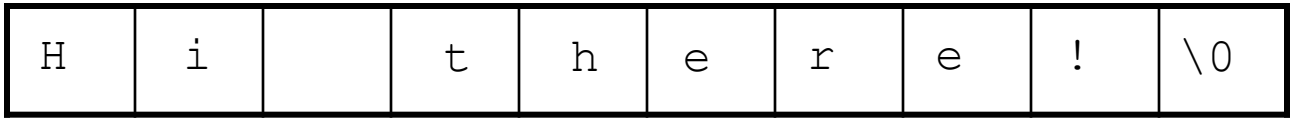
Strings

Cheng-Chin Chiang

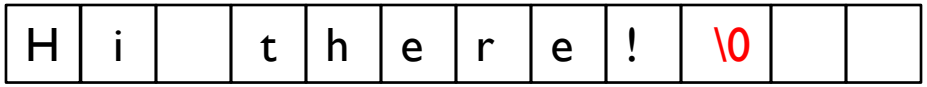
Strings

- Strings
  - 一串儲存在連續記憶體之字元串
  - 表示法：以雙引號圍起
    - “This is a book”，“I love programming”，“12234”
  - 字串須有一結束字元‘\0’（NULL)在字串尾，NULL在C++內為一個內定常數值



An Array Type of Strings – Character Array (字元陣列)

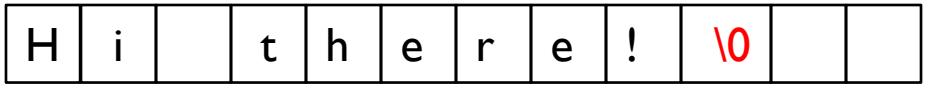
- 沿用C語言的陣列型態，陣列元素為字元，因此常被稱為C-String
- 例如：
  - char str[10];　　//一個可以放10個字元的陣列
  - char message[12];　　//一個可以放12個字元的陣列
  - char message[12]="Hi there!"　//宣告和初始化一次完成



  - char mystring[ ]="abc"　　//利用初始化來決定陣列大小(=4)
  - char mystring[ ]={'a','b','c'}　//和上例不一樣，那裏不一樣？
- 宣告技巧：
  - Char ArrayName[Maximum_C-String_Size+1];

An Array Type of Strings – Character Array (字元陣列)

- 沿用C語言的陣列型態，陣列元素為字元，因此常被稱為C-string
- 例如：
  - char str[10];　　//一個可以放10個字元的陣列
  - char message[12];　　//一個可以放12個字元的陣列
  - char message[12]="Hi there!"　//宣告和初始化一次完成



  - char mystring[ ]="abc"　　//利用初始化來決定陣列大小(3 or 4?)
  - char mystring[ ]={'a','b','c'}　//和上例不一樣，那裏不一樣？
- 宣告技巧：
  - **Char ArrayName[Maximum_C-String_Size+1];**

Pointers to Character Array

- 指標(pointer)經過動態記憶體配置(dynamic memory allocation)後，也可儲存陣列資料，所以也可以用來儲存一個字元陣列
- 例如：

```
char *s；                    //需配置記憶體，大小=字串最長長度+1
s = new char[5];
char *str=”ABCDE”;           //宣告、配置、初始化三工作一體成型
delete [] str;
```

String Array

- 字元二維陣列：
  - char s[4][6]={“Black”, ”White”, ”Red”, ”Green”}；
- 佔用24 Bytes

	0	1	2	3	4	5
0	‘B’	‘l’	‘a’	‘c’	‘k’	‘\0’
1	‘W’	‘h’	‘i’	‘t’	‘e’	‘\0’
2	‘R’	‘e’	‘d’	‘\0’		
3	‘G’	‘r’	‘e’	‘e’	‘n’	‘\0’

String Array

- 指標陣列：
  - char *s[4]={“Black”, ”White”, ”Red”, ”Green”};
- 佔用22 Bytes
  - char *s[ ]={“Black”, ”White”, ”Red”, ”Green”};

