

规格严格 功夫到家



# 第11讲 字符串处理

## 教材10.1~10.5节

## MOOC第10周

哈尔滨工业大学

苏小红

sxh@hit.edu.cn

# 如何输出一个字符串？

```
#define LEN 80  
char str[LEN+1];
```

```
printf("%s\n", str);
```

```
puts(str);
```

```
for (i=0; str[i]!='\0'; i++)  
{  
    //putchar(str[i]);  
    printf("%c", str[i]);  
}  
//putchar('\n');  
printf("\n");
```

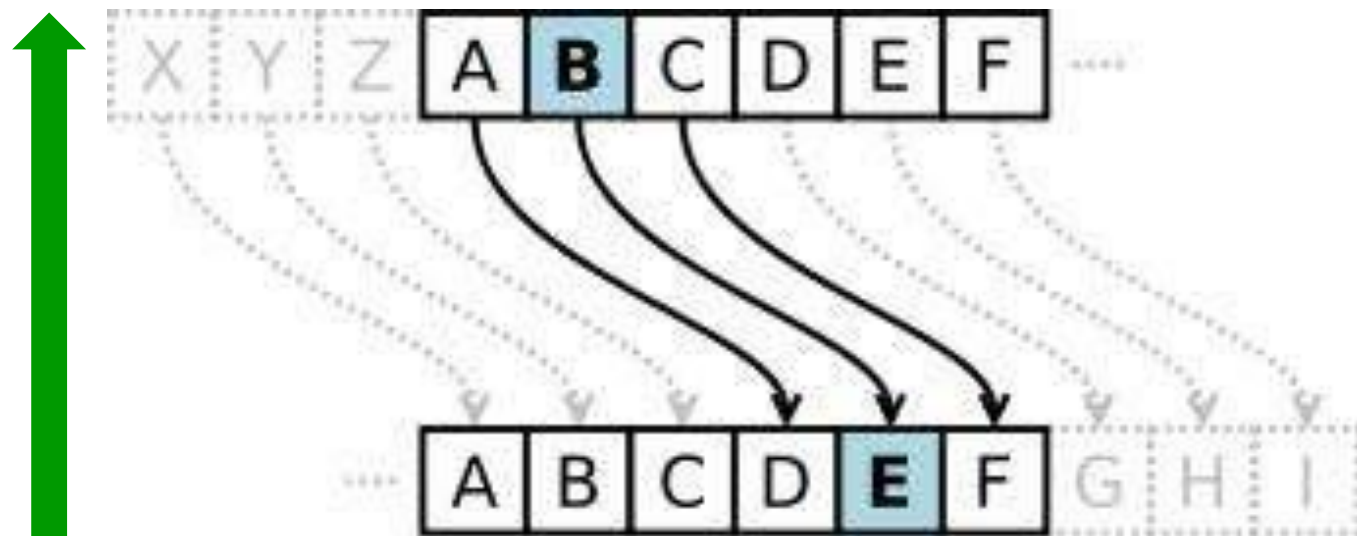
```
len = strlen(str);  
for (i=0; i<len; i++)  
{  
    //putchar(str[i]);  
    printf("%c", str[i]);  
}  
//putchar('\n');  
printf("\n");
```

检测字符串结束标志  
不用  $i < \text{STR\_LEN}$

先计算字符串长度

# 凯撒密码

- 凯撒密码——代换加密技术，原理是将字母加上一个定值产生密文
- 请根据这一线索编程**破解字条上的密文**，将用户从键盘输入的密文字符串（只包含a~z或A~Z 的英文字母且长度小于等于100）进行解密后输出。



?

BhvLoryhbrx

//函数功能: 计算凯撒密码, 如果不是英文字母, 则返回0, 否则将字母减去m进行解密处理

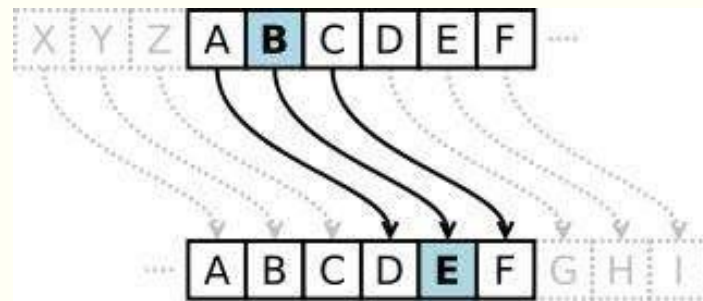
```
int Caesar(char c[], int m)
{
    int i;
    for (i = 0; _____; i++)
    {
        if (c[i]是大写英文字母)
        {
            c[i] = c[i] - m;

            对超出大写字母左边界的字符进行处理

        }
        else if (c[i]是小写英文字母)
        {
            c[i] = c[i] - m;

            对超出小写字母左边界的字符进行处理

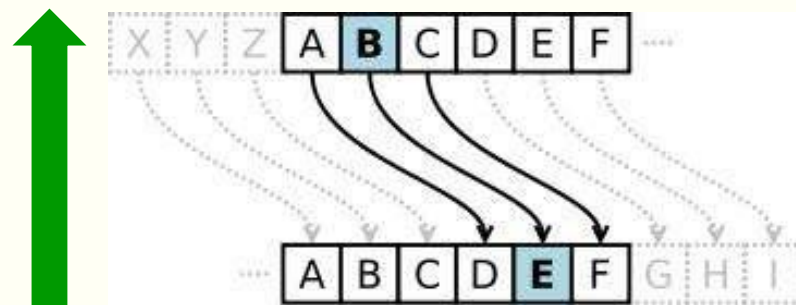
        }
        else
        {
            return 0;
        }
    }
    return 1;
}
```



```
#include <stdio.h>
#define N 100
int Caesar(char c[], int m);
int main()
{
    char c[N];
    printf("Input a string:");
    gets(c);
    if (Caesar(c, 3))
    {
        printf("%s", c);
    }
    else
    {
        printf("Input error!\n");
    }
    return 0;
}
```

