



# C++ Programming: From Problem Analysis to Program Design, Fifth Edition

## Chapter 7: User-Defined Functions II

برمجة C++ من تحليل المشكلات إلى تصميم البرنامج  
الطبعة الخامسة

الفصل 7: وظائف محددة من قبل المستخدم II

- ✓ Download textbook
- ✓ Read through
- ✓ Try examples using Dev C++
- ✓ Try End of chapter problems
- ✓ Do that in groups

✓ تنزيل الكتاب المدرسي

✓ من خلال قراءة

✓ جرب أمثلة باستخدام Dev C ++

✓ جرب مشاكل نهاية الفصل

✓ افعل ذلك في مجموعات

# Void Functions

- Void functions and value-returning functions have similar structures
  - Both have a heading part and a statement part
- User-defined void functions can be placed either before or after the function `main`
- If user-defined void functions are placed after the function `main`
  - The function prototype must be placed before the function `main`

• الدوال الفارغة ووظائف إرجاع القيمة لها هياكل مماثلة

– كلاهما يحتوي على جزء رئيسي وجزء بيان

• يمكن وضع وظائف الفراغ المعرفة من قبل المستخدم إما قبل الوظيفة أو بعدها رئيسي

• إذا تم وضع وظائف خالية من قبل المستخدم بعد الوظيفة رئيسي

– يجب وضع النموذج الأولي للوظيفة قبل الوظيفة رئيسي

# Void Functions (cont'd.)

- A void function does not have a return type
  - `return` statement without any value is typically used to exit the function early
- Formal parameters are optional
- A call to a void function is a stand-alone statement

- لا يوجد نوع إرجاع للدالة الفارغة
  - إرجاع عادةً ما يتم استخدام العبارة بدون أي قيمة للخروج من الوظيفة مبكرًا
- المعلومات الرسمية اختيارية
- استدعاء دالة باطل هو بيان مستقل

# Void Functions (cont'd.)

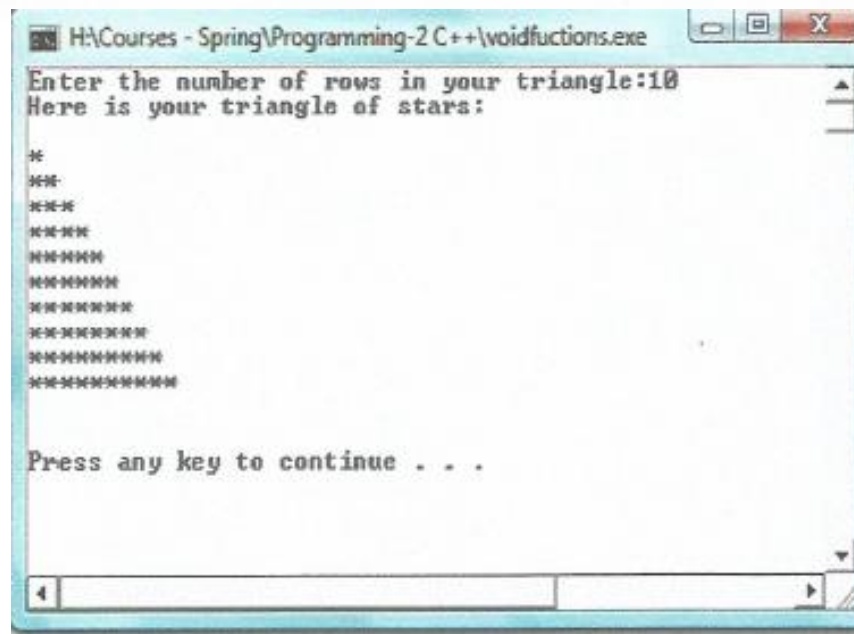
Example 7-3 P364 in textbook

وظائف باطلة (تابع)  
مثال 7-3 P364 في الكتاب المدرسي

Ex: write a program that print a triangle of stars of  $n$  rows. Your program should ask the user for  $n$  and then uses a function *printStars* to draw the required triangle.



Expected output  
for  $n=10$



```

1 #include<iostream>
2 using namespace std;
3
4 void printStars(int n);
5
6 int main()
7 {
8     int nRows;
9
10    cout<<"Enter the number of rows in your triangle:";
11    cin>>nRows;
12    cout<<"Here is your triangle of stars: "<<endl<<endl;
13    printStars(nRows);
14
15    return 0;
16 }
17
18 void printStars(int n)
19 {
20     int i,j;
21
22     //using nested FOR to print
23     //the triangle of stars
24     for (i = 1; i <= n; i++)
25     {
26         for (j = 1; j <= i; j++)
27             cout << '*';
28         cout << endl;
29     }
30 }

```

▷ Notice the way we call the void fns. !

# Value and Reference Parameters

معلومات القيمة والمرجع

- Value parameter: a formal parameter that receives a copy of the content of corresponding actual parameter
- Reference parameter: a formal parameter that receives the location (memory address) of the corresponding actual parameter

- قيمة معامل: معلمة رسمية تتلقى نسخة من محتوى المعلمة الفعلية المقابلة
- المرجعي معامل: معلمة رسمية تتلقى الموقع (عنوان الذاكرة) للمعلمة الفعلية المقابلة

# Function Value Parameters

- If a formal parameter is a **value** parameter
  - The value of the corresponding actual parameter is copied into it
- The value parameter has its own copy of the data
- During program execution
  - The value parameter manipulates the data stored in its own memory space
- إذا كانت المعلمة الرسمية هي **القيمة** معامل
  - يتم نسخ قيمة المعلمة الفعلية المقابلة فيه
  - معلمة القيمة لها نسختها الخاصة من البيانات
  - أثناء تنفيذ البرنامج
  - تعالج معلمة القيمة البيانات المخزنة في مساحة الذاكرة الخاصة بها



**EXAMPLE 7-4**

The following program shows how a formal parameter of a primitive data type works.

```
//Example 7-4
//Program illustrating how a value parameter works.

#include <iostream>

using namespace std;

void funcValueParam(int num);

int main()
{
    int number = 6;                                //Line 1

    cout << "Line 2: Before calling the function "
         << "funcValueParam, number = " << number
         << endl;                                    //Line 2

    funcValueParam(number);                          //Line 3

    cout << "Line 4: After calling the function "
         << "funcValueParam, number = " << number
         << endl;                                    //Line 4

    return 0;
}
```

```

void funcValueParam(int num)
{
    cout << "Line 5: In the function funcValueParam, "
          << "before changing, num = " << num
          << endl;                                //Line 5

    num = 15;                                       //Line 6

    cout << "Line 7: In the function funcValueParam, "
          << "after changing, num = " << num
          << endl;                                //Line 7
}

