CHAPTER

18 C 與 C++的差異

18.1 C 與 C++的差異簡介

前一章已經知道 C 和 C++ 是使用不同的方式來處理格式化輸出入。C++是 C 語言的超集合，因此 C 語言的相關語法及函式也可以在 C++中使用，下表我們列出 C 語言與 C++的差異處：

功能	C 語言	C++語言
註解	/* … */ 為單行註解	//…、/*…*/ 為單行註解； /*…*/ 也可以當做多行註解
變數宣告	必須在程式或函式的最開頭先宣告才能使用	可以在任何位置宣告變數
資料型別	int、char、float、double、指標、陣列、union、struct、enum	除了可使用 C 語言提供的資料型別，還提供 bool、參考型別、類別(class)…等，本章介紹
多載函式 或 多載運算子	無	可讓相同名稱的函式或運算子擁有不同的功能。(多載函式本章介紹，多載運算子第 20 章介紹)
inline 函式	無	inline 函式內的敘述會直接取代該函式，如此會省略呼叫函式時所須往返的時間，因此執行時的速度較快。(本章介紹)
常數定義	設定 const 常數時允許可以不指定初值	設定 const 常數時一定要同時指定初值

功能	C 語言	C++語言
格式化 I/O	使用 scanf()和 printf()函式	使用 cout 和 cin 物件，並配合 ios 類別的格式化旗標以及 I/O 控制器。(第 17 章介紹)
string 類別	無	透過 string 類別可以建立字串物件。(本章介紹)
動態配置記憶體	使用 malloc()函式動態配置記憶體；透過 free()函式釋放記憶體。	使用 new 動態配置記憶體；透過 delete 釋放記憶體。(第 19 章介紹)
物件導向技術	無	C++物件導向程式設計提供封裝、繼承、多型的特性。(封裝第 19 章介紹；繼承與多型第 20 章介紹)
樣版	無	透過樣版的技術只要撰寫一個函式或類別即可以讓不同的型別的資料做相同的處理動作。(第 21 章介紹)
例外處理	無	透過例外處理機制讓您可以即時回應程式執行時期的錯誤。(第 21 章介紹)

下表為 C 和 C++共用的關鍵字：

auto	do	goto	signed	unsigned
break	double	if	sizeof	void
case	else	int	static	volatile
char	enum	long	struct	while
const	extern	register	switch	
continue	float	return	typedef	
default	for	short	union	

下表為 C++專屬的關鍵字：

asm	explicit	operator	this	using
bool	false	private	throw	virtual
catch	friend	protected	true	wchar_t
class	inline	public	try	
const_cast	mutable	reinterpret_cast		
delete	namespace	static_cast	typeid	
dynamic_cast	new	template	typename	

18.2 布林資料型別

C++ 定義了布林資料型別，它可以用來存放 true(真)和 false(假)，bool 布林資料型別常被使用在關係運算式或邏輯運算式的條件式中。在 C 語言可以使用數值來表示布林值，在 C++ 也接受，即用「0」表示「false」；用非零值表示「true」。下表即是做 And、Or、Not 布林運算的結果。

x	y	x && y (x And y)	x || y (x Or y)	! x (Not x)
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

譬如宣告 t 和 f 為 bool 型別變數，t 為 true，f 為 false，其寫法如下：

```
bool t = true, f = false ;
```

18.3 多載函式

所謂的「多載函式」(Overloaded Function)，就是允許函式在 C++程式(*.cpp)或同一類別中可以使用相同函式名稱，多載函式是藉由不同的引數串列個數或引數的不同資料型別來加以區隔相同名稱的函式。

如下簡例中，定義兩個名稱同為「add」的函式，一個 add 函式用來傳回兩個整數相加的結果，另一個 add 函式用來傳回三個浮點數(倍精確度)相加的結果。

```
int add(int, int);
double add(double, double, double);
int main(void)
{
        int total1, x, y;
        x=10 ; y=15;
        double total2, i, j ,k ;
        i=1.3 ; j=5.6 ; k=45.3;
        total1 = add(x, y)  ;          //total=25
        total2 = add(i, j ,k);         //total=52.2
        ……
    return 0;
}

int add(int a, int b)
{
        return   a+b;          //傳回兩個整數相加的結果
}

double add(double a , double b , double c)
{
        return a+b+c;          //傳回三個單精確度相加的結果
}
```

