

C语言——面向过程编程

单元四、字符串

第一次课 字符串及字符串API

一、字符、字符数组与字符串

① 【回顾：】 字符类型、ASCII码值及显示原理

十进制	二进制	符号	十进制	二进制	符号	十进制	二进制	符号	十进制	二进制	符号
0	0000 0000	NUL	32	0010 0000	[空格]	64	0100 0000	@	96	0110 0000	`
1	0000 0001	SOH	33	0010 0001	!	65	0100 0001	A	97	0110 0001	a
2	0000 0010	STX	34	0010 0010	"	66	0100 0010	B	98	0110 0010	b
3	0000 0011	ETX	35	0010 0011	#	67	0100 0011	C	99	0110 0011	c
4	0000 0100	EOT	36	0010 0100	\$	68	0100 0100	D	100	0110 0100	d
5	0000 0101	ENQ	37	0010 0101	%	69	0100 0101	E	101	0110 0101	e
6	0000 0110	ACK	38	0010 0110	&	70	0100 0110	F	102	0110 0110	f
7	0000 0111	BEL	39	0010 0111	\	71	0100 0111	G	103	0110 0111	g
8	0000 1000	BS	40	0010 1000	(72	0100 1000	H	104	0110 1000	h
9	0000 1001	HT	41	0010 1001)	73	0100 1001	I	105	0110 1001	i
10	0000 1010	LF	42	0010 1010	*	74	0100 1010	J	106	0110 1010	j
11	0000 1011	VT	43	0010 1011	+	75	0100 1011	K	107	0110 1011	k
12	0000 1100	FF	44	0010 1100	,	76	0100 1100	L	108	0110 1100	l
13	0000 1101	CR	45	0010 1101	-	77	0100 1101	M	109	0110 1101	m
14	0000 1110	SO	46	0010 1110	.	78	0100 1110	N	110	0110 1110	n
15	0000 1111	SI	47	0010 1111	/	79	0100 1111	O	111	0110 1111	o
16	0001 0000	DLE	48	0011 0000	0	80	0101 0000	P	112	0111 0000	p
17	0001 0001	DC1	49	0011 0001	1	81	0101 0001	Q	113	0111 0001	q
18	0001 0010	DC2	50	0011 0010	2	82	0101 0010	R	114	0111 0010	r
19	0001 0011	DC3	51	0011 0011	3	83	0101 0011	S	115	0111 0011	s
20	0001 0100	DC4	52	0011 0100	4	84	0101 0100	T	116	0111 0100	t
21	0001 0101	NAK	53	0011 0101	5	85	0101 0101	U	117	0111 0101	u
22	0001 0110	SYN	54	0011 0110	6	86	0101 0110	V	118	0111 0110	v
23	0001 0111	ETB	55	0011 0111	7	87	0101 0111	W	119	0111 0111	w
24	0001 1000	CAN	56	0011 1000	8	88	0101 1000	X	120	0111 1000	x
25	0001 1001	EM	57	0011 1001	9	89	0101 1001	Y	121	0111 1001	y
26	0001 1010	SUB	58	0011 1010	:	90	0101 1010	Z	122	0111 1010	z
27	0001 1011	ESC	59	0011 1011	;	91	0101 1011	[123	0111 1011	{
28	0001 1100	FS	60	0011 1100	<	92	0101 1100	\	124	0111 1100	
29	0001 1101	GS	61	0011 1101	=	93	0101 1101]	125	0111 1101	}
30	0001 1110	RS	62	0011 1110	>	94	0101 1110	^	126	0111 1110	~
31	0001 1111	US	63	0011 1111	?	95	0101 1111	_	127	0111 1111	DEL

字符在屏幕上的显示原理



48	0011 0000	0
49	0011 0001	1
50	0011 0010	2
51	0011 0011	3
52	0011 0100	4
53	0011 0101	5
54	0011 0110	6
55	0011 0111	7
56	0011 1000	8
57	0011 1001	9

在显卡中找到字符编码
找到对应的像素点：“点亮”

② 字符数组与'\0'

```
char word[]={'h','e','l','l','o' };  
printf("%s\n",word);
```

```
char word[]={'h','e','l','l','o' };  
printf("%s\n",word);  
/*字符串*/ %s 打印字符串的格式
```

选择C:\WINDOWS\system32\cmd.exe

hello烫烫烫? S渣?
请按任意键继续. . .

