一、获取字符串长度函数

```
#include <string.h>
size_t strlen(const char *s);
功能：计算一个字符串的长度
参数：
 s：指定的字符串
返回值：
 当前字符串的长度
注意：strlen获取的字符串长度遇到第一个\0结束且\0不算做字符串长度之中
```

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
 //使用strlen函数获取字符串的长度
```

```
7   //strlen获取的字符串的长度遇到第一个\0结束
8   char s1[100] = "hel\0lo";
9
10   printf("s1_len = %d\n", strlen(s1));
11   printf("s1_size = %d\n", sizeof(s1));
12
13   char *s2 = "hello";
14
15   printf("s2_len = %d\n", strlen(s2));
16   printf("s2_size = %d\n", sizeof(s2));
17
18   return 0;
19  }
20
```

执行结果

```
Starting C:\Users\lzx\Desktop\src
\debug\01_strlen.exe...
s1_len = 3
s1_size = 100
s2_len = 5
s2_size = 4
C:\Users\lzx\Desktop\src\build-01
```

二、字符串拷贝函数

```
1  #include <string.h>
2  char *strcpy(char *dest, const char *src);
3  功能：将src复制给dest
4  参数：
5   dest：目的字符串
6   src：源字符串
7  返回值：
8   保存dest字符串的首地址
9  注意：使用strcpy函数复制字符串时必须保证dest足够大，否则会内存溢出
10   strcpy是将src字符串中第一个\0之前包括\0复制给dest
11
12  char *strncpy(char *dest, const char *src, size_t n);
13  函数的说明：
14   将src指向的字符串前n个字节，拷贝到dest指向的内存中
15  返回值:
16   目的内存的首地址
```

```
注意：
 1、strncpy不拷贝 ‘\0’
 2、如果n大于src指向的字符串中的字符个数，则在dest后面填充n-strlen(src)个’\0’
```

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
 //使用strcpy函数拷贝字符串
 char s1[32] = "hello world";
 //使用strcpy函数时，必须保证第一个参数的内存足够大
 //char s1[5] = "abcd";
  char s2[32] = "abcdefg";

  strcpy(s1, s2);

  printf("s1 = %s\n", s1);

  int i;
  for(i = 0; i < 32; i++)
  {
  printf("[%c] - %d\n", s1[i], s1[i]);
  }

  return 0;
}

```

执行结果

```
\debug\02_strcpy.exe...
s1 = abcdefg
[a] - 97
[b] - 98
[c] - 99
[d] - 100
[e] - 101
[f] - 102
[g] - 103
[] - 0
[r] - 114
[l] - 108
[d] - 100
[] - 0
[] - 0
```

三、字符串追加函数

```
#include <string.h>
char *strcat(char *dest, const char *src);
功能：将src追加到dest的后面
参数：
 dest：目的字符串
 src：源字符串
返回值：
 保存dest字符串的首地址

char *strncat(char *dest, const char *src, size_t n);
追加src指向的字符串的前n个字符，到dest指向的字符串的后面。
注意如果n 大于src的字符个数，则只将src字符串追加到dest指向的字符串的后面
追加的时候会追加'\0'
```

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
 //使用strcat函数追加字符串
 char s1[32] = "hello world";
 char s2[32] = "abcdef";

 //strcat是从s1的\0的位置开始追加，直到s2的第一个\0复制完毕后结束
 strcat(s1, s2);

 printf("s1 = %s\n", s1);

 return 0;
}

```

执行结果

```
Starting C:\Users\lzx\Desktop\
\debug\03_strcat.exe...
s1 = hello worldabcdef
C:\Users\lzx\Desktop\src\build
```

四、字符串比较函数

```
#include <string.h>
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);
功能：strcmp是比较两个字符串的内容，strncmp是比较两个字符串的前n个字节是否一样
参数：
 s1、s2：要比较的两个字符串
 n：strncmp中的参数n表示要比较的字节数
返回值：
 0 s1 = s2
  >0 s1 > s2
  <0 s1 < s2
```

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
 //使用strcmp比较两个字符串的内容是否一致
 //strcmp函数一个字符一个字符比较，只要出现不一样的，就会立即返回
 char s1[] = "hello";
 char s2[] = "w";

  int ret = strcmp(s1, s2);

  if(ret == 0)
  {
  printf("s1 = s2\n");
  }
  else if(ret > 0)
  {
  printf("s1 > s2\n");
  }
  else
  {
  printf("s1 < s2\n");
  }

  return 0;
```

```
27 }
```