範例：overloading.cpp

練習撰寫兩個名稱皆為 getmin()的函式。第一個 getmin()函式用來取得兩個浮點數中的最小數；第二個 getmin()函式用來取得浮點數陣列中的最小數。

執行結果

程式碼 FileName : overloading.cpp

```
01 #include <cstdlib>
02 #include <iostream>
03 #include <iomanip>
04 #include <string>
05 using namespace std;
06
07 double getmin(double, double);
08 double getmin(double[], int);
09 int main(int argc, char *argv[])
10 {
11     double a=21.3, b=14.8;
12     cout << a << " 和 " << b << " 最小的數值為 " << getmin(a, b) << "\n\n";
13     double f[]={12.1, 54.33, 7.2, 40, 65.1};
14     cout << "陣列元素 [12.1, 54.33, 7.2, 40, 65.1] 中最小的數值為 "
              << getmin(f, 5) << "\n\n";
15     system("PAUSE");
16     return 0;
17 }
18
19 double getmin(double x, double y)
20 {
21     if (x < y)
22     {
23        return x;
24   }
```

```
25       else
26       {
27           return y;
28       }
29 }
30
31 double getmin(double varray[], int n)
32 {
33   double min=varray[0];
34   for(int i=1; i<n-1; i++)
35   {
36      if(varray[i]<min)
37       {
38         min=varray[i];
39       }
40   }
41   return min;
42 }
```

說明

1. 第 19~29 行：這個 getmin()函式用來傳回兩個浮點數(倍精確數)的最小數。
2. 第 31~42：這個 getmin()函式用來傳回浮點數陣列中的最小數。

18.4 inline 函式

傳統的 C 語言並未提供 inline 這個函式。如果函式內定義的程式敘述相當短，只寫成一行敘述，可將該函式定義成 inline 函式。inline 函式的特點如下：

1. inline 函式類似 C 語言參數化巨集的運作方式，其優點是 C++編譯時會將呼叫函式直接替換成 inline 函式內的敘述，因此執行時不必像呼叫一般函式，必須跳到函式所在處，待執行完再返回原呼叫處的下一個敘述，因此 inline 函式的執行時間較快。

2. inline 函式的缺點是：如果 inline 函式本身敘述很長，而且呼叫次數相當頻繁，則程式的長度會變長。
3. inline 函式比巨集展開更能被 C++ 編譯器最佳化，因此建議在 C++可以使用 inline 函式取代巨集來避免呼叫函式時往返所需的時間。
4. 若 inline 函式內有迴圈敘述、switch 敘述、goto 敘述或 static 變數，或是 inline 函式內的敘述太長，則此 inline 函式會被 C++編譯器視為一般函式來處理。
5. 巨集內的參數不會做資料型別的檢查；但 inline 函式因為有引數串列，因此呼叫 inline 函式時會檢查所傳入引數串列的資料型別是否有錯誤。

下面即是 inline 函式的宣告語法，宣告和定義 inline 函式時，必須在傳回值型別之前加上「inline」，呼叫 inline 函式與呼叫一般函式的方式相同。

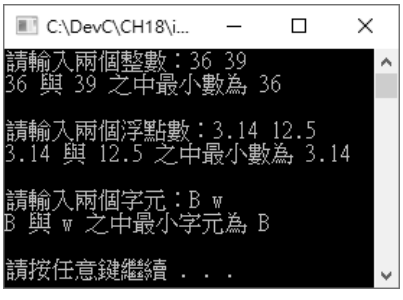
語法

```
inline 傳回值型別 函式名稱( [引數串列]);
```

範例：inlinefun.cpp

利用 inline 配合多載函式撰寫三個相同名稱的 getmin()函式。其中呼叫 getmin()函式可以用來取得兩個整數或兩個浮點數，或兩個字元之中的最小數。