//函数功能: 计算凯撒密码, 如果不是英文字母, 则返回0, 否则将字母减去m进行解密处理

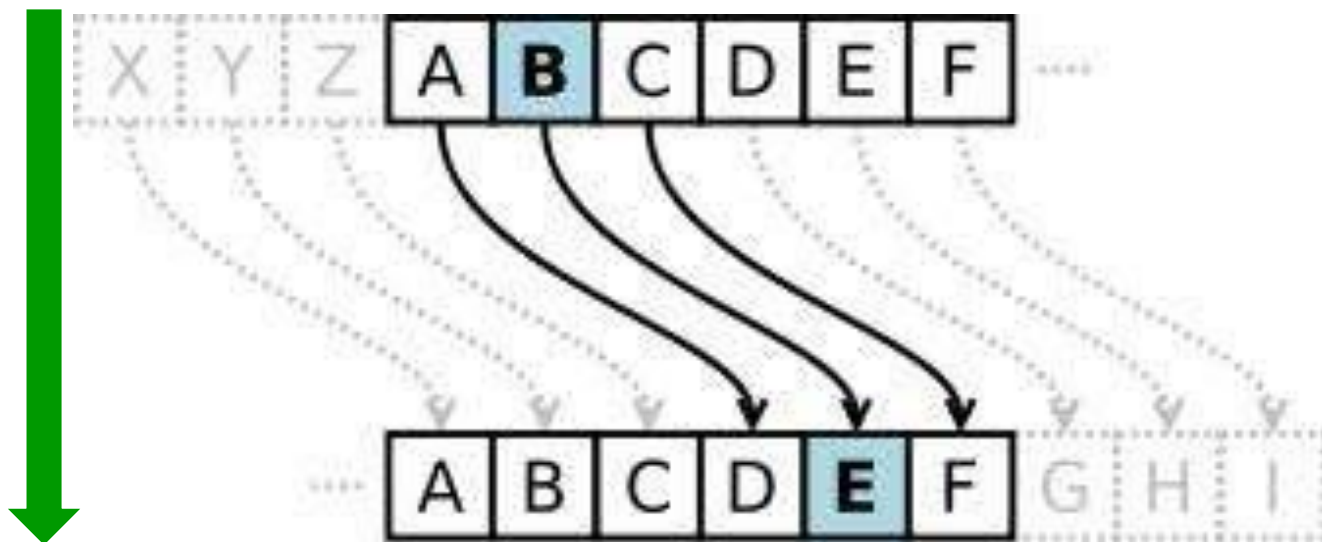
```
int Caesar(char c[], int m)
{
    int i;
    for (i = 0; c[i] != '\0'; i++)
    {
        if (c[i] >= 'A' && c[i] <= 'Z')
        {
            c[i] = c[i] - m;
            if (c[i] < 'A') //超过左边界
            {
                c[i] = c[i] + 26;
            }
        }
        else if (c[i] >= 'a' && c[i] <= 'z')
        {
            c[i] = c[i] - m;
            if (c[i] < 'a') //超过左边界
            {
                c[i] = c[i] + 26;
            }
        }
        else
        {
            return 0;
        }
    }
    return 1;
}
```



```
#include <stdio.h>
#define N 100
int Caesar(char c[], int m);
int main()
{
    char c[N];
    printf("Input a string:");
    gets(c);
    if (Caesar(c, 3))
    {
        printf("%s", c);
    }
    else
    {
        printf("Input error!\n");
    }
    return 0;
}
```

# 凯撒密码

- 凯撒密码——代换加密技术，原理是将字母加上一个定值产生密文
- 请根据这一加密规则，编程对女生的回答进行加密处理。



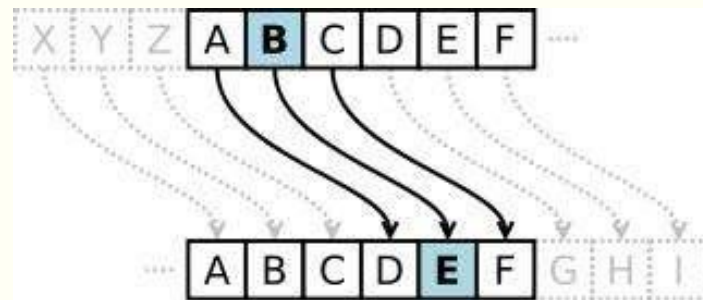
YesIloveyou



?