打印结果有乱码

【原因：】%s字符串格式在打印时，会从数组首地址开始依次打印每一个字符。它会以一个特殊的字符'\0'作为字符串的结束标记。由于word数组没有'\0'这个字符，它会一致打印每个字符，直到遇到'\0'为止。

【注意：】'\0'中的'\为转义字符 比如\n','\t'。'\0'转义后其实就是0 0 '\0'。之所以这么写，就是约定的特殊字符——即字符串的结束标记。

因此，需要如下修改。

```
char word[]={'h','e','l','l','o','\0' };// '\0'也占一个位置空间  
printf("%s\n",word);
```

C:\WINDOWS\system32\cmd.exe

hello
请按任意键继续. . .

③ 字符串对字符数组赋初值

```
char word[]={ 'h','e','l','l','o','\0' };//当字符赋值方式（麻烦）
char word2[]="hello";//字符串赋值方式（快捷）
printf("%s\n",word);
printf("%s\n",word2);
```

名称	值	类型
word	0x00f3f7b8 "hello"	char[6]
[0]	104 'h'	char
[1]	101 'e'	char
[2]	108 'l'	char
[3]	108 'l'	char
[4]	111 'o'	char
[5]	0 '\0'	char
word2	0x00f3f7a8 "hello"	char[6]
[0]	104 'h'	char
[1]	101 'e'	char
[2]	108 'l'	char
[3]	108 'l'	char
[4]	111 'o'	char
[5]	0 '\0'	char

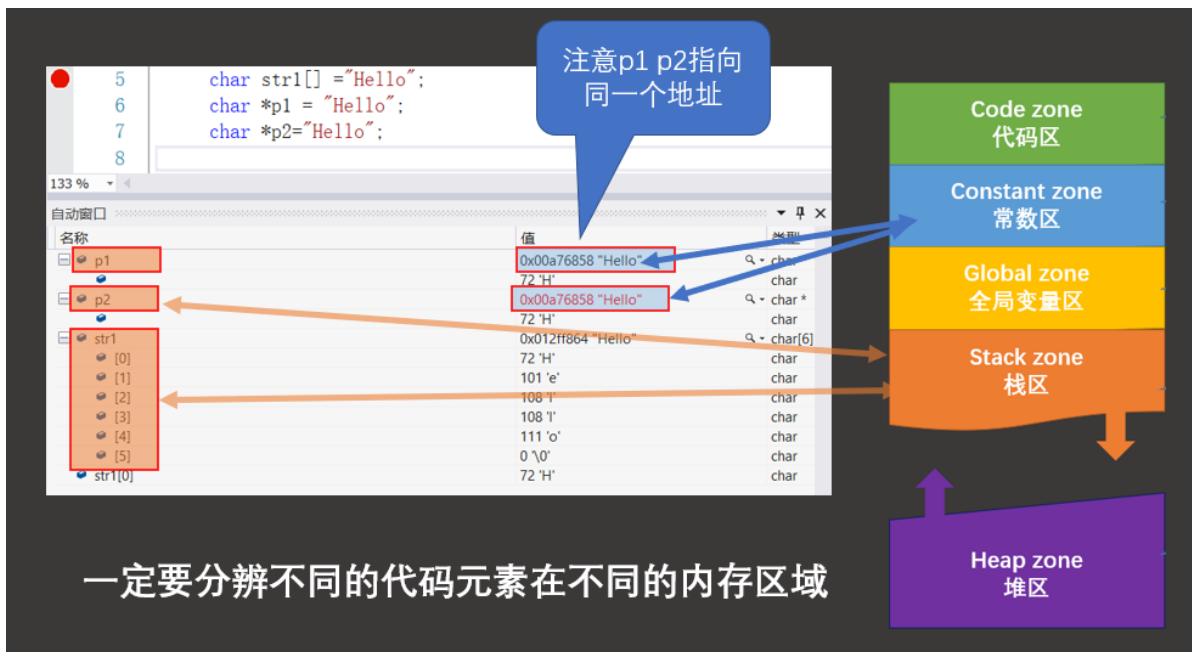
名称	值	类型
word	0x00b9f7b0 "hello"	char[6]
[0]	104 'h'	char
[1]	101 'e'	char
[2]	108 'l'	char
[3]	108 'l'	char
[4]	111 'o'	char
[5]	0 '\0'	char
word2	0x00b9f7a0 "hello"	char[6]
[0]	104 'h'	char
[1]	101 'e'	char
[2]	108 'l'	char
[3]	108 'l'	char
[4]	111 'o'	char
[5]	0 '\0'	char