執行結果



程式碼 FileName : overloading.cpp

```
01 #include <cstdlib>
02 #include <iomanip>
```

```
03 #include <iostream>
04 #include <string>
05
06 using namespace std;
07 inline int getmin(int, int);
08 inline double getmin(double, double);
09 inline char getmin(char, char);
10
11 int main(int argc, char *argv[])
12 {
13   int n1, n2;
14  cout << "請輸入兩個整數：" ;
15  cin >> n1 >> n2;
16  cout << n1 << " 與 " << n2 << " 之中最小數為 " << getmin(n1, n2) << "\n\n";
17
18  double d1, d2;
19  cout << "請輸入兩個浮點數：" ;
20  cin >> d1 >> d2;
21  cout << d1 << " 與 " << d2 << " 之中最小數為 " << getmin(d1, d2) << "\n\n";
22
23  char c1, c2;
24  cout << "請輸入兩個字元：" ;
25  cin >> c1 >> c2;
26  cout << c1 << " 與 " << c2 << " 之中最小字元為 " << getmin(c1, c2) << "\n\n";
27
28  system("PAUSE");
29  return 0;
30 }
31
32 inline int getmin(int a, int b)
33 {
34  return a<b ? a : b;
35 }
36
37 inline double getmin(double a, double b)
38 {
39  return a<b ? a : b;
40 }
41
```

```
42 inline char getmin(char a, char b)
43 {
44  return a<b ? a : b;
45 }
```

說明

1. 第 32~35 行：呼叫此函式會傳回兩個整數之中的最小數。
2. 第 37~40 行：呼叫此函式會傳回兩個浮點數之中的最小數。
3. 第 42~45 行：呼叫此函式會傳回兩個字元之中的最小字元。

18.5 參考變數

18.5.1 參考變數的使用

在 C++提供的參考(reference)，讓我們可以為變數或常數建立新的別名，也就是說新的別名和原來的變數名稱佔用相同的記憶體位址，一般我們統稱為「參考變數」，參考變數的宣告方式就是在變數名稱之前加上「&」位址運算子。下面即是參考變數的宣告方式與使用方式：

```
int a = 5;     // 宣告整數變數 a 的值為 5
int &b = a;    // 宣告整數型別的參考變數 b，並設定 b 與 a 佔用相同記憶體位址,
               // 此時 a 和 b 兩者的值皆為 5
b = 10;        // 參考變數 b 設為 10，因為 a 和 b 佔用相同記憶體位址,
               // 此時 a 和 b 兩者的值皆為 10
```

乍看之下，感覺參考變數就好像指標變數一樣。其實不是，前面我們說過參考變數和一般變數佔用相同記憶體位址，因此兩者皆可以直接取得記憶體位址的內容；但指標變數和其他變數是分別存放在不同的記憶體位址，目前指標變數本身存放的是某一個變數的記憶體位址，表示指標變數指向某一個變數的位址，因此指標變數要取得所指向變數的值，必須在指標變數之前加上「*」間接運算子。

18.5.2 參考呼叫

函式中的虛引數之前若加上&，表示將此函式的引數傳遞方式設為參考呼叫(call by reference)。所謂的「參考呼叫」就是呼叫程式的實引數與被呼叫程式的虛引數兩者佔用同一個記憶體位址，也就是說在做參數傳遞時，呼叫程式中的實引數是將自己本身的記憶體位址傳給被呼叫程式的虛引數，因此虛引數即可以直接取值並不用像傳址呼叫一樣必須在變數之前加上「*」間接運算子才能取值。因此，以參考呼叫傳遞引數的好處就是被呼叫程式可以透過該引數將值傳回給呼叫程式。下面即為參考呼叫函式宣告與定義的寫法。

```
void fun(int &, double &);     //函式的宣告，虛引數資料型別後加上&
      ⋮
void fun(int &n, double &d)  //函式的定義，虛引數之前加上&
{
      ⋮  // 函式主體
}
```