执行结果

```
Starting C:\Users\lzx\Desktop
\debug\04_strcmp.exe...
s1 < s2
C:\Users\lzx\Desktop\src\bui
\04_strcmp.exe exited with c
```

五、字符查找函数

```
#include <string.h>
char *strchr(const char *s, int c);
功能：在字符指针s指向的字符串中，找ascii 码为c的字符
参数：
 s：指定的字符串
 c：要查找的字符
返回值：
 成功：找到的字符的地址
 失败：NULL
注意：s指向的字符串中有多个ASCII为c的字符，则找的是第1个字符

char *strrchr(const char *s, int c);
功能：在s指向的字符串中，找最后一次出现的ASCII为c的字符，
```

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
 //使用strchr函数在一个字符串中查找字符
 char s[] = "hel6lo wor6ld";
 //找第一个匹配的字符
 char *ret = strchr(s, '6');
 //找最后一个匹配的字符
 //char *ret = strrchr(s, '6');

 if(ret == NULL)
 {
 printf("没有找到\n");
 }
 else
```

```
18  {
19  printf("找到了，在数组的第%d个位置\n", ret - s);
20  }
21
22  return 0;
23 }
```

执行结果

```
\debug\05_strchr.exe...
找到了，在数组的第3个位置
C:\Users\lzx\Desktop\src\b
```

六、字符串匹配函数

```
1 #include <string.h>
2 char *strstr(const char *haystack, const char *needle);
3 函数说明：
4  在haystack指向的字符串中查找needle指向的字符串，也是首次匹配
5 返回值：
6  找到了：找到的字符串的首地址
7  没找到：返回NULL
```

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(int argc, char *argv[])
5  {
6   //使用strstr函数在一个字符串中查找另一个字符串
7   char s[] = "1234:4567:666:789:666:7777";
8
9   //strstr查找的时候，查找的是第二个参数的第一个\0之前的内容
10   char *ret = strstr(s, "666");
11
12   if(ret == NULL)
13   {
14   printf("没找到\n");
15   }
16   else
17   {
18   printf("找到了，在当前字符串的第%d个位置\n", ret - s);
19   }
```

```
20   return 0;
21  }
```

执行结果

```
Starting C:\Users\lzx\Desktop\s
\debug\06_strstr.exe...
找到了，在当前字符串的第10个位置
C:\Users\lzx\Desktop\src\build-
```

七、字符串转换数值

```
1  #include <stdlib.h>
2  int atoi(const char *nptr);
3  功能：将一个数字型字符串转化为整形数据
4  参数：
5   nptr：指定的字符串
6  返回值：
7   获取到的整形数据
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[])
5  {
6   //使用atoi将数字型字符串转化为整形数据
7   char s1[] = "7856";
8   int ret1 = atoi(s1);
9
10   printf("ret1 = %d\n", ret1);
11
12   //使用atof将浮点型的字符串转化为浮点型数据
13   char s2[] = "3.1415926";
14   double ret2 = atof(s2);
15
16   printf("ret2 = %lf\n", ret2);
17
18   return 0;
19  }
```

执行结果

```
Starting C:\Users\lzx\Des
\debug\07_atoi.exe...
ret1 = 7856
ret2 = 3.141593
C:\Users\lzx\Desktop\src\
```

八、字符串切割函数

```
#include <string.h>
char *strtok(char *str, const char *delim);
功能：对字符串进行切割
参数：
 str：要切割的字符串
 第一次切割，就传入指定的字符串，后面所有次的切割传NULL
 delim：标识符，要根据指定的delim进行切割，切割的结果不包含delim
返回值：
 返回切割下来的字符串的首地址，如果都切割完毕，则返回NULL
```

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
 //使用strtok函数切割字符串
 char s[] = "111:22222:33:4444444444:5555555555555";
 char *ret;

 //第一次切割
 ret = strtok(s, ":");
 printf("ret = %s\n", ret);

