字符函数与内存函数（使用与模拟实现）

1.字符函数与字符串函数

1.1 strlen()函数

**size\_t** **strlen(** **const** **char** **\****string* **);//接收字符串某地址，返回字符串长度，地址开始到’\0’或0结束。由此字符串必须要有’\0’。**

利用函数的思想，模拟实现strlen()的功能。

#include<stdio.h>

#include<assert.h>

int My\_strlen(const char\* str)//内容不能改变const

{

assert(str != NULL);//断言str不为空指针

int count = 0;

while (\*str != '\0')

{

count++;

str++;

}

return count;

}

int main()

{

int len = 0;

//char str[] = "abcd";

char str[] = { 'a','b','c',0 };//0存到char中被看作ASCII码，而字符'\0'的ASCII码就是0

len = My\_strlen(str);

printf("%d\n", len);

return 0;

}

三种方法：计数器法，递归法，指针-指针法

**注：库函数strlen()返回的是size\_t无符号整型，当两个strlen()进行计算时，结果为无符号整型。**

1.2 strcpy()函数

**char** **\*strcpy(** **char** **\****strDestination***,** **const** **char** **\****strSource* **);//传参数，目的字符串地址，源字符串地址；将源字符串复制到目的字符串中，结束符最后一次拷贝，所以必须要有结束符；返回目的字符串地址。目标空间足够大，且内容可以被改变。**

#include<stdio.h>

#include<assert.h>

#include<string.h>

char\* My\_strcpy(char\* str1, const char\* str2)

{

assert(str1 != NULL && str2 != NULL);

char\* start = str1;

while (\*str2 != '\0')

{

\*start = \*str2;

start++;

str2++;

}

\*start = \*str2;

return str1;

}

int main()

{

char str1[20] = { 0 };

char str2[] = "love you";

char\* p;

p = My\_strcpy(str1, str2);

//strcpy(str1, str2);

printf("%s\n", p);

return 0;

}

1.3 strcat()函数

**char** **\*strcat(** **char** **\****strDestination***,** **const** **char** **\****strSource* **);//在目的字符串后面追加源字符串，从目的字符串的\0位置开始追加，到源字符串的\0位置追加结束，所以两者都要有\0。且目标空间足够大。**

**注：此函数不能给自己追加自己，因为自己都是同一块空间，第一次覆盖会使\0消失从而不能停止。**

#include<stdio.h>

#include<assert.h>

#include<string.h>

char\* My\_strcat(char\* dest, const char\* src)

{

assert(dest != NULL && src != NULL);

char\* start = dest;

while (\*start != '\0')

{

start++;

}

while (\*src != '\0')

{

\*start = \*src;

start++;

src++;

}

\*start = \*src;

return dest;

}

int main()

{

char str1[20] = "hello ";

char str2[] = "world!";

//My\_strcat(str1, str2);

printf("%s\n", My\_strcat(str1, str2));

return 0;

}

1.4 strcmp()函数

**int** **strcmp(** **const** **char** **\****string1*, **const** **char** **\****string2* **);//两个字符串比较，从第一个字符开始依次比较，如果string1>string2返回正数，string1<string2返回负数，string1=string2返回0.**

#include<stdio.h>

#include<assert.h>

#include<string.h>

int My\_strcmp(const char\* str1, const char\* str2)

{

assert(str1 != NULL && str2 != NULL);

while (\*str1 != '\0' && str2 != '\0')

{

if (\*str1 > \*str2)

{

return 1;

}

else if (\*str1 < \*str2)

{

return -1;

}

str1++;

str2++;

}

if (\*str1 > \*str2)

{

return 1;

}

else if (\*str1 < \*str2)

{

return -1;

}

else

return 0;

}

int main()

{

char\* p = "word";

char\* q = "world";

if (My\_strcmp(p, q) > 0)

{

printf("字符串p大于字符串q\n");

}

else if (My\_strcmp(p, q) < 0)

{

printf("字符串p小于字符串q\n");

}

else

{

printf("字符串p等于字符串q\n");

}

return 0;

}

注：strcpy、strcat、strcmp属于长度不受限制的字符串函数.

strncpy、strncat、strncmp属于长度受限制的字符串函数.