自动补上'\0' →

二、字符串的存储原理

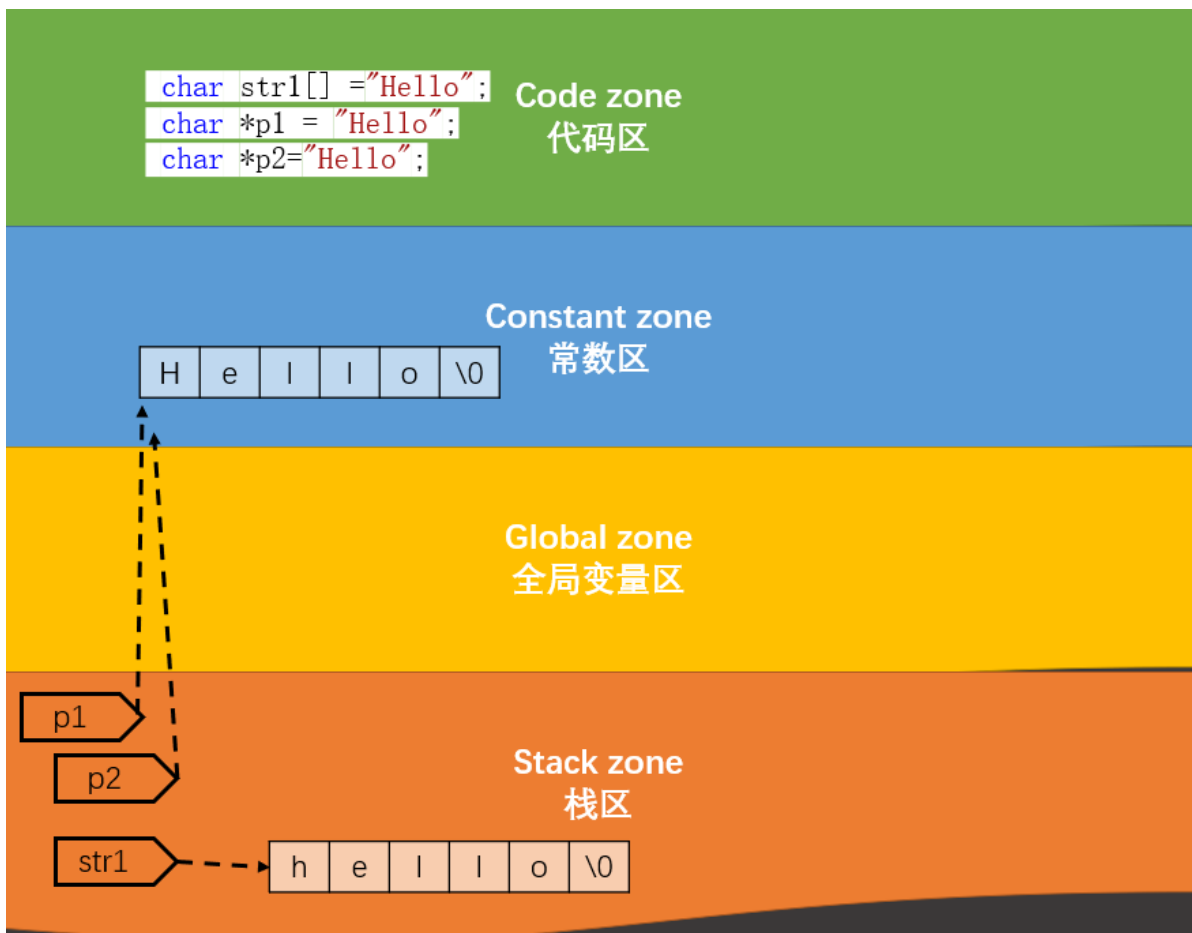
【重点理解：】字符数组与字符指针对字符串处理的区别

```
char str1[] ="Hello";
char *p1 = "Hello";
char *p2="Hello";
```