	0	1	2	3	4	5
0	‘B’	‘l’	‘a’	‘c’	‘k’	‘\0’
1	‘W’	‘h’	‘i’	‘t’	‘e’	‘\0’
2	‘R’	‘e’	‘d’	‘\0’		
3	‘G’	‘r’	‘e’	‘e’	‘n’	‘\0’

= and == with C-strings

- C-strings 特殊之處
  - 無法直接指定(assign)或比較(compare)

```
char aString[10];
aString = "Hello";    // ILLEGAL!
```
("=" 只能用在宣告的初始化!)

```
char aString[10] = "Hello";
char anotherString[10] = "Goodbye";
aString == anotherString;   // NOT allowed!
```

- 若要做指定，必須呼叫字串函數
  strcpy(aString, "Hello");
  - 內建函數(定義在標頭檔 <cstring>)
  - "cpy" 意為拷貝(copy)
  - 拷貝時電腦不負責檢查陣列大小是否足夠!
    - 程式設計師的責任
- 若要做比較，必須呼叫字串函數
  strcmp(cStringA, cStringB);
  - 回傳值
    - >0 ➔ cStringA>cStringB
    - =0 ➔ cStringA==cStringB
    - <0 ➔ cStringA<cStringB

C-string Functions: strlen()

- 取得字串長度（不包含NULL）

```
char myString[10] = "dobedo";

cout << strlen(myString);
```

  - 傳回6

C-string Functions: strcat()

- 字串串接

  char stringVar[20] = "The rain";

  strcat(stringVar, "in Spain");

  - 結果：

    stringVar 變成 “The rainin Spain” (記得要為stringVar備好足夠的記憶體空間)

Other Predefined Functions in <cstring>

Display 9.1 Some Predefined C-String Functions in `<cstring>`

FUNCTION	DESCRIPTION	CAUTIONS
`strcpy(`*Target_String_Var*`,` *Src_String*`)`	Copies the C-string value *Src_String* into the C-string variable *Target_String_Var*.	Does not check to make sure *Target_String_Var* is large enough to hold the value *Src_String*.
`strcpy(`*Target_String_Var*`,` *Src_String, Limit*`)`	The same as the two-argument `strcpy` except that at most *Limit* characters are copied.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcpy`. Not implemented in all versions of C++.
`strcat(`*Target_String_Var*`,` *Src_String*`)`	Concatenates the C-string value *Src_String* onto the end of the C-string in the C-string variable *Target_String_Var*.	Does not check to see that *Target_String_Var* is large enough to hold the result of the concatenation.

(continued)

Other Predefined Functions in <cstring>

Display 9.1 Some Predefined C-String Functions in <cstring>

FUNCTION	DESCRIPTION	CAUTIONS
`strcat(`*Target_String_Var*`,` *Src_String, Limit*`)`	The same as the two argument `strcat` except that at most *Limit* characters are appended.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcat`. Not implemented in all versions of C++.
`strlen(`*Src_String*`)`	Returns an integer equal to the length of *Src_String*. (The null character, `'\0'`, is not counted in the length.)	
`strcmp(`*String_1*`,` *String_2*`)`	Returns 0 if *String_1* and *String_2* are the same. Returns a value < 0 if *String_1* is less than *String_2*. Returns a value > 0 if *String_1* is greater than *String_2* (that is, returns a nonzero value if *String_1* and *String_2* are different). The order is lexicographic.	If *String_1* equals *String_2*, this function returns 0, which converts to `false`. Note that this is the reverse of what you might expect it to return when the strings are equal.
`strcmp(`*String_1*`,` *String_2, Limit*`)`	The same as the two-argument `strcat` except that at most *Limit* characters are compared.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcmp`. Not implemented in all versions of C++.

