C++ Programming

Chapter 6 Strings

Zheng Guibin
（郑贵滨）

6.1 C-strings（C 风格字符串）

- **C风格字符串： 空字符结尾的字符数组 （null character '\0' ）**

char greetings[6]={'H','e','l','l','o','\0'};

'H'	'e'	'l'	'l'	'o'	'\0'

- 字符串字**(string literal)**

　　双引号括起来的字符序列，不需要指定字符数量，编译器自动在末尾插入空字符**'\0';**

```
 char greetings[]= "Hello";
 char  *greetingsptr = "Hello";
```

- 字符串字**"Hello"**包含**6**个字符，而不是**5**个。

```
 cin>>setw[6]>>greetings;
```

6.1 C-strings

- 如果初始化的字符串长度比字符数组小，数组中多出的元素用空字符'\0'填充

```
char greetings[9]= "Hello";
```

'H'	'e'	'l'	'l'	'o'	'\0'	'\0'	'\0'	'\0'

- 转义字符：\

```
char greetings[]= "\"Hello\", I said.";
```

6.1 C-strings

- 换行符'\n' 可以用 *endl* 代替

```
cout  <<  "Some text.\n";

cout  <<  "Some text." << endl;
```

- 若字符串太长，无法在一行写完，可分成多行来写。

```
char long_string[] = "This is the first half of the string"
                     "and this is the second half.";
```

- 字符数组的元素个数必须至少比字符串中的字符数多1;

6.2 C-string input and output

- **Program Example P6A: a simple demonstration of C-string input and output.**

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

main()
{
  const int MAX_CHARACTERS = 10 ;
  char first_name[ MAX_CHARACTERS + 1 ] ;

  cout << "Enter your first name (maximum "
       << MAX_CHARACTERS << " characters) " ;
  cin >> first_name ;
  cout << "Hello " << first_name << endl ;
}
```

- **The extraction operator >> read the characters up to(but not including) the space character after John.**

输入字符超过10个，发生溢出，….

使用getline()：避免溢出、能读入空白字符

6.2 C-string input and output

getline()

- *从cin 读入字符串，直到用户按下*'\n' 或已经读入了 10 字符
- '\n' 定界符（分隔符，delimiter），若省略该项，默认是 '\n'
- '\0' 自动加到字符串末尾。

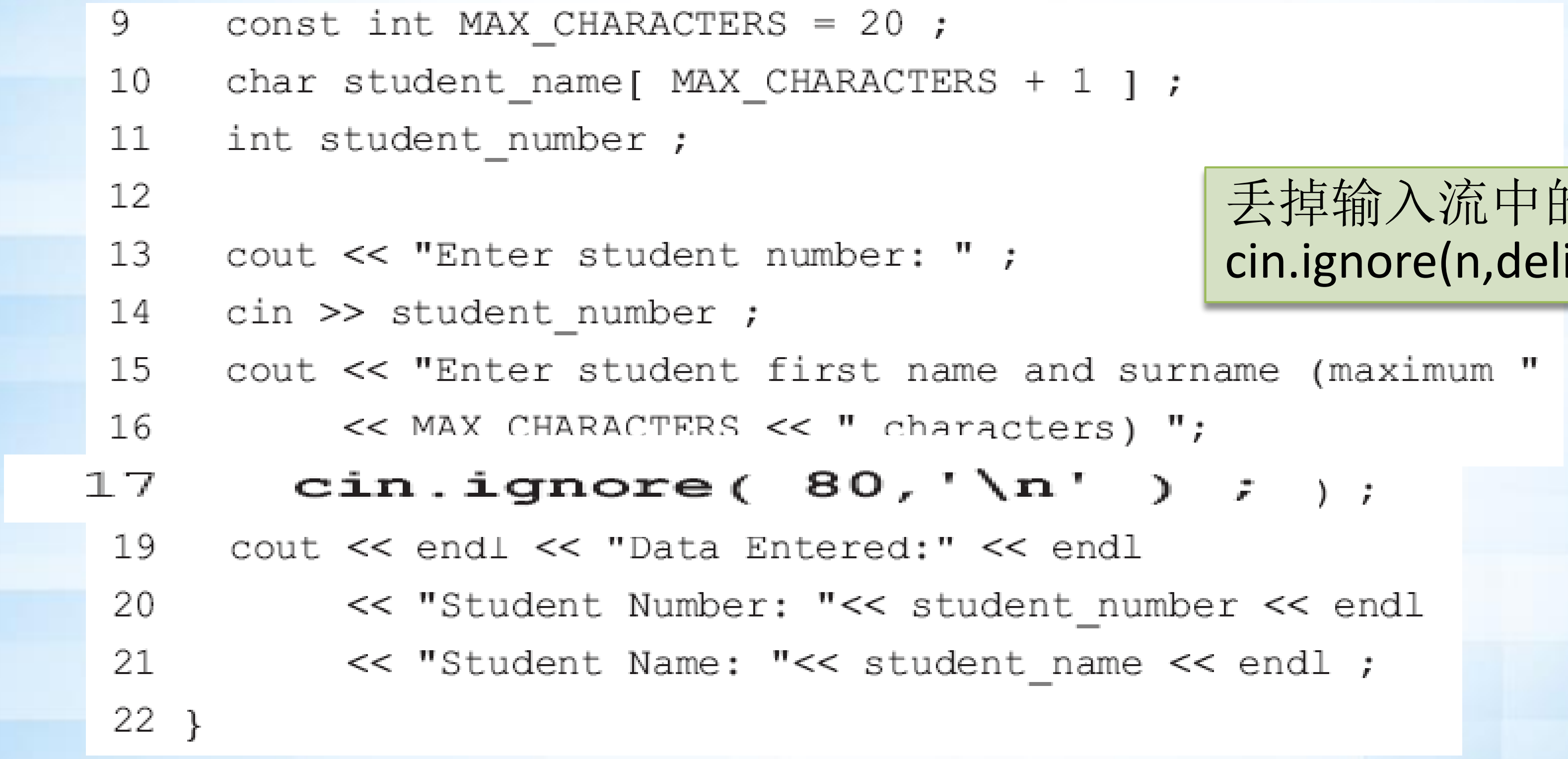
Program Example P6B

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

main()
{
  const int MAX_CHARACTERS = 10 ;
  char first_name[ MAX_CHARACTERS + 1 ] ;

  cout << "Enter your first name(maximum "
       << MAX_CHARACTERS << " characters) " ;
   cin.getline( first_name, MAX_CHARACTERS + 1, '\n' )
  cout << "Hello " << first_name << endl ;
}
```