通过下图可知：hello字符串常量存在于常量区，“hello”字符串常量返回的就是它所在内存中的首地址。

`char str1[] = "Hello";` 当此句执行时，系统会在“常量区”寻找“hello”字符串常量，如果没有就创建一个。由于左侧是字符串数组，它会在栈区创建字符串数组并把每个字符存入栈区的每个数组成员。但是当执行 `char *p1 = "Hello";` 时，发现“常量区”有hello就把其首地址赋给p1，因此通过p1只能访问的第一个字符h，`char *p2 = "Hello";` 亦如此。所以p1 和 p2指向同一个地址。



名称	值	类型
p1	0x00a76858 "Hello"	char *
	72 'H'	char
p2	0x00a76858 "Hello"	char *
	72 'H'	char
str1	0x012ff864 "Gello"	char[6]
[0]	71 'G'	char
[1]	101 'e'	char
[2]	108 'l'	char
[3]	108 'l'	char
[4]	111 'o'	char
[5]	0 '\0'	char
str1[0]	71 'G'	char

三、字符串常用API

API是什么？

API（应用程序编程接口）

API（Application Programming Interface，应用程序接口）是一些预先定义的接口（如函数、HTTP接口），或指软件系统不同组成部分衔接的约定。用来提供应用程序与开发人员基于某软件或硬件得以访问的一组例程，而又无需访问源码，或理解内部工作机制的细节。

简单来说：

API就是现有的函数，拿来就用。会用即可、懂原理更佳，能查手册是王道。例如：rand()，printf()都是API。

根据API的来源可分为：

- 第一方：即系统自带的，stdio.h、stdlib.h、string.h、math.h里面的函数。
- 第二方：即自我封装的；包括所在公司、团队、个人具有产权的代码。
- 第三方：即开源免费或通过购买许可得到的使用权，easyX就属于第三方API。

字符串常用于文字处理，常用且实用。因此其常用功能被封装成了一套API。可通过string.h进行使用。如下就是常见的功能及用法。

手册可以看微软“特色”的：<https://docs.microsoft.com/zh-cn/cpp/c-runtime-library/string-manipulation-crt?view=msvc-160>

也可以看通用的：<https://www.cplusplus.com/reference/cstring/>

常用String字符串API用法如下：

① strlen

功能：计算字符串长度

简例：

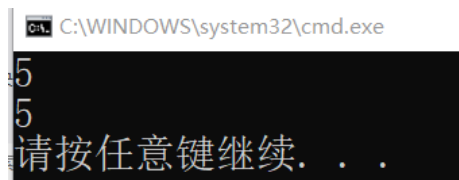

```

#include <stdio.h>
#include <string.h>
int my_strlen(char * str);
int main ()
{
    char str1[] ="Hello";
    char *p1="Hello";
    printf("%d\n", strlen(str1));
    printf("%d\n", strlen(p1));

    return 0;
}

```

结果:



```

C:\WINDOWS\system32\cmd.exe
5
5
请按任意键继续. . .

```

【注意：】'\0'不被计算在字符串长度内。但在数组中，'\0'是占一个数组成员空间的。即str1的数组长度为6，但字符串长度为5。

② strcpy、strcpy_s、strncpy、strncpy_s

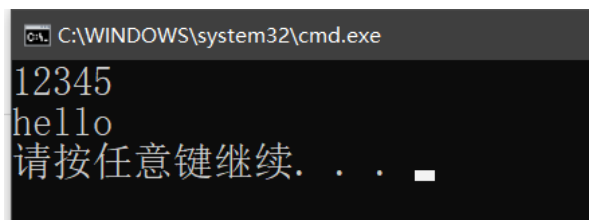
功能：字符串的复制

简例 1:

```

#include <stdio.h>
#include <string.h>
int main ()
{
    /*字符串复制函数*/
    char * p="12345";
    char a[20];
    /*strcpy 字符串的复制*/
    strcpy(a,p);//把p指向的字符串，复制到 a数组
    printf("%s\n",a);
    strcpy(a,"hello");//把字符串常量，复制到 a数组
    printf("%s\n",a);
    return 0;
}

