C++ Programming:

From Problem Analysis to Program Design, Eighth Edition

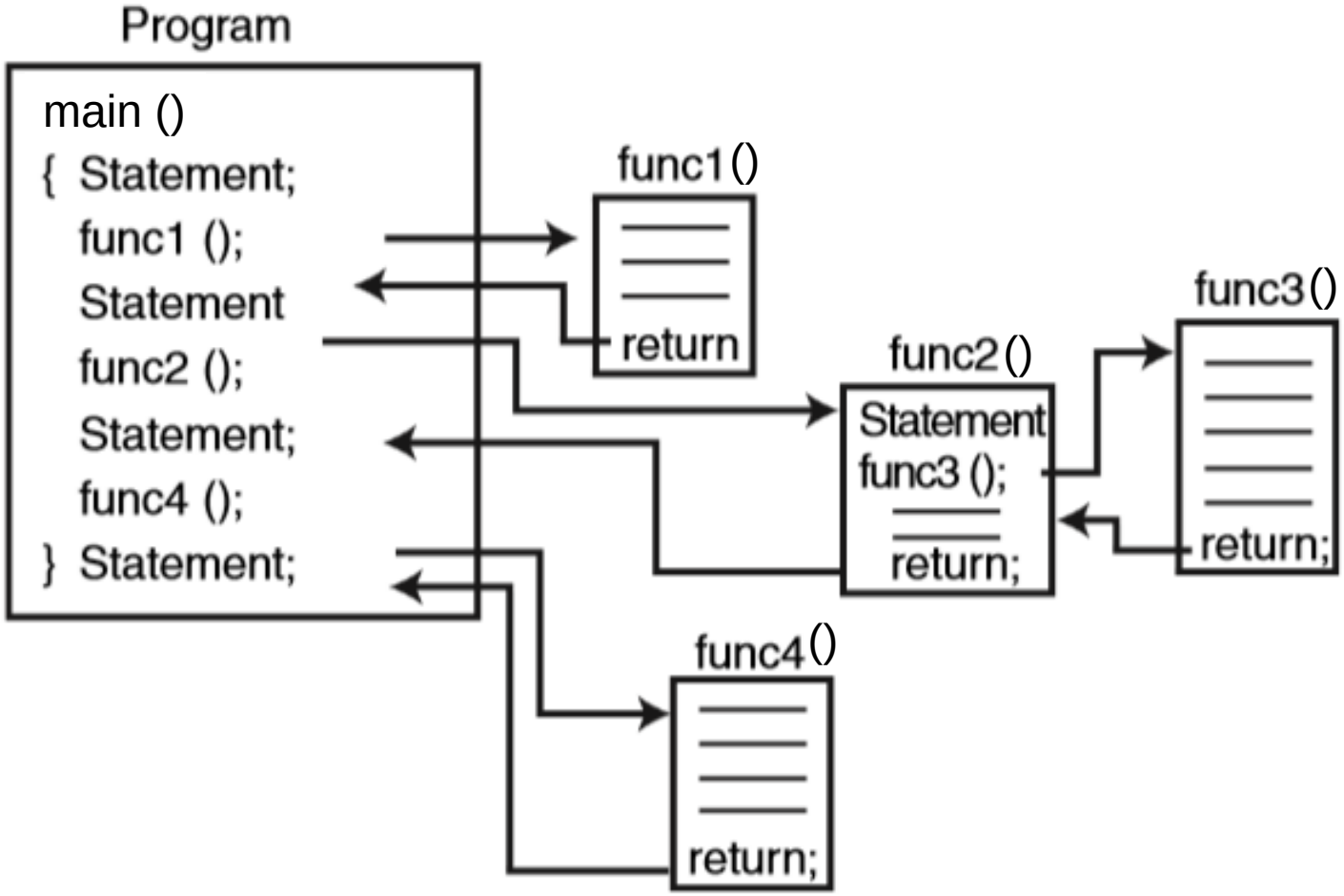
الدوال
Functions

Lecturer Dr. Ibrahim Al-Surmi

مقدمة

- **Functions** are like building blocks.
- الدوال (Function) تشبة بناء البلوك
- They allow complicated programs to be **divided** into manageable pieces
- يتم استخدامها من اجل تقسيم البرامج(programs) المعقدة والكبيرة الى اجزاء صغيرة سهلة من اجل تسهل عملية ادارتة.
- تقسيم البرنامج الى دوال هي احدى المبادئ الرئيسية للبرامج المهيكلة باتباع اسلوب من الاعلى الى الاسفل Top Down، وهي مفيدة نظراً لإمكانية استدعائها واستخدامها في اماكن مختلفة في البرنامج.
- لذلك يمكن ان يحتوي البرنامج على العديد من الدوال التي تستدعي بعضها البعض– سوف نرى ذلك لاحقاً

Calling a function استدعاء الدوال



مقدمة

- الدوال (Functions) هي واحدة من كتل البناء الاساسية في لغة C++, وهي عبارة عن مجموعة من الخطوات (التعليمات) تحت اسم واحد.

- الدالة (Function) تسمح لك بأنشاء مجاميع منطقية من الشفرات, فهي جزء من برنامج يعمل على البيانات ويعيد قيمة.

- يجب أن يكون اسم الدالة ذو معنى وان يكون وحيداً (أي لا توجد دالة بنفس الأسم – ما عدا بعض الأستثائات التي سوف نراها لاحقاً).

