字符串程序示例

```
    title [0] = 't';
    title [3] = 0;  //等效于  title [3] = '\0';
    cout << title << endl;
    return 0;
}
```



字符串程序示例

```
    title [0] = 't';
    title [3] = 0;  //等效于  title [3] = '\0';
    cout << title << endl;
    return 0;
}
```

What's the name of the prison in Prison Break
Fox-River↙
Yeah! Do you love Michael Scofield
tri

What's the name of the prison in Prison Break
Shark↙
It seems you haven't watched it!
tri

用scanf读入字符串

- 用scanf可以将字符串读入字符数组
- scanf会自动添加结尾的'\0'
- scanf读入到空格为止

```
char line[100];
scanf("%s",line);   //注意，不是 &line
printf("%s",line);
```

用scanf读入字符串

- 用scanf可以将字符串读入字符数组
- scanf会自动添加结尾的'\0'
- scanf读入到空格为止

```
char line[100];
scanf("%s",line);   //注意，不是 &line
printf("%s",line);
```

Fox River↙
Fox

用scanf读入字符串

- 在数组长度不足的情况下，scanf可能导致数组越界

```
char line[5];
scanf("%s",line);
```

若输入"12345"，则数组越界！

用scanf读入字符串

- 在数组长度不足的情况下，scanf可能导致数组越界

```
char line[5];
scanf("%s",line);
```

若输入"12345"，则数组越界！

- cin输入字符串的情况和scanf相同

```
char line[5];
cin >> line; //若输入"12345"，则数组越界！
```

读入一行到字符数组

- **cin.getline(char buf[], int bufSize);**

读入一行（行长度不超过bufSize-1）或bufSize-1个字符到buf，自动添加'\0'
回车换行符不会写入buf,但是会从输入流中去掉

读入一行到字符数组

- **cin.getline(char buf[], int bufSize);**

读入一行（行长度不超过bufSize-1）或bufSize-1个字符到buf, 自动添加'\0'
回车换行符不会写入buf,但是会从输入流中去掉

```
char line[10];
cin.getline(line, sizeof(line));
//或 cin.getline(line,10); 读入最多9个字符到 line
cout << line;
```

读入一行到字符数组

- **cin.getline(char buf[], int bufSize);**

读入一行（行长度不超过bufSize-1）或bufSize-1个字符到buf，自动添加'\0'
回车换行符不会写入buf,但是会从输入流中去掉

```
char line[10];
cin.getline(line, sizeof(line));
//或 cin.getline(line,10); 读入最多9个字符到 line
cout << line;
```

```
A b c↙
A b c
```

读入一行到字符数组

- **cin.getline(char buf[], int bufSize);**

读入一行（行长度不超过bufSize-1）或bufSize-1个字符到buf，自动添加'\0'
回车换行符不会写入buf,但是会从输入流中去掉

```
char line[10];
cin.getline(line, sizeof(line));
//或 cin.getline(line,10); 读入最多9个字符到 line
cout << line;
```

```
A b c↙
A b c
```

```
A b c1234567↙
A b c1234
```

读入一行到字符数组

- **gets(char buf[]);**

读入一行，自动添加'\0'
回车换行符不会写入buf,但是会从输入流中去掉。可能导致数组越界！

```
char s[10];
while( gets(s) ) {
    printf("%s\n",s);
}
```

读入一行到字符数组

- **gets(char buf[]);**

读入一行，自动添加'\0'
回车换行符不会写入buf，但是会从输入流中去掉。可能导致数组越界！

```
char s[10];
while( gets(s) ) {
    printf("%s\n",s);
}
```

```
A b c↙
A b c
12 34↙
12 34
^Z↙
```

字符串库函数

- 使用字符串函数需要 #include <cstring>

字符串库函数

- 使用字符串函数需要 #include <cstring>
- 字符串函数都根据'\0'来判断字符串结尾

字符串库函数

- 使用字符串函数需要 #include <cstring>
- 字符串函数都根据'\0'来判断字符串结尾
- 形参为char []类型，则实参可以是char数组或字符串常量

字符串库函数

- 使用字符串函数需要 #include <cstring>
- 字符串函数都根据'\0'来判断字符串结尾
- 形参为char []类型，则实参可以是char数组或字符串常量

字符串拷贝

```
strcpy(char dest [],char src [] ); //拷贝src到dest
```

字符串库函数

- 使用字符串函数需要 #include <cstring>
- 字符串函数都根据'\0'来判断字符串结尾
- 形参为char []类型，则实参可以是char数组或字符串常量

字符串拷贝

```
strcpy(char dest [],char src[]); //拷贝src到dest
```

字符串比较大小

```
int strcmp(char s1 [],char s2 []); //返回0则相等
```

字符串库函数

- 使用字符串函数需要 #include <cstring>
- 字符串函数都根据'\0'来判断字符串结尾
- 形参为char []类型，则实参可以是char数组或字符串常量

