

第16讲：字符函数和字符串函数

目录：

1. 字符分类函数
2. 字符转换函数
3. strlen的使用和模拟实现
4. strcpy的使用和模拟实现
5. strcat的使用和模拟实现
6. strcmp的使用和模拟实现
7. strncpy函数的使用
8. strncat函数的使用
9. strncmp函数的使用
10. strstr的使用和模拟实现
11. strtok函数的使用
12. strerror函数的使用

正文开始

1. 字符分类函数

C语言中有一系列的函数是专门做字符分类的，也就是一个字符是属于什么类型的字符的。

这些函数的使用都需要包含一个头文件是 `ctype.h`

	A	B
1	函数	如果他的参数符合下列条件就返回真
2	isctrl	任何控制字符
3	isspace	空白字符：空格 ‘ ’ ，换页 ‘\f’ ，换行‘\n’，回车 ‘\r’ ，制表符‘\t’或者垂直制表符‘\v’
4	isdigit	十进制数字 0~9
5	isxdigit	十六进制数字，包括所有十进制数字，小写字母a~f，大写字母A~F
6	islower	小写字母a~z
7	isupper	大写字母A~Z
8	isalpha	字母a~z或A~Z
9	isalnum	字母或者数字，a~z,A~Z,0~9
10	ispunct	标点符号，任何不属于数字或者字母的图形字符（可打印）
11	isgraph	任何图形字符
12	isprint	任何可打印字符，包括图形字符和空白字符

这些函数的使用方法非常类似，我们就讲解一个函数的事情，其他的非常类似：

```
1 int islower ( int c );
```

`islower` 是能够判断参数部分的c是否是小写字母的。

通过返回值来说明是否是小写字母，如果是小写字母就返回非0的整数，如果不是小写字母，则返回0。

使用代码演示：写一个代码，讲字符串中的小写字母转大写，其他字符不变。

```
1 #include <stdio.h>
2 #include <ctype.h>
3 int main ()
4 {
5     int i = 0;
6     char str[] = "Test String.\n";
7     char c;
8     while (str[i])
9     {
10         c = str[i];
11         if (islower(c))
12             c -= 32;
13         putchar(c);
```

```
14     i++;
15 }
16 return 0;
17 }
```

2. 字符转换函数

C语言提供了2个字符转换函数：

```
1 int tolower ( int c ); //将参数传进去的小写字母转大写
2 int toupper ( int c ); //将参数传进去的大写字母转小写
```

上面的代码，我们将小写转大写，是-32完成的效果，有了转换函数，就可以直接使用 `tolower` 函数。

```
1 #include <stdio.h>
2 #include <ctype.h>
3 int main ()
4 {
5     int i = 0;
6     char str[] = "Test String.\n";
7     char c;
8     while (str[i])
9     {
10         c = str[i];
11         if (islower(c))
12             c = toupper(c);
13         putchar(c);
14         i++;
15     }
16     return 0;
17 }
```

3. strlen的使用和模拟实现

```
1 size_t strlen ( const char * str );
```

- 字符串以 `'\0'` 作为结束标志，strlen函数返回的是在字符串中 `'\0'` 前面出现的字符个数（不包含 `'\0'`）。

- 参数指向的字符串必须要以 `\0` 结束。
- 注意函数的返回值为 `size_t`, 是无符号的 (易错)
- 学会 `strlen` 函数的模拟实现

```

1 #include <stdio.h>
2 int main()
3 {
4     const char*str1 = "abcdef";
5     const char*str2 = "bbb";
6     if(strlen(str2)-strlen(str1)>0)
7     {
8         printf("str2>str1\n");
9     }
10    else
11    {
12        printf("str1>str2\n");
13    }
14    return 0;
15 }

```

`strlen` 的模拟实现:

方式1:

```

1 //计数器方式
2 int my_strlen(const char * str)
3 {
4     int count = 0;
5     assert(str);
6     while(*str)
7     {
8         count++;
9         str++;
10    }
11    return count;
12 }

```

方式2:

```

1 //不能创建临时变量计数器
2 int my_strlen(const char * str)

```

```
3 {
4     assert(str);
5     if(*str == '\0')
6         return 0;
7     else
8         return 1+my_strlen(str+1);
9 }
```