انواع الدوال (Function Type) من حيث الانشاء

- تنقسم الدوال (function) الى قسمين هما:

1) **دوال مبنية مسبقاً (Pre-defined Functions) في لغة البرمجة**
   - **يجب تضمين المكتبة التي يوجد فيها بناء الدالة المطلوبة قبل استخدامها وذلك باستخدام include**
   - **مثلاً: توجد في لغة ال C++ مكتبة تسمى cmath خاصة بالدوال الرياضية (كما سنرى لاحقاً)**

2) **دوال يتم بنائها بواسطة المبرمج (User-defined Functions)**

فوائد استخدام الدوال

- تساعد الدوال المبنية مسبقاً (Pre-defined Functions) أو التي يكتبها المبرمج (User-defined Functions) على تلافي عملية التكرار في خطوات البرنامج التي تتطلب عملا مشابها لعمل تلك الدوال.
- تساعد الدوال الجاهزة على تسهيل عملية البرمجة نفسها.
- تساعد على اختصار زمن البرمجة وزمن تنفيذ البرنامج.
- أمكانية اعادة استخدام الدوال
- عندما يكون برنامج C++ مكون من أجزاء (دوال) مستقلة واضحة المعالم، فأن البرنامج نفسة يكون واضحا لكل من المبرمج والقارئ والمستخدم على حد سواء.

الدالة الرئيسة Main Function

- الدالة main() هي الدالة الرئيسية لأي برنامج, معنى ذلك أن كل برنامج بلغة C++ يجب ان يحتوي على الدالة main()

- يبدأ تنفيذ الدالة main() عند تنفيذ اي برنامج C++.

Pre-defined Functions

1) دوال مبنية مسبقاً

Function	Header File	Purpose	Parameter(s) Type	Result
`abs(x)`	`<cmath>`	Returns the absolute value of its argument: `abs(-7) = 7`	`int (double)`	`int (double)`
`ceil(x)`	`<cmath>`	Returns the smallest whole number that is not less than `x`: `ceil(56.34) = 57.0`	`double`	`double`
`cos(x)`	`<cmath>`	Returns the cosine of angle `x`: `cos(0.0) = 1.0`	`double (radians)`	`double`
`exp(x)`	`<cmath>`	Returns e^x, where `e = 2.718`: `exp(1.0) = 2.71828`	`double`	`double`
`fabs(x)`	`<cmath>`	Returns the absolute value of its argument: `fabs(-5.67) = 5.67`	`double`	`double`

Function	Header File	Purpose	Parameter(s) Type	Result
`floor(x)`	`<cmath>`	Returns the largest whole number that is not greater than `x`: `floor(45.67) = 45.00`	`double`	`double`
`islower(x)`	`<cctype>`	Returns 1 (`true`) if `x` is a lowercase letter; otherwise, it returns 0 (`false`); `islower('h')` is 1 (`true`)	`int`	`int`
`isupper(x)`	`<cctype>`	Returns 1 (`true`) if `x` is an uppercase letter; otherwise, it returns 0 (`false`); `isupper('K')` is 1 (`true`)	`int`	`int`
`pow(x, y)`	`<cmath>`	Returns x^y; if `x` is negative, `y` must be a whole number: `pow(0.16, 0.5) = 0.4`	`double`	`double`
`sqrt(x)`	`<cmath>`	Returns the nonnegative square root of `x`; `x` must be nonnegative: `sqrt(4.0) = 2.0`	`double`	`double`
`tolower(x)`	`<cctype>`	Returns the lowercase value of `x` if `x` is uppercase; otherwise, it returns `x`	`int`	`int`
`toupper(x)`	`<cctype>`	Returns the uppercase value of `x` if `x` is lowercase; otherwise, it returns `x`	`int`	`int`

EXAMPLE 6-1

مثال 1

```
//How to use predefined functions.
#include <iostream>
#include <cmath>
#include <cctype>
#include <cstdlib>

using namespace std;

int main()
{
    int    x;
    double u, v;

    cout << "Line 1: Uppercase a is "
         << static_cast<char>(toupper('a'))
         << endl;                                          //Line 1

    u = 4.2;                                               //Line 2
    v = 3.0;                                               //Line 3
    cout << "Line 4: " << u << " to the power of "
         << v << " = " << pow(u, v) << endl;               //Line 4

    cout << "Line 5: 5.0 to the power of 4 = "
         << pow(5.0, 4) << endl;                           //Line 5

    u = u + pow(3.0, 3);                                   //Line 6
    cout << "Line 7: u = " << u << endl;                   //Line 7

    x = -15;                                               //Line 8
    cout << "Line 9: Absolute value of " << x
         << " = " << abs(x) << endl;                       //Line 9

    return 0;
}
```

sample run:

```
Line 1: Uppercase a is A
Line 4: 4.2 to the power of 3 = 74.088
Line 5: 5.0 to the power of 4 = 625
Line 7: u = 31.2
Line 9: Absolute value of -15 = 15
```