字符串拷贝

```
strcpy(char dest [],char src[]); //拷贝src到dest
```

字符串比较大小

```
int strcmp(char s1[],char s2[]); //返回0则相等
```

求字符串长度

```
int strlen(char s[]);
```

字符串库函数

- 使用字符串函数需要 #include <cstring>
- 字符串函数都根据'\0'来判断字符串结尾
- 形参为char []类型，则实参可以是char数组或字符串常量

字符串拷贝

```
strcpy(char dest[],char src[]); //拷贝src到dest
```

字符串比较大小

```
int strcmp(char s1[],char s2[]); //返回0则相等
```

求字符串长度

```
int strlen(char s[]);
```

字符串拼接

```
strcat(char s1[],char s2[]); //s2拼接到s1后面
```

字符串库函数

- 使用字符串函数需要 #include <cstring>
- 字符串函数都根据'\0'来判断字符串结尾
- 形参为char []类型，则实参可以是char数组或字符串常量

字符串拷贝

```
strcpy(char dest [],char src[]); //拷贝src到dest
```

字符串比较大小

```
int strcmp(char s1[],char s2[]); //返回0则相等
```

求字符串长度

```
int strlen(char s[]); //不算结尾的'\0'
```

字符串拼接

```
strcat(char  s1[],char s2[]); //s2拼接到s1后面
```

字符串转成大写

```
strupr(char s[]);
```

字符串转成小写

```
strlwr(char s[]);
```

字符串库函数用法示例

```
#include <iostream>
#include <cstring> //要使用字符串库函数需要包含此头文件
using namespace std;
void PrintSmall( char s1[],char s2[]) //输出词典序小的字符串
{
        if( strcmp( s1,s2) <= 0) //如果s1小于等于s2
                cout << s1 ;
        else
                cout << s2;
}
int main() {
    char s1[30];        char s2[40];    char s3[100];
    strcpy( s1,"Hello"); // 拷贝 "Hello" 到s1 ,  s1 = "Hello"
    strcpy( s2,s1);       // 拷贝s1到s2, s2 = "Hello"
    cout << "1) " << s2 << endl;  //输出 1) Hello
    strcat( s1,",world"); // 连接 ",world"到s1尾部。s1 = "Hello,world"
    cout << "2) " << s1 << endl;  //输出 2) Hello,world
```

字符串库函数用法示例

```
    cout << "3) "; PrintSmall("abc",s2); cout << endl; //输出 3) Hello
    cout << "4) "; PrintSmall("abc","aaa"); cout << endl;  //输出 4) aaa
    int n = strlen( s2 ); //求s2长度
    cout <<"5) " << n << "," << strlen("abc") << endl; //输出 5) 5,3
    strupr(s1);  // 把s1变成大写，s1 = "HELLO,WORLD"
    cout <<"6) " << s1 << endl; //输出 6) HELLO,WORLD
    return 0;
}
```

strlen常见糟糕用法

```
char s[100]="test";
for( int i = 0; i < strlen(s); ++ i ) {
      s[i] = s[i]+1; //此句不重要，只是为了说明要访问s[i]
}
```

strlen常见糟糕用法

```
char s[100]="test";
for( int i = 0; i < strlen(s); ++ i ) {
       s[i] = s[i]+1;
}
```

- strlen函数的执行是需要时间的，且时间和字符串的长度成正比

- 每次循环，都调用strlen函数，这是效率上的很大浪费

strlen常见糟糕用法

- 应取出s的长度存放在一个变量里面，然后在循环的时候使用该变量

```
char s[100] = "test";
int len = strlen(s);
for( int i = 0; i < len; ++ i ) {
        s[i] = s[i]+1;
}
```

strlen常见糟糕用法

- 应取出s的长度存放在一个变量里面，然后在循环的时候使用该变量

```
char s[100] = "test";
int len = strlen(s);
for( int i = 0; i < len; ++ i ) {
        s[i] = s[i]+1;
}
```

- 或：

```
char s[100] = "test";
for( int i = 0; s[i]; ++ i ) {
        s[i] = s[i]+1;
}
```

例题：编写判断子串的函数

- 编写一个函数：

```
int Strstr(char s1[],char s2[]);
```

如果s2不是s1的字串，返回 -1
如果s2是s1的子串，返回其在s1中第一次出现的位置
空串是任何串的子串，且出现位置为0

例题：编写判断子串的函数

```
int Strstr(char s1[],char s2[]) {
        if( s2[0] == 0)
                return 0;
        for(int i = 0; s1[i]; ++i) { //枚举比较起点
                int k = i,j = 0;
                for( ; s2[j]; ++j,++k) {
                        if(s1[k] != s2[j])
                                break;
                }
                if( s2[j] == 0)
                        return i;
        }
        return -1;
}
```