演示：Program Example P6C

6.2 C-string input and output

- **Program Example P6D**



```
const int MAX_CHARACTERS = 20 ;
char student_name[ MAX_CHARACTERS + 1 ] ;
int student_number ;

cout << "Enter student number: " ;
cin >> student_number ;
cout << "Enter student first name and surname (maximum "
     << MAX_CHARACTERS << " characters) ";
  cin.ignore( 80,'\n' ) ;  ) ;
cout << endl << "Data Entered:" << endl
     << "Student Number: "<< student_number << endl
     << "Student Name: "<< student_name << endl ;
}
```

丢掉输入流中的字符:
cin.ignore(n,delimiter);

```
#include <iostream>
using namespace std ;
main()
{
  const int MAX_CHARACTERS = 20 ;
  char student_name[ MAX_CHARACTERS + 1 ] ;
  int student_number ;
  cout << "Enter student number: " ;
  cin >> student_number ;
  cout << "Enter student first name and surname (maximum "
       << MAX_CHARACTERS << " characters) ";
  cin.ignore( 80,'\n' ) ; //include '\n'  cin.ignore( 80,'n' ) ;
  cin.getline( student_name, MAX_CHARACTERS + 1 ) ;
  cout << endl << "Data Entered:" << endl
       << "Student Number: "<< student_number << endl
       << "Student Name: "<< student_name << endl ;
}
```

6.4 C-string functions

- **C++ 继承了C字符串的函数库**

#include <cstring>

- ***strlen():*** 字符串长度，不含末尾的'\0'.
- ***strcpy***(destination, source);
  - ✓ 字符串必须 '\0'结尾；
  - ✓ 目的操作数‘destination’有足够空间。

6.4 C-string functions

- ***strcat***(str1, str2)