مثال 2

يوضح استخدام بعض الدوال الجاهزة في لغة ال C++

```cpp
#include <iostream>
#include <cmath>
#include <cctype>
#include <iomanip>
using namespace std;
int main()
{
    int num; double firstNum, secondNum;  char ch = 'T';
    cout << fixed << showpoint << setprecision (2)<< endl;
    cout << "Line 12: Is " << ch<< " a lowercase letter? "<< islower(ch) << endl;
    cout << "Line 13: Uppercase a is "<< static_cast<char>(toupper('a')) << endl;
    cout << "Line 14: 4.5 to the power 6.0 = "<< pow(4.5, 6.0) << endl;
    cout << "Line 15: Enter two decimal numbers: ";
    cin >> firstNum >> secondNum;
    cout << endl;
    cout << "Line 18: " << firstNum<< " to the power of " << secondNum
    << " = " << pow(firstNum, secondNum) << endl;
    cout << "Line 19: 5.0 to the power of 4 = "<< pow(5.0, 4) << endl;
    firstNum = firstNum + pow(3.5, 7.2);
    cout << "Line 21: firstNum = " << firstNum << endl;
    num = -32;
    cout << "Line 23: Absolute value of " << num<< " = " << abs(num) << endl;
    cout << "Line 24: Square root of 28.00 = "<< sqrt(28.00) << endl;
return 0; }
```

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

```
Line 12: Is T a lowercase letter? 0
Line 13: Uppercase a is A
Line 14: 4.5 to the power 6.0 = 8303.77
Line 15: Enter two decimal numbers: 24.7 3.8

Line 18: 24.70 to the power of 3.80 = 195996.55
Line 19: 5.0 to the power of 4 = 625.00
Line 21: firstNum = 8290.60
Line 23: Absolute value of -32 = 32
Line 24: Square root of 28.00 = 5.29
```

2) الدوال يتم بنائها بواسطة المبرمج User-defined Functions

- **تنقسم الدوال التي يتم بنائها بواسطة المبرمج User-defined Functions الى قسمين :**

(1) دوال تُعيد قيمة **value-returning functions**:

- يكون لديها نوع بيانات (int, float, char…) يحدد نوع القيمة التي سوف تُعيدها الدالة.
- تحتوي على جملة return التي تقوم باسترجاع القيمة الناتجة من الدالة الى المكان الذي تم فيه مناداة الدالة.
- تحجز مكان في الذاكرة.
  - مثل: دالة تجمع عددين, سوف تُرجع حاصل جمع العددين

(2) دوال لا تُعيد قيمة **void functions**:

- لا يكون لديها نوع بيانات للقيمة وتكون من نوع void
- لا تُعيد اي قيمة ولا تحتوي على جملة return
- لا تحجز اي مكان في الذاكرة.
  - مثل :دالة لطباعة رسالة على الشاشة.

الدوال التي تُعيد قيمة...value-returning functions

- **يمكن استخدام القيم المُسترجعة من هذه الدوال كما يلي:**
  - حفظها في متغير واستخدامها اكثر من مرة لاحقاً
  - استخدامها في عمليات حسابية اخرى
  - طباعتها

- يتم استخدام الدوال التي تعيد قيمة في:
  - جمل الاسناد مثل : x=sum(3,5);
  - جمل الطباعة مثل: cout<<sum(4,5);

Function Definition تعريف الدالة

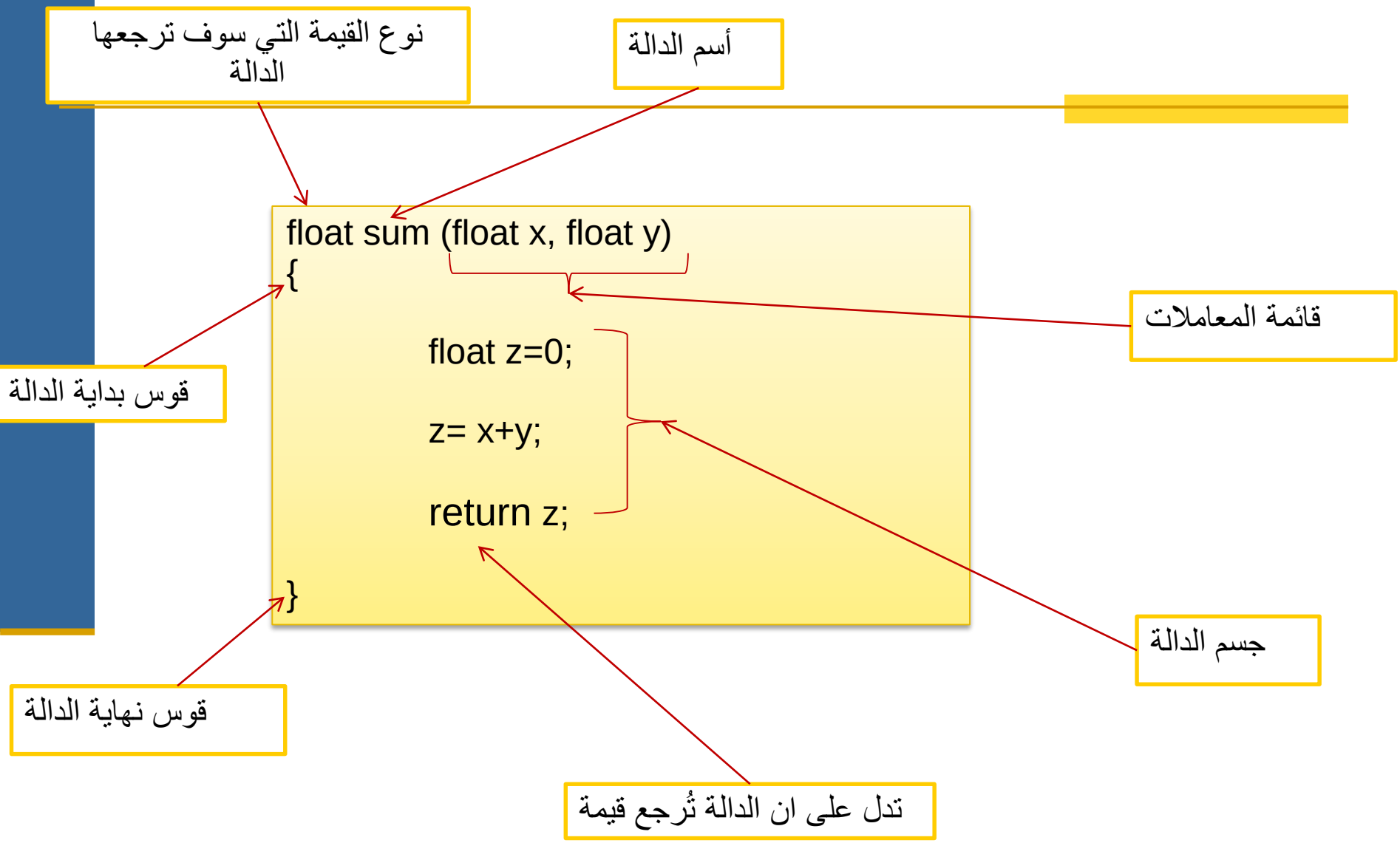
- تتكون الدالة من رأس وجسم، والدالة تأخذ الصيغة او الشكل العام التالي:

```
functionType functionName(formal parameter list)
{
    statements
}
```

