



## § 5. 利用数组处理批量数据

### 5.1. 一维数组的定义和使用

★ 复习与回顾视频（时长69:29）

<https://www.bilibili.com/video/BV1Ea411F76j/>

### 5.2. 二维数组的定义和使用

★ 复习与回顾视频（时长 22:13）

<https://www.bilibili.com/video/BV1rK4y1j73E/>

### 5.3. 字符数组

★ 复习与回顾视频（时长 28:36）

<https://www.bilibili.com/video/BV1iy4y1q71r/>

### 5.4. 常用的字符串处理函数

### 5.5. C++处理字符串的方法 – 字符串类与字符串变量

★ 上课视频（时长 93:26）

<https://www.bilibili.com/video/BV1ip4y1r7ky/>

★ 课件：见后

★ 本页所列的四个视频，要求本周内学习完成（11.29止）



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 操作对象

一维字符数组表示的字符串

##### ★ 对应头文件

- 使用前加 `#include <string.h>` //C方式  
`#include <cstring>` //C++方式

◆ VS2019下可以不加这个头文件

- VS2019认为部分函数不安全，使用前需要加 `#define _CRT_SECURE_NO_WARNINGS`

##### ★ 常用字符串处理函数

- ① `strlen (const char s[]);`
  - ② `strcat (char dest[], const char src[]);`
  - ③ `strncat(char dest[], const char src[], const unsigned int len);`
  - ④ `strcpy (char dest[], const char src[]);`
  - ⑤ `strncpy(char dest[], const char src[], const unsigned int len);`
  - ⑥ `strcmp (const char s1[], const char s2[]);`
  - ⑦ `strncmp(const char s1[], const char s2[], const unsigned int len);`
- 更多的字符串处理函数通过作业完成并理解
  - 教材/参考资料/其它老师的课件中，很多形式是 `const char *s`，暂时忽略，待学习指针后再进一步理解
  - 先不要考虑这些函数的返回值，待学习指针后再进一步理解



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

##### ① strlen(const char s[])

功 能：求字符串的长度

输入参数：存放字符串的字符数组

返 回 值：整型值表示的长度

注意事项：返回第一个'\0'前的字符数量, 不含'\0'

**//例：字符数组与字符串长度**

**//读操作，不需要加\_CRT\_SECURE\_NO\_WARNINGS**

```
#include <iostream>
```

```
#include <cstring>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char str1[]="Hello";
```

```
    cout << sizeof(str1) << endl;
```

```
    cout << strlen(str1) << endl;
```

**6 数组长度**

**5 字符串长度**

```
    char str2[]="china\0Hello\0\0";
```

```
    cout << sizeof(str2) << endl;
```

```
    cout << strlen(str2) << endl;
```

**14 数组长度**

**5 字符串长度**

```
    return 0;
```

```
}
```



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

##### ② strcat(char dst[], const char src[])

功 能：将字符串src连接到字符串dest的尾部

输入参数：存放字符串dest的字符数组dest

存放字符串src的字符数组src (只读)

返 回 值：改变后的字符数组dest

注意事项：字符数组dest要有足够的空间 (两串总长+1)

//例：字符串连接

#define \_CRT\_SECURE\_NO\_WARNINGS //VS2019需要

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

char str1[30]="Tongji "; //不能缺省，至少18字节!!!

char str2[]="University"; //缺省长度为11

cout << strcat(str1, str2) << endl;

return 0;

}

cout输出为strcat函数的返回值，  
也证明了返回值是第1个字符数组

输出为：  
Tongji University

//例：字符串连接

#define \_CRT\_SECURE\_NO\_WARNINGS //VS2019需要

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

char str1[]="Tongji ";

char str2[]="University"; //缺省长度为11

cout << strcat(str1, str2) << endl;

return 0;