```

### Sample Run:

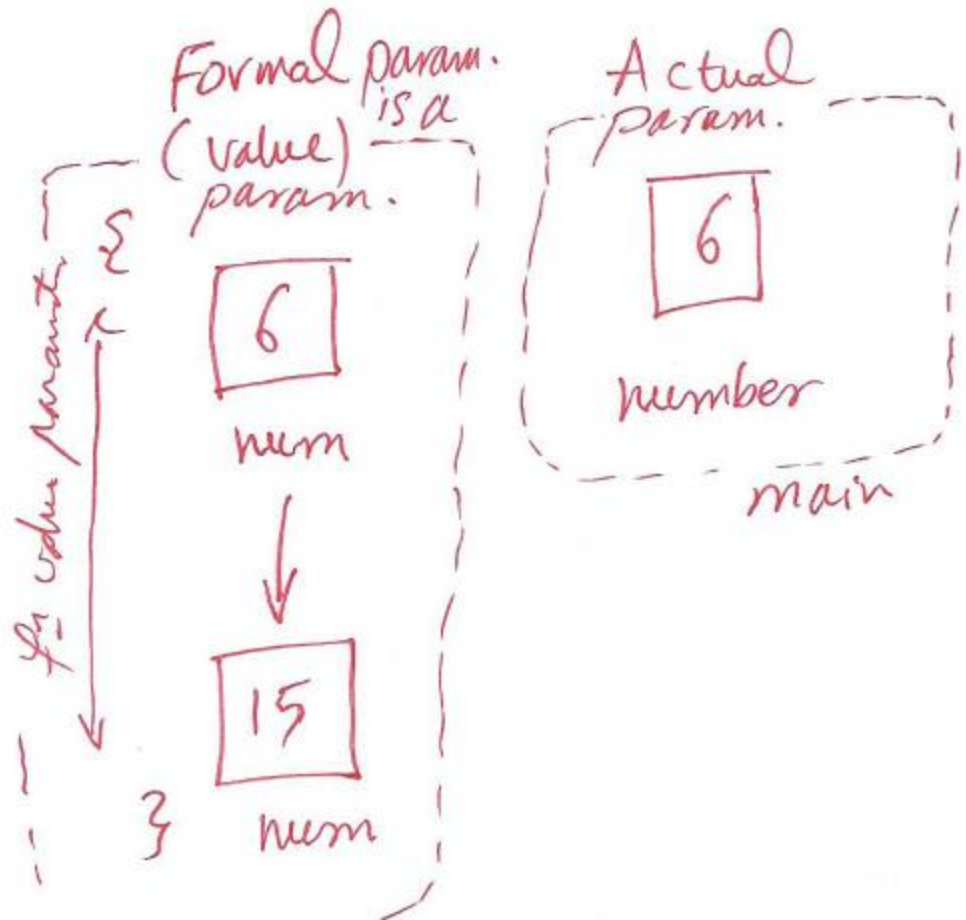
```

Line 2: Before calling the function funcValueParam, number = 6
Line 5: In the function funcValueParam, before changing, num = 6
Line 7: In the function funcValueParam, after changing, num = 15
Line 4: After calling the function funcValueParam, number = 6

```

The changes that happened in the fn. to the copy "num" didn't affect the value of number, because:

VALUE parameters take a copy



# Before using Call by Reference

## We Check how **Pointers** work in C++

قبل استخدام الاتصال بالمرجع

نتحقق من كيفية القيام بذلك **المؤشرات** العمل في C++

- Recall the data type `int`. The set of values belonging to this data type includes integers that range between  $-2147483648$  and  $2147483647$ , and the operations allowed on these values are the arithmetic operators.
- Now, we describe the **pointer** data type.

- أذكر نوع البيانات `int`. تتضمن مجموعة القيم التي تنتمي إلى نوع البيانات هذا أعدادًا صحيحة تتراوح بين  $-2147483648$  و  $2147483647$ ، والعمليات المسموح بها على هذه القيم هي العوامل الحسابية.
- الآن ، نصف **المؤشر** نوع البيانات.

## Pointers in C++

- The values belonging to pointer data types are the memory addresses of your computer.
- The set of values of a pointer data type—is the addresses (memory locations), a pointer variable is a variable whose content is an address, that is, a memory location.
- There is no name associated with the pointer data type in C++.
- القيم التي تنتمي إلى أنواع بيانات المؤشر هي عناوين ذاكرة جهاز الكمبيوتر الخاص بك.
- مجموعة القيم الخاصة بنوع بيانات المؤشر —هي العناوين (مواقع الذاكرة) ، متغير المؤشر هو متغير محتواه عنوان ، أي موقع ذاكرة.
- لا يوجد اسم مرتبط بنوع بيانات المؤشر في C++.

# المؤشرات في C++ Pointers in C++

- In C++, you declare a pointer variable by using the asterisk symbol (\*) between the data type and the variable name. The asterisk is called *dereferencing operator* or *indirection operator*.

- Example:

```
int *p;
```

- In C++, the ampersand (&), called the *referencing operator, address of operator*, is a unary operator that returns the address of its operand.

• في C++، تقوم بتعريف متغير مؤشر باستخدام رمز العلامة النجمية (\*) بين نوع البيانات واسم المتغير. النجمة تسمى *عامل الإسناد* أو *عامل المراوغة*.

• مثال:

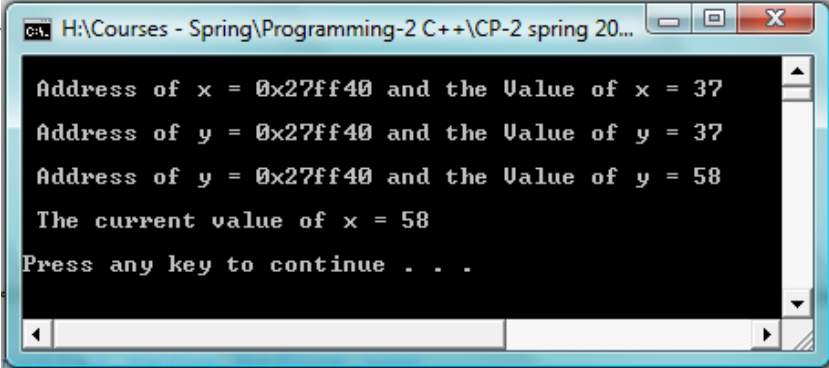
```
int * p
```

• في C++، علامة العطف (&)، ودعا *عامل الإشارة*، *عنوان المشغل*، هو عامل تشغيل أحادي يقوم بإرجاع عنوان معاملة.

