

字符串函数的使用

C语言中对字符和字符串的处理很是频繁，但是C语言本身是没有字符串类型的，字符串通常放在常量字符串中 或者 字符数组中。

字符串常量适用于那些对它不做修改的字符串函数

江山如此多娇

@[TOC](#)

一、strcat

strcat 是字符串追加函数，将src里的字符串追加到Destination目标字符串中。

```
char * strcat ( char * destination, const char * source );
```

参数


- strDestination*
以NULL中止的目的字符串。
- strSource*
null 终止的源字符串。

返回值

这些函数都返回一个目标字符串 (*strDestination*)。 没有保留任何返回值以指示错误。

备注

strcat 功能追加 *strSource* 到 *strDestination* 并停止使用 null 字符的结果字符串。 *strSource* 的初始字符覆盖 *strDestination*终止 null 字符。 如果源和目标字符串重叠，则 **strcat** 的行为未定义。

 **安全说明**

由于 **strcat** 在追加 *strSource* 之前不会检查是否在 *strDestination* 有足够空间，这是一个可能导致缓冲区溢出的原因。 考虑改用 [strncat](#) 代替。

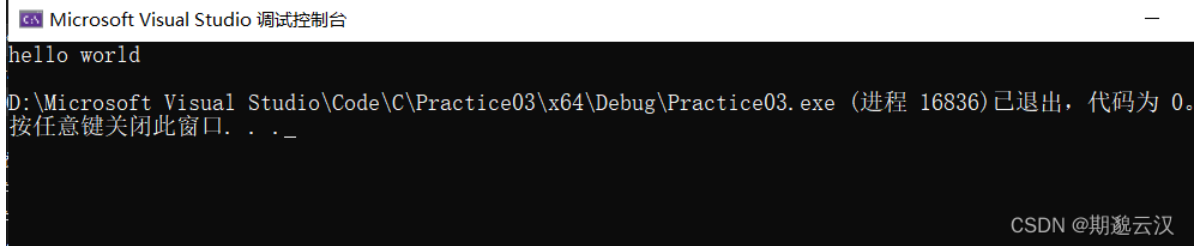
wcscat 和 **_mbscat** 是宽字符，属于 **strcat** 的多字节字符版本。 参数和 **wcscat** 的返回值是宽字符串；**_mbscat** 的参数和返回值为多字节字符串。 否则这三个函数否则具有相同行为。 在 C++ 中，这些函数有调用较新的、较安全的替代版本的模板重载。 有关详细信息，请参阅 [安全模板重载](#)。

一般文本例程映射

TCHAR.H 例程	未定义 _UNICODE & _MBCS	已定义 _MBCS	已定义 _UNICODE
_tcscat	strcat	_mbscat	wcscat

CSDN @期遛云汉

```
//strcat字符串追加函数
#include <string.h>
int main() {
    char arr1[20] = "hello ";
    char arr2[] = "world";
    strcat(arr1, arr2);
    printf("%s\n", arr1);
    return 0;
}
```



源字符串必须以 '\0' 结束。

模拟实现strcat

代码如下（示例）：

```
#include <string.h>
//实现strcat函数
char* myStrcat(char* dest, char* src) {
    //先找到目标字符串里的'\0'
    char* cur = dest;
    while (*cur != '\0') {
        cur++;
    }
    //进行拷贝
    while (*cur++ = *src++) {
        ;
    }
    return dest;
}

int main() {
    char arr1[20] = "hello ";
    char arr2[] = "world";

    char dest[20] = "江山如此";
    char src[] = "多娇";
    strcat(arr1, arr2);
    printf("%s\n", arr1);

    printf("%s\n", myStrcat(dest, src));
    return 0;
}
```

```
//实现strcat函数
char* myStrcat(char* dest, char* src) {
    //先找到目标字符串里的'\0'
    char* cur = dest;
    while (*cur != '\0') {
        cur++;
    }
    //进行拷贝
    while (*cur++ = *src++) {
        ;
    }
    return dest;
}

int main() {
    char arr1[20] = "hello ";
    char arr2[] = "world";

    char dest[20] = "江山如此";
    char src[] = "多娇";
    strcat(_Destination: arr1, _Source: arr2);
    printf(_Format: "%s\n", arr1);

    printf(_Format: "%s\n", myStrcat(dest, src));
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
hello world
江山如此多娇

D:\Microsoft Visual Studio\Code\C\Practice03\x64\Debug\Practice03.exe (进程 10220) 已退出，代码为 0。
按任意键关闭此窗口。 . . _
```

CSDN @期邈云汉

考虑字符串给自己追加

代码如下（示例）：

```
//考虑自己追加
//例如：h e l l o \0 那么追加时 自己修改自己，那么src将不会找到\0停下，肯定会报错
```

二、strcmp

字符串比较函数

```
int strcmp ( const char * str1, const char * str2 );
```

比较的时比较对应位置上的字符，与字符串长度无关

比较规则:

第一个字符串大于第二个字符串，则返回大于0的数字

第一个字符串等于第二个字符串，则返回0

第一个字符串小于第二个字符串，则返回小于0的数字

且同样的思路，字符串里必须都有 \0，否则函数将不会停止比较，引起错误。

模拟实现strcmp函数

代码如下（示例）：

```
//实现strcmp函数
int myStrcmp(char* s1, char* s2) {

    while (*s1 == *s2) {
        if (*s1 == '\0')
            return 0; //知道检索到字符串末尾的\0,说明像相等了
        s1++;
        s2++;
    }
    return *s1 - *s2;
}

int main() {
    char s1[] = "acbhg";
    char s2[] = "abjj";
    printf("%d\n", myStrcmp(s1, s2));
}
```

```
//实现strcmp函数
int myStrcmp(char* s1, char* s2) {
    while (*s1 == *s2) {
        if (*s1 == '\0')
            return 0; //知道检索到字符串末尾的\0,说明像相等了
        s1++;
        s2++;
    }
    return *s1 - *s2;
}

int main() {
    char s1[] = "acbhg";
    char s2[] = "abjj";
    printf(_Format: "%d\n", myStrcmp(s1, s2));
}
```