範例：swap.cpp

使用參考呼叫與傳址呼叫兩種方式來交換兩個變數內的值，並觀察兩者之間虛引數與實引數變化的情形。

執行結果

```
C:\DevC\CH18\swap\swap.exe
==使用參考呼叫進行兩數交換==
請輸入整數 x 的值：10
請輸入整數 y 的值：5
x=10, y=5, x位址=0x6ffe3c, y位址=0x6ffe38
a=10, b=5, a位址=0x6ffe3c, b位址=0x6ffe38, 進入swapbyref()函式進行兩數交換

a=5, b=10, a位址=0x6ffe3c, b位址=0x6ffe38, 離開swapbyref()函式完成兩數交換

兩數交換完成, 結果如下
x=5, y=10, x位址=0x6ffe3c, y位址=0x6ffe38


==使用傳址呼叫進行兩數交換==
請輸入整數 k 的值：10
請輸入整數 z 的值：5
k=10, z=5, k位址=0x6ffe34, z位址=0x6ffe30
進入swapbyadd()函式進行兩數交換
a=0x6ffe34, b=0x6ffe30 *a=10, *b=5, a位址=0x6ffe10, b位址=0x6ffe18
離開swapbyadd()函式完成兩數交換
a=0x6ffe34, b=0x6ffe30 *a=5, *b=10, a位址=0x6ffe10, b位址=0x6ffe18
兩數交換完成, 結果如下
k=5, z=10, k位址=0x6ffe34, z位址=0x6ffe30
請按任意鍵繼續 . . .
```

參考呼叫中實引數與虛引數兩者佔用相同記憶體位址

傳址呼叫中虛引數是指標，因此虛引數指標變數會指向實引數的記憶體位址，且虛引數本身也擁有一個記憶體

程式碼 FileName : swap.cpp

```
01 #include <cstdlib>
02 #include <iostream>
03 #include <iomanip>
04 #include <string>
05 using namespace std;
06
07 void swapbyref(int &, int &);
08 void swapbyadd(int *, int *);
09
10 int main(int argc, char *argv[])
11 {
12   cout << " ==使用參考呼叫進行兩數交換==" << endl;
13   int x, y;
14   cout << " 請輸入整數 x 的值：" ;
15   cin >> x;
16   cout << " 請輸入整數 y 的值：" ;
17   cin >> y;
18   cout << " x=" << x << ", y=" << y
          << ", x 位址=" << &x << ", y 位址=" << &y << endl;
19   swapbyref(x, y);
20   cout << " 兩數交換完成，結果如下" << endl;
21   cout << " x=" << x << ", y=" << y
          << ", x 位址=" << &x << ", y 位址=" << &y << endl;
22
23   cout << endl << endl;
24
25   cout << " ==使用傳址呼叫進行兩數交換==" << endl;
26   int k, z;
27   cout << " 請輸入整數 k 的值：" ;
28   cin >> k;
29   cout << " 請輸入整數 z 的值：" ;
30   cin >> z;
31   cout << " k=" << k << ", z=" << z
          << ", k 位址=" << &k << ", z 位址=" << &z << endl;
32   swapbyadd(&k, &z);
33   cout << " 兩數交換完成，結果如下" << endl;
34   cout << " k=" << k << ", z=" << z
          << ", k 位址=" << &k << ", z 位址=" << &z << endl;
```

```
35   system("PAUSE");
36   return 0;
37 }
38
39 void swapbyref(int &a, int &b)
40 {
41   int t;
42   cout << " a=" << a << ", b=" << b
          << ", a 位址=" << &a << ", b 位址=" << &b
          << ", 進入 swapbyref()函式進行兩數交換" << endl;
44   t=a;
45   a=b;
46   b=t;
47   cout << " a=" << a << ", b=" << b
          << ", a 位址=" << &a << ", b 位址=" << &b
          << ", 離開 swapbyref()函式完成兩數交換" << endl;
48 }
49
50 void swapbyadd(int *a, int *b)
51 {
52   int t;
53   cout << " 進入 swapbyadd()函式進行兩數交換" << endl;
54   cout << " a=" << a << ", b=" << b
          << " *a=" << *a << ", *b=" << *b
          << ", a 位址=" << &a << ", b 位址=" << &b << endl ;
55   t=*a;
56   *a=*b;
57   *b=t;
58   cout << " 離開 swapbyadd()函式完成兩數交換" << endl;
59   cout << " a=" << a << ", b=" << b
          << " *a=" << *a << ", *b=" << *b
          << ", a 位址=" << &a << ", b 位址=" << &b  << endl;
60 }
```