例題：Using strcpy and strncpy

```
11  int main()
12  {
13     char x[] = "Happy Birthday to You"; // string length 21
14     char y[ 25 ];
15     char z[ 15 ];
16
17     strcpy( y, x ); // copy contents of x into y
18
19     cout << "The string in array x is: " << x
20        << "\nThe string in array y is: " << y << '\n';
21
22     // copy first 14 characters of x into z
23     strncpy( z, x, 14 ); // does not copy null character
24     z[ 14 ] = '\0'; // append '\0' to z's contents
25
26     cout << "The string in array z is: " << z << endl;
27     return 0; // indicates successful termination
28  } // end main
```

```
The string in array x is: Happy Birthday to You
The string in array y is: Happy Birthday to You
The string in array z is: Happy Birthday
```

例題：Concatenating Strings with strcat and strncat

```
11 int main()
12 {
13    char s1[ 20 ] = "Happy "; // length 6
14    char s2[] = "New Year "; // length 9
15    char s3[ 40 ] = "";
16
17    cout << "s1 = " << s1 << "\ns2 = " << s2;
19    strcat( s1, s2 ); // concatenate s2 to s1 (length 15)
21    cout << "\n\nAfter strcat(s1, s2):\ns1 = " << s1 << "\ns2 = " << s2;
23    // concatenate first 6 characters of s1 to s3
24    strncat( s3, s1, 6 ); // places '\0' after last character
25
26    cout << "\n\nAfter strncat(s3, s1, 6):\ns1 = " << s1
27       << "\ns3 = " << s3;
28
29    strcat( s3, s1 ); // concatenate s1 to s3
30    cout << "\n\nAfter strcat(s3, s1):\ns1 = " << s1
31       << "\ns3 = " << s3 << endl;
32    return 0; // indicates successful termination
33 } // end main
```

例題：Concatenating Strings with strcat and strncat

s1 = Happy
s2 = New Year

After strcat(s1, s2):
s1 = Happy New Year
s2 = New Year

After strncat(s3, s1, 6):
s1 = Happy New Year
s3 = Happy

After strcat(s3, s1):
s1 = Happy New Year
s3 = Happy Happy New Year

例題：Comparing Strings with strcmp and strncmp

```
14 int main()
15 {
16    char *s1 = "Happy New Year";
17    char *s2 = "Happy New Year";
18    char *s3 = "Happy Holidays";
19
20    cout << "s1 = " << s1 << "\ns2 = " << s2 << "\ns3 = " << s3
21       << "\n\nstrcmp(s1, s2) = " << setw( 2 ) << strcmp( s1, s2 )
22       << "\nstrcmp(s1, s3) = " << setw( 2 ) << strcmp( s1, s3 )
23       << "\nstrcmp(s3, s1) = " << setw( 2 ) << strcmp( s3, s1 );
24
25    cout << "\n\nstrncmp(s1, s3, 6) = " << setw( 2 )
26       << strncmp( s1, s3, 6 ) << "\nstrncmp(s1, s3, 7) = " << setw( 2 )
27       << strncmp( s1, s3, 7 ) << "\nstrncmp(s3, s1, 7) = " << setw( 2 )
28       << strncmp( s3, s1, 7 ) << endl;
29    return 0; // indicates successful termination
30 } // end main
```

例題：Comparing Strings with strcmp and strncmp

s1 = Happy New Year
s2 = Happy New Year
s3 = Happy Holidays

strcmp(s1, s2) = 0
strcmp(s1, s3) = 1
strcmp(s3, s1) = -1

strncmp(s1, s3, 6) = 0
strncmp(s1, s3, 7) = 1
strncmp(s3, s1, 7) = -1

例題：Tokenizing a String with strtok

```
10  int main()
11  {
12     char sentence[] = "This is a sentence with 7 tokens";
13     char *tokenPtr;
14
15     cout << "The string to be tokenized is:\n" << sentence
16        << "\n\nThe tokens are:\n\n";
17
18     // begin tokenization of sentence
19     tokenPtr = strtok( sentence, " " );
20
21     // continue tokenizing sentence until tokenPtr becomes NULL
22     while ( tokenPtr != NULL )
23     {
24        cout << tokenPtr << '\n';
25        tokenPtr = strtok( NULL, " " ); // get next token
26     } // end while
28     cout << "\nAfter strtok, sentence = " << sentence << endl;
29     return 0; // indicates successful termination
30  } // end main
```