# Pointers in C++

## Example 14-3 P799 in textbook

```
1 #include <iostream>
2
3 using namespace std;
4 int main()
5 {
6     int *y;
7     int x;
8
9     x = 37;
10
11     cout << "\n Address of x = " << &x << " and the Value of x = " << x<<endl<<endl;
12
13     y = &x;           //must set initial value before use
14                       //Any change in Y will affect x and ViceVersa
15     cout << " Address of y = " << y << " and the Value of y = " << *y<<endl<<endl;
16
17     *y = 58;          //must use dereferencing operator to access content
18     cout << " Address of y = " << y << " and the Value of y = " << *y<<endl<<endl;
19
20     cout << " The current value of x = " << x << endl<<endl;
21
22     system("PAUSE");
23     return 0;
24 }
25
```

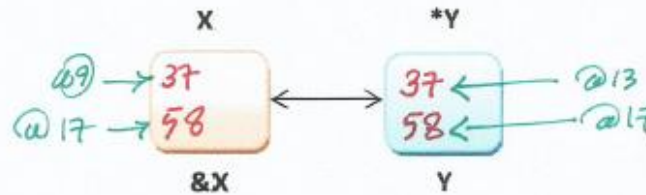


```
Address of x = 0x27ff40 and the Value of x = 37
Address of y = 0x27ff40 and the Value of y = 37
Address of y = 0x27ff40 and the Value of y = 58
The current value of x = 58
Press any key to continue . . .
```

```

1 #include <iostream>
2
3 using namespace std;
4 int main()
5 {
6     int *y;
7     int x;
8
9     ✓ x = 37;
10
11     cout << "\n Address of x = " << &x << " and the Value of x = " << x << endl << endl;
12
13     ✓ y = &x;           //must set initial value before use
14                        //Any change in Y will affect x and ViceVersa
15     cout << " Address of y = " << y << " and the Value of y = " << *y << endl << endl;
16
17     ✓ *y = 58;          //must use dereferencing operator to access content
18     cout << " Address of y = " << y << " and the Value of y = " << *y << endl << endl;
19
20     cout << " The current value of x = " << x << endl << endl;
21
22     system("PAUSE");
23     return 0;
24 }
25

```



```

H:\Courses - Spring\Programming-2 C++\CP-2 spring 20...
Address of x = 0x27ff40 and the Value of x = 37
Address of y = 0x27ff40 and the Value of y = 37
Address of y = 0x27ff40 and the Value of y = 58
The current value of x = 58
Press any key to continue . . .

```



# Reference Variables as Parameters to functions

- المتغيرات المرجعية كمعاملات للوظائف  
If a formal parameter is a reference parameter
  - It receives the memory address of the corresponding actual parameter
- A reference parameter stores the address of the corresponding actual parameter
- During program execution to manipulate data
  - The address stored in the reference parameter directs it to the memory space of the corresponding actual parameter
- إذا كانت المعلمة الرسمية هي معلمة مرجعية
  - يتلقى عنوان الذاكرة للمعلمة الفعلية المقابلة
- تخزن المعلمة المرجعية عنوان المعلمة الفعلية المقابلة
- أثناء تنفيذ البرنامج لمعالجة البيانات
  - العنوان المخزن في المعلمة المرجعية يوجهها إلى مساحة الذاكرة للمعلمة الفعلية المقابلة

# Reference Variables as Parameters

(cont'd.) المتغيرات المرجعية كمعاملات (تابع)

- Reference parameters can:
  - Pass one or more values from a function
  - Change the value of the actual parameter
- Reference parameters are useful in three situations:
  - Returning more than one value
  - Changing the actual parameter
  - When passing the address would save memory space and time
- يمكن للمعاملات المرجعية :
  - قم بتمرير قيمة واحدة أو أكثر من دالة
  - غير قيمة المعلمة الفعلية
- المعاملات المرجعية مفيدة في ثلاث حالات :
  - إرجاع أكثر من قيمة
  - تغيير المعلمة الفعلية
  - عند تمرير العنوان سيوفر مساحة الذاكرة والوقت

# Example 7-5

## P369 in Text

مثال 7-5  
P369 في النص

Write a program that reads a course score (a value between 0 and 100) and determines the student's course grade using the following function prototypes:

- void getScore(int& score);
- void printGrade(int score);

اكتب برنامجًا يقرأ درجة الدورة التدريبية (قيمة بين 0 و 100) ويحدد درجة المقرر الدراسي للطالب باستخدام نماذج الوظائف الأولية التالية:

- getScore باطلة (int & Score)؛
- printGrade باطلة (int) درجة؛

مثال 5-7: احسب الدرجة  
//This program reads a course score and prints the  
//associated course grade.

# Grade

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

void getScore(int& score);
void printGrade(int score);

int main()
{
    int courseScore;

    cout << "Line 1: Based on the course score, \n"
         << "    this program computes the "
         << "course grade." << endl;

    getScore(courseScore);

    printGrade(courseScore);

    return 0;
}
```

Notice the  
Structured  
Programming  
Approach!

//Line 1

//Line 2

//Line 3

```

void getScore(int& score)
{
    cout << "Line 4: Enter course score: ";
    cin >> score;
    cout << endl << "Line 6: Course score is "
        << score << endl;
}

```

//Line 4  
//Line 5

# Grade

مثال 7-5: حساب التقدير (تابع)

```

void printGrade(int cScore)
{
    cout << "Line 7: Your grade for the course is "; //Line 7

    if (cScore >= 90) //Line 8
        cout << "A." << endl;
    else if (cScore >= 80)
        cout << "B." << endl;
    else if (cScore >= 70)
        cout << "C." << endl;
    else if (cScore >= 60)
        cout << "D." << endl;
    else
        cout << "F." << endl;
}

```

**Sample Run:** In this sample run, the user input is shaded.

Line 1: Based on the course score,  
this program computes the course grade.

Line 4: Enter course score: 85

Line 6: Course score is 85

Line 7: Your grade for the course is B.

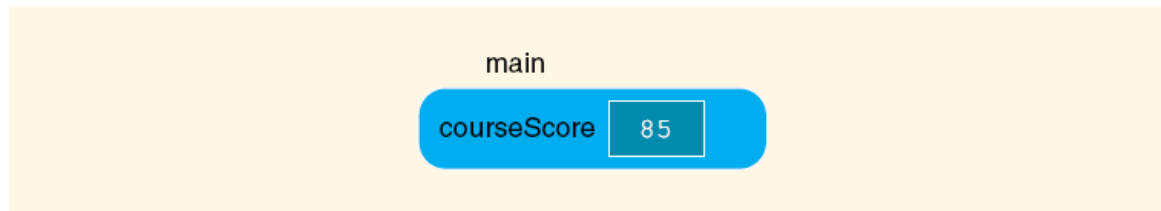
executes `main`



cont'd.)

مثال 7-5: حساب التقدير (تابع)

**FIGURE 7-4** Variable `courseScore` and the parameter `cScore` executes



**FIGURE 7-3** Variable `courseScore` after the statement in Line 6 is executed and control goes back to `main`

# Value and Reference Parameters and Memory Allocation

معلومات القيمة والمرجع وتخصيص الذاكرة

- When a function is called
  - Memory for its formal parameters and variables declared in the body of the function (called local variables) is allocated in the function data area
- In the case of a value parameter
  - The value of the actual parameter is copied into the memory cell of its corresponding formal parameter
- عندما يتم استدعاء وظيفة
  - يتم تخصيص ذاكرة المعلومات والمتغيرات الرسمية المعلنة في جسم الوظيفة (تسمى المتغيرات المحلية) (في منطقة بيانات الوظيفة)
- في حالة معلمة القيمة
  - يتم نسخ قيمة المعلمة الفعلية في خلية الذاكرة للمعلمة الرسمية المقابلة لها

# Value and Reference Parameters and Memory Allocation (cont'd.)

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

- In the case of a reference parameter
  - The address of the actual parameter passes to the formal parameter
- Content of formal parameter is an address
- During execution, changes made by the formal parameter permanently change the value of the actual parameter
- Stream variables (e.g., `ifstream`) should be passed by reference to a function
- في حالة وجود معلمة مرجعية
  - يمر عنوان المعلمة الفعلية إلى المعلمة الرسمية
  - محتوى المعلمة الرسمية هو عنوان
  - أثناء التنفيذ ، تؤدي التغييرات التي يتم إجراؤها بواسطة المعلمة الرسمية إلى تغيير قيمة المعلمة الفعلية بشكل دائم
  - متغيرات التدفق (على سبيل المثال ، `ifstream`) بالرجوع إلى دالة



# Value and Reference Parameters and Memory Allocation (cont'd.)

## EXAMPLE 7-6

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

The following program shows how reference and value parameters work.