- وهذا يسمى **تعريف definition** الداله التي تحتوي على شفرة البرنامج اللازمة لانجاز عمل معين اضافة الى رأس الدالة.

مكونات الدوال الأساسية

- في لغة ال C++ وايضا في معظم لغات البرمجة, تعريف الدوال يمتلك اربعة وحدات اساسية:
  - اسم الدالة The name of the function
  - قائمة المعاملات A list of parameters
  - نوع الكائن أو نوع البيانات التي سوف ترجعها الدالة The type of object or primitive type returned by the function
  - جسم الدالة The body of the function


نوع القيمة التي سوف ترجعها الدالة
أسم الدالة
float sum (float x, float y)
{
float z=0;
z= x+y;
return z;
}
قائمة المعاملات
قوس بداية الدالة
جسم الدالة
قوس نهاية الدالة
تدل على ان الدالة تُرجع قيمة

ماهو الفرق بين تعريفdefinition والتصريح declaration عن الدالة

- **التصريح عن الدالة declaration:**
  - هو كتابة رأس الدالة فقط وقائمة المعاملات وتحديد ما اذا كانت الدالة (تُعيد قيمة اولاتعيد قيمة) قبل دالة main()

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

void display();

main()
{
   display();
}

void display()
{
   cout<<"Welcome";
}
```

التصريح عن دالة لطباعة رساله على الشاشة

- **تعريف الدالة definition:**
  - يعني كتابة الدالة كاملة بما في ذلك جسم الدالة

تعريف دالة لطباعة رساله على الشاشة

الدوال التي تُعيد قيمة...value-returning functions

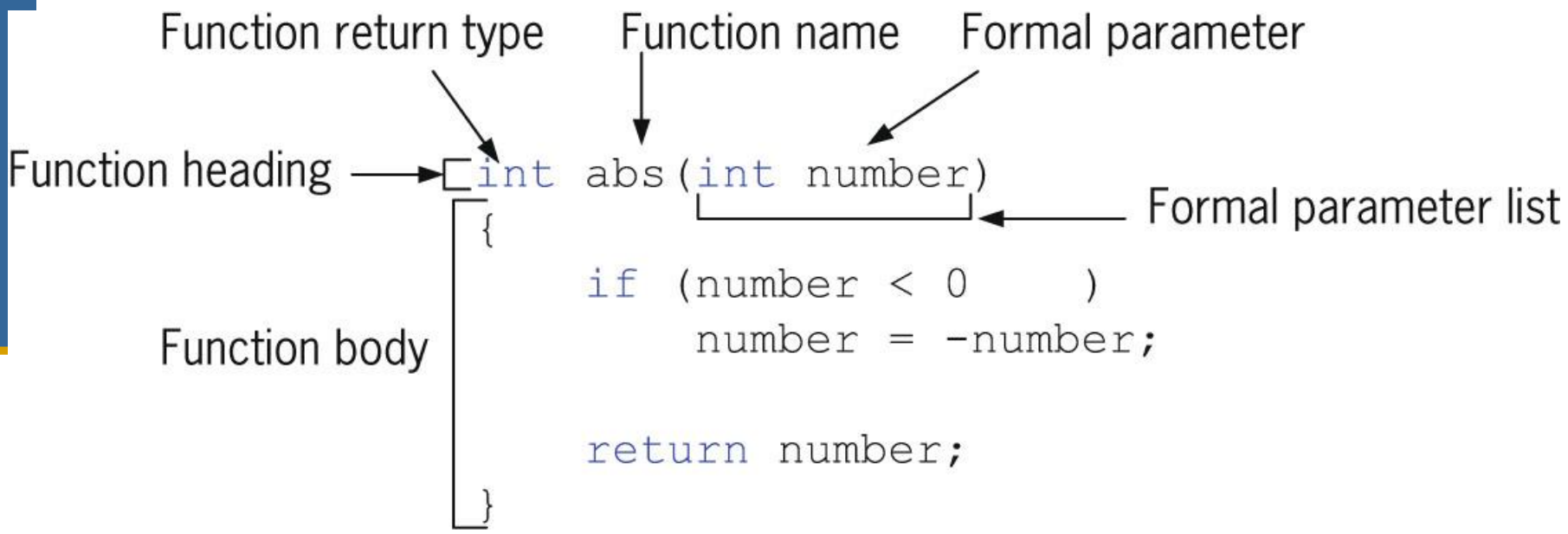
- رأس الدالة function header: وهو السطر الأول من الدالة
  - مثال : `int abs(int number);`

- Formal parameter: variable declared in the heading
  - Example: `number`

- Actual parameter: variable or expression listed in a call to a function
  - Example: `x = pow(u, v)`

الدوال التي تُعيد قيمة...value-returning functions