 //后面所有切割时都要将strtok的第一个参数传NULL
 while((ret = strtok(NULL, ":")) != NULL)
 {
 printf("ret = %s\n", ret);
 }

 return 0;
}
```

执行结果

```
\debug\08_strtok.exe...
ret = 111
ret = 22222
ret = 33
ret = 4444444444
ret = 5555555555555
C:\Users\lzx\Desktop\src\build-
```

九、格式化字符串操作函数

```
#include <stdio.h>
int sprintf(char *str, const char *format, ...);
功能：将按照格式保存的字符串复制给str
参数：
 str：保存字符串
 format：同printf
返回值：
 保存的字符串的字节数


#include <stdio.h>
int sscanf(const char *str, const char *format, ...);
功能：scanf是从终端读取数据并赋值给对应变量，而sscanf是从第一个参数中读取数据
参数：
 str：指定要获取内容的字符串
 format：按照格式获取数据保存在变量中
返回值：
 成功获取的个数
```

9.1 sprintf和sscanf的基本用法

```
//sprintf和sscanf的基本用法
void test1()
{
 char buf[20];
 int a, b, c;

 sprintf(buf,"%d:%d:%d",2013,10,1);
 printf("buf = %s\n",buf);

  sscanf("2013:10:1", "%d:%d:%d", &a, &b, &c);
  printf("a=%d,b=%d,c=%d\n",a,b,c);
}
```

执行结果

```
Desktop_Qt_5_8_0_M
buf = 2013:10:1
a=2013,b=10,c=1
C:\Users\lzx\Deskt
```

9.2 sscanf高级用法

```
1  //sscanf高级用法
2  void test2()
3  {
4   //1、跳过数据：%*s或%*d
5   char buf1[20];
6   sscanf("1234 5678","%*d %s",buf1);
7   printf("%s\n",buf1);
8
9   //2、读指定宽度的数据：%[width]s
10   char buf2[20];
11   sscanf("12345678","%4s ",buf2);
12   printf("%s\n",buf2);
13
14   //3、支持集合操作：只支持获取字符串
15   // %[a-z] 表示匹配a到z中任意字符(尽可能多的匹配)
16   // %[aBc] 匹配a、B、c中一员，贪婪性
17   // %[^aFc] 匹配非a、F、c的任意字符，贪婪性
18   // %[^a-z] 表示读取除a-z以外的所有字符
19   char buf3[20];
20   sscanf("agcd32DajfDdFF","%[a-z]",buf3);
21   printf("%s\n",buf3);
22  }
```

执行结果

```
Starting C:\Users\lzx\Desk
Desktop_Qt_5_8_0_MinGW_32b
5678
1234
agcd
C:\Users\lzx\Desktop\src\b
\debug\09_sprintf_sscanf.e
```

十、const

```
1  #include <stdio.h>
2
```

```c
//const修饰全局变量
//此时全局变量只能使用但是不能修改，
//如果直接拿全局变量修改值，编译直接报错
//如果使用全局变量的地址修改值，运行时程序异常结束
const int a = 100;
void test1()
{
  printf("a = %d\n", a);

  //a = 666;
  //printf("a = %d\n", a);

  int *p = &a;
  *p = 888;
  printf("a = %d\n", a);
}

//const修饰普通局部变量
//可以读取变量的值
//不能直接通过变量进行修改值，编译报错
//可以通过变量的地址修改值
void test2()
{
  const int b = 100;
  printf("b = %d\n", b);

  //b = 666;
  //printf("b = %d\n", b);

  int *p = &b;
  *p = 888;
  printf("b = %d\n", b);
}

//const修饰指针变量
//如果const修饰指针变量的类型，无法通过指针变量修改地址里面的值
//如果const修饰指针变量，无法修改指针变量保存的地址
//如果const既修饰指针变量的类型，又修饰指针变量，则只能通过原本变量修改值
void test3()
{
```

```
 int c = 100;
 //const修饰指针变量的类型
 //const int * p = &c;
 //const修饰指针变量
 //int * const p = &c;
 //const既修饰指针变量的类型，又修饰指针变量
 const int * const p = &c;
 printf("*p = %d\n", *p);

 c = 666;
 printf("*p = %d\n", *p);

 *p = 777;
 printf("*p = %d\n", *p);

 int d = 888;
 p = &d;
 printf("*p = %d\n", *p);
}

int main(int argc, char *argv[])
{
 test3();

 return 0;
}

```