}

缺省为8

观察输出是否有错



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

③ `strncat(char dest[], const char src[], const unsigned int n)`

功 能：将字符串src的**前n个字符**连接到字符串dest的尾部

输入参数：存放字符串dest的字符数组dest

存放字符串src的字符数组src**(只读)**

要复制的长度n**(只读，如果n超过src长度，则只连接src个)**

返 回 值：改变后的字符数组dest

注意事项：字符数组dest要有足够的空间**(原dest长度+n+1)**

**//例：字符串连接前n个字符**

`#define _CRT_SECURE_NO_WARNINGS` **//VS2019需要**

`#include <iostream>`

`#include <cstring>`

`using namespace std;`

`int main()`

`{`

`char str1[30]="Tongji ";`

`char str2[30]="Tongji ";`

`char str3[]="University";` **//缺省长度为11**

`cout << strncat(str1, str3, 3) << '*' << endl;`

`cout << strncat(str2, str3, 300) << '*' << endl;`

`return 0;`

`}`

**输出为：**

**Tongji Uni\***

**Tongji University\***



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

##### ④ strcpy(char dest[], const char src[])

功 能：将字符串src复制到字符串dest中, 覆盖原dest串

输入参数：存放字符串dest的字符数组dest

存放字符串src的字符数组src (只读)

返 回 值：改变后的字符数组dest

注意事项：字符数组dest要有足够的空间 (串src长+1)

● 字符串复制时包含' \0'，到' \0' 为止

//例：字符串拷贝

#define \_CRT\_SECURE\_NO\_WARNINGS //VS2019需要

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

int i;

char a[]="student", b[]="hello";

strcpy(a, b);

cout << a << endl;

for(i=0;i<8;i++)

cout << (int)a[i] << ' ';

cout << endl;

return 0;

}

a/b数组的缺省长度8/6

复制前a	s	t	u	d	e	n	t	\0
复制后a	h	e	l	l	o	\0	t	\0

复制到\0为止  
a[6]以后保持原值

输出为：  
hello  
104 101 108 108 111 0 116 0

假设2:

char a[]="student", b[]="hellochina";

1、为什么错?

2、仅修改a的定义使正确, 如何做?

假设1:

char a[]="student", b[]="hello\0china";

则: a/b数组的缺省长度8/12

但结果与本例完全相同,  
复制的字符个数也相同



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

##### ⑤ strncpy(char dest[], const char src[], unsigned int n)

功 能：将字符串src的**前n个**复制到字符串dest中, 覆盖原dest串

输入参数：存放字符串dest的字符数组dest

存放字符串src的字符数组src (只读)

继续复制导致出错

要复制的长度n (只读, 如果n超过src长度, 则只复制src个)

返 回 值：改变后的字符数组dest

n+1

注意事项：字符数组dest要有足够的空间 (**min(串src长, n)+1**)

● strncpy复制时不包含'\0', 且长度超过时出错, 要人为保证n正确

**//例：字符串拷贝前n个字符**

#define \_CRT\_SECURE\_NO\_WARNINGS //VS2019需要

#include <iostream>

#include <cstring>

using namespace std;

int main()

{ int i;

char a[]="student", b[]="hello";

strncpy(a, b, 2);

cout << a << endl;

for(i=0; i<8; i++)

cout << (int)a[i] << ' ';

cout << endl;

return 0;

}

输出为:

heudent

104 101 117 100 101 110 116 0

证明了未加\0

**//例：字符串拷贝前n个字符**

#define \_CRT\_SECURE\_NO\_WARNINGS //VS2019需要

#include <iostream>

#include <cstring>

using namespace std;

int main()

{ int i;

char a[]="student", b[]="hello";

strncpy(a, \_\_\_\_\_, 2);

cout << a << endl;

for(i=0; i<8; i++)

cout << (int)a[i] << ' ';

cout << endl;

return 0;

}

提示：数组名代表数组的首地址  
即：b ⇔ &b[0]

如果想从b[2]开始复制  
2个字符到a中, 如何做?  
即期望输出: lludent



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

##### ⑤ strncpy(char dest[], const char src[], unsigned int n)

功 能：将字符串src的**前n个**复制到字符串dest中, 覆盖原dest串

输入参数：存放字符串dest的字符数组dest

存放字符串src的字符数组src(只读)

继续复制导致出错

要复制的长度n(只读, 如果n超过src长度, 则只复制src个)

返 回 值：改变后的字符数组dest

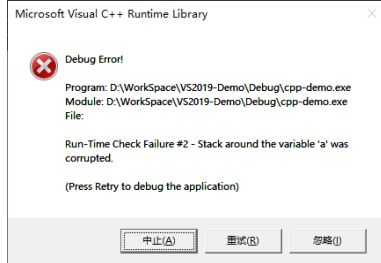
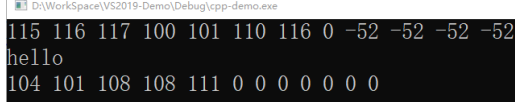
n+1