//函数功能: 计算凯撒密码, 如果不是英文字母, 则返回0, 否则将字母加上m进行加密处理

```
int Caesar(char c[], int m)
{
    int i, len;
    len = strlen(c);
    for (i = 0; i < len; i++)
    {
        if (c[i] >= 'A' && c[i] <= 'Z')
        {
            c[i] = c[i] + m;
            if (c[i] > 'Z') //超过右边界
            {
                c[i] = c[i] - 26;
            }
        }
        else if (c[i] >= 'a' && c[i] <= 'z')
        {
            c[i] = c[i] + m;
            if (c[i] > 'z') //超过右边界
            {
                c[i] = c[i] - 26;
            }
        }
        else
        {
            return 0;
        }
    }
    return 1;
}
```



```
#include <stdio.h>
#include <string.h>
#define N 100
int Caesar(char c[], int m);
int main()
{
    char c[N];
    printf("Input a string:");
    gets(c);
    if (Caesar(c, 3))
    {
        printf("%s", c);
    }
    else
    {
        printf("Input error!\n");
    }
    return 0;
}
```

# 最牛微信

- 什么可以决定我们100%的人生，或100%地影响我们的生活？
  - 输入一个单词，输出其百分比
  - 大小写对应相同的编码值



是哪位高人琢磨出的这条微信？太牛了？！

如果26个英文字母 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 分别等于：

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26。那么：

Knowledge (知识) :  $K+N+O+W+L+E+D+G+E = 11+14+15+23+12+5+4+7+5=96\%$ 。

Workhard (努力工作) :  $W+O+R+K+H+A+R+D = 23+15+18+11+8+1+18+4 = 98\%$ 。

也就是说知识和努力工作对我们人生的影响可以达到96%和98%。

Luck (好运)  $L+U+C+K = 12+21+3+11=47\%$ 。

Love (爱情)  $L+O+V+E = 12+15+22+5=54\%$ 。

看来，这些我们通常认为重要的东西却并没起到最重要的作用。

那么，什么可以决定我们100%的人生呢？

是Money (金钱) 吗？

$M+O+N+E+Y=13+15+14+5+25=72\%$ 。

看来也不是。

是Leadership (领导能力)吗？

$L+E+A+D+E+R+S+H+I+P=12+5+1+4+5+18+19+9+16=89\%$

a	b	c	d	e	f	g	h	i	j	k	l	m
97	98	99	100	101	102	103	104	105	106	107	108	109
n	o	p	q	r	s	t	u	v	w	x	y	z
110	111	112	113	114	115	116	117	118	119	120	121	122

**`str[i] - 'a' + 1`**



//函数功能：将字符数组str中的字符串转换为英文字母对应的数字，然后累加求和并返回

```
int LetterSum(char str[])
```

```
{
    int i, sum = 0;
    for (i=0; str[i]!='\0'; i++)
    {
        if (str[i]>='a' && str[i]<='z')
        {
            sum += str[i] - 'a' + 1;
        }
        else if (str[i]>='A' && str[i]<='Z')
        {
            sum += str[i] - 'A' + 1;
        }
        else
        {
            return -1;
        }
    }
    return sum;
}
```

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

```
int LetterSum(char str[]);
```

```
int main()
```

```
{
```

```
    char a[80];
```

```
    int sum;
```

```
    printf("Input a word:");
```

```
    gets(a);
```

```
    sum = LetterSum(a);
```

```
    if (sum != -1)
```

```
    {
```

```
        printf("%s=%d%%\n", a, sum);
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("Input error!\n");
```

```
    }
```

```
    return 0;
```

```
}
```

a	b	c	d	e	f	g	h	i	j	k	l	m
97	98	99	100	101	102	103	104	105	106	107	108	109
n	o	p	q	r	s	t	u	v	w	x	y	z
110	111	112	113	114	115	116	117	118	119	120	121	122

如果26个英文字母 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 分别等于：

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26。那么：

Knowledge (知识)：K+N+O+W+L+E+D+G+E= 11+14+15+23+12+5+4+7+5=96%。

Workhard (努力工作)：W+O+R+K+H+A+R+D= 23+15+18+11+8+1+18+4 =98%。

# 藏头诗

---

请输入藏头诗：

一叶轻舟向东流✓

帆梢轻握杨柳手✓

风纤碧波微起舞✓

顺水任从雅客悠✓

一帆风顺

```

#include <stdio.h>
#define N 20
void GetFirst(char s[][N],char t[]);
int main()
{
    int i;
    char s[4][N], t[N];
    printf("请输入藏头诗: \n");
    for (i=0; i<4; i++)
    {
        scanf("%s", s[i]);
    }
    GetFirst(s, t);
    puts(t);
    return 0;
}

```

```

#include <stdio.h>
#define N 20
char *GetFirst(char s[][N],char t[]);
int main()
{
    int i;
    char s[4][N], t[N];
    printf("请输入藏头诗: \n");
    for (i=0; i<4; i++)
    {
        scanf("%s", s[i]);
    }
    puts(GetFirst(s, t));
    return 0;
}

```

```

void GetFirst(char s[][N],char t[])
{
    int i;
    for (i=0; i<4; i++)
    {
        t[2*i] = s[i][0];
        t[2*i+1] = s[i][1];
    }
    t[2*i] = '\0';
}

```

```

char *GetFirst(char s[][N],char t[])
{
    int i;
    for (i=0; i<4; i++)
    {
        t[2*i] = s[i][0];
        t[2*i+1] = s[i][1];
    }
    t[2*i] = '\0';
    return t;
}