方式3:

```
1 //指针-指针的方式
2 int my_strlen(char *s)
3 {
4     assert(str);
5     char *p = s;
6     while(*p != '\0' )
7         p++;
8     return p-s;
9 }
```

4. strcpy的使用和模拟实现

```
1 char* strcpy(char * destination, const char * source );
```

- Copies the C string pointed by source into the array pointed by destination, including the terminating null character (and stopping at that point).
- 源字符串必须以 `'\0'` 结束。
- 会将源字符串中的 `'\0'` 拷贝到目标空间。
- 目标空间必须足够大, 以确保能存放源字符串。
- 目标空间必须可变。
- 学会模拟实现。

strcpy的模拟实现:

```
1 //1. 参数顺序
```

```
2 //2.函数的功能, 停止条件
3 //3.assert
4 //4.const修饰指针
5 //5.函数返回值
6 //6.题目出自《高质量C/C++编程》书籍最后的试题部分
7 char *my_strcpy(char *dest, const char*src)
8 {
9     char *ret = dest;
10    assert(dest != NULL);
11    assert(src != NULL);
12
13    while((*dest++ = *src++))
14    {
15        ;
16    }
17    return ret;
18 }
```

5. strcat的使用和模拟实现

- Appends a copy of the source string to the destination string. The terminating null character in destination is overwritten by the first character of source, and a null-character is included at the end of the new string formed by the concatenation of both in destination.
- 源字符串必须以 `'\0'` 结束。
- 目标字符串中也得有 `\0` , 否则没办法知道追加从哪里开始。
- 目标空间必须有足够的大, 能容纳下源字符串的内容。
- 目标空间必须可修改。
- 字符串自己给自己追加, 如何?

模拟实现strcat函数:

```
1 char *my_strcat(char *dest, const char*src)
2 {
3     char *ret = dest;
4     assert(dest != NULL);
5     assert(src != NULL);
6     while(*dest)
7     {
8         dest++;
```

```
9     }
10    while((*dest++ = *src++))
11    {
12        ;
13    }
14    return ret;
15 }
```

6. strcmp 的使用和模拟实现

- This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached.
- 标准规定:
 - 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
 - 那么如何判断两个字符串? 比较两个字符串中对应位置上字符ASCII码值的大小。

strcmp函数的模拟实现:

```
1 int my_strcmp (const char * str1, const char * str2)
2 {
3     int ret = 0 ;
4     assert(src != NULL);
5     assert(dest != NULL);
6     while(*str1 == *str2)
7     {
8         if(*str1 == '\0')
9             return 0;
10        str1++;
11        str2++;
12    }
13    return *str1-*str2;
14 }
```

7. strncpy 函数的使用

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

- Copies the first num characters of source to destination. If the end of the source C string (which is signaled by a null-character) is found before num characters have been copied, destination is padded with zeros until a total of num characters have been written to it.
- 拷贝num个字符从源字符串到目标空间。
- 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。

8. strncat 函数的使用

- Appends the first num characters of source to destination, plus a terminating null-character.
- If the length of the C string in source is less than num, only the content up to the terminating null-character is copied.

```
1 /* strncat example */
2 #include <stdio.h>
3 #include <string.h>
4
5 int main ()
6 {
7     char str1[20];
8     char str2[20];
9     strcpy (str1, "To be ");
10    strcpy (str2, "or not to be");
11    strncat (str1, str2, 6);
12    printf("%s\n", str1);
13    return 0;
14 }
```

9. strncmp 函数的使用

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

比较str1和str2的前num个字符, 如果相等就继续往后比较, 最多比较num个字母, 如果提前发现不一样, 就提前结束, 大的字符所在的字符串大于另外一个。如果num个字符都相等, 就是相等返回0。

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>

10. strstr的使用和模拟实现

```
1 char * strstr ( const char *, const char * );
```

Returns a pointer to the first occurrence of *str2* in *str1*, or a null pointer if *str2* is not part of *str1*.