```



```

C:\WINDOWS\system32\cmd.exe
12345
hello
请按任意键继续. . .

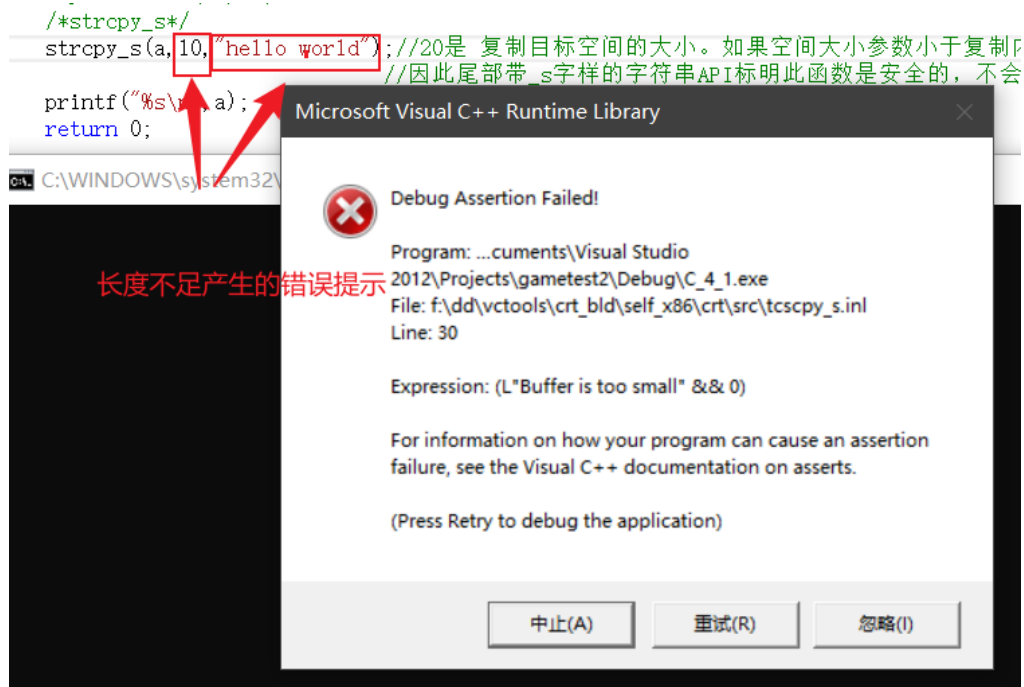
```

简例 2:

```
/*strcpy_s 字符串的复制(安全版)*/
strcpy_s(a,20,"hello world");//20是 复制目标空间的大小。如果空间大小参数小于复制内容会产生
错误提示。
//因此尾部带_s字样的字符串API标明此函数是安全的，不会溢出或目标空间不足的潜在隐患。
printf("%s\n",a);
```

```
C:\WINDOWS\system32\cmd.exe
hello world
请按任意键继续. . .
```

【注意：】

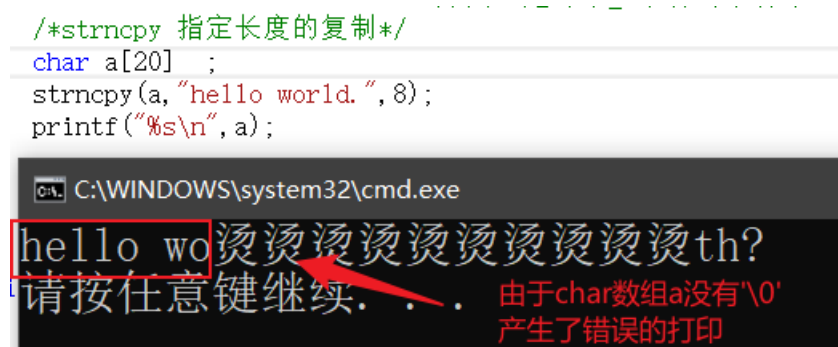


简例 3:

```
/*strncpy 指定长度的复制*/
char a[20] = {'\0'} ;//务必把数组所有成员赋值为'\0'
strncpy(a,"hello world.",8);
printf("%s\n",a);
```

```
C:\WINDOWS\system32\cmd.exe
hello wo
请按任意键继续. . .
```

【注意：】如果数组a的成员没有被初始化清零，会有如下结果。



简例 4:

```
char a[20] ;
//指定长度的复制（安全版） a数组不用全部初始'\0'
strncpy_s(a,10,"hello world",8);
printf("%s\n",a);
```