CSDN @期邈云汉

Microsoft Visual Studio 调试控制台

```
1
D:\Microsoft Visual Studio\Code\C\Practice03\x64\Debug\Practice03.exe (进程 17608)已退出, 代码为 0。
按任意键关闭此窗口. . .
```

CSDN @期邈云汉

长度受限的字符串函数

长度不受限制的字符串函数：

strcpy
strcat
strcmp

\0

长度受限的字符串函数

strncpy
strncat
strncmp

CSDN @期邈云汉

例如：strncpy

拷贝num个字符从源字符串到目标空间。

```
char * strncpy ( char * destination, const char * source, size_t num );
```

如果源字符串的长度小于num，则拷贝完源字符串之后，在目标的后边追加\0，直到num个

```
int main() {
    char arr1[] = "dadxxp";
    char arr2[] = "xs";
    strncpy(_Destination: arr1, _Source: arr2, _Count: 3);
    printf(_Format: "%s\n", arr1);
}
```

CSDN @期邈云汉

Microsoft Visual Studio 调试控制台

XS

D:\Microsoft Visual Studio\Code\C\Practice03\x64\Debug\Practice03.exe (进程 18352) 已退出，代码为 0。
按任意键关闭此窗口。 . . . _

CSDN @期邈云汉

strncat

追加num个字符

```
char * strncat ( char * destination, const char * source, size_t num );
```

```
#include <stdio.h>
#include <string.h>
int main ()
{
    char str1[20];
    char str2[20];
    strcpy (str1, "To be ");
    strcpy (str2, "or not to be");
    strncat (str1, str2, 6);
    puts (str1);
    return 0;
}
```

实际上还是会再放入 \0

Microsoft Visual Studio 调试控制台

To be or not

D:\Microsoft Visual Studio\Code\C\Practice03\x64\Debug\Practice03.exe (进程 18352) 已退出，代码为 0。
按任意键关闭此窗口。 . . .

CSDN @期邈云汉

strncmp

比较限定个长度的字符

Return Value

Returns an integral value indicating the relationship between the strings:

return value	indicates
<0	the first character that does not match has a lower value in <i>str1</i> than in <i>str2</i>
0	the contents of both strings are equal
>0	the first character that does not match has a greater value in <i>str1</i> than in <i>str2</i>

```
int strncmp ( const char * str1, const char * str2, size_t num )
```

```
int main ()
{
    char str[][5] = { "R2D2" , "C3PO" , "R2A6" };
    int n;
    puts ("Looking for R2 astromech droids...");
    for (n=0 ; n<3 ; n++)
        if (strncmp (str[n],"R2xx",2) == 0)
        {
            printf ("found %s\n",str[n]);
        }
    return 0;
}
```

```
int main()
{
    char str[][5] = { "R2D2" , "C3PO" , "R2A6" };
    int n;
    puts(Buffer:"Looking for R2 astromech droids...");
    for (n = 0; n < 3; n++)
        if (strncmp(_Str1:str[n], _Str2:"R2xx", _MaxCount:2) == 0)
        {
            printf(_Format:"found %s\n", str[n]);
        }
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
Looking for R2 astromech droids...
found R2D2
found R2A6
```

CSDN @期邈云汉

strstr判断子串是否存在

```
char * strstr ( const char *str1, const char * str2);
```

判断一个字符串是否存在与另一个字符串，如果存在，返回子串第一次出现的位置，不存在返回NULL

实现strstr函数

```
char* myStrstr(const char* str1, const char* str2) {
    //考虑两个指针来控制检索两个字符串
```

```

const char* s1 = str1;
const char* s2 = str2;
const char* p = str1;
while (*p) {
    s1 = p;
    s2 = str2;
    while ((*s1 == *s2) && *s1 != '\0' && *s2 != '\0') {
        s1++;
        s2++;
    }
    p++; //记录开始匹配的位置
    if (*s2 == '\0') {
        return (char*)p; //找到子串
    }
}
return NULL; //找不到子串
}

int main() {
    char arr1[] = "dadwadwa";
    char arr2[] = "adw";
    printf("%s\n", myStrstr(arr1, arr2));
    return 0;
}

```

```

char* myStrstr(const char* str1, const char* str2) {
    //考虑两个指针来控制检索两个字符串
    const char* s1 = str1;
    const char* s2 = str2;
    const char* p = str1;
    while (*p) {
        s1 = p;
        s2 = str2;
        while ((*s1 == *s2) && *s1 != '\0' && *s2 != '\0') {
            s1++;
            s2++;
        }
        p++; //记录开始匹配的位置
        if (*s2 == '\0') {
            return (char*)p; //找到子串
        }
    }
    return NULL; //找不到子串
}

int main() {
    char arr1[] = "dadwadwa";
    char arr2[] = "adw";
    printf(_Format: "%s\n", myStrstr(str1: arr1, str2: arr2));
    return 0;
}

```

Microsoft Visual Studio 调试控制台

dwadwa

D:\Microsoft Visual Studio\Code\C\Practice03\x64\Debug\Practice03.exe (进程 18064) 已退出, 代码为 0。
按任意键关闭此窗口。 . . _ CSDN @期邈云汉

其实这样的算法比较满，暴力算法，KMP算法可以很好解决这样的匹配问题。