  字符串连接将str2追加到 str1的末尾，默认是空字符结尾，

str1空间足够。

- ***strcmp***(str1, str2)
  - 两个C风格字符串比较大小.
  - 返回值:

```
< 0   ——  if str1 < str2,
0     ——  if str1 == str2,
>0    ——  if str1 > str2.
```

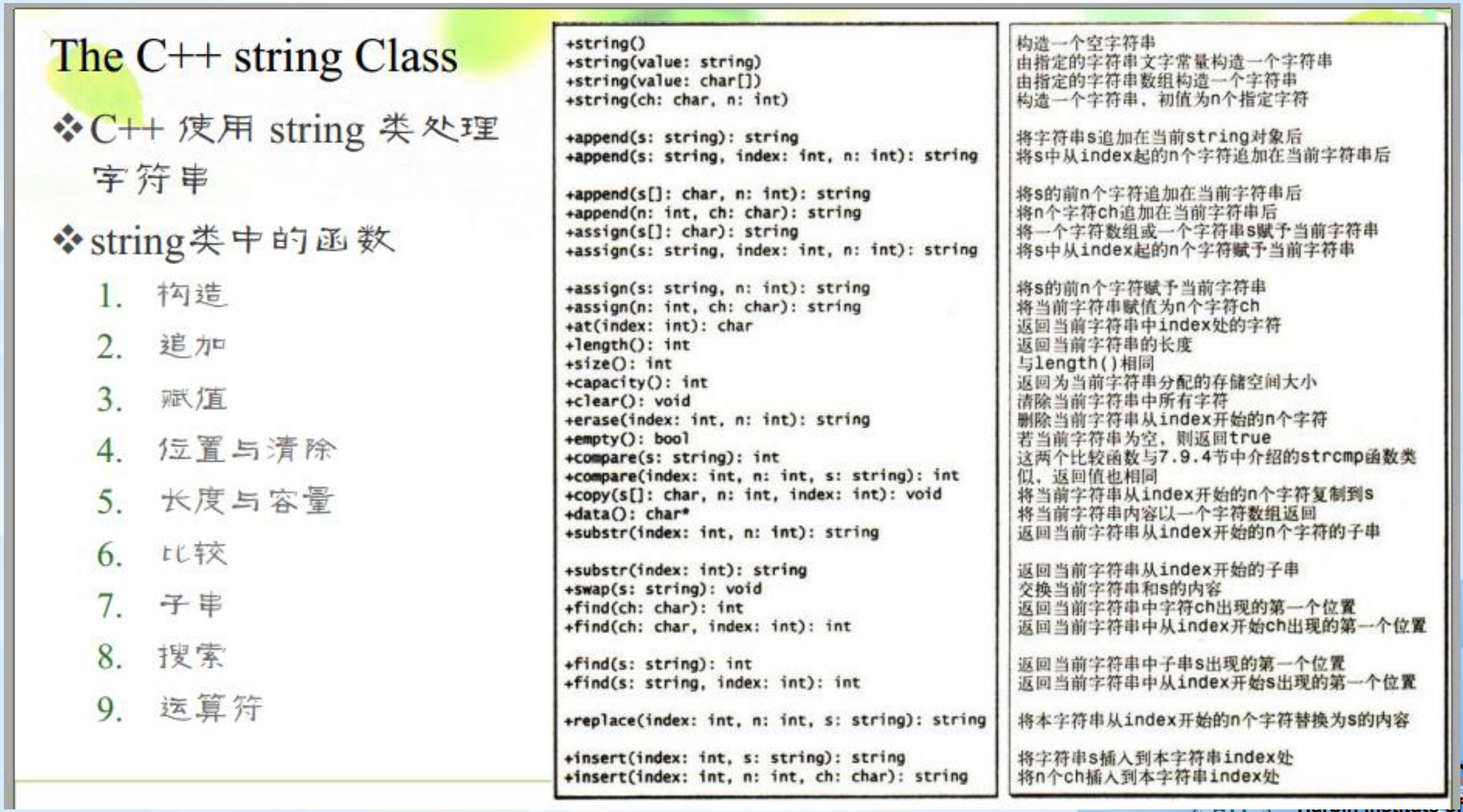
- strncat(str1, str2, n)
- strncmp(str1, str2, n)
- strncpy(str1, str2, n)

6.4 C-string functions

Function	Remark
strlen(str)	Finding the length of a C-string str
strcpy(str1, str2)	copies the contents of a C-string str2 to str1
strcat(str1, str2)	concatenates a C-string str2 to the end of the C-string str1
strcmp(str1, str2)	compares two null-terminated C-strings str1 and str2.
strncat(str1, str2, n)	Appends the first n characters of the C-string str2 to str1.
strncmp(str1, str2, n)	Identical to strcmp(str1, str2), except that at most, n characters are compared.
strncpy(str1, str2, n)	Copies the first n characters of str2 into str1.