1.5 strncpy()函数

**char** **\*strncpy(** **char** **\****strDest***,** **const** **char** **\****strSource***,** **size\_t** *count* **);//从Source中拷贝count个字符（可能会覆盖目的）到目的串中，比strcpy()多了一个长度限制，如果长度比源字符串长，则后面拷贝\0。**

1.6 strncat()函数

**char** **\*strncat(** **char** **\****strDest***,** **const** **char** **\****strSource***,** **size\_t** *count* **);//从源字符串中的count个字符追加到目的字符串后面，如果长度长，追加源字符串结束和一个\0就停止了。**

1.7 strncmp()函数

**int** **strncmp(** **const** **char** **\****string1***,** **const** **char** **\****string2***,** **size\_t** *count* **);// 从源字符串中的count个字符与目的字符串进行比较。**

1.8 strstr()函数

**char** **\*strstr(** **const** **char** **\****string***,** **const** **char** **\****strCharSet* **);//在str1中查找是否包含str2，如果包含，则返回第一次包含的地址，否则返回空指针。**

#include<stdio.h>

#include<string.h>

int main()

{

char str1[] = "abcdef";

char str2[] = "cde";

char\* ret = strstr(str1, str2);

if (ret == NULL)

{

printf("没找到\n");

}

else

{

printf("找到了:%s\n", ret);

}

return 0;

}

模拟实现：将str1从头遍历是否包含str2，返回包含的地址，否则返回空指针。

#include<stdio.h>

#include<assert.h>

char\* My\_strstr(const char\* s1, const char\* s2)

{

assert(s1 && s2);

const char\* start1;

const char\* start2;

const char\* cp = s1;

if (\*s2 == '\0')

{

return NULL;

}

while (\*cp)//1.从s1的第一个字符依次查看是否包含s2直到\0结束

{

start1 = cp;

start2 = s2;

//2.看是否匹配成功

while (\*start1 && \*start2 && (\*start1 == \*start2))

{//当为\0时

start1++;

start2++;

}

//3.匹配成功返回匹配地址

if (\*start2 == '\0')

{

return (char\*)cp;

}

//4.没匹配成功进入下次循环

cp++;

}

return NULL;

}

int main()

{

char str1[] = "abbbbbcdef";

char str2[] = "bbcd";

char\* ret = My\_strstr(str1, str2);

if (ret == NULL)

{

printf("没找到\n");

}

else

{

printf("找到了:%s\n", ret);

}

return 0;

}

1.9 strtok()函数

**char** **\*strtok(** **char** **\****strToken***,** **const** **char** **\****strDelimit* **);//字符串分割函数**

1. *strToken* 是原字符串的拷贝（防止被改变），待分割。如123@qq.com

2. *strDelimit* 是分割符的集合如@和.

3.运行过程，从字符串开始到每一个分割符处，将分隔符赋值为\0，并记录\0的位置（猜测用static关键字），方便寻找下一个分割位置，并返回相应段的地址。char\* ret = strtok(str, “@.”); char\* ret = strtok(NULL, “@.”); char\* ret = strtok(NULL, “@.”);

#include<stdio.h>

#include<string.h>

int main()

{

char str[] = "123@qq.com";

char\* p = "@.";

char arr[20] = { 0 };

char\* ret = NULL;

strcpy(arr, str);

for (ret = strtok(arr, p); ret != NULL; ret = strtok(NULL, p))

{

printf("%s\n", ret);

}

/\*ret = strtok(arr, p);

printf("%s\n", ret);

ret = strtok(NULL, p);

printf("%s\n", ret);

ret = strtok(NULL, p);

printf("%s\n", ret);\*/

return 0;

}

1.10 strerror()函数

**char** **\*strerror(** **int** *errnum* **);//查看错误信息函数，传入错误码参数，返回错误信息的地址。**

int errno;//错误码，是C语言内置的变量，当程序出现错误时就会被赋值错误码，相应的错误码对应不同的错误信息。

#include<stdio.h>

#include<string.h>

#include<errno.h>

int main()

{

//printf("%s\n", strerror(0));

//printf("%s\n", strerror(1));

//printf("%s\n", strerror(2));

//printf("%s\n", strerror(3));

//printf("%s\n", strerror(4));

FILE\* fp = fopen("test.txt", "r");//读取文件test.txt

if (fp == NULL)//如果没有读取成功

{

printf("%s\n", strerror(errno));//打印错误信息

return 1;

}

fclose(fp);//关闭文件

fp = NULL;//置空指针

return 0;