由這個範例可以知道參考變數比指標變數更容易使用，優點如下：

1. 使用參考呼叫時不必將實引數的位址傳給虛引數。
2. 參考呼叫的函式寫法比傳址呼叫更加簡潔，不必使用「*」間接運算子來取值。

3. 參考呼叫的實引數與虛引數佔用相同記憶體位址，而傳址呼叫必須額外產生存放指標變數的記憶體位址，因此參考呼叫較節省記憶體空間。

18.6 string 字串類別

18.6.1 使用 string 類別建立字串物件

傳統 C 語言是使用字元陣列來建立字串，使用 string.h 標頭檔內的字串函式來處理字串；在 C++ 中您可以使用 string 類別來建立字串物件，以及透過 string 類別所提供的成員函式來處理字串；因此使用 C++ 中的 string 類別讓您可以簡化傳統 C 語言的字串處理方式。在 C++ 中若要使用 string 類別必須在程式最開頭先含入 #include <string> ，在 C++ 中若要使用傳統 C 所提供的字串處理函式必須在程式最開頭撰寫 #include <cstring> 或 #include <string.h>。下面是使用 string 類別建立字串物件的四種寫法：

```
string s1("");                  // 建立 s1 字串物件，s1 是一個空字串
string gotop="碁峰";            // 建立 gotop 字串物件，字串內容為 "碁峰"
string movie("可愛巧虎島"); // 建立 movie 字串物件, 字串內容是 "可愛巧虎島"
string name(gotop);             // 建立 name 字串物件，其內容是使用 gotop 字串物件的內容
```

18.6.2 string 類別的運算子

string 類別中定義了許多運算子，讓您不需使用傳統 C 語言字串處理函式即可以直接處理字串在運算式上的使用。例如在 C 語言要連接兩個字串必須使用 strcat()函式，但是在 C++中您只要使用 + 運算子即可以將兩個字串做連接的動作。string 類別常用的運算子說明如下。我們以 str1="abc"、str2="def" 為例。

運算子	功能說明	例	結果
=	將等號右邊的字串指定給左邊的字串物件	str1=str2	str1= "abc"
==	比較兩個字串是否相等	str1==str2	false
!=	比較兩個字串是否不相等	str1!=str2	true
+	連接字串	str1+str2	"abcdef"
+=	連接字串後再指定	str1+=str2	str1="abcdef"
<	比較此運算子左邊的字串是否小於右邊的字串	str1<str2	true
<=	比較此運算子左邊的字串是否小於等於右邊的字串	str1<=str2	true
>	比較此運算子左邊的字串是否大於右邊的字串	str1>str2	false
>=	比較此運算子左邊的字串是否大於等於右邊的字串	str1>=str2	false

範例 ：stropr.cpp

讓使用者輸入帳號與密碼並存入 string 類別的 id 及 pwd，接著再使用== 運算子比較 id 是否為 "gotop" 且 pwd 是否為 "168" ，若兩者同時成立表示帳號與密碼兩者皆正確，即顯示歡迎光臨的訊息，否則顯示離開系統的訊息。

執行結果

程式碼 FileName : stropr.cpp

```
01 #include <cstdlib>
02 #include <iostream>
03 #include <iomanip>
04 #include <string>
05
```

```
06 using namespace std;
07
08 int main(int argc, char *argv[])
09 {
10  string id, pwd;
11  cout << "請輸入帳號：";
12  cin >> id;
13  cout << "請輸入密碼：";
14  cin >> pwd;
15  cout << endl;
16  if(id=="gotop" && pwd=="168")
17  {
18       cout << "帳號：" + id + ", 密碼：" + pwd + " 正確, 歡迎光臨" ;
19  }
20  else
21  {
22       cout << "帳號：" + id + ", 密碼：" + pwd + " 錯誤, 請離開系統" ;
23  }
24  cout << endl;
25  system("PAUSE");
26  return 0;
27 }
```

18.6.3 string 類別常用的成員函式(方法)