例題： Tokenizing a String with strtok

The string to be tokenized is:
This is a sentence with 7 tokens

The tokens are:

This
is
a
sentence
with
7
tokens

After strtok, sentence = This

C-string Arguments and Parameters

- 因為**c-string parameter** 是陣列
  - C-strings 事宜call by reference的方式傳遞
  - 其值可被呼叫函數的內部指令改變
  - 如果想避免被修改，可以 “**const**” 宣告函數引數
- 一般陣列需要把元素個數當作參數傳遞，**C-strings**可有彈性
  - 傳遞元素個素
  - 不傳遞，找出結束字元即可算出元素個數

Input/Output of Character Arrays

- 除了可用陣列方式搭配迴圈逐一元素輸出入(勿忘加上結束字元)外，也可使用簡易輸出入
- 例如：

```
char st[5];
for (n=0；n<5；n++) {
    cout << st[n]；
    cin>>st[n];
}
char str[10]="My god!";
char message[12];
cout << str;
cin >> message;
```

- 使用字元陣列儲存字串時切勿覆蓋掉結束字元

String/Numeric Conversion Functions

- 須#include <cstdlib>

FUNCTION	PARAMETER	ACTION
`atoi`	C-string	converts C-string to an `int` value, returns the value
`atol`	C-string	converts C-string to a `long` value, returns the value
`atof`	C-string	converts C-string to a `double` value, returns the value
`itoa`	`int`, C-string, `int`	converts 1st `int` parameter to a C-string, stores it in 2nd parameter. 3rd parameter is base of converted value

String/Numeric Conversion Functions

```
int iNum;
long lNum;
double dNum;
char intChar[10];
iNum = atoi("1234"); // puts 1234 in iNum
lNum = atol("5678"); // puts 5678 in lNum
dNum = atof("35.7"); // puts 35.7 in dNum
itoa(iNum, intChar, 8); // puts the string
    // "2322" (base 8 for 1234_10) in
intChar
```

String/Numeric Conversion Functions - Notes

- 如果 **C-string** 中含有非數字字元，回傳結果就變成未定義
  - 可能狀況一：以非數字字元前的數字字元作為回傳結果
  - 可能狀況二：回傳**0**
- `itoa` 函數不做陣列長度檢查，請自備足夠的長度空間來儲存回傳的字串

Output of C-strings

- cout之直接輸出：字串之輸出
  - cout << message;
- cout.put()之輸出：單一字元之輸出
  - cout.put(message[5]);
- cout.write()之輸出：功用為輸出字串前n個字元，並不因碰到字串之結束字元 '\0'而結束
  - cout.write(message, 21);

Input of C-strings

- cin之直接輸入
- cin.getline()之輸入：整行輸入或限定字數輸入
- cin.get()之輸入：逐字輸入或限定字數輸入

Direct C-string Input

- 使用extraction operator >>
- 注意事項
  - Tab, space, line breaks 會被跳過
  - 輸入讀取多個變數時會以空白和跳行作為分隔
  - 字串變數須事先備好足夠的記憶體空間（C++不會警告陣列長度不夠）

```
char a[80], b[80];
cout << "Enter input: ";
cin >> a >> b;
cout << a << b << "END OF OUTPUT\n";
```

結果：

```
Enter input: Do be do to you!
DobeEND OF OUTPUT
```

C-string Input with getline()