```

# 使用指针的原则

- 在茫茫存储器中追逐指针，一不小心就会迷失方向啊！

- 基本原则

- \* 永远清楚指针指向了哪块内存——初始化
- \* 永远清楚指针指向了什么内容——基类型
- \* x型指针指向x型变量——同一基类型初始化

- 总纲

- \* 永远清楚你正在操作哪块内存
- \* 永远清楚这种操作是否合理、合法



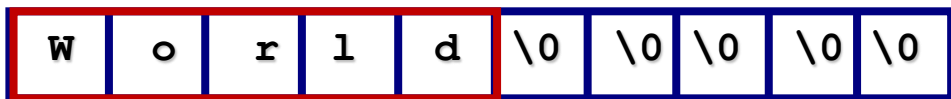
```
int str[10] = "world";  
printf("%d\n", strlen(str));  
请问打印结果是（）
```

- ☒ A 5
- ☐ B 6
- ☐ C 10

提交

# 计算字符串长度

**strlen**(字符串);



↑  
pStr

不包括 '\0' 的实际字符的个数

字符数组/指针做函数参数  
都是接收字符串的首地址



```
unsigned int MyStrlen(char str[])
{
    int i;
    unsigned int len = 0;
    for (i=0; str[i]!='\0'; i++)
    {
        len++;
    }
    return len;
}
```

```
unsigned int MyStrlen(char *pStr)
{
    unsigned int len = 0;
    for (; *pStr!='\0'; pStr++)
    {
        len++;
    }
    return len;
}
```

# 亲爱的，谁截走了我们的字符串？

```
#include <stdio.h>
#define N 20
unsigned int  MyStrlen(char str[]);
int main()
{
    char t[N] = "world";
    printf("main:%u bytes\n", MyStrlen(t));
    return 0;
}
```

字符数组/指针做函数参数  
都是接收字符串的首地址

```
unsigned int  MyStrlen(char str[])
{
    int i;
    unsigned int len = 0;
    printf("message:%s\n", str);
    printf("MyStrlen:%d bytes\n", sizeof(str));
    for (i=0; str[i]!='\0'; i++)
    {
        len++;
    }
    return len;
}
```

C:\Users\sxh\Desktop\c\coi

```
message:world
MyStrlen:4 bytes
main:5 bytes
```

把数组传给函数时，数组将退化为指针

# sizeof究竟计算的是什么？

```
#include <stdio.h>
#define N 20
unsigned int  MyStrlen(char str[]);
int main()
{
    char t[N] = "world";
    printf("main:%u bytes\n", MyStrlen(t));
    printf("t:%d bytes\n", sizeof(t));
    return 0;
}
```

F:\c\test\bin\Debug\test.exe

```
message:world
MyStrlen:4 bytes
main:5 bytes
t:20 bytes
```

```
unsigned int  MyStrlen(char str[])
{
    int i;
    unsigned int len = 0;
    printf("message:%s\n", str);
    printf("MyStrlen:%d bytes\n", sizeof(str));
    for (i=0; str[i]!='\0'; i++)
    {
        len++;
    }
    return len;
}
```



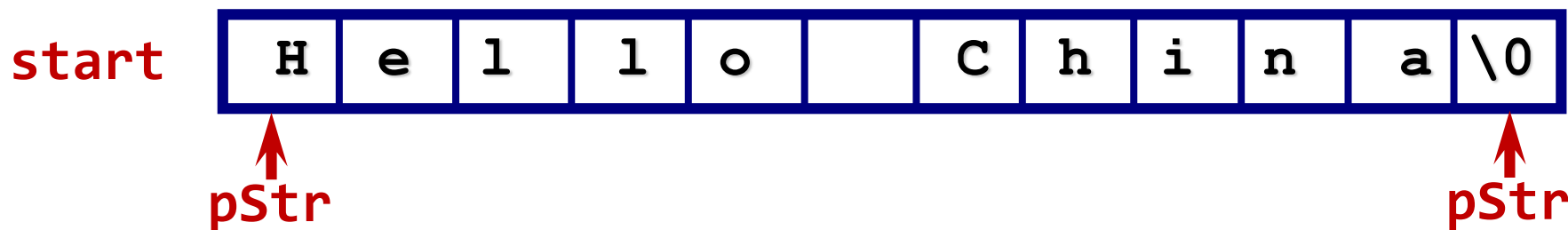
# 讨论

## ■ 这两个函数有区别吗？都能正确计算字符串长度吗？

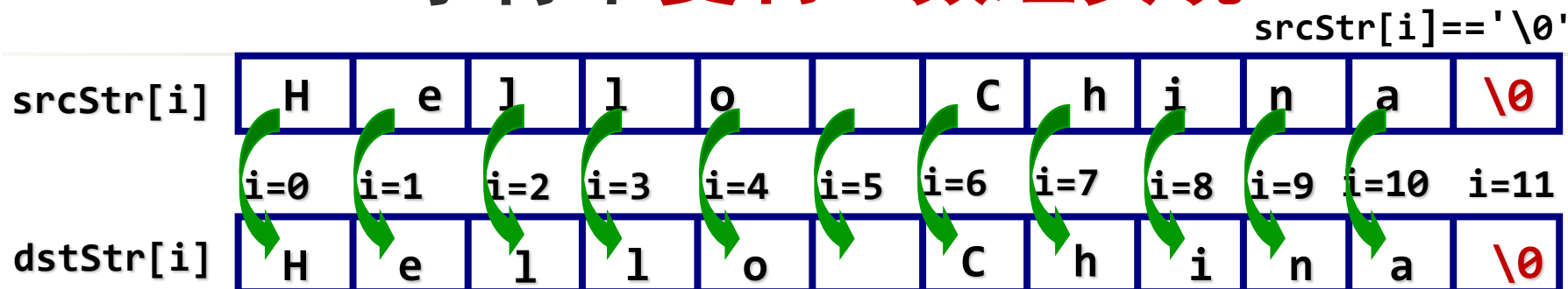
```
unsigned int MyStrlen(const char *pStr)
{
    const char *start = pStr;
    while (*pStr != '\0')
    {
        pStr++;
    }
    return pStr - start;
}
```