名称	值	类型
返回 strncpy_s	0	int
a	0x0078fa8c "hello wo"	char[20]
[0]	104 'h'	char
[1]	101 'e'	char
[2]	108 'l'	char
[3]	108 'l'	char
[4]	111 'o'	char
[5]	32 ' '	char
[6]	119 'w'	char
[7]	111 'o'	char
[8]	0 '\0'	char
[9]	-2 '?'	char
[10]	-52 '?'	char
[11]	-52 '?'	char
[12]	-52 '?'	char

自动添加

③ strcat、strcat_s、strncat、strncat_s

功能：字符串拼接函数

简历1：

```
#include <stdio.h>
#include <string.h>
int main ()
{
    /*字符串拼接函数*/
    char a[50]="hello ";
    char *p="hi ";
    /*strcat字符串拼接函数(经典版)*/
    strcat(a,p);
    printf("%s\n",a);
}
```

```
C:\WINDOWS\system32\cmd.exe
hello hi
请按任意键继续. . .
```

简例 2：

```
/*strcat_s字符串拼接函数(安全版)*/
strcat_s(a,20,p); //20即保证 a 具有的保障性的空间大小
printf("%s\n",a);
```

```
C:\WINDOWS\system32\cmd.exe
hello hi
请按任意键继续. . .
```

简例3：


```
/*指定长度的字符串连接（经典版）*/
strncat(a,p,1);//只把p指向字符串的 1 个长度的字符串连接到a数组。
printf("%s\n",a);
```

A terminal window titled 'C:\WINDOWS\system32\cmd.exe' showing the output 'hello h' and the prompt '请按任意键继续. . .'.

简例4:

```
/*指定长度的字符串连接（安全版）*/
strncat_s(a,15,p,1);// 实参15 即保证 a 具有的保障性的空间大小
printf("%s\n",a);
```

A terminal window titled 'C:\WINDOWS\system32\cmd.exe' showing the output 'hello h' and the prompt '请按任意键继续. . .'.

④ strcmp

功能：字符串比较函数，返回 1 代表 大于关系； 返回 0代表等于关系；返回-1代表小于关系。

简例:

```
#include <stdio.h>
#include <string.h>
int main ()
{
    /*字符串比较函数*/
    /*strcmp*/
    char *p="abcd";
    char *q="abcde";
    printf("%d\n",strcmp(p,q));

    return 0;
}
```

A terminal window titled 'C:\WINDOWS\system32\cmd.exe' showing the output '1' and the prompt '请按任意键继续. . .'.

打印 1 说明：p字符串大于q字符串

A screenshot showing a code snippet where 'abcd' is assigned to both 'p' and 'q', and a terminal window showing the output '0'. A red arrow points from the text '打印 0 说明完全相同' to the '0' in the terminal.

```
char *p="abcd";
char *q="abcd";
printf("%d\n",strcmp(p,q));
```

打印 0 说明完全相同

请按任意键继续. . .

```
/*strcmp*/
char *p="Abcd";
char *q="a";
printf("%d\n", strcmp(p, q));
```

A的ASCII码值小于a,
因此返回负数

C:\WINDOWS\system32\cmd.exe 说明: p字符串小于a字符串

-1
请按任意键继续. . .

⑤ strstr

功能: 字符串匹配函数; 搜索字符串在另一个字符串中首次出现的位置。

```
#include <stdio.h>
#include <string.h>
int main ()
{
    /*字符串匹配函数
    搜索字符串在另一个字符串中首次出现的位置。
    */
    char str1[]="this is a simple string";
    char *str2="is";
    char *p ;
    p = strstr(str1,str2); //在第一个字符串中找第二个字符串首次出现的指针并返回
    printf("%c\n", *p ); //输出p指向的字符
    printf("%s\n", p ); //输出从p开始的字符串
    return 0;
```

```
C:\WINDOWS\system32\cmd.exe

i
is is a simple string
请按任意键继续. . .
```

⑥ itoa

功能: 按指定的进制, 将数字转成字符串

简例

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main ()
{
    char s[25];
    itoa(10,s,2); //把10转换成二进制的字符串
    printf("%s\n",s);
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
1010
请按任意键继续. . .
```

⑦: atoi

功能：将字符串转成十进制形式的数字

简例：