```
//Example 7-6: Reference and value parameters

#include <iostream>

using namespace std;

void funOne(int a, int& b, char v);
void funTwo(int& x, int y, char& w);

int main()
{
    int num1, num2;
    char ch;

    num1 = 10; //Line 1
    num2 = 15; //Line 2
    ch = 'A'; //Line 3

    cout << "Line 4: Inside main: num1 = " << num1
         << ", num2 = " << num2 << ", and ch = "
         << ch << endl; //Line 4

    funOne(num1, num2, ch); //Line 5

    cout << "Line 6: After funOne: num1 = " << num1
         << ", num2 = " << num2 << ", and ch = "
         << ch << endl; //Line 6

    funTwo(num2, 25, ch); //Line 7

    cout << "Line 8: After funTwo: num1 = " << num1
         << ", num2 = " << num2 << ", and ch = "
         << ch << endl; //Line 8

    return 0;
}
```

# Value and Reference Parameters and Memory Allocation (cont'd.)

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

```
void funOne(int a, int& b, char v)
{
    int one;

    one = a; //Line 9
    a++; //Line 10
    b = b * 2; //Line 11
    v = 'B'; //Line 12

    cout << "Line 13: Inside funOne: a = " << a
          << ", b = " << b << ", v = " << v
          << ", and one = " << one << endl; //Line 13
}

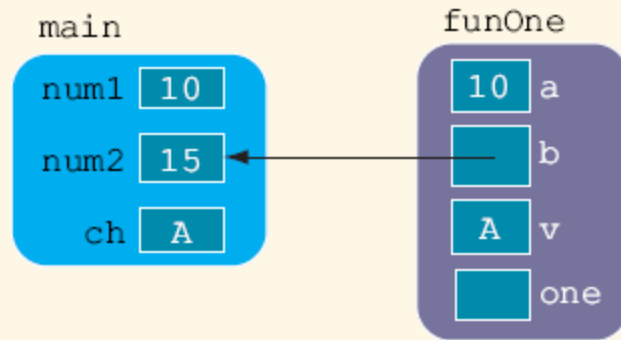
void funTwo(int& x, int y, char& w)
{
    x++; //Line 14
    y = y * 2; //Line 15
    w = 'G'; //Line 16

    cout << "Line 17: Inside funTwo: x = " << x
          << ", y = " << y << ", and w = " << w
          << endl; //Line 17
}
```

## Sample Run:

```
Line 4: Inside main: num1 = 10, num2 = 15, and ch = A
Line 13: Inside funOne: a = 11, b = 30, v = B, and one = 10
Line 6: After funOne: num1 = 10, num2 = 30, and ch = A
Line 17: Inside funTwo: x = 31, y = 50, and w = G
Line 8: After funTwo: num1 = 10, num2 = 31, and ch = G
```

eters  
t'd.)



معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

**FIGURE 7-6** Values of the variables just before the statement in Line 9 executes

# Value and Reference Parameters and Memory Allocation (cont'd.)

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

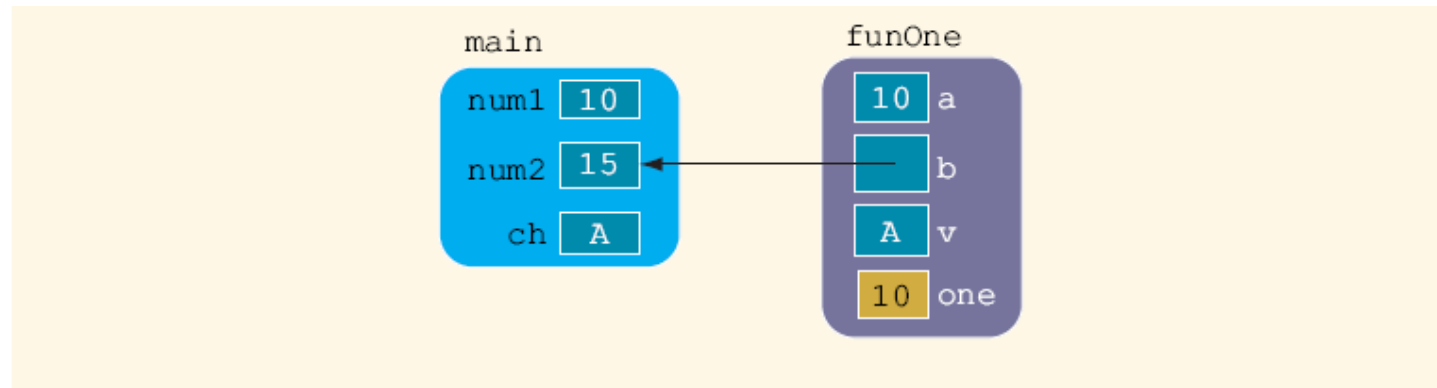


FIGURE 7-7 Values of the variables after the statement in Line 9 executes

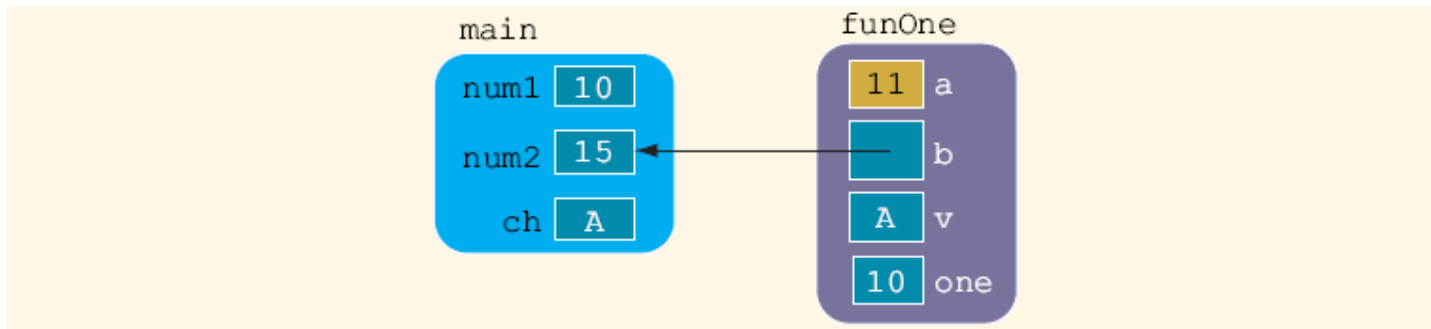


FIGURE 7-8 Values of the variables after the statement in Line 10 executes

# Value and Reference Parameters and Memory Allocation (cont'd.)

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

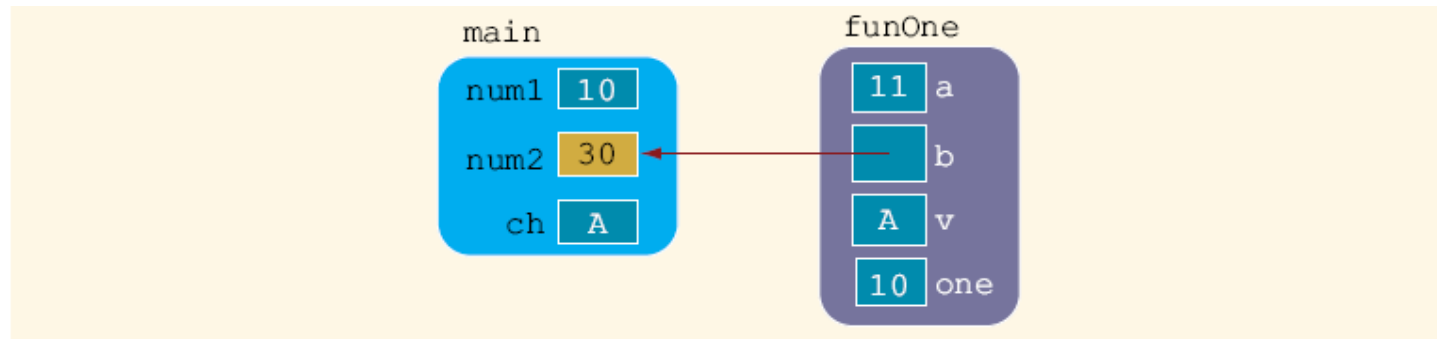


FIGURE 7-9 Values of the variables after the statement in Line 11 executes

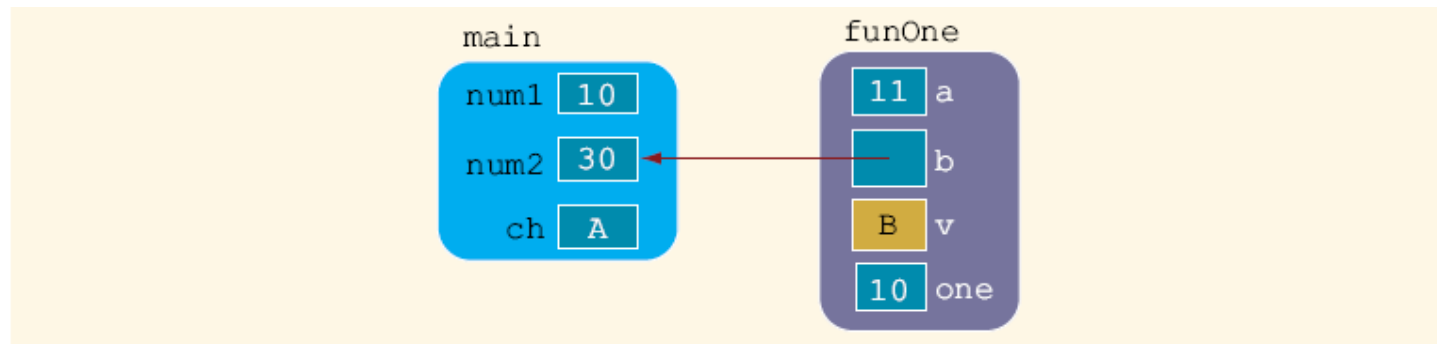


FIGURE 7-10 Values of the variables after the statement in Line 12 executes

# Value and Reference Parameters and Memory Allocation (cont'd.)

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

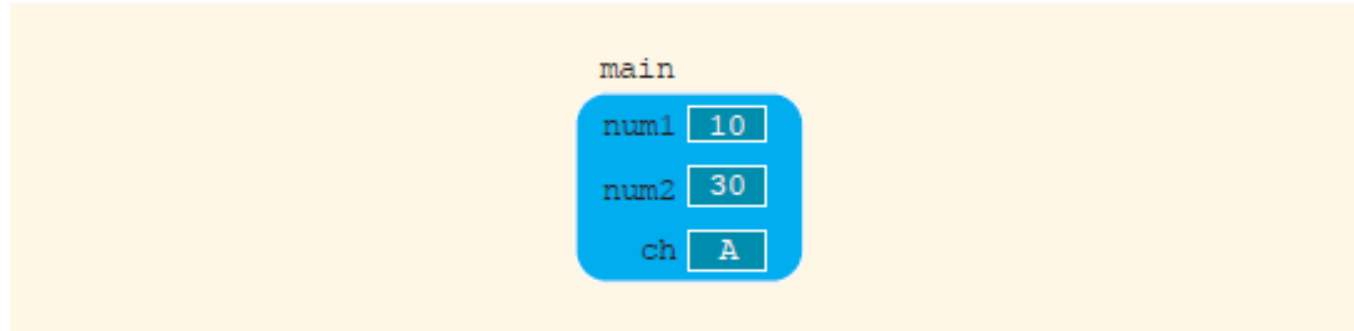


FIGURE 7-11 Values of the variables when control goes back to Line 6

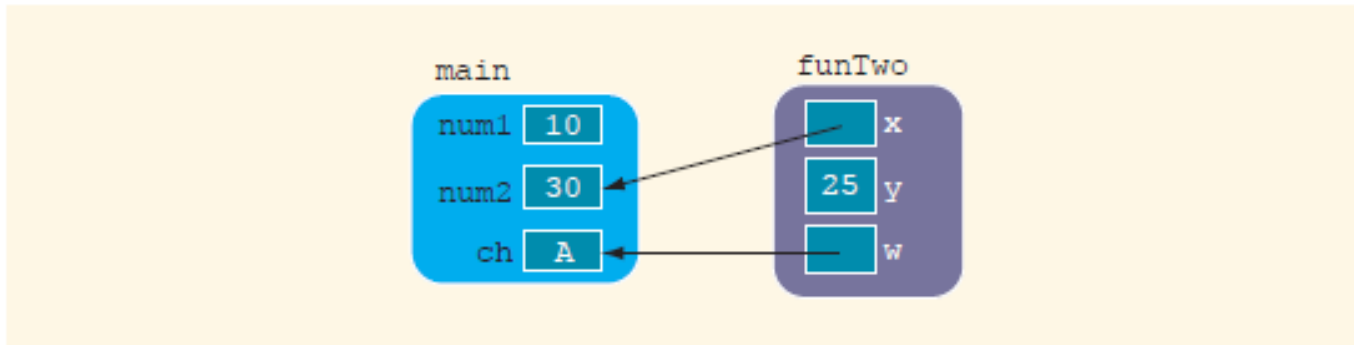


FIGURE 7-12 Values of the variables before the statement in Line 14 executes

# Value and Reference Parameters and Memory Allocation (cont'd.)

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

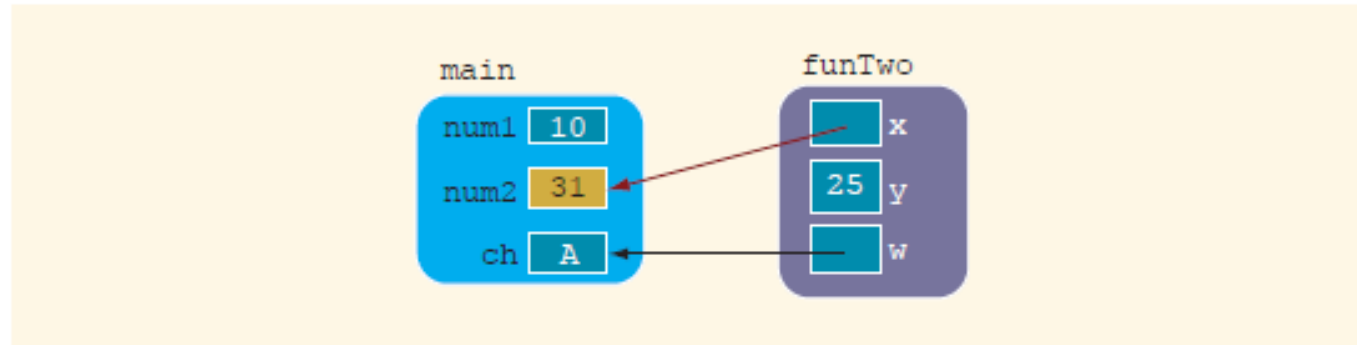


FIGURE 7-13 Values of the variables after the statement in Line 14 executes

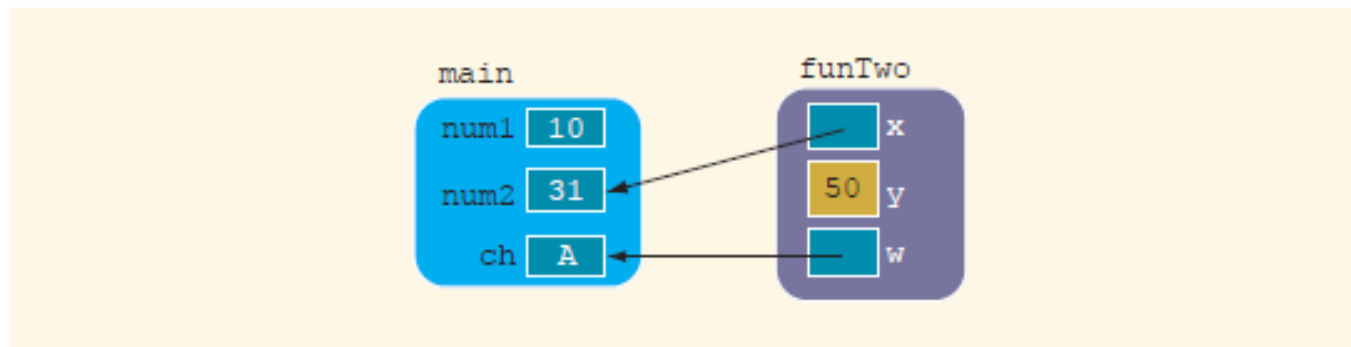


FIGURE 7-14 Values of the variables after the statement in Line 15 executes

# Value and Reference Parameters and Memory Allocation (cont'd.)

معلومات القيمة والمرجع وتخصيص الذاكرة (تابع)

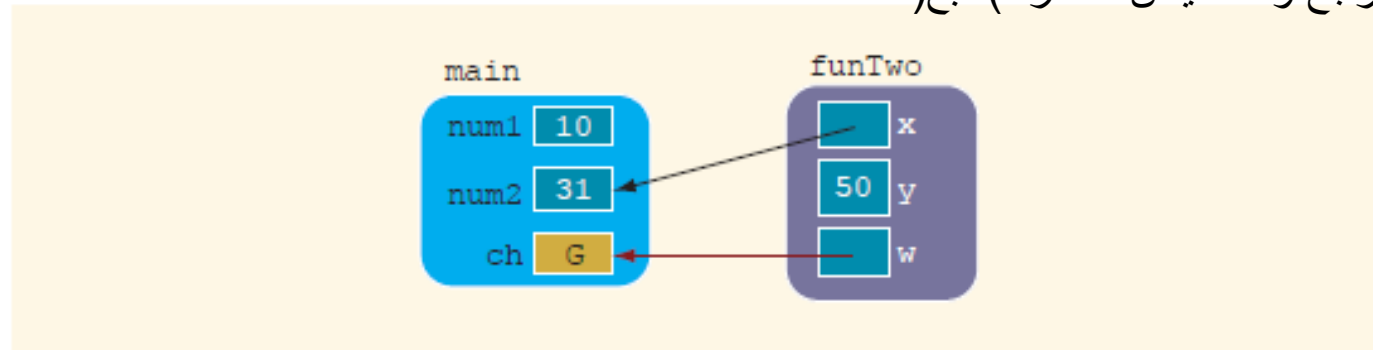


FIGURE 7-15 Values of the variables after the statement in Line 16 executes

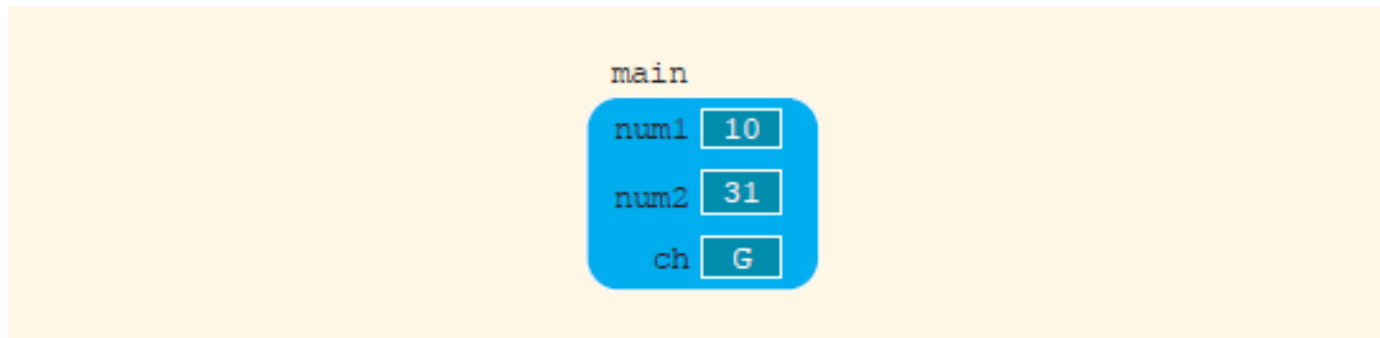


FIGURE 7-16 Values of the variables when control goes to Line 8



# Reference Parameters and Value-Returning Functions

المعاملات المرجعية ووظائف إرجاع القيمة

- You can also use reference parameters in a value-returning function
  - Not recommended
- By definition, a value-returning function returns a single value
  - This value is returned via the return statement
- If a function needs to return more than one value, you should change it to a void function and use the appropriate reference parameters to return the values
- يمكنك أيضًا استخدام المعاملات المرجعية في دالة إرجاع القيمة
  - لا ينصح
- حسب التعريف ، تقوم دالة إرجاع القيمة بإرجاع قيمة واحدة
  - يتم إرجاع هذه القيمة عبر بيان الإرجاع
- إذا احتاجت إحدى الوظائف إلى إرجاع أكثر من قيمة واحدة ، فيجب عليك تغييرها إلى دالة باطلة واستخدام معلمات المرجع المناسبة لإرجاع القيم

# Scope of an Identifier

نطاق المعرف

- The scope of an identifier refers to where in the program an identifier is accessible
- Local identifier: identifiers declared within a function (or block)
- Global identifier: identifiers declared outside of every function definition
- C++ does not allow nested functions
  - The definition of one function cannot be included in the body of another function

• يشير نطاق المعرف إلى المكان الذي يمكن فيه الوصول إلى المعرف في البرنامج

• المعرف المحلي: المعارف المعلنة داخل دالة (أو كتلة)

• المعرف العام: المعارف المعلنة خارج كل تعريف دالة

• لا يسمح C++ بالوظائف المتداخلة

– لا يمكن تضمين تعريف وظيفة واحدة في جسم وظيفة أخرى

# Scope of an Identifier (cont'd.)

- Rules when an identifier accessed:
  - Global identifiers
    - Declared before function definition
    - Function name different from identifier
    - Parameters to function have different names
    - All local identifiers have different names
  - القواعد عند الوصول إلى المعرف:
    - المعارف العالمية
      - أعلن قبل تعريف الوظيفة
      - اسم الوظيفة مختلف عن المعرف
      - المعلومات لتعمل لها أسماء مختلفة
      - جميع المعارف المحلية لها أسماء مختلفة

# Scope of an Identifier (cont'd.)

- Nested block
  - Identifier accessible from declaration to end of block
  - Within nested blocks if no identifier with same name exists
- Scope of function name similar to scope of identifier declared outside any block

– كتلة متداخلة

- المعرف يمكن الوصول إليه من الإعلان إلى نهاية الكتلة
- داخل الكتل المتداخلة في حالة عدم وجود معرف بنفس الاسم
- نطاق اسم الوظيفة مشابه لنطاق المعرف المعلن خارج أي كتلة

# Scope of an Identifier (cont'd.)

- Some compilers initialize global variables to default values
- The operator `::` is called the scope resolution operator
- By using the scope resolution operator
  - A global variable declared before the definition of a function (block) can be accessed by the function (or block)
  - Even if the function (or block) has an identifier with the same name as the variable

• يقوم بعض المترجمين بتهيئة المتغيرات العامة إلى القيم الافتراضية

• المشغل `::` يسمى مشغل تحليل النطاق

• باستخدام عامل تحليل النطاق

– تم الإعلان عن متغير عام قبل أن يمكن الوصول إلى تعريف دالة (كتلة (بواسطة الوظيفة) أو الكتلة (

– حتى لو كانت الوظيفة (أو الكتلة (لها معرف بنفس اسم المتغير

# Scope of an Identifier (cont'd.)

- C++ provides a way to access a global variable declared after the definition of a function
  - In this case, the function must not contain any identifier with the same name as the global variable
- يوفر C++ طريقة للوصول إلى متغير عام تم الإعلان عنه بعد تعريف الوظيفة
  - في هذه الحالة ، يجب ألا تحتوي الوظيفة على أي معرف يحمل نفس اسم المتغير العام

# Global Variables, Named Constants, and Side Effects

المتغيرات العامة والثوابت المسماة والآثار الجانبية

- Using global variables causes side effects
- A function that uses global variables is not independent
- If more than one function uses the same global variable and something goes wrong
  - It is difficult to find what went wrong and where
  - Problems caused in one area of the program may appear to be from another area
- Global named constants have no side effects

- استخدام المتغيرات العالمية يسبب آثاراً جانبية
- الوظيفة التي تستخدم المتغيرات العالمية ليست مستقلة
- إذا كانت هناك أكثر من دالة تستخدم نفس المتغير العام وحدث خطأ ما
  - من الصعب العثور على الخطأ الذي حدث وأين
  - قد يبدو أن المشكلات التي تحدث في أحد مجالات البرنامج ناتجة عن منطقة أخرى
- الثوابت العالمية المسماة ليس لها آثار جانبية

# Static and Automatic Variables

المتغيرات الثابتة والتلقائية

- Automatic variable: memory is allocated at block entry and de-allocated at block exit
  - By default, variables declared within a block are automatic variables
- Static variable: memory remains allocated as long as the program executes
  - Variables declared outside of any block are static variables
  - Declare a static variable within a block by using the reserved word `static`

- متغير تلقائي: يتم تخصيص الذاكرة عند إدخال الكتلة وإلغاء تخصيصها عند خروج الكتلة
  - بشكل افتراضي ، المتغيرات المعلنة داخل الكتلة هي متغيرات تلقائية
- متغير ثابت: تظل الذاكرة مخصصة ما دام البرنامج يعمل
  - المتغيرات المعلنة خارج أي كتلة هي متغيرات ثابتة
  - قم بتعريف متغير ثابت داخل كتلة باستخدام الكلمة المحجوزة ثابتة



# Static and Automatic Variables (cont'd.)

المتغيرات الثابتة والتلقائية (تابع)

- The syntax for declaring a static variable is:

- The statement