- cin.getline() 讀資料時只有在輸入字串後遇到Enter鍵('\n')後才取得資料交給字串變數，並捨去 **'\n'** 後在字串尾加上 **'\0'**。

  cin.getline(str，num_of_char);

  如要以getline來輸入20個字元之字串（第21個字元為結束字元），就用

  cin.getline(str,21)；

  - 此敘述會將整行字串放入**name**內，但輸入字元需小於**20**
  - 如果在20各字元內遇到 '\n'，則停止讀取 '\n' 的後續字元

C-string Input with get()

- cin.get() :
  - int get()； //讀取下一字元，傳回整數型態
  - istream & get(char&);
    讀取單一字元放入參數內
    - get()可用來讀取space, '\n'等特殊字元

```
char c1, c2, c3, c4;
cin.get(c1);
cin.get(c2);
cin.get(c3);
cin.get(c4);
cout<<c1<<c2<<c3;
```

結果：

```
AB
CD
AB
C
```

  - istream & get(char*, int len, char = '\n');
    讀取（len-1)個字元，當中時若碰到輸入鍵('\n')就結束('\n'還留在input stream中)

例題:以get()檢查輸入行的結束

```
cout << “Enter a line of input:\n”;
char symbol;
do
{
    cin.get(symbol);
    cout <<symbol;
} while (symbol != ‘n’);
cout << “That’s all.\n”;
```

例題:以get()參數輸入後輸出

```
#include <iostream>
using namespace std;
const int SIZE=21;
int main( ){
int age;
char sport[SIZE];
cout <<“輸入年齡:";
cin<<age;
cout <<"喜好運動名:";
cin.get(sport,SIZE);
cout <<“\n”<< age <<“歲的你 ,喜歡”
      <<sport<<“，我也很喜歡\n";
return 0;
}
```

輸入年齡：18
喜好運動名：
18歲的你，喜歡我也很喜歡

Why運動名sport無法輸入?

原因乃在讀取年齡後輸入鍵(Enter)(‘\n’)保留在input stream內，因此下一輸入cin.get(sport，20)就讀到‘\n’ (別忘了get()不會跳過‘\n’和空格），所以造成sport得到一NULL字元而已

解決cin.get()讀取資料時留住’\n’在輸入緩衝區之困擾

- 利用第一種格式無參數之cin.get()；在接下來讀取’\n’，因無參數之get()之功能乃讀取下一字元，讓輸入緩衝區空無一物後，下一輸入才能正常運作，即：

```
cin.get(name，SIZE)；    //讀取Bill Gates
cin.get()；                    //讀取 \n
```

  - 或將上述兩列合併如下：

```
cin.get(name,SIZE).get()；
```

例題：數字輸入與getline()輸入

```
#include <iostream>          //cout
using namespace std;
const int SIZE=21;
int main( ){
    char *sentence;
    short n;
    cout <<"字串長度=";
    cin >> n;
    sentence = new char[n-1];//配置記憶體
    cout <<"輸入字串=";
    cin.getline(sentence,n);
    cout <<"輸入字串為 "<<sentence<<endl;
    delete sentence;          //釋放記憶體
    return 0;
}
```

字串長度=30
輸入字串=
輸入字串為

為何無法輸入字串呢?sentence="\0"

More Member Functions

- putback()
  - 退回上一個讀取字元到input stream中（下次讀取會讀到此退回字元）

    cin.putback(lastChar);
- peek()
  - 傳回下個字元，但仍然讓該字元留在input stream中

    peekChar = cin.peek();
- ignore()
  - 最多跳過指定個數的字元，直到特定字元出現

    cin.ignore(1000, '\n');
    - Skips at most 1000 characters until '\n'

例題：輸入數字和文字的判斷（版本一）

```
cout << "Enter a number or a word: ";
c = cin.get();

if ( (c >= '0') && (c <= '9') )
{
  cin.putback (c);
  cin >> n;
  cout << "You have entered number " << n << endl;
}
else
{
  cin.putback (c);
  cin >> str;
  cout << " You have entered word " << str << endl;
}
```