6.5 C++ strings

◆ C++中有字符串类string（ 非内置数据类型）



The C++ string Class

❖C++ 使用 string 类处理字符串

❖string类中的函数

1. 构造
2. 追加
3. 赋值
4. 位置与清除
5. 长度与容量
6. 比较
7. 子串
8. 搜索
9. 运算符

+string()	构造一个空字符串
+string(value: string)	由指定的字符串文字常量构造一个字符串
+string(value: char[])	由指定的字符串数组构造一个字符串
+string(ch: char, n: int)	构造一个字符串，初值为n个指定字符
+append(s: string): string	将字符串s追加在当前string对象后
+append(s: string, index: int, n: int): string	将s中从index起的n个字符追加在当前字符串后
+append(s[]: char, n: int): string	将s的前n个字符追加在当前字符串后
+append(n: int, ch: char): string	将n个字符ch追加在当前字符串后
+assign(s[]: char): string	将一个字符数组或一个字符串s赋予当前字符串
+assign(s: string, index: int, n: int): string	将s中从index起的n个字符赋予当前字符串
+assign(s: string, n: int): string	将s的前n个字符赋予当前字符串
+assign(n: int, ch: char): string	将当前字符串赋值为n个字符ch
+at(index: int): char	返回当前字符串中index处的字符
+length(): int	返回当前字符串的长度
+size(): int	与length()相同
+capacity(): int	返回为当前字符串分配的存储空间大小
+clear(): void	清除当前字符串中所有字符
+erase(index: int, n: int): string	删除当前字符串从index开始的n个字符
+empty(): bool	若当前字符串为空，则返回true
+compare(s: string): int	这两个比较函数与7.9.4节中介绍的strcmp函数类
+compare(index: int, n: int, s: string): int	似，返回值也相同
+copy(s[]: char, n: int, index: int): void	将当前字符串从index开始的n个字符复制到s
+data(): char*	将当前字符串内容以一个字符数组返回
+substr(index: int, n: int): string	返回当前字符串从index开始的n个字符的子串
+substr(index: int): string	返回当前字符串从index开始的子串
+swap(s: string): void	交换当前字符串和s的内容
+find(ch: char): int	返回当前字符串中字符ch出现的第一个位置
+find(ch: char, index: int): int	返回当前字符串中从index开始ch出现的第一个位置
+find(s: string): int	返回当前字符串中子串s出现的第一个位置
+find(s: string, index: int): int	返回当前字符串中从index开始s出现的第一个位置
+replace(index: int, n: int, s: string): string	将本字符串从index开始的n个字符替换为s的内容
+insert(index: int, s: string): string	将字符串s插入到本字符串index处
+insert(index: int, n: int, ch: char): string	将n个ch插入到本字符串index处

```
// Program example P6G
// Program to demonstrate C++ strings.
#include <iostream>
#include <string>
using namespace std ;

main()
{
  string password = "secret" ;
  string user_input ;
  cout << "Enter Password: " ;
  cin >> user_input ;
  if ( password == user_input )
    cout << "Correct password. Welcome to the system ..." << endl ;
  else
    cout << "Invalid password" << endl ;
}
```