```
static int static dataType identifier;
```

declares `x` to be a static variable of the type `int`

- Static variables declared within a block are local to the block
  - Their scope is the same as any other local identifier of that block

• صيغة التصريح عن متغير ثابت هي:

• البيان

ثابت `int x`؛

يعلن `x` ليكون متغير ثابت من النوع `int`

- المتغيرات الثابتة المعلنه داخل كتلة محلية بالنسبة للكتلة

– نطاقها هو نفسه مثل أي معرف محلي آخر لتلك الكتلة

# Static and Automatic Variables (cont'd.)

المتغيرات الثابتة والتلقائية (تابع)

مثال 7-9 في كتاب نصي ص 391

Example 7-9 in Text Book P391

# Debugging: Using Drivers and Stubs

التصحيح :استخدام برامج التشغيل و Stubs

- Driver program: separate program to test a function
  - Make sure that each function is working properly
- When results calculated by one function are needed in another function
  - Use a stub: function that is not fully coded

- برنامج السائق :برنامج منفصل لاختبار وظيفة
  - تأكد من أن كل وظيفة تعمل بشكل صحيح
- عندما تكون النتائج المحسوبة بواسطة دالة واحدة مطلوبة في دالة أخرى
  - استخدم كعب :دالة غير مشفرة بالكامل

# Function Overloading: An Introduction

وظيفة التحميل الزائد :مقدمة

- In a C++ program, several functions can have the same name
  - This is called function overloading or overloading a function name

- في برنامج C ++ ، يمكن أن يكون للعديد من الوظائف نفس الاسم
  - هذا يسمى وظيفة الزائد أو زيادة التحميل على اسم الوظيفة

# Function Overloading (cont'd.)

وظيفة التحميل الزائد (تابع)

- Two functions are said to have different formal parameter lists if both functions have:
  - A different number of formal parameters, or
  - If the number of formal parameters is the same, then the data type of the formal parameters, in the order you list them, must differ in at least one position

• يقال أن وظيفتين لهما قوائم معلمات رسمية مختلفة إذا كان لكلتا الوظيفتين:

- عدد مختلف من المعلمات الرسمية ، أو
- إذا كان عدد المعلمات الرسمية هو نفسه ، فيجب أن يختلف نوع بيانات المعلمات الرسمية ، بالترتيب الذي تدرجها فيه ، في موضع واحد على الأقل