The matching process does not include the terminating null-characters, but it stops there.

```
1 /* strstr example */
2 #include <stdio.h>
3 #include <string.h>
4
5 int main ()
6 {
7     char str[] = "This is a simple string";
8     char * pch;
9     pch = strstr (str, "simple");
10    strncpy (pch, "sample", 6);
11    printf ("%s\n", str);
12    return 0;
13 }
```

strstr的模拟实现：

```
1 char * strstr (const char * str1, const char * str2)
2 {
3     char *cp = (char *) str1;
4     char *s1, *s2;
5
6     if ( !*str2 )
7         return((char *)str1);
8
9     while (*cp)
```

```
10     {
11         s1 = cp;
12         s2 = (char *) str2;
13
14         while ( *s1 && *s2 && !(*s1-*s2) )
15             s1++, s2++;
16
17         if (!*s2)
18             return(cp);
19
20         cp++;
21     }
22
23     return(NULL);
24 }
```

11. strtok函数的使用

```
1 char * strtok ( char * str, const char * sep);
```

- sep参数指向一个字符串, 定义了用作分隔符的字符集合
- 第一个参数指定一个字符串, 它包含了0个或者多个由sep字符串中一个或者多个分隔符分割的标记。
- strtok函数找到str中的下一个标记, 并将其用 `\0` 结尾, 返回一个指向这个标记的指针。(注: strtok函数会改变被操作的字符串, 所以在使用strtok函数切分的字符串一般都是临时拷贝的内容并且可修改。)
- strtok函数的第一个参数不为 `NULL`, 函数将找到str中第一个标记, strtok函数将保存它在字符串中的位置。
- strtok函数的第一个参数为 `NULL`, 函数将在同一个字符串中被保存的位置开始, 查找下一个标记。
- 如果字符串中不存在更多的标记, 则返回 `NULL` 指针。

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int main()
5 {
6     char arr[] = "192.168.6.111";
7     char* sep = ".";
```

```

8     char* str = NULL;比特就业课主页：https://m.cctalk.com/inst/s9yewhfr
9     for (str = strtok(arr, sep); str != NULL; str = strtok(NULL, sep))
10    {
11        printf("%s\n", str);
12    }
13    return 0;
14 }

```

12. strerror函数的使用

```
1 char * strerror ( int errnum );
```

strerror函数可以把参数部分错误码对应的错误信息的字符串地址返回来。

在不同的系统和C语言标准库的实现中都规定了一些错误码，一般是放在 `errno.h` 这个头文件中说明的，C语言程序启动的时候就会使用一个全面的变量errno来记录程序的当前错误码，只不过程序启动的时候errno是0，表示没有错误，当我们在使用标准库中的函数的时候发生了某种错误，就会讲对应的错误码，存放在errno中，而一个错误码的数字是整数很难理解是什么意思，所以每一个错误码都是有对应的错误信息的。strerror函数就可以将错误对应的错误信息字符串的地址返回。

```

1 #include <errno.h>
2 #include <string.h>
3 #include <stdio.h>
4
5 //我们打印一下0~10这些错误码对应的信息
6 int main()
7 {
8     int i = 0;
9     for (i = 0; i <= 10; i++) {
10        printf("%s\n", strerror(i));
11    }
12    return 0;
13 }

```

在Windows11+VS2022环境下输出的结果如下：

```

1 No error
2 Operation not permitted

```

```
3 No such file or directory
4 No such process
5 Interrupted function call
6 Input/output error
7 No such device or address
8 Arg list too long
9 Exec format error
10 Bad file descriptor
11 No child processes
```

举例:

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <errno.h>
4 int main ()
5 {
6     FILE * pFile;
7     pFile = fopen ("unexist.ent","r");
8     if (pFile == NULL)
9         printf ("Error opening file unexist.ent: %s\n", strerror(errno));
10    return 0;
11 }
```

输出:

```
1 Error opening file unexist.ent: No such file or directory
```

也可以了解一下perror函数, perror函数相当于一次将上述代码中的第9行完成了, 直接将错误信息打印出来。

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <errno.h>
4 int main ()
5 {
6     FILE * pFile;
7     pFile = fopen ("unexist.ent","r");
8     if (pFile == NULL)
9         perror("Error opening file unexist.ent");
10    return 0;
```

11 }

比特就业课主页：<https://m.cctalk.com/inst/s9yewhfr>

输出：

```
1 Error opening file unexist.ent: No such file or directory
```

完

比特就业课