注意事项：字符数组dest要有足够的空间(**min(串src长, n)+1**)

● strncpy复制时不包含'\0', 且长度超过时出错, 要人为保证n正确

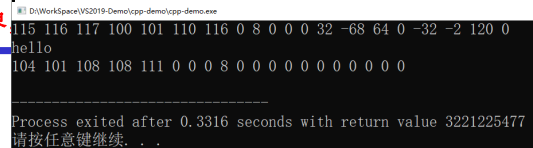

深度讨论

```
#define _CRT_SECURE_NO_WARNINGS //VS2019需要
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int i;
    char a[] = "student", b[] = "hello";
    for (i = 0; i < 12; i++) //12已越界, 目的?
        cout << (int)a[i] << ' ';
    cout << endl;
    strncpy(a, b, 200);
    cout << a << endl;
    for (i = 0; i < 12; i++) //12已越界, 目的?
        cout << (int)a[i] << ' ';
    cout << endl;
    return 0;
}
```



VS输出为：上图+弹窗

```
#define _CRT_SECURE_NO_WARNINGS //VS2019需要
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int i;
    char a[] = "student", b[] = "hello";
    for (i = 0; i < 12; i++) //12已越界
        cout << (int)a[i] << ' ';
    cout << endl;
    strncpy(a, b, 200);
    cout << a << endl;
    for (i = 0; i < 12; i++) //12已越界, 目的?
        cout << (int)a[i] << ' ';
    cout << endl;
    return 0;
}
```



Dev输出为：上图  
正确吗？怎么体现错？  
12换20或更大数试试





## § 5. 利用数组处理批量数据

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

##### ⑥ strcmp(const char s1[], const char s2[])

功 能：比较字符串s1和字符串s2的大小

输入参数：存放字符串s1的字符数组s1 (只读)

存放字符串s2的字符数组s2 (只读)

返 回 值：整型值(0:相等 >0:串1大 <0:串1小)

```
#include <iostream>
using namespace std;
int main()
{
    char str1[] = "house", str2[] = "horse";
    char str3[] = "abcd", str4[] = "abcde";
    char str5[] = "abcd", str6[] = "abc";
    char str7[] = "abcd", str8[] = "abcd";
    char str9[] = "abcd", str10[] = "abcd\0efgh";
    cout << strcmp(str1, str2) << endl;
    cout << strcmp(str3, str4) << endl;
    cout << strcmp(str5, str6) << endl;
    cout << strcmp(str7, str8) << endl;
    cout << strcmp(str9, str10) << endl;
} //仅比较无赋值，因此不需加_CRT_SECURE_NO_WARNINGS
```

1	串1>串2
-1	<
1	>
0	==
0	==

```
#include <iostream>
using namespace std;
```

另一种输出形式:  
串1 < 串2

```
int main()
{
    char str1[]="abcd", str2[]="abcde";
    int k = strcmp(str1, str2);
    if (k==0)
        cout << "串1 = 串2" << endl;
    else if (k<0)
        cout << "串1 < 串2" << endl;
    else
        cout << "串1 > 串2" << endl;

    return 0;
}
```



## § 5. 利用数组处理批量数据

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

⑥ strcmp(const char s1[], const char s2[])

● 两串相等的条件是长度相等且对应位置字符的ASCII码值相等

● 字符串的比较流程如下：两串首字符对应比较，若**不等则返回非0**，若相等则继续比较下一个字符，重复到两串对应字符均为'\0'则结束比较，**返回0(相等)**

strcmp("house", "horse"); 到第3个字符结束

strcmp("abcd", "abcde"); 到第5个字符结束

strcmp("abcd", "abc"); 到第4个字符结束

strcmp("abcd", "abcd"); 到第5个字符结束

● 不相等返回时，有些系统返回-1/1，有些系统返回第一个不相等字符的ASCII差值，因此一般不比较具体值，而只是判断 >0 / <0 / ==0

strcmp("house", "horse"); VS2019/DevC++编译器返回1  
某些编译器可能返回3



## § 5. 利用数组处理批量数据

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

##### ⑥ strcmp(const char s1[], const char s2[])

- 不能直接用比较运算符比较字符串的大小 (但直接用比较运算符语法不错, 只是含义不同)

