第五章 程序设计复合类型

模块5.3：string类型
（C++风格的字符串）

主讲教师：同济大学电子与信息工程学院 陈宇飞
同济大学电子与信息工程学院 龚晓亮

目录

- C++字符串初始化
- C++字符串赋值、合并和附加操作
- string类的其他操作
- string类的I/O
- 其他形式的字符串字面值
- 字符串使用专用函数进行操作

目录

1.1 C++字符串初始化

- ✓ 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

1.1 C++字符串初始化

- ✓ ISO/ANSI C++98标准通过添加string类扩展了C++库，使用string类型的变量（对象）而不是字符串数组来存储字符串
- ✓ 要使用string类，必须在程序中包含头文件string
- ✓ string类位于命名空间std中，需要使用using编译指令，或std::string
- ✓ string类定义隐藏了字符串的数组性质，从而可以像变量那样处理字符串

1.1 C++字符串初始化

- ✓ string类数组的定义和使用
  - ❖ 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" };
```

1.1 C++字符串初始化

- ✓ string类数组的定义和使用
  - ❖ 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 str2[3] = { "Zhang", "Li", "Wang" };
```

str占用连续的90个字节

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

str2[0]占5个字节、
str2[1]占2个字节、
str2[2]占4个字节，
但整体上不保证连续

Z	h	a	n	g
L	i			
W	a	n	g	

1.1 C++字符串初始化

- ✓ C++允许将列表初始化用于C风格字符串和string对象

```
char first_date[] ={"Le Chapon Dodu"}; // initialized array
char second_date[] {"The Elegant Plate"}; // initialized array
string third_date = {"The Bread Bowl"}; // initialized string
string fourth_date {"The Bistro Francais"}; // initialized string
```

```
// strtype1.cpp -- using the C++ string class
#include <iostream>
#include <string>      // make string class available
int main()
{
    using namespace std;
    char charr1[20];              // create an empty array
    char charr2[20] = "jaguar"; // create an initialized array
    string str1;                  // create an empty string object
    string str2 = "panther";     // create an initialized string
    cout << "Enter a kind of feline: ";
    cin >> charr1;
    cout << "Enter another kind of feline: ";
    cin >> str1;                          // use cin for input
    cout << "Here are some felines:\n";
    cout << charr1 << " " << charr2 << " " << str1 << " " << str2  << endl;
    // use cout for output
    cout << "The third letter in " << charr2 << " is " << charr2[2] << endl;
    cout << "The third letter in " << str2 << " is "<< str2[2] << endl;
    // use array notation
    return 0;
}
```

```
Microsoft Visual Studio 调试控
Enter a kind of feline: ocelot
Enter another kind of feline: tiger
Here are some felines:
ocelot jaguar tiger panther
The third letter in jaguar is g
The third letter in panther is n
```

1.1 C++字符串初始化

- ✓ string对象与字符数组的相同点：
  - ❖ 可以使用C风格字符串来初始化string对象
  - ❖ 可以使用cin来将键盘输入存储到string对象中
  - ❖ 可以使用cout来显示string对象
  - ❖ 可以使用数组表示方法来访问存储在string对象中的字符

1.1 C++字符串初始化

✓ string对象与字符数组的不同点：

❖ 可以将string对象声明为简单变量，而不是数组

```
string str1;                // create an empty string object
string str2 = "panther";    // create an initialized string
```

❖ 类设计让程序能够自动处理string的大小

```
cin >> str1; // str1 resized to fit input
```

❖与使用数组相比，使用string对象更方便，也更安全

目录

- C++字符串赋值、合并和附加操作

2.1 C++字符串赋值

- ✓ 使用string类时，某些操作比使用数组时更简单
- ✓ 可以将一个string对象赋给另一个string对象（但不能将一个数组赋给另个一数组）

```
char charr1[20];              // create an empty array
char charr2[20] = "jaguar"; // create an initialized array
string str1;                  // create an empty string object
string str2 = "panther";    // create an initialized string
charr1 = charr2;
//error: incompatible types in assignment of 'char [20]' to 'char [20]'
str1 = str2;  //valid, str1 is a string object and str2 is a string object
```

2.1 C++字符串赋值

✓ 使用string类时，某些操作比使用数组时更简单

通过提示及时发现错误，并解决

```
char charr1[20];
char charr2[20] = "jagukkkkkkkkkkkkkkkkkkkkar";
string str1;
string str2 = "jagukkkkkkkkkkkkkkkkkkkkar";
charr1 = charr2;
str1 = str2;
```

(局部变量) char charr1[20]
联机搜索
未初始化本地变量。
显示可能的修补程序 (Alt+Enter或Ctrl+.)