指针的作用：避免副本，共享数据  
**const**保护指针指向的内容不被修改

```
unsigned int MyStrlen(const char *pStr)
{
    const char *start = pStr;
    while (*pStr++ != '\0') //如何理解*pStr++?
    {
        ;
    }
    return pStr - start;
}
```



# 字符串复制：数组实现



```
void MyStrcpy(char dstStr[], const char srcStr[])
{
    int i = 0;
    while (srcStr[i] != '\0')
    {
        dstStr[i] = srcStr[i];
        i++;
    }
    dstStr[i] = '\0';
}
```

`dstStr`  `srcStr`;

**dstStr 须足够大!**

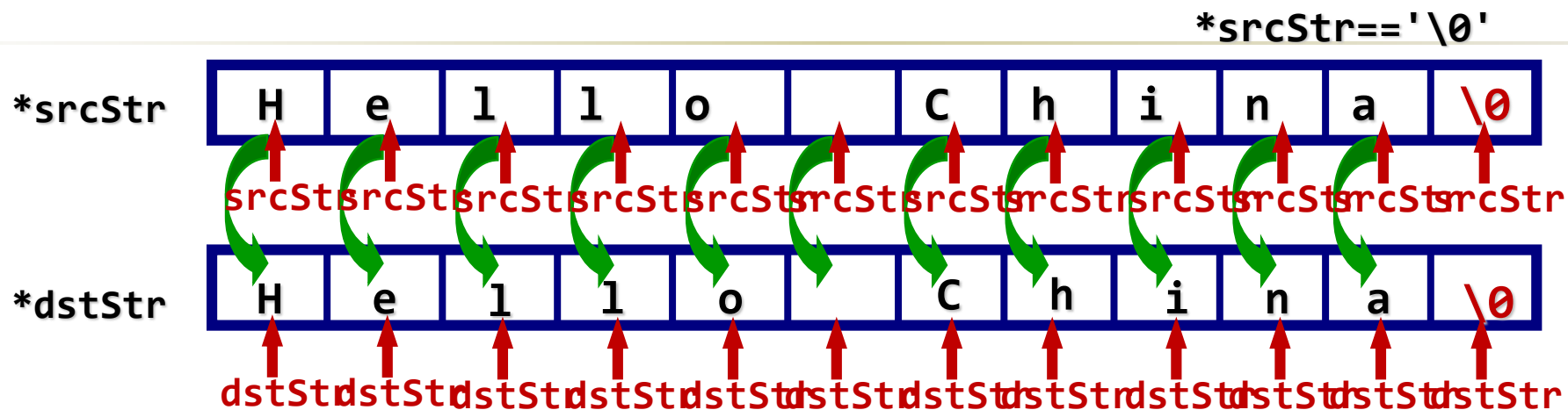
**n族函数确保不  
发生数组越界!**

```
void MyStrncpy(char dstStr[], const char srcStr[], int n)
{
    int i;
    for (i=0; i<n; i++)
    {
        dstStr[i] = srcStr[i];
    }
}
```

F:\c\test\bin\Debug\test.exe

```
12345
12
12345阳貌☐@
12345阳貌☐@
```

# 字符串复制：指针实现



```
void MyStrcpy(char dstStr[], const char srcStr[])
{
    int i = 0;
    while (srcStr[i] != '\0')
    {
        dstStr[i] = srcStr[i];
        i++;
    }
    dstStr[i] = '\0';
}
```



```
MyStrncpy(str, s, n);
puts(str);
```

```
void MyStrcpy(char *dstStr, const char *srcStr)
{
    while (*srcStr != '\0')
    {
        *dstStr = *srcStr;
        srcStr++;
        dstStr++;
    }
    *dstStr = '\0';
}
```