```
dataType identifier, dataType identifier, ...
```



Syntax: `return` Statement

- Syntax:

```
return expr;
```

- في لغة ال C++, تعتبر جملة ال return كلمة محجوزة
- عندما يتم تنفيذ جملة ال return داخل الدالة function:
  - يتم انهاء عمل الدالة function مباشرة
  - يعود التحكم الى المكان الذي تم فيه مناداة الدالة
- عندما يتم تنفيذ جملة ال return داخل main():
  - يتم انهاء عمل البرنامج Program مباشرة

Syntax: `return` Statement (cont'd.)

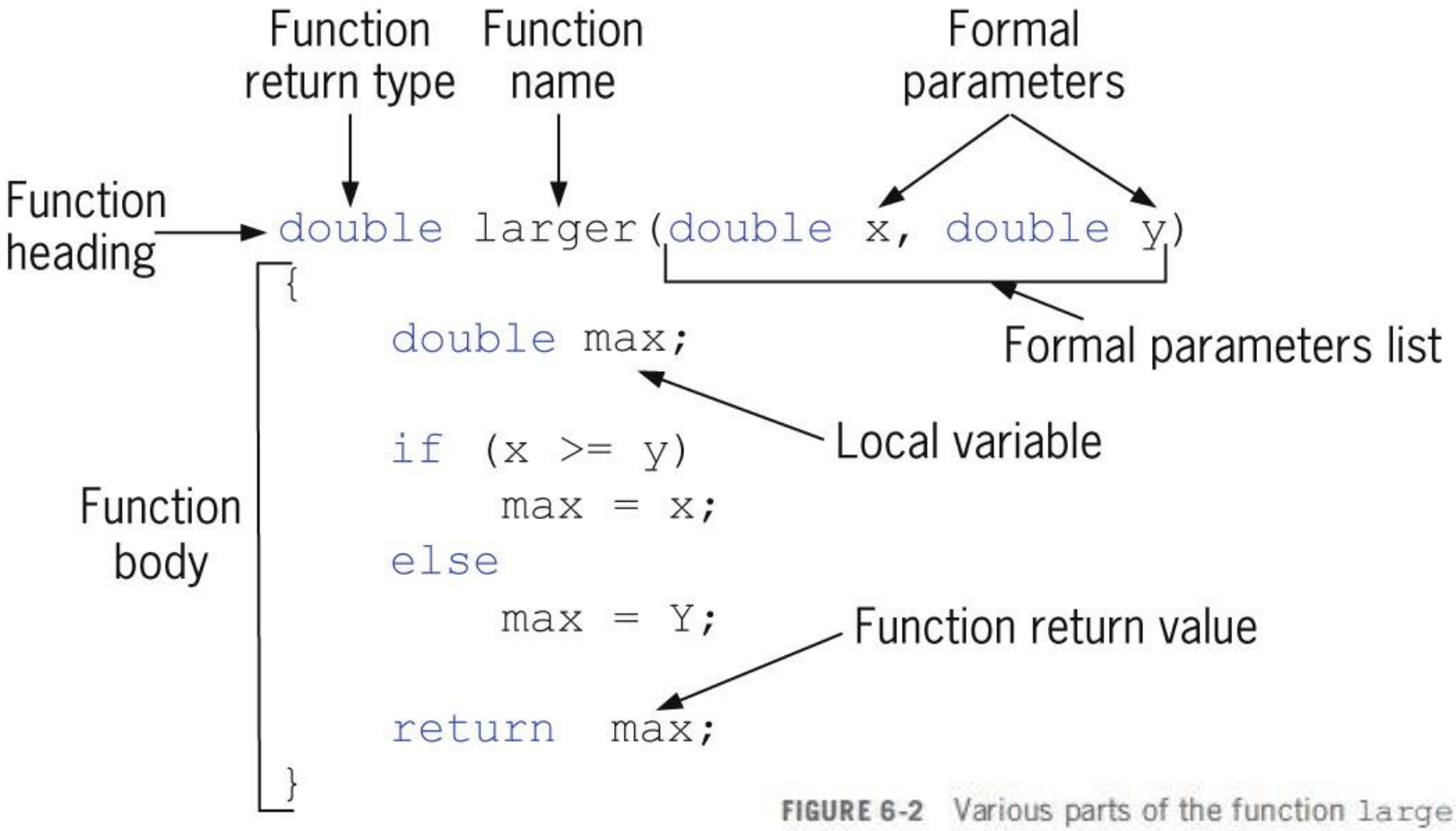


FIGURE 6-2 Various parts of the function `larger`

```
double larger(double x, double y)
{
    double max;

    if (x >= y)
        max = x;
    else
        max = y;

    return max;
}
```

You can also write this function as follows:

```
double larger(double x, double y)
{
    if (x >= y)
        return x;
    else
        return y;
}
```

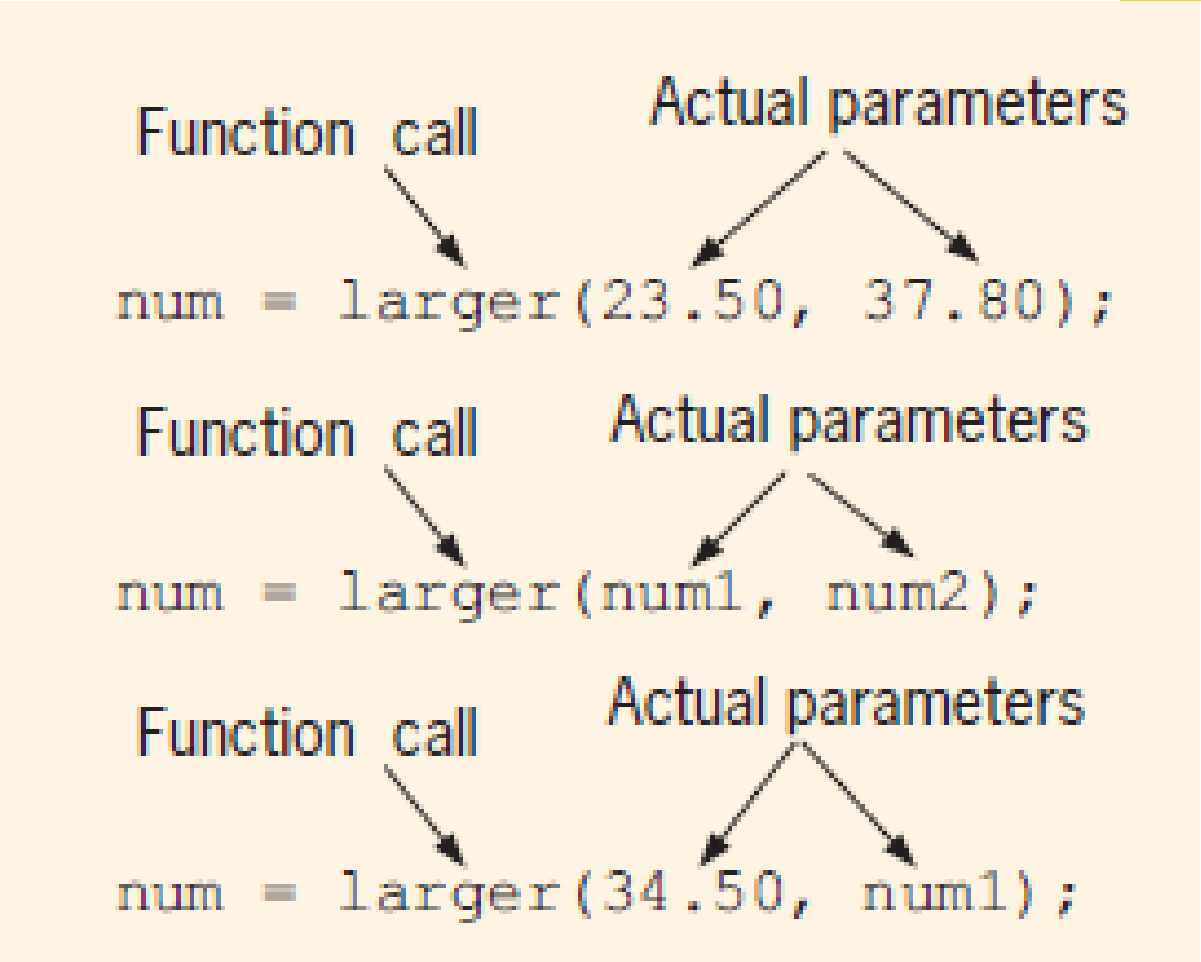
```
double larger(double x, double y)
{
    if (x >= y)
        return x;

    return y;
}
```

NOTE

1. In the definition of the function `larger`, `x` and `y` are formal parameters.
2. The `return` statement can appear anywhere in the function. Recall that once a `return` statement executes, all subsequent statements are skipped. Thus, it's a good idea to return the value as soon as it is computed.

استدعاءات الدوال



Function Prototype

- Function prototype: هي عبارة عن رأس الدالة function heading (تصريح) بدون كتابة جسمها (بدون بناء)

- Syntax

```
functionType functionName(parameter list);
```

- ليس من الضرورة تحديد اسم المتغير في قائمة المعاملات.
- لكن يجب تحديد نوع البيانات الخاص بكل معامل Data type of each parameter must be specified

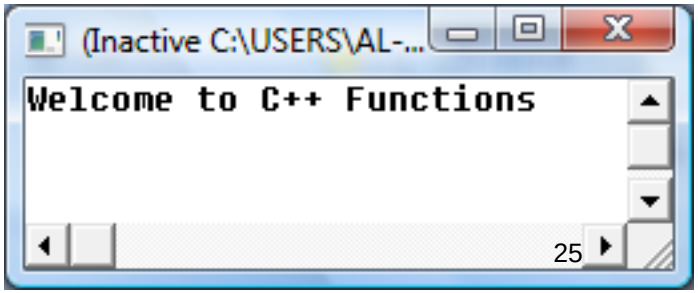
مثال 3

- برنامج يحتوي على دالة (لا تُعيد قيمة) تطبع رسالة على الشاشة