(局部变量) char charr1[20]
联机搜索
表达式必须是可修改的左值
联机搜索

"const char [23]" 类型的值不能用于初始化 "char [20]" 类型的实体
联机搜索

2.2 C++字符串合并和附加

- ✓ 使用string类简化了字符串合并操作
- ✓ 可以使用运算符+将两个string对象合并起来
- ✓ 可以使用运算符+=将字符串附加到string对象的末尾

```
string str3;
str3 = str1 + str2; // assign str3 the joined string of str1 and str2
str1 += str2; // add str2 to the end of str1
```

```cpp
// strtype2.cpp -- assigning, adding, and appending
#include <iostream>
#include <string>       // make string class
available
int main()
{
    using namespace std;
    string s1 = "penguin";
    string s2, s3;

    cout << "You can assign one string "
         "object to another: s2 = s1\n";
    s2 = s1;
    cout << "s1: " << s1 << ", s2: " << s2 << endl;
    cout << "You can assign a C-style string "
         "to a string object.\n";
    cout << "s2 = \"buzzard\"\n";
    s2 = "buzzard";
    cout << "s2: " << s2 << endl;
    cout << "You can concatenate strings: "
         "s3 = s1 + s2\n";
    s3 = s1 + s2;
    cout << "s3: " << s3 << endl;
    cout << "You can append strings.\n";
    s1 += s2;
    cout << "s1 += s2 yields s1 = "
         << s1 << endl;
    s2 += " for a day";
    cout << "s2 += \" for a day\" yields
s2 = " << s2 << endl;

    return 0;
}
```

```
Microsoft Visual Studio 调试控
You can assign one string object to another: s2 = s1
s1: penguin, s2: penguin
You can assign a C-style string to a string object.
s2 = "buzzard"
s2: buzzard
You can concatenate strings: s3 = s1 + s2
s3: penguinbuzzard
You can append strings.
s1 += s2 yields s1 = penguinbuzzard
s2 += " for a day" yields s2 = buzzard for a day
```

目录

- string类的其他操作

3.1 string类的其他操作

✓对于C-风格字符串，头文件cstring（以前为string.h）提供了很多专用函数，例如：

- ❖ 函数strcpy()将字符串复制到字符数组中
- ❖ 使用函数strcat()将字符串附加到字符数组末尾
- ❖ 使用strlen()接受一个C-风格字符串作为参数，并返回该字符串包含的字符数

```
strcpy(charr1,charr2); // copy charr2 to charr1
strcat(charr1,charr2); // append contents of charr2 to charr1
strncpy() strncat() //第三个参数指出最大允许长度
int len2 = strlen(charr1); // obtain length of charr1
```

3.1 string类的其他操作

✓处理string对象的语法通常比使用C-风格字符串函数的简单，尤其是执行比较复杂的操作时

```
str3=str1+str2;
等价于：strcpy(charr3,charr1); strcat(charr3,charr2);
```

✓使用字符数组时，总是存在目标数组过小，无法存储指定信息的危险

```
char site[10] = "house";
strcat(site," of pancakes");  //目标数组过小
```

✓确定字符串中字符数的方法

```
int len1 = str1.size(); // str1是一个对象，而size()是一个类方法
int len2 = strlen(charr1); // strlen()接受一个C-风格字符串作为参数
```

```
// strtype3.cpp -- more string class features
#include <iostream>
#include <string> // make string class available
#include <cstring> // C-style string library

int main()
{
    using namespace std;

    char charr1[20];
    char charr2[20] = "jaguar";
    string str1;
    string str2 = "panther";
    // assignment for string objects and character arrays
    str1 = str2;              // copy str2 to str1
    strcpy(charr1, charr2); // copy charr2 to charr1

    // appending for string objects and character arrays
    str1 += " paste";     // add paste to end of str1
    strcat(charr1, " juice");   // add juice to end of charr1

    // finding the length of a string object and a C-style string
    int len1 = str1.size();    // length of str1
    int len2 = strlen(charr1); // length of charr1

    cout << "The string " << str1 << " contains "
         << len1 << " characters.\n";
    cout << "The string " << charr1
      << " contains " << len2 << " characters.\n";
    return 0;
}
```