下表介紹 string 類別常用的成員函式。其說明如下：

函式	功能說明

assign	語法：string &assign(string &str, size_type start, size_type n);
	功能：由 str 字串的第 start 個字開始取出 n 個字並存放到呼叫 assign 成員函式的 string 字串物件內。
	簡例：① string s("");
	s.assign("David", 1, 4); // s="avid"
	② string s("");
	string s2("七夜怪談");
	s.assign(s2, 2, 4); // s="夜怪"，一個中文字表示兩個字元

append	語法：string &append(string &str, size_type start, size_type n);
	功能：由 str 字串的第 start 個字開始取出 n 個字並連接到呼叫 append 成員函式的 string 字串物件後面。
	簡例：string s1("Peter");
	string s2("七夜怪談");
	① string s("好可怕");
	s2.append(s, 0, 6); // s2="七夜怪談好可怕"
	② s1.append("is△good", 2, 7); // s1="Peter△good"

insert	語法：string &insert(size_type start, string &str, size_type s, size_type n); 功能：將 str 字串的第 s 個字到第 n 個字之間的字串插入到 string 字串物件的第 start 個字。 範例：string s1="C&C++"; string s2="is△good"; s1.insert(5, s2, 2, 8); // s1="C&C++△good"
erase	語法：iterator erase(iterator first, iterator n); 功能：將 string 字串物件的第 first 個字開始刪除 n 個字。 範例：string s1="C&C++程式設計經典"; s1.erase(1,4); //s1="C 程式設計經典"
empty	語法：bool empty() const; 功能：判斷字串是否為空字串，若為空字串傳回 true，反之傳回 false。

find	語法：size_type find(const basic_string& str, size_type pos=0)const;
	功能：由 pos 位置開始往前尋找 string 字串物件中 str 子字串出現的位置，若傳回-1 表示找不到子字串。
	範例：string s1="C&C++程式設計經典 C&C++";
	int a=s1.find("C++"); // a=2
	int b=s1.find("C++", 10); // b=19
	int c=s1.find("C++", 20); //c=-1 因第 20 個字之後並沒 "C++" 子字串

rfind	語法：size_type rfind(const basic_string& str, size_type pos=npos)const;
	功能：由 pos 位置開始往後尋找 string 字串物件中 str 子字串出現的位置，若傳回-1 表示找不到子字串。
	範例：string s1="C&C++程式設計經典 C&C++";
	int a=s1.rfind("C++"); // a=19
	int b=s1.rfind("C++", 10); // a=2
	int c=s1.rfind("C++", 1); // a=-1

c_str	語法：const E *c_str() const;
	功能：將 string 字串轉換成傳統 C 語言字串型別。
	範例：char *s1;
	string s2("VC 2017");
	s1=s2.c_str(); // 將 s2 之 string 字串轉換成傳統
	// C 語言字串，然後再指定給 s1 字元指標

substr	語法：basic_string substr(size_type s, size_type n=npos)const;
	功能：由 string 字串物件中第 s 個字開始取得 n 個字，然後再傳回。
	範例：string s1("可愛巧虎島 YA") ;
	string s2, s3;
	s2=s1.substr(10); // s2="YA"
	s3=s1.substr(4, 6); // s3="巧虎島"

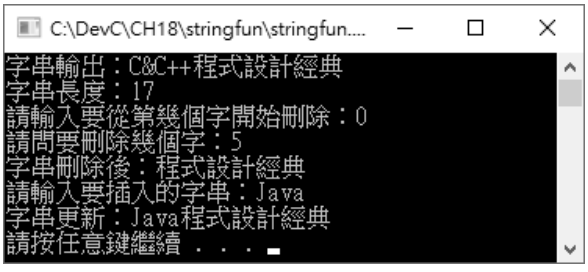
length	語法：size_type length()const;
	功能：取得 string 字串物件的長度。
	範例：string s1("C&C++");
	string s2="程式設計經典" ;
	int a=s1.length(); // a=5
	int b=s2.length(); // b=12

swap	語法：void swap(basic_string& str); 功能：將 string 字串物件與指定的 str 進行互換。兩個字串物件的長度必須相同。 範例：string s1="gotop"; string s2="碁峰"; s1.swap(s2); // s1="碁峰", s2="gotop"
begin	語法：const_iterator begin()const; 功能：傳回 string 字串物件的起始指標。
end	語法：const_iterator end()const; 功能：傳回 string 字串物件的終止指標。 範例：string s1("C&C++"); string s2("△is△good"); s1.append(s2.begin(),s2.end()); // s1="C&C++△is△good"