- C++ string 可以用运算符 ==, <, >,etc进行比较;
- string有很多有用的成员函数(^_^)

```
// Program example P6H-string initialisation and assignment
#include <iostream>
#include <string>
using namespace std ;
main()
{  // String initialisation examples.
  string str1 = "ABCDEFGHI" ; // Define a string and initialise it.
  string str2( 11, '-' ) ;    // Define a string of 11 dashes.
  string str3 = "This is the first part"
                     " and this is the second part." ;
  string str4 = str2 ;  // Initialise str4 with str2.
  string str5 ; // str5 has no initial value.
  cout << "After initialisations:" << endl
       << "  str1=" << str1 << endl
       << "  str2=" << str2 << endl
       << "  str3=" << str3 << endl
       << "  str4=" << str4 << endl
       << "  str5=" << str5 << endl ;
....
// String assignment examples.
 str1 = "ABCD" ;
 str2.assign( 3, '.' ) ; // Assign 3 dots to str2.
 cout << "After the 1st and 2nd assignments:" << endl
      << "  str1=" << str
      << "  str2=" << str

 str5.assign( str1, 1, 3 )
 cout << "After the 3r
         << "  str5=" << st

 cout << "Before swapping str1 and str2:"  << endl
      << "  str1=" << str1 << endl
      << "  str2=" << str2 << endl ;
 str1.swap( str2 ) ; // swap str1 and str2.
 cout << "After swapping str1 and str2:"  << endl
      << "  str1=" << str1 << endl
      << "  str2=" << str2 << endl ;
}
```

- str5.assign() :
- **str1** is the string to assign from,
- **1** is the starting position
- **3** is the number of characters to assign

6.5 C++ strings

6.5.2 string concatenation

```
//Program Example P6I
 string str1 = "ABCD", str2, str3 ;
  str2.assign( 3, '.' ) ; // Assign 3 dots to str2.
 // Concatenate str2 to str1 and assign to str3.
  str3 = str1 + str2 ;  // With strings, + means concatenate.
```

```
main()
{//string length, string indexing and sub-strings
  string str1 = "ABCDEFGH" ;
  int len1 ;

  len1 = str1.length() ;
  str1[0] = '*' ; // Index start at 0 and ends at (len1-1).
  str1[len1-1] = '*' ; // No index checking is done using [].

  // It is much safer to check the index value to ensure it is
  // not out of range by using the string member function at().
  str1.at( 0 ) = 'A' ;
  str1.at( len1 - 1 ) = 'H' ;

  // Display a space between each character of str1.
  cout << str1 << " with a space between each character:" << endl ;
  for ( int i = 0 ; i < len1 ; i++ )
    cout << str1.at( i ) << ' ' ;
   cout << endl ;
```

6.5 C++ strings

◆ Program Example P6J

substr() 取子串.

- 第1个参数：开始位置
- 第2个参数：取的字符数

```
//main(){
...

string str2 = "ABCDEFGH" ;
  cout << "Demonstration of substr:" << endl << "   " ;
  cout << "The first four characters of " << str2<< " are "
       <<  str2.substr( 0, 4 ) << endl << "   "
       << "The middle two characters of " << str2 << " are "
       <<  str2.substr( 3, 2 ) << endl << "   "
       << "The last three characters of " << str2 << " are "
       <<  str2.substr( 5,3 ) << endl ;
}
```

```
main()
{
  string str1 = "ABCDEFGH" ;
  int len1 ;

  len1 = str1.length() ;
  str1[0] = '*' ; // Index start at 0 and ends at (len1-1).
  str1[len1-1] = '*' ; // No index checking is done using [].

  // It is much safer to check the index value to ensure it is
  // not out of range by using the string member function at().
  str1.at( 0 ) = 'A' ;
  str1.at( len1 - 1 ) = 'H' ;

  // Display a space between each character of str1.
  cout << str1 << " with a space between each character:" << endl ;
  for ( int i = 0 ; i < len1 ; i++ )
    cout << str1.at( i ) << ' ' ;
   cout << endl ;
```

6.5 C++ strings

- **6.5.4 String replace, erase, insert and empty strings**

  Program example P6K，

- **6.5.5 string searching**

  Program example P6L

- **6.5.6 string comparisons**

  Program Example P6L

•C++ strings 可以用标准的比较运算符: ==, !=, <=, >= < 和 >

compare()

• 按字典顺序规则行比较

提示

- 字符串字（字符串直接量）初始化char * 变量时，一些编译器将字符串放在常量数据区保存，无法修改。如需修改字符串字，应将其存放在字符数组中。
- C++允许存储任意长度字符串，注意越界问题！
- 把字符当成字符串（参数传递），很危险！

Q & A