目录

- string类I/O

4.1 string类I/O

✓用cin对string类的输入和字符数组的输入是一样的

```
cin >> str;

cin >> arr;
```

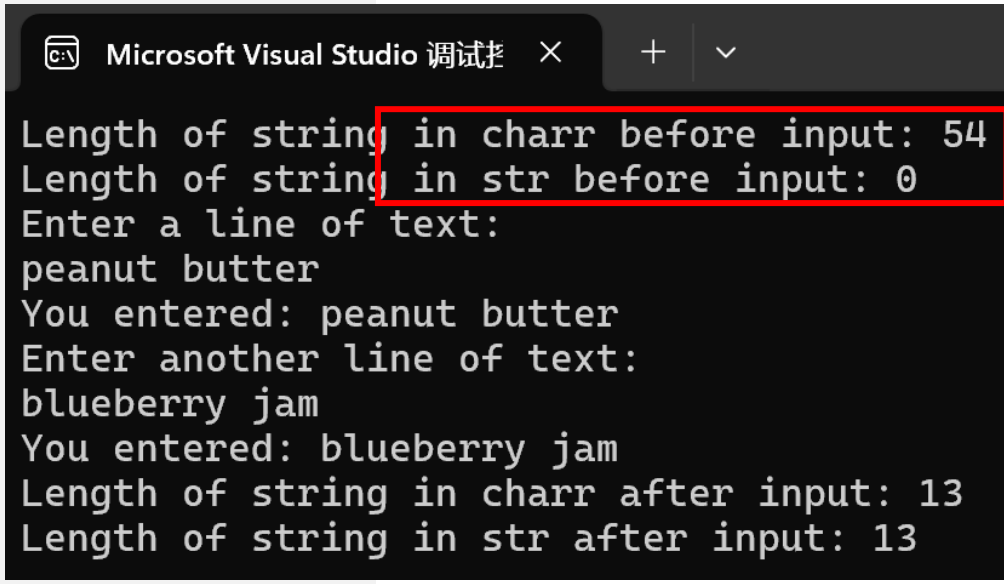
✓不一样的是读取一行字符串的时候string类的句法是

```
getline(cin,str);
```

```
// strtype4.cpp -- line input
#include <iostream>
#include <string>
#include <cstring>

int main()
{
    using namespace std;
    char charr[20];
    string str;

    cout << "Length of string in charr before input: "
         << strlen(charr) << endl;
    cout << "Length of string in str before input: "
         << str.size() << endl;
    cout << "Enter a line of text:\n";
    cin.getline(charr, 20);   // indicate maximum length
    cout << "You entered: " << charr << endl;
    cout << "Enter another line of text:\n";
    getline(cin, str); // cin now an argument; no length specifier
    cout << "You entered: " << str << endl;
    cout << "Length of string in charr after input: "
         << strlen(charr) << endl;
    cout << "Length of string in str after input: "
         << str.size() << endl;
    return 0;
}
```



目录

- 其他形式的字符串字面值

5.1 其他形式的字符串字面值

✓除char类型外，C++还有类型wchar_t, char16_t和char32_t,可以创建这些类型的字符常量和字符串字面值

✓C++分别使用前缀L,u,U来表示对应类型的字符常量或字符串字面值

```
wchar_t title[] = L"Chief Astrogator"; //w_char string
char16_t name[] = u"Felonia Ripova";  //char16_t string
char32_t car[] = U"Humber Super Snipe"; //char32_t string
```

5.2 原始字符串

✓问题引入：当我们想cout出″的时候，可不可以不要用\″转义序列来输出，那样将会把一个输出字符串增加得很难看或难懂

✓原始(raw)字符串：在原始字符串中，字符表示的就是字符本身，例如，\n在原始字符串中不是换行的意思，就是\和n字符，″也是一个字符，可以放在字符串字面值里，因此，就不能再使用它来表示字符串的开头和结尾。因此，原始字符串将″(和)″用做界定符，并使用前缀R来标识原始字符串

```
cout << R"(Jim "King" Tutt uses a "\n" in his name.)" << endl;//raw string
```
→ Jim "King" Tutt uses a "\n" in his name.

5.2 原始字符串

✓如果要在原始字符串中包含)",该怎么做？

编译器遇到第一个)"时，会不会认为字符串到此结束？会的！

✓使用R"+*(标识原始字符串的开头，使用)+*"标识原始字符串的结尾

```
cout << R"+*("(Who wouldn't?)",she whispered.)+*" << endl;//raw string
```

```
"(Who wouldn't?)",she whispered.
```

✓自定义定界符时，在默认定界符之间添加任意数量的基本字符，但空格、左右括号、斜杠和控制字符（如制表符和换行符）除外

✓可将前缀R与其他字符串前缀结合使用，以标识wchar_t等类型的原始字符串：Ru、UR

目录

- 字符串使用专用函数进行操作

操作	字符数组	字符串变量
适用	C、C++	C++
头文件	#include<string.h> #include<ctring>	#include<string>
定义	char 变量名 char s1[10];	string 变量名 string s1;
定义时赋初值	char s1[10]="hello";长度不超过9	string s1="hello"; 长度无限制

赋值	char s1[10];
	strcpy(s1,"hello");长度不超过9
	
	char s1[10]="hello",s2[10];
	strcpy(s2,s1);
		string s1;
		s1="hello";长度无限制
		string s1="hello",s2;
		s2=s1;

单字符操作	char s1[10]="hello"; s1[2]='p';	string s1="hello"; s1[2]='p';

操作	字符数组	字符串变量
输入	cin、scanf	cin
输出	cout、printf	cout
字符串复制	char s1[10],s2[10]; ... strcpy(s1,s2);防止溢出	string s1,s2; ... s1=s2;不必考虑溢出
字符串连接	char s1[20],s2[10]; ... strcat(s1,s2);防止溢出	string s1,s2; ... s1=s1+s2;不必考虑溢出

字符串比较	用函数
	if(!strcmp(s1,s2))
	if(strcmp(s1,s2)>0)
	if(!strcmp(s1,s2)>=0)
		用比较运算符
		s1==s2;
		s1>s2;
		s1>=s2;

总结

- C++字符串初始化
- C++字符串赋值、合并和附加操作
- string类的其他操作
- string类的I/O
- 其他形式的字符串字面值
- 字符串使用专用函数进行操作