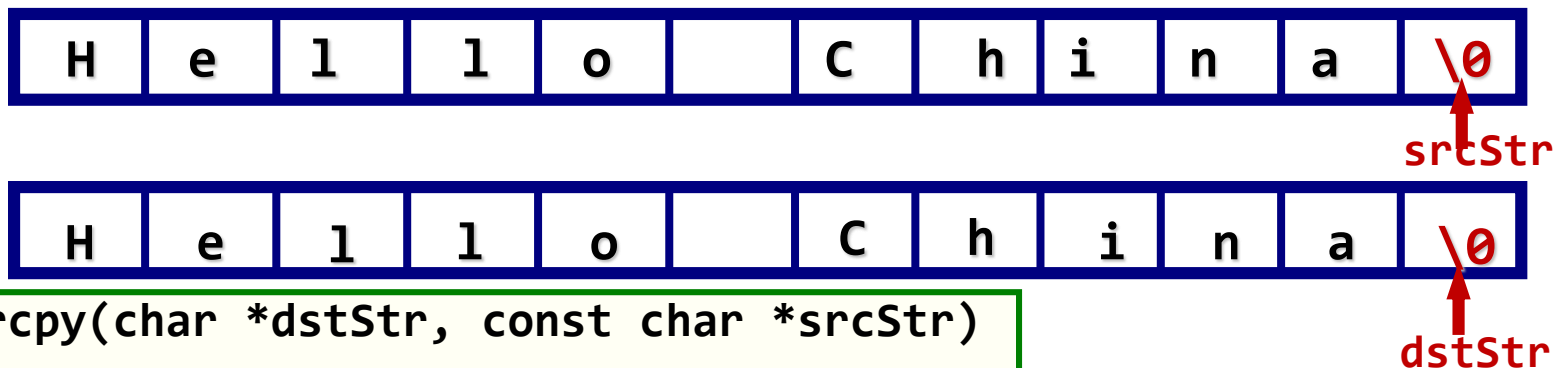
# 字符串复制：n族数组实现

srcStr[i]	H	e	l	l	o		C	h	i	n	a	\0
dstStr[i]	H	e	l	l	o		C	h	i	n	a	\0

```
char *MyStrcpy(char dstStr[], const char srcStr[])
{
    int i = 0;
    while (srcStr[i] != '\0')
    {
        dstStr[i] = srcStr[i];
        i++;
    }
    dstStr[i] = '\0';
    return dstStr;
}
```

```
char *MyStrncpy(char dstStr[], const char srcStr[], int n)
{
    int i;
    for (i=0; i<n; i++)
    {
        dstStr[i] = srcStr[i];
    }
    //dstStr[i] = '\0';
    return dstStr;
}
```

# 字符串复制：n族指针实现

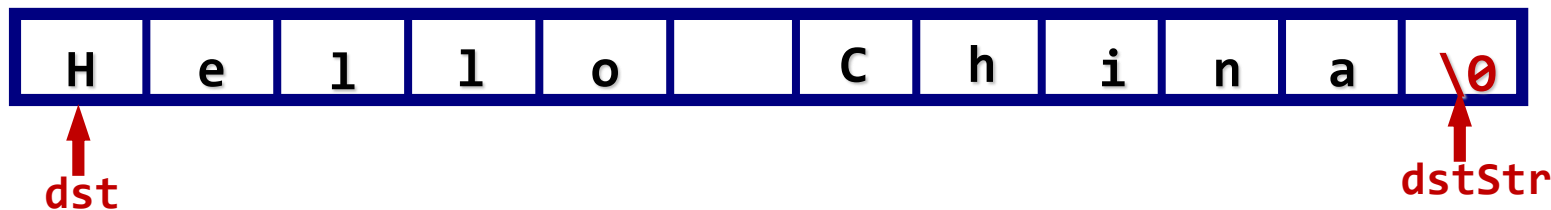


```
char *MyStrcpy(char *dstStr, const char *srcStr)
{
    while (*srcStr != '\0')
    {
        *dstStr++ = *srcStr++;
    }
    /*dstStr = '\0';
    return dstStr;
}
```

```
char *MyStrncpy(char *dstStr, const char *srcStr, int n)
{
    int i;
    for (i=0; i<n; i++)
    {
        *dstStr++ = *srcStr++;
    }
    /*dstStr = '\0';
    return dstStr;    //函数能返回目标字符串的指针吗?
}
```



# 字符串复制：n族指针实现



`puts(MyStrncpy(str, s, n));`

```
char *MyStrcpy(char *dstStr, const char *srcStr)
{
    char *dst = dstStr;
    while (*srcStr != '\0')
    {
        *dstStr++ = *srcStr++;
    }
    *dstStr = '\0';
    return dst;
}
```

↑  
`MyStrncpy(str, s, n);`  
`puts(str);`

```
char *MyStrncpy(char *dstStr, const char *srcStr, int n)
{
    int i;
    char *dst = dstStr; // 保存原来的目标字符串首地址
    for (i=0; i<n; i++)
    {
        *dstStr++ = *srcStr++;
    }
    *dstStr = '\0';
    return dst; // 函数返回字符指针的好处是什么?
}
```