}

**void perror(const char \* str)函数**，打印错误信息函数。1.将错误码转换成错误信息；2.打印错误信息，包含自定义信息和本身的错误信息。



2 字符操作函数

2.1 字符分类函数



如果为真，返回非零；否则返回0。**int** **isdigit(** **int** *c* **);//ASCII码**

#include<stdio.h>

#include<ctype.h>

int main()

{

char ch = 'a';

int ret = isdigit(ch);

printf("%d\n", ret);

return 0;

}

2.2 字符转换函数

**int** **tolower(** **int** *c* **);//输入字符，转换成int型，即ASCII，返回ASCII**

**int** **toupper(** **int** *c* **);**

#include<stdio.h>

#include<ctype.h>

int main()

{

char str[20] = { 0 };

gets(str);

int i = 0;

while (str[i])

{

if (isupper(str[i]))

{

str[i] = tolower(str[i]);

}

printf("%c", str[i]);

i++;

}

return 0;

}

3 内存函数

3.1 memcpy()函数

**void** **\*memcpy(** **void** **\****dest***,** **const** **void** **\****src***,** **size\_t** *count* **);//内存拷贝函数，将源内存中的count个字节的内容拷贝到目的内存中，并返回目的内存的地址，可以拷贝各种类型数据。**

#include<stdio.h>

#include<assert.h>

void\* My\_memcpy(void\* dest, void\* src, int num)

{

assert(dest && src);

void\* ret = dest;

while (num--)

{

\*(char\*)dest = \*(char\*)src;//每一字节进行拷贝

dest = (char\*)dest + 1;

src = (char\*)src + 1;

}

return ret;

}

int main()

{

int arr1[10] = { 1,2,3,4,5,6,7,8,9,10 };

int arr2[10] = { 0 };

My\_memcpy(arr2, arr1, 20);

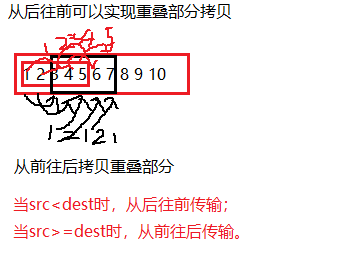
return 0;

}

注：该内存拷贝函数不能对具有重叠部分的两个内存直接进行拷贝，会覆盖。memcpy-只要实现不重叠拷贝就可以了，而VS实现了重叠拷贝。

3.2 memmove()函数

由上述函数提出的问题，**void** **\*memmove(** **void** **\****dest***,** **const** **void** **\****src***,** **size\_t** *count* **);//类似与内存拷贝函数，但可以解决重叠拷贝问题。**



#include<stdio.h>

#include<assert.h>

void\* My\_memmove(void\* dest, void\* src, int num)

{

assert(dest && src);

void\* ret = dest;

if (src < dest)

{

while (num--)

{//num = 19

\*((char\*)dest + num) = \*((char\*)src + num);//从后往前拷贝

}

}

else

{

while (num--)

{

\*(char\*)dest = \*(char\*)src;//从前往后拷贝

dest = (char\*)dest + 1;

src = (char\*)src + 1;

}

}

return ret;

}

int main()

{

int arr1[10] = { 1,2,3,4,5,6,7,8,9,10 };

//int arr2[10] = { 0 };

My\_memmove(arr1, arr1+2, 20);

return 0;

}

3.3 memcmp()函数

**int** **memcmp(** **const** **void** **\****buf1***,** **const** **void** **\****buf2***,** **size\_t** *count* **);//比较两块内存空间count字节中的值的大小，返回>0，<0，=0。**

#include<stdio.h>

#include<memory.h>

int main()

{

int arr1[10] = { 2,3,4,4,5 };

int arr2[10] = { 2,3,4,5 };

if (memcmp(arr1, arr2, 12) > 0)

{

printf(">\n");

}

else if (memcmp(arr1, arr2, 12) < 0)

{

printf("<\n");

}

else

printf("=\n");

return 0;

}

3.4 memset()函数

**void** **\*memset(** **void** **\****dest***,** **int** *c***,** **size\_t** *count* **);//对内存进行赋值，目标空间，设置值，以及设置空间的大小（按字节单位进行设置），返回空间的地址。**

#include<stdio.h>

#include<memory.h>

int main()

{

char c[10] = { 0 };

int i = 0;

char\* p = c;

for (i = 0; i < 10; i++)

{

memset(p + i, 65 + i, 1);//每个字节设置成大写字母65+i为ASCII码

}

return 0;

}