```

void functionSix(int x, double y, char ch)
void functionSeven(int one, double u, char firstCh)
void functionThree(double y, int x)
int functionFour(char ch, int x, double y)
int functionFive(char ch, int x, string name)

```

## • Loading (cont'd.)

- تحتوي جميع الوظائف التالية على قوائم معلمات رسمية مختلفة:
- The following functions all have different formal parameter lists:
- الوظائف التالية لها نفس قائمة المعلمات الرسمية:
- The following functions have the same formal parameter list:

# Function Overloading (cont'd.)

- Function overloading: creating several functions with the same name
- The signature of a function consists of the function name and its formal parameter list
- Two functions have different signatures if they have either different names or different formal parameter lists
- Note that the signature of a function does not include the return type of the function

- وظيفة التحميل الزائد: إنشاء عدة وظائف بنفس الاسم
- يتكون توقيع الوظيفة من اسم الوظيفة وقائمة المعلمات الرسمية الخاصة بها
- وظيفتان لهما توقيعان مختلفان إذا كانت لهما أسماء مختلفة أو قوائم معلمات رسمية مختلفة
- لاحظ أن توقيع الدالة لا يتضمن نوع إرجاع الدالة

# Function Overloading (cont'd.)

- تصحيح الوظيفة الزائدة:  
• Correct function overloading:

```
void functionXYZ()  
void functionXYZ(int x, double y)  
void functionXYZ(double one, int y)  
void functionXYZ(int x, double y, char ch)
```

- خطأ في بناء الجملة:  
• Syntax error:

```
void functionABC(int x, double y)  
int functionABC(int x, double y)
```



# Function Overloading (cont'd.)

// وظيفة التحميل الزائد أكبر

## // Overloading Function LARGER

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

// Function arguments are of different data type
int larger(int x, int y);
char larger(char first, char second);

int main()
{
    int a, b;
    char c, d;

    cout << "Enter two integers\n";
    cin >> a >> b;
```

```
cout << "The larger number is: " << larger(a,b) << endl<<endl;
```

```
cout << "Enter two characters\n";
```

```
cin >> c >> d;
```

```
cout << "The larger character is: " << larger(c,d) << endl<<endl;
```

```
}
```

```
int larger(int x, int y)
```

```
{
```

```
    if(x>y)
```

```
        return x;
```

```
    return y;
```

```
}
```

```
char larger(char x, char y)
```

```
{
```

```
    if(x>y)
```

```
        return x;
```

```
    return y;
```

```
}
```

# Programming Example: Classify Numbers

مثال البرمجة: تصنيف الأعداد

- In this example, we use functions to rewrite the program that determines the number of odds and evens from a given list of integers
- Main algorithm remains the same:
  - Initialize variables, `zeros`, `odds`, `evens` to 0
  - Read a number
  - If number is even, increment the even count
    - If number is also zero, increment the zero count; else increment the odd count
  - Repeat Steps 2-3 for each number in the list

• في هذا المثال ، نستخدم الدوال لإعادة كتابة البرنامج الذي يحدد عدد الاحتمالات ويسوي من قائمة معينة من الأعداد الصحيحة

• تظل الخوارزمية الرئيسية كما هي:

– تهيئة المتغيرات ، الأصفار و احتمال و يسوي إلى 0

– اقرأ عددًا

– إذا كان الرقم زوجيًا ، قم بزيادة العدد الزوجي

• إذا كان الرقم صفرًا أيضًا ، فقم بزيادة العدد الصفري ؛ وإلا زيادة العدد الفردي

# Programming Example: Classify Numbers (cont'd.)

مثال البرمجة :تصنيف الأرقام (تابع)

- The program functions include:
  - `initialize`: initialize the variables, such as `zeros`, `odds`, and `evens`
  - `getNumber`: get the number
  - `classifyNumber`: determine if number is odd or even (and whether it is also zero); this function also increments the appropriate count
  - `printResults`: print the results
- تشمل وظائف البرنامج:
  - تهيئة: تهيئة المتغيرات ، مثل الأصفار و احتمال ، ويسوي
  - `getNumber`: احصل على الرقم
  - صنف: تحديد ما إذا كان الرقم فرديًا أم زوجيًا (وما إذا كان أيضًا صفرًا ؛) تزيد هذه الوظيفة أيضًا من العدد المناسب
  - طباعة النتائج :اطبع النتائج

```
void getNumber(int& num, int& oddCount, int& evenCount)
{
    cin >> num;
}
evenCount = 0;
}
```

NUMBERS (CONT'D.)

# Classify

مثال البرمجة :تصنيف الأرقام (تابع)

```

void printResults(int zeroCount, int oddCount, int evenCount)
{
    cout << "There are " << evenCount << " evens, "
        << "which includes " << zeroCount << " zeros"
        << endl;

    cout << "The number of odd numbers is: " << oddCount
        << endl;
} //end printResults

```

```

{
    switch (num % 2)
    {
        case 0:
            evenCount++;
            if (num == 0)
                zeroCount++;
            break;
        case 1:
        case -1:
            oddCount++;
    } //end switch
} //end classifyNumber

```

# Classify

مثال البرمجة :تصنيف الأرقام (تابع)

ddCount,

# Programming Example: Main Algorithm

مثال البرمجة :الخوارزمية الرئيسية

- Call `initialize` to initialize variables
- Prompt the user to enter 20 numbers
- For each number in the list
  - Call `getNumber` to read a number
  - Output the number
  - Call `classifyNumber` to classify the number and increment the appropriate count
- Call `printResults` to print the final results

- مكاملة تهيئة لتهيئة المتغيرات
- اطلب من المستخدم إدخال 20 رقمًا
- لكل رقم في القائمة
  - مكاملة `getNumber` لقراءة رقم
  - إخراج الرقم
  - مكاملة صنف لتصنيف الرقم وزيادة العدد المناسب
- مكاملة طباعة النتائج لطباعة النتائج النهائية

# Programming Example: Classify Numbers (cont'd.)

مثال البرمجة: تصنيف الأرقام (تابع)

مثال، P402 في نص

Example, P402 in Text



# Main

مثال البرمجة :الخوارزمية الرئيسية (تابع)

```
int main()
{
    //Variable declaration
    int counter; //loop control variable
    int number;  //variable to store the new number
    int zeros;   //variable to store the number of zeros
    int odds;    //variable to store the number of odd integers
    int evens;   //variable to store the number of even integers

    initialize(zeros, odds, evens);           //Step 1

    cout << "Please enter " << N << " integers."
         << endl;                             //Step 2
    cout << "The numbers you entered are: "
         << endl;

    for (counter = 1; counter <= N; counter++) //Step 3
    {
        getNumber(number);                     //Step 3a
        cout << number << " ";                 //Step 3b
        classifyNumber(number, zeros, odds, evens); //Step 3c
    } // end for loop

    cout << endl;

    printResults(zeros, odds, evens);          //Step 4

    return 0;
}
```