範例： stringfun.cpp

本例執行時會有一個 str 字串物件，其內容為 "C&C++程式設計經典"，且會顯示該字串的長度。然後會要求您指定 str 由第幾個字開始刪除幾個字，再要求您輸入要插入的子字串，最後將子字串插入到 str 字串之前。執行結果如下圖：

執行結果



程式碼 FileName : stringfun.cpp

```
01 #include <cstdlib>
02 #include <iostream>
03 #include <iomanip>
04 #include <string>
05 using namespace std;
06
07 int main(int argc, char *argv[])
08 {
09   string str("C&C++程式設計經典");
10   int s, e;
11   cout << "字串輸出：" << str << endl;
12   cout << "字串長度：" << str.length() << endl;
13   cout << "請輸入要從第幾個字開始刪除：";
14   cin >> s;
15   cout << "請問要刪除幾個字：";
16   cin >> e;
17   str.erase(s, e);                          //將 str 字串中第 s 個字開始刪除 e 個字
18   cout << "字串刪除後：" << str << endl;
19   string input;
20   cout << "請輸入要插入的字串：" ;
21   cin >> input;
```

```
22   str.insert(0, input, 0, input.length()); //在 str 字串中指定的位置插入子字串
23   cout << "字串更新：" << str << endl;
24   system("PAUSE");
25   return 0;
26 }
```

18.7 習題

一. 選擇題

1. 請問下列何者說明有誤？
   (1) 傳統 C 語言的變數可以在任何位置宣告
   (2) C++的變數可以在任何位置宣告
   (3) 傳統 C 語言變數必須在程式最開頭先宣告才能使用
   (4) C++可以使用 /*…*/ 當多行註解

2. 下例何者是 C++有的而 C 語言沒有的功能？(複選)
   (1) 多載函式　　(2) 多載運算子
   (3) 物件導向技術　　(4) 結構

3. 請問多載的意義？
   (1) 可以讓相同名稱的函式或運算子擁有不同的功能
   (2) 無法讓相同名稱的函式或運算子擁有不同的功能

4. 下例何者敘述正確？
   (1) C 語言可以使用指標變數、參考型別變數、陣列
   (2) C 語言設定 const 變數時一定要設定初值
   (3) C++設定 const 變數時一定要設定初值
   (4) C 語言可以使用 string 類別

5. 如下 C++程式，結果印出？

```
int main(int argc, char *argv[])
{
    int a=5;
    int &b=a;
    b++;
```

```
        cout << a << b;
    }
```

(1) 5　　(2) 55　　(3) 66　　(4) 以上皆非

6. 下列何者是 C++特有的功能？(複選)
   (1) 多載函式　　(2) 使用 cin 輸入資料
   (3) 使用 cin 物件輸出資料　　(4) 物件導向程式設計

7. 下列何者正確？
   (1) C 語言可以使用 bool 型別　　(2) C++無法使用 bool 型別
   (3) C++可以多載函式　　(4) C 語言可以多載函式

8. 欲定義 inline 函式必須在函式傳回值型別之前加上下面哪個保留字？
   (1) inline　　(2) extern　　(3) global　　(4) auto

9. 如何要宣告參考變數，宣告變數時必須在變數名稱之前加上？
   (1) static　　(2) extern　　(3) &　　(4) *

10. 使用 "+" 運算子進行兩個 string 字串物件相加，其意義為？
    (1) 比較兩個字串　　(2) 合併兩個字串
    (3) 指定字串　　(4) 傳回字串的指標

二. 程式設計

1. 利用 inline 配合多載函式撰寫三個相同名稱的 getmax()函式，其中呼叫 getmax()函式可以用來取得兩個整數或兩個浮點數，或兩個字元之中的最大數。

2. 使用多載函式定義三個相同名稱 abs()函式，呼叫 abs()函式可傳入整數、浮點數或長整數的引數，結果會傳回所傳入引數的絕對值。

3. 輸入身份證號碼並指定給 string 字串物件，然後透過 string 類別所提供的成員函式判斷所輸入的身份證是否合法。關於身份證號碼的演算法請參閱 12.5.2 節。