例題：輸入數字和文字的判斷（版本二）

```
cout << "Enter a number or a word: ";
 c=cin.peek();

 if ( (c >= '0') && (c <= '9') )
 {
   cin >> n;
   cout << "You have entered number " << n << endl;
 }
 else
 {
   cin >> str;
   cout << " You have entered word " << str << endl;
 }
```

String Manipulation Functions

- 須先 #include <cctype>

Display 9.3 Some Functions in <cctype>

FUNCTION	DESCRIPTION	EXAMPLE
`toupper(`*Char_Exp*`)`	Returns the uppercase version of *Char_Exp* (as a value of type `int`).	`char c = toupper('a');` `cout << c;` **Outputs:** A
`tolower(`*Char_Exp*`)`	Returns the lowercase version of *Char_Exp* (as a value of type `int`).	`char c = tolower('A');` `cout << c;` **Outputs:** *a*
`isupper(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is an uppercase letter; otherwise, returns `false`.	`if (isupper(c))` `  cout << "Is uppercase.";` `else` `  cout << "Is not uppercase.";`

String Manipulation Functions

- 須先 #include <cctype>

Display 9.3 Some Functions in <cctype>

FUNCTION	DESCRIPTION	EXAMPLE
`islower(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a lowercase letter; otherwise, returns `false`.	`char c = 'a';` `if (islower(c))` `  cout << c << " is lowercase.";` **Outputs:** `a is lowercase.`
`isalpha(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is a letter of the alphabet; otherwise, returns `false`.	`char c = '$';` `if (isalpha(c))` `  cout << "Is a letter.";` `else` `  cout << "Is not a letter.";` **Outputs:** `Is not a letter.`
`isdigit(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is one of the digits `'0'` through `'9'`; otherwise, returns `false`.	`if (isdigit('3'))` `  cout << "It's a digit.";` `else` `  cout << "It's not a digit.";` **Outputs:** `It's a digit.`
`isalnum(`*Char_Exp*`)`	Returns `true` provided *Char_Exp* is either a letter or a digit; otherwise, returns `false`.	`if (isalnum('3') && isalnum('a'))` `  cout << "Both alphanumeric.";` `else` `  cout << "One or more are not.";` **Outputs:** `Both alphanumeric.`

String Manipulation Functions

- 須先 #include <cctype>

`isspace(Char_Exp)`	Returns `true` provided *Char_Exp* is a whitespace character, such as the blank or newline character; otherwise, returns `false`.	`//Skips over one "word" and sets c` `//equal to the first whitespace` `//character after the "word":` `do` `{` `  cin.get(c);` `} while (! isspace(c));`
`ispunct(Char_Exp)`	Returns `true` provided *Char_Exp* is a printing character other than whitespace, a digit, or a letter; otherwise, returns `false`.	`if (ispunct('?'))` `  cout << "Is punctuation.";` `else` `  cout << "Not punctuation.";`
`isprint(Char_Exp)`	Returns `true` provided *Char_Exp* is a printing character; otherwise, returns `false`.	
`isgraph(Char_Exp)`	Returns `true` provided *Char_Exp* is a printing character other than whitespace; otherwise, returns `false`.	
`isctrl(Char_Exp)`	Returns `true` provided *Char_Exp* is a control character; otherwise, returns `false`.	

Standard Class “string”

- C++內建的字串物件類別
  - 須#include <string>
  - 須using namespace std;
  - 可以視同一般的基本型態（如int, float, double, char等）使用
- 可以指定(assign)、比較(compare)、串接(concatenation)：

```
string s1, s2, s3;
s3 = s1 + s2;                  //Concatenation
If (s1==s2)
    s3 = "Hello Mom!“; //Assignment
```

例題：string的使用

Display 9.4 Program Using the Class string

```
1  //Demonstrates the standard class string.
2  #include <iostream>
3  #include <string>
4  using namespace std;

5  int main( )
6  {
7       string phrase;
8       string adjective("fried"), noun("ants");
9       string wish = "Bon appetite!";

10      phrase = "I love " + adjective + " " + noun + "!";
11      cout << phrase << endl
12           << wish << endl;

13      return 0;
14 }
```

Initialized to the empty string.