<pre>#include &lt;iostream&gt; using namespace std;  int main() {   char str1[]="house", str2[]="horse";     int k;     k = str1 &lt; str2;     cout &lt;&lt; k &lt;&lt; endl;     return 0; }</pre>	输出: ?	<pre>#include &lt;iostream&gt; using namespace std;  int main() {   char str1[]="horse", str2[]="house";//互换     int k;     k = str1 &lt; str2;     cout &lt;&lt; k &lt;&lt; endl;     return 0; }</pre>	输出: ?
<pre>#include &lt;iostream&gt; using namespace std;  char str1[]="house", str2[]="horse"; int main() {   int k;     k = str1 &lt; str2;     cout &lt;&lt; k &lt;&lt; endl;     return 0; }</pre>	输出: ?	<pre>#include &lt;iostream&gt; using namespace std;  char str1[]="horse", str2[]="house";//互换 int main() {   int k;     k = str1 &lt; str2;     cout &lt;&lt; k &lt;&lt; endl;     return 0; }</pre>	输出: ?

问题:

- 1、四个例子输出分别是什么? 为什么?
- 2、为什么局部和全局的定义, 结果相反?



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.1. C方式的字符串处理函数

##### ★ 常用字符串处理函数

⑦ `strncmp(const char s1[], const char s2[], const unsigned int n)`

功 能：比较字符串s1和字符串s2的前n个字符的大小

输入参数：存放字符串s1的字符数组s1 (只读)

存放字符串s2的字符数组s2 (只读)

要比较的长度n (只读, n/s1长/s2长三者关系有影响吗?)

返 回 值：整型值(0:相等 >0:串1大 <0:串1小)

● n大于短串长度(短串长度+1)时，一定会结束

//例：字符串比较前n个字符，不需加\_CRT\_SECURE\_NO\_WARNINGS

```
#include <iostream>
```

```
#include <cstring>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char str1[] = "abcd", str2[] = "abcde";
```

```
    cout << strncmp(str1, str2, 3) << endl; //1~4均可
```

```
    cout << strncmp(str1, str2, 5) << endl;
```

```
    cout << strncmp(str1, str2, 100) << endl;
```

```
    return 0;
```

```
}
```

0

-1

-1



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.2. C方式的字符处理函数

- ① isdigit(char ch) //判断是否数字(0-9)
- ② isalpha(char ch) //判断是否字母(A~Z, a-z)
- ③ isalnum(char ch) //判断是否字母或数字(A~Z, a-z, 0-9)
- ④ isxdigit(char ch) //判断是否16进制数字(0-9, A-F, a-f)
- ⑤ isspace(char ch) //判断是否空格(含回车/换行/换页/tab/竖向tab)
- ⑥ islower(char ch) //判断是否小写字母(a-z)
- ⑦ isupper(char ch) //判断是否大写字母(A-Z)
- ⑧ tolower(char ch) //大写转小写, 其余原值返回
- ⑨ toupper(char ch) //小写转大写, 其余原值返回

★ 需要包含头文件<ctype.h>或<cctype>

这些判断函数满足条件返回非零, 不满足返回0  
具体的返回值涉及到位运算概念, 暂时忽略(荣誉)

//实现也相对比较简单, 不再深入讨论

```
int isdigit(int ch)
{
    return (ch>='0' && ch<='9') ? 1 : 0;
}

int tolower(int ch)
{
    return (ch>='A' && ch<='Z') ? ch+32 : ch;
}
```



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.3. C方式的字符串与格式化输入输出

##### 5.4.3.1. 将格式化数据输出到字符串中

#### ★ 标准输出函数

`int printf("格式串", 输出表列);`

#### ★ 向字符串输出函数

`int sprintf(字符数组, "格式串", 输出表列);`

- 返回值是输出字符的个数
- 字符数组要有足够空间容纳输出的数据
- 格式串同printf
- VS2019下需加 `#define _CRT_SECURE_NO_WARNINGS`