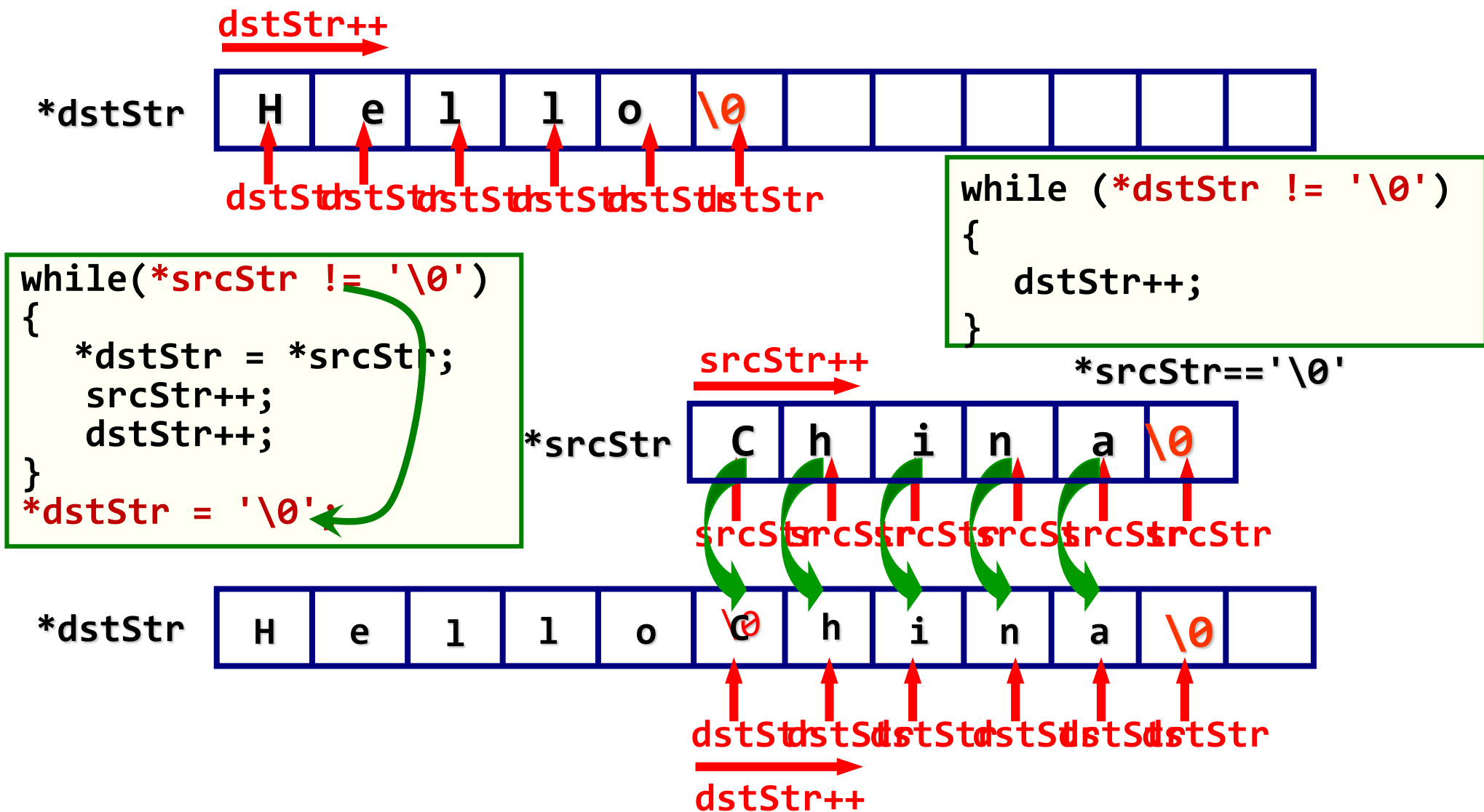
# 字符串连接

```
#include <stdio.h>
#define N 80
char * MyStrcat(char *dstStr, const char *srcStr);
int main()
{
    char first[2*N+1]; //应足够大
    char second[N+1];
    char *result = NULL;
    printf("Input the first string:");
    gets(first);
    printf("Input the second string:");
    gets(second);
    result = MyStrcat(first, second); //字符指针保存连接后字符串的首地址
    printf("The result is: %s\n", result);
    return 0;
}
```

H	e	l	l	o	\0	\0	\0	\0	\0	\0	\0
					C	h	i	n	a	\0	
H	e	l	l	o	C	h	i	n	a	\0	\0

```
strcat(third, strcat(first, second));
```