```
#include <stdio.h>
#include <string.h>
#include<stdlib.h>
int main ()
{
    char s[25]="8848";
    int h=atoi(s); //把字符串 转换成int型整数。
    printf("%d\n",h);
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
8848
请按任意键继续. . .
```

```
8
9 char s[25]="884a8";
0 int h=atoi(s);
1 printf("%d\n",h);
2
3
4 }
5
6
7
```

遇到第一个非法字符就停止转换

```
C:\WINDOWS\system32\cmd.exe
884
请按任意键继续. . .
```

类似的函数还有：atof把字符串转成float类型、atol把字符串转成long类型

⑧: getchar

功能：从键盘输入缓冲区读取一个字符：包括 回车\n 空格等特殊字符。

简例：

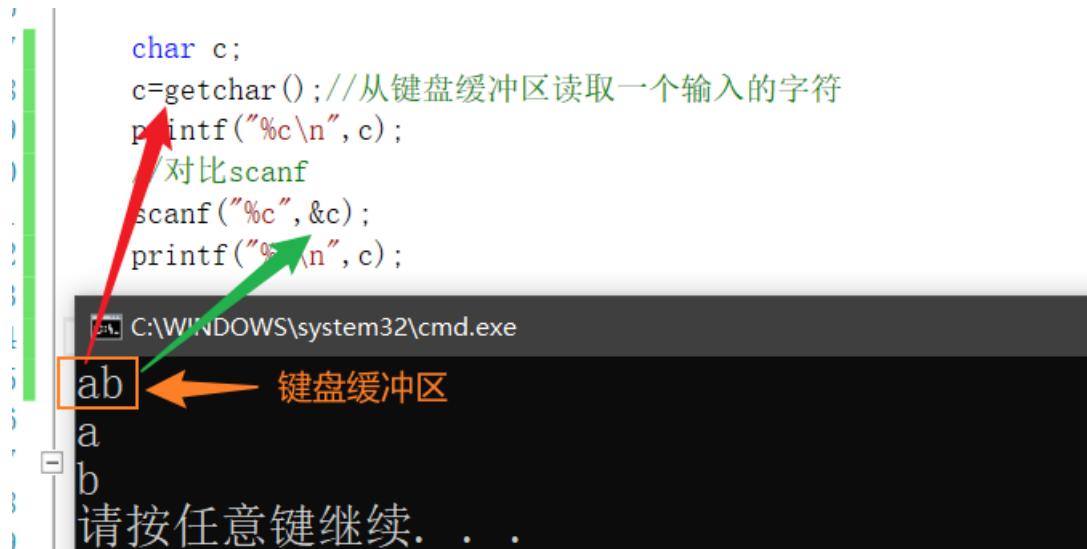
```
#include <stdio.h>
#include <string.h>
#include<stdlib.h>
int main ()
{
    char c;
```

```

c=getchar();//从键盘缓冲区读取一个输入的字符
printf("%c\n",c);
//对比scanf的用法区别
scanf("%c",&c);
printf("%c\n",c);

return 0;
}

```



如果你输入的内容不考虑回车符号，建议用conio.h的getch()或getche()

```

#include <stdio.h>
#include <string.h>
#include<stdlib.h>
#include <conio.h> //注意： getch getche包含此头文件
int main ()
{

    char c;
    c=getch(); //不回显你输入的字符
    printf("你输入了: %c\n",c);
    c=getche();//回显你输入的字符
    printf("你输入了: %c\n",c);
    return 0;
}

```

⑨：gets、gets_s

功能：从键盘缓冲区读取一串字符

```

#include <stdio.h>
#include <string.h>
#include<stdlib.h>
int main ()
{
    char a[300];
    gets(a);//<经典版>从键盘缓冲区读取一行字符放入到a数组
    printf("%s\n",a);
}

```

```
gets_s(a,10);//<安全版>从键盘缓冲区读取一行字符放入到a数组，10代表缓冲区的大小 超出提示  
错误  
printf("%s\n",a);  
//对比scanf  
scanf("%s",a);  
printf("%s\n",a);  
return 0;  
}
```

C:\WINDOWS\system32\cmd.exe

abc 输入的字符串

abc

123

123

456

456

请按任意键继续. . .