#### //例: sprintf的基本使用

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
using namespace std;
```

```
int main()
{
```

```
    char str[80];
    int k=123, ret;
    double pi=3.1415925;
```

```
    ret = sprintf(str, "k=%-4d*pi=%.2f#", k, pi);
    printf("ret : %d\n", ret);
    printf("str : %s\n", str);
```

```
    return 0;
```

```
}
```

Microsoft Visual Studio 调试控制台

```
ret : 15
str : k=123 *pi=3.14#
```



## § 5. 数组

### 5. 4. 常用的字符串处理函数

#### 5. 4. 3. C方式的字符串与格式化输入输出

##### 5. 4. 3. 2. 从字符串中输入格式化数据

#### ★ 标准输入函数

`int scanf("格式串", 输入地址表列);`

#### ★ 从字符串中输入函数

`int sscanf(字符串数组, "格式串", 输入地址表列);`

- 返回值是正确读入的输入数据的个数
- 格式串同scanf
- VS2019下需加 `#define _CRT_SECURE_NO_WARNINGS`

**//例: sscanf的基本使用**

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char str[80] = "Hello 123 11.2", s[10];
```

```
    int i, ret;
```

```
    double d;
```

```
    ret = sscanf(str, "%s %d %lf", s, &i, &d);
```

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

```
    printf("s=%s i=%d d=%f\n", s, i, d);
```

```
    return 0;
```

```
}
```

Microsoft Visual Studio 调试控制台

```
ret : 3
```

```
s=Hello i=123 d=11.200000
```



## § 5. 数组

### 5.4. 常用的字符串处理函数

#### 5.4.3. C方式的字符串与格式化输入输出

例：若已知变量day要输出的数据宽度为n(变量)，如何输出？(多个作业中)

答：C++方式：cout << setw(n) << day << endl;

C方式：

```
switch(n) {
    case 6:
        printf("%6d", day);
        ...;
    case 8:
        printf("%8d", day);
        ...;
}
```

//因为%后的宽度要求常量

```
if (n==6) {
    if (day < 10)
        printf("      %d", day);
    else
        printf("      %d", day);
}
else if (n==8) {
    ...;
}
```

//例：用sprintf将变量转常量后用于printf格式串

```
#define _CRT_SECURE_NO_WARNINGS
```

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

```
int main()
```

```
{
```

```
    char fmt[16];
```

```
    int d1 = 3, d2 = 17, n;
```

```
    printf("01234567890123456789\n");
```

```
    n = 6;
```

```
    sprintf(fmt, "%%%dd\n", n); //%%=>%, %d=>6
```

```
    printf(fmt, d1);
```

```
    printf(fmt, d2);
```

```
    n = 8;
```

```
    sprintf(fmt, "%%%dd\n", n);
```

```
    printf(fmt, d1);
```

```
    printf(fmt, d2);
```

```
    return 0;
```

```
}
```

Microsoft Visual Studio 调试控制台

```
01234567890123456789
      3
     17
          3
         17
```





## § 5. 数组

### 5. 4. 常用的字符串处理函数

#### 5. 4. 4. 用C++方式的cin/cout成员函数处理字符串

★ cin.get(); //多种形式

★ cin.getline();