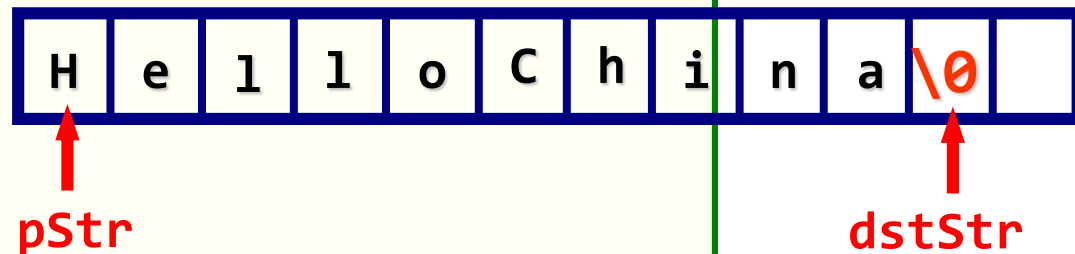
# 字符串连接



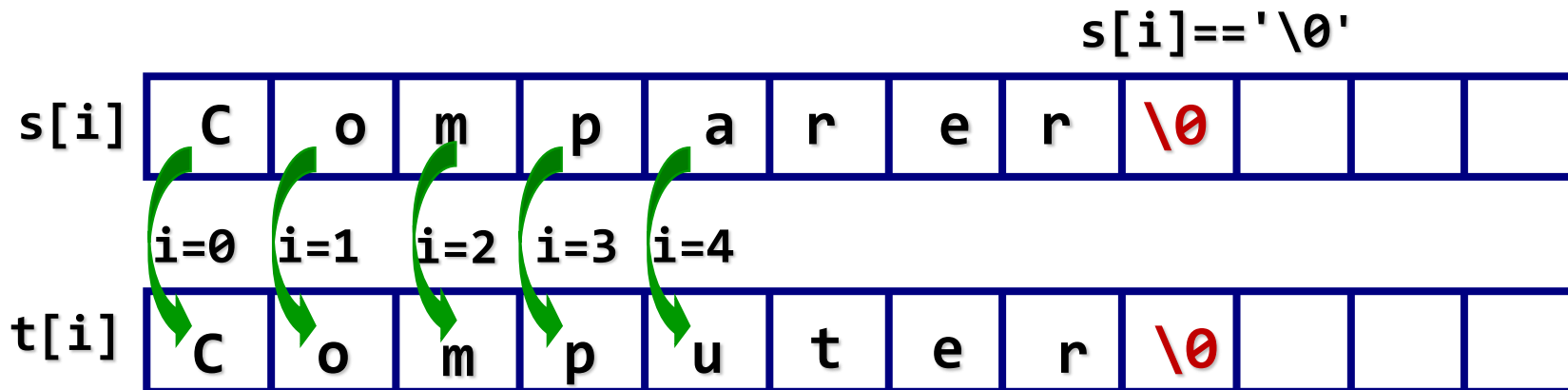


# 字符串连接


```
char *MyStrcat(char *dstStr, const char *srcStr)
{
    char *pStr = dstStr;
    while (*dstStr != '\0')
    {
        dstStr++;
    }
    while(*srcStr != '\0')
    {
        *dstStr = *srcStr;
        srcStr++;
        dstStr++;
    }
    *dstStr = '\0';
    return pStr; //返回字符串的首地址
}
```



# 字符串比较



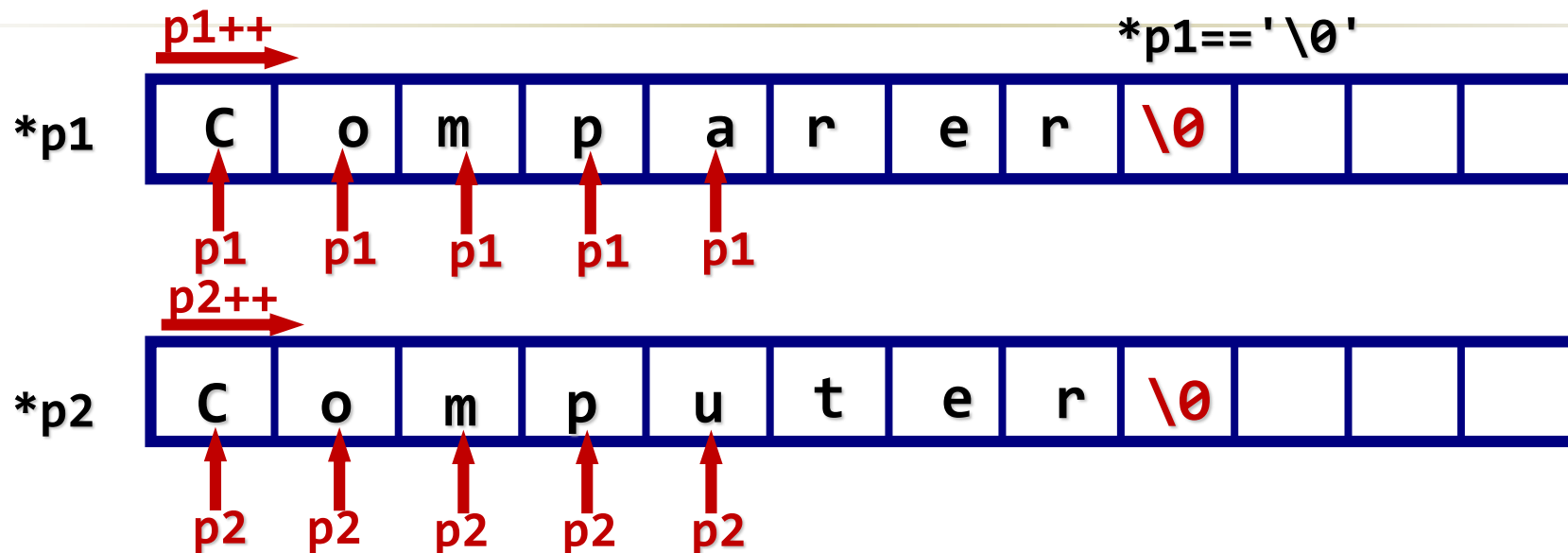
```
int MyStrcmp(char s[], char t[])
{
    int i;
    for (i=0; s[i] == t[i]; i++ )
    {
        if (s[i] == '\0') return 0 ;
    }
    return s[i] - t[i];
}
```

$\text{if (str1 < str2)}$  

$\text{if (strcmp(str1, str2) < 0)}$

$\text{if (strncmp(str1, str2, n) < 0)}$

# 字符串比较



```
int MyStrcmp(char s[], char t[])
{
    int i;
    for (i=0; s[i] == t[i]; i++ )
    {
        if (s[i] == '\0') return 0;
    }
    return s[i] - t[i];
}
```

```
int MyStrcmp(char *p1, char *p2)
{
    for (; *p1 == *p2; p1++, p2++)
    {
        if (*p1 == '\0') return 0;
    }
    return *p1 - *p2;
}
```

# 删除字符

- 用字符数组作函数参数编程，在从键盘输入的字符串（假设长度小于80）中删除与某字符相同的字符，然后输出删掉该字符后的字符串。

```
#include <stdio.h>
#include <string.h>
#define N 80
void Squeeze(char s[], char c);
int main()
{
    char str[N], ch;
    printf("Input a string:\n");
    gets(str);
    printf("Input a character:\n");
    ch = getchar();//scanf("%c",&ch);
    Squeeze(str, ch);
    printf("Results:%s\n", str);
    return 0;
}
```

```
void Squeeze(char s[], char c)
{
    int i, j;
    for (i=0,j=0; s[i]!='\0'; i++)
    {
        if (s[i] != c)
        {
            s[j] = s[i];
            j++;
        }
    }
    s[j] = '\0';
}
```

# 香港中文大学微情书一等奖：你还在我身旁