Two equivalent ways of initializing a string variable

SAMPLE DIALOGUE

I love fried ants!
Bon appetite!

I/O with Class string

- 與其他基本型態相同

```
string s1, s2;
cin >> s1;
cin >> s2;
```

結果：

User types in:
May the hair on your toes grow long and curly

➔s1 receives value "May"
➔s2 receives value "the"

getline() with Class string

- 一個全域性函數，輸入完畢後回傳一個cin物件的reference
- 與 cin 的 getline()使用相同

```
string line;
cout << "Enter a line of input: ";
getline(cin, line);
cout << line << "END OF OUTPUT";
```

結果:

- Enter a line of input: Do be do to you!
  Do be do to you!END OF INPUT

Other getline() Versions

- 可以指定分隔符號

```
string line;
cout << "Enter input: ";
getline(cin, line, "?");
```

  - 讀取輸入字元直到“？”出現

- getline() 函數回傳一個cin的reference

```
string s1, s2;
getline(cin, s1) >> s2;
```

  - 相當于： (cin) >> s2;

Class string Processing

- 具備c-strings所有運算，且還更多函數（超過100個）

EXAMPLE	REMARKS
Constructors	
`string str;`	Default constructor; creates empty `string` object `str`.
`string str("string");`	Creates a `string` object with data `"string"`.
`string str(aString);`	Creates a `string` object `str` that is a copy of `aString`. `aString` is an object of the class `string`.
Element access	
`str[i]`	Returns read/write reference to character in `str` at index `i`.
`str.at(i)`	Returns read/write reference to character in `str` at index `i`.
`str.substr(position, length)`	Returns the substring of the calling object starting at `position` and having `length` characters.
Assignment/Modifiers	
`str1 = str2;`	Allocates space and initializes it to `str2`'s data, releases memory allocated for `str1`, and sets `str1`'s size to that of `str2`.
`str1 += str2;`	Character data of `str2` is concatenated to the end of `str1`; the size is set appropriately.
`str.empty( )`	Returns `true` if `str` is an empty `string`; returns `false` otherwise.

(continued)

Class string Processing

Display 9.7 Member Functions of the Standard Class `string`

EXAMPLE	REMARKS
`str1 + str2`	Returns a `string` that has `str2`'s data concatenated to the end of `str1`'s data. The size is set appropriately.
`str.insert(pos, str2)`	Inserts `str2` into `str` beginning at position `pos`.
`str.remove(pos, length)`	Removes substring of size `length`, starting at position `pos`.
Comparisons	
`str1 == str2  str1 != str2`	Compare for equality or inequality; returns a Boolean value.
`str1 < str2  str1 > str2`	Four comparisons. All are lexicographical comparisons.
`str1 <= str2  str1 >= str2`	
`str.find(str1)`	Returns index of the first occurrence of `str1` in `str`.
`str.find(str1, pos)`	Returns index of the first occurrence of string `str1` in `str`; the search starts at position `pos`.
`str.find_first_of(str1, pos)`	Returns the index of the first instance in `str` of any character in `str1`, starting the search at position `pos`.
`str.find_first_not_of (str1, pos)`	Returns the index of the first instance in `str` of any character *not* in `str1`, starting search at position `pos`.

C-string and string Object Conversions

- 自動轉換 c-string 到 string object:

```
char aCString[ ] = “My C-string”;
string stringVar;
stringVar = aCstring;  // OK
```

- 不允許從string到c-string

```
aCString = stringVar;
```

  - 必須使用strcopy進行copy運算

```
strcpy(aCString, stringVar.c_str());
```