后期完成相应的作业并理解其中的差异



## § 5. 利用数组处理批量数据

5.5. C++处理字符串的方法 - 字符串类与字符串变量

5.5.1. 用一维字符数组来表示字符串变量的不足

★ 字符串的长度受限于数组定义时的大小

=> 数组定义过大导致浪费

=> 数组定义过小导致不够

=> 数组定义的大小不能动态变化

★ 赋值、复制、连接等必须用专用函数

5.5.2. 字符串类(**string**类)的引入

string类是C++引入的字符串处理的新方法，通过定义类及运算符重载的方式实现



## § 5. 利用数组处理批量数据

### 5.5. C++处理字符串的方法 – 字符串类与字符串变量

#### 5.5.3. string类简单变量的定义和使用（与C方式对比）

项目	字符串变量	字符数组
适用	C++	C、C++
头文件	#include <string>	#include <string.h> #include <cstring>
定义	string 变量名 string s1;	char 变量名 char s1[10];
定义时赋初值	string s1="hello"; 长度无限制	char s1[10]="hello"; 长度不超过9
赋值	string s1; s1="hello"; 长度无限制 string s1="hello", s2; s2=s1;	char s1[10]; strcpy(s1, "hello"); 长度不超过9 char s1[10]="hello", s2[10]; strcpy(s2, s1);
单字符操作	string s1="hello"; s1[2]='p';	char s1[10]="hello"; s1[2]='p';
输入	cin	cin、scanf
输出	cout	cout、printf
字符串复制	string s1, s2; ... s1=s2; 不必考虑溢出	char s1[10], s2[10]; ... strcpy(s1, s2); 防止溢出
字符串连接	string s1, s2; ... s1=s1+s2; 不必考虑溢出	char s1[20], s2[10]; ... strcat(s1, s2); 防止溢出
	注: s1+s2 ≠ strcat; s1=s1+s2/s1+=s2 ⇔ strcat	
字符串比较	用比较运算符 s1==s2; s1>s2; s1>=s2;	用函数 if (!strcmp(s1, s2)) if (strcmp(s1, s2)>0) if (strcmp(s1, s2)>=0)



## § 5. 利用数组处理批量数据

### 5.5. C++处理字符串的方法 - 字符串类与字符串变量

#### 5.5.3. string类简单变量的定义和使用（与C方式对比）

//观察string的所占空间及存放的字符串长度的关系

```
#include <iostream>
using namespace std;
int main()
{
    string s1 = "abc";
    string s2 = "abcdefghijklmnopqrstuvwxyz0123456789";
    cout << sizeof(s1) << ' ' << s1.length() << endl;
    cout << sizeof(s2) << ' ' << s2.length() << endl;
    for (int i = 0; i < 10000; i++)
        s1 += s2;
    cout << sizeof(s1) << ' ' << s1.length() << endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
28 3
28 36
28 360003
```

- ★ string变量存放的字符串的长度会自动调整，不是固定大小
- ★ string变量的sizeof大小与存放串的长度无关，涉及到一种新的内存存储和分配方式(动态内存的申请与释放-荣誉课)
- ★ string变量能用+、=、>等运算符来实现连接、赋值、比较等运算，具体的实现原理及实现方法待后续内容(运算符的重载-荣誉课)学习完成后再进一步了解
- ★ 在本节中仅需要了解与一维字符数组表示字符串在表示及使用方法的差异即可
- ★ 除特定题目外，string类本学期都不准使用



## § 5. 利用数组处理批量数据

### 5.5. C++处理字符串的方法 - 字符串类与字符串变量

#### 5.5.4. string类数组的定义和使用（与C方式对比）

##### ★ 形式

```
string 数组名[正整型常量表达式];  
string name[5];
```

##### ★ 含义

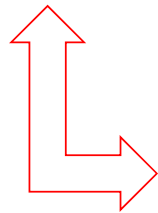
数组有若干元素，每个元素是一个字符串变量

##### ★ 定义时赋初值

```
string name[5]={"Zhang", "Li", "Wang", "Lin", "Zhao"};
```

##### ★ 与二维字符数组的内存表示比较(定义时赋初值)

```
char str[3][30]={"Zhang", "Li", "Wang"};  
string name[3]={"Zhang", "Li", "Wang"};
```



Z	h	a	n	g	\0	.....	\0
L	i	\0	\0	\0	\0	.....	\0
W	a	n	g	\0	\0	.....	\0

str占用连续的90个字节

Z	h	a	n	g
L	i			
W	a	n	g	

name[0]、name[1]、name[2]  
分别占用连续的5、2、4个字节，但整体上不保证连续



## § 5. 利用数组处理批量数据

5.5. C++处理字符串的方法 - 字符串类与字符串变量

5.5.4. string类数组的定义和使用（与C方式对比）

★ 与二维字符数组的比较 (通过键盘输入、通过strcpy/赋值语句赋值)

```
char str[3][30];  
string name[3];  
for (i=0; i<3; i++) {  
    cin >> str[i];  
    cin >> name[i];  
}
```

通过键盘输入赋值

假设键盘输入:

Zhang Zhang Li Li Wang Wang

```
char str[3][30];  
string name[3];  
strcpy(str[0], "Zhang");  
strcpy(str[1], "Li");  
strcpy(str[2], "Wang");  
name[0]="Zhang";  
name[1]="Li";  
name[2]="Wang";
```

通过语句赋值