```
#include <iostream.h>
using namespace std;
void display()
 {
  cout<<"Welcome to C++ Functions";
  }

int main ()
  {
    display();
    return 0;
  }
```

مخرجات البرنامج



مثال 4

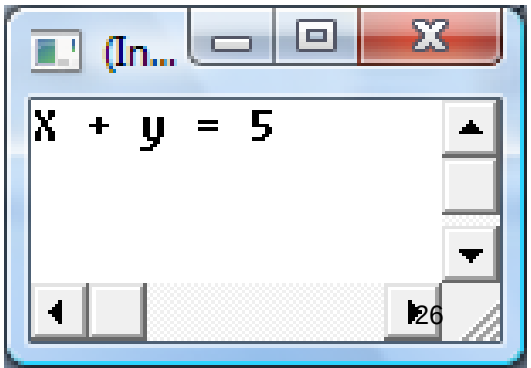
- أكتب برنامج يحتوي على دالة (تُعيد قيمة) لجمع عددين

```
#include <iostream.h>
using namespace std;
float sum(float x, float y)
 {
   float z=0;

   z= x+y;
   return z;
  }

int main ()
  {
    cout<<"X + y = " <<sum(2,3);
    return 0;
  }
```

مخرجات البرنامج



تمرير معاملات الدالة بالقيمة والمرجع
Value and reference parameter

- **Value parameter:** هو عبارة عن معامل parameter يستقبل **نسخة** من قيمة المعامل الفعلي (actual parameter) اثناء استدعاء الدالة.
  - **لا يتم التأثير** على قيمة المعامل الفعلي لأنه فقط استلم نسخة من القيمة.

- **Reference parameter:** وهو عبارة عن معامل parameter يستقبل **عنوان** (عنوان الذاكرة) المعامل الفعلي (actual parameter).
  - **تتأثر** قيمة المعامل الفعلي لأن العمليات سوف تتم على القيمة الفعلية للمعامل.

- يتم استخدام ال reference parametersفي الحالات التالية:
  - عند استرجاع اكثر من قيمة.
  - في حالة الرغبة في تغيير قيمة المعامل الفعلي.
  - للمحافظة على المساحة والزمن.

Reference and value parameters

مثال 5

EXAMPLE 7-7

```
#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;
}
```

```
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
```

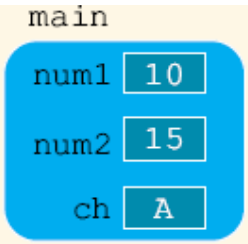


FIGURE 7-5 Values of the variables after the statement in Line 3 executes

```
one = a;                                    //Line 9
```

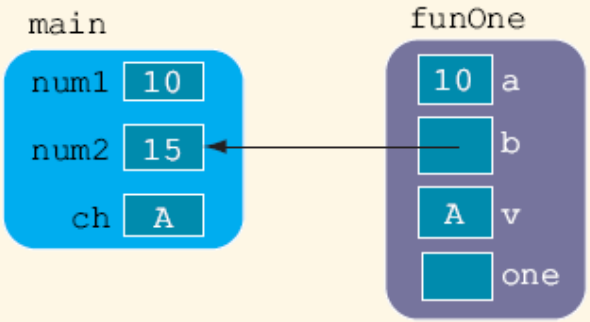


FIGURE 7-6 Values of the variables just before the statement in Line 9 executes

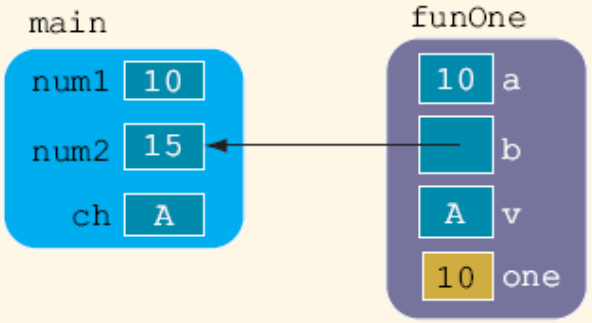


FIGURE 7-7 Values of the variables after the statement in Line 9 executes

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

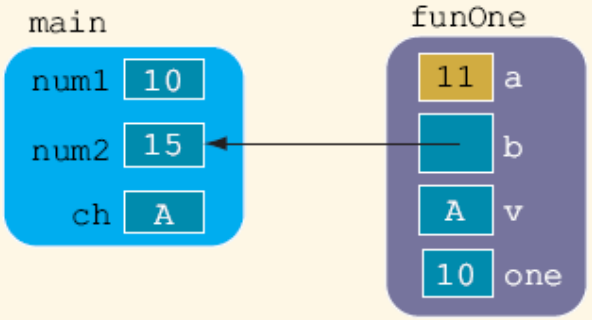


FIGURE 7-8 Values of the variables after the statement in Line 10 executes

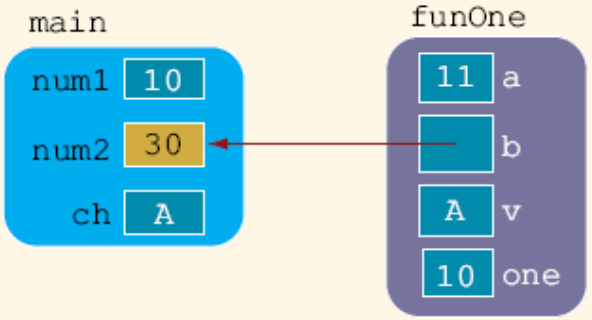


FIGURE 7-9 Values of the variables after the statement in Line 11 executes

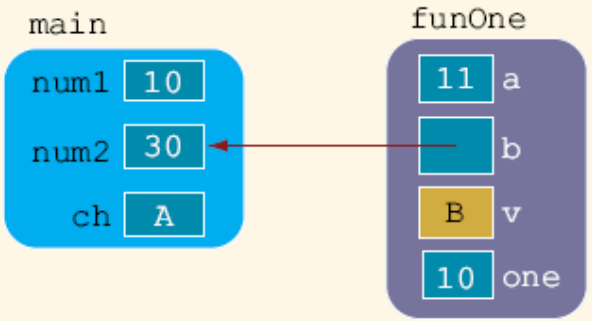


FIGURE 7-10 Values of the variables after the statement in Line 12 executes

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

The statement in Line 13 produces the following output:

```
Line 13: Inside funOne: a = 11, b = 30, v = B, and one = 10
```

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

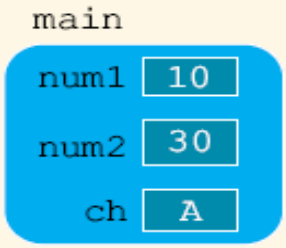


FIGURE 7-11 Values of the variables when control goes back to Line 6

Line 6 produces the following output:

```
Line 6: After funOne: num1 = 10, num2 = 30, and ch = A
```

```
x++;                                                    //Line 14
y = y * 2;                                              //Line 15
```

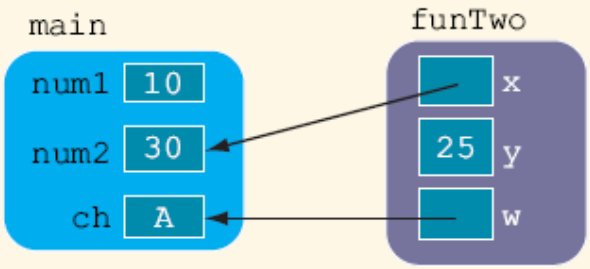


FIGURE 7-12 Values of the variables before the statement in Line 14 executes

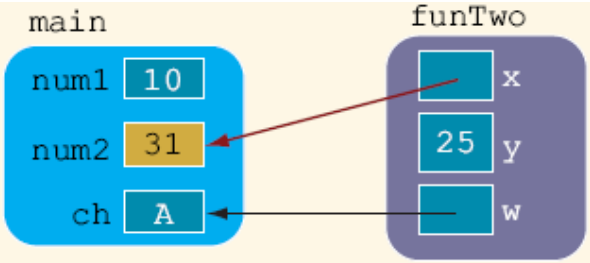


FIGURE 7-13 Values of the variables after the statement in Line 14 executes

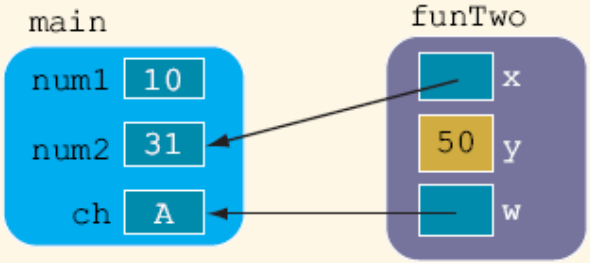


FIGURE 7-14 Values of the variables after the statement in Line 15 executes

```
w = 'G';                                                    //Line 16

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

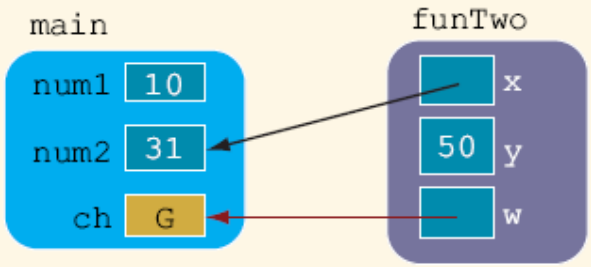


FIGURE 7-15 Values of the variables after the statement in Line 16 executes

Line 17 produces the following output:

```
Line 17: Inside funTwo: x = 31, y = 50, and w = G
```

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

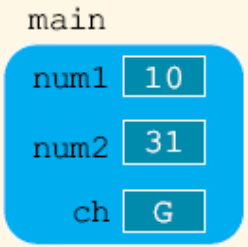


FIGURE 7-16 Values of the variables when control goes to Line 8

```
Line 8: After funTwo: num1 = 10, num2 = 31, and ch = G
```

المتغيرات الساكنه والديناميكية
Static and Automatic Variables

- **المتغير الديناميكى automatic variable:**

- يتم تخصيص مكان المتغير في الذاكرة عند الدخول الى البلوك { } ويتم الغائها عند الخروج من البلوك.

- By default, variables declared within a block are automatic variables

- المتغيرات التي تعرف خارج داخل اي بلوك تعتبر متغيرات ديناميكية.

- **المتغير الساكن static variable:**

- يتم حجز الذاكرة للمتغير طيلة تنفيذ البرنامج.

- Global variables declared outside of any block are static variables

- المتغيرات العامة التي تعرف خارج اي بلوك تعتبر متغيرات ساكنة.

- الصيغة العامة لتعريف المتغير الساكن:

```
static dataType identifier;
```

المتغيرات الساكنه والديناميكية

Static and Automatic Variables (cont..)

مثال 6

```
#include <iostream>
using namespace std;
void test();
int main()
{
    int count;
    for (count = 1; count <= 5; count++)
    test();
    return 0;
}
void test()
{
    static int x = 0;
    int y = 10;
    x = x + 2;
    y = y + 1;
    cout << "Inside test x = " << x << " and y = "
    << y << endl;
}
```

Sample Run:

```
Inside test x = 2 and y = 11
Inside test x = 4 and y = 11
Inside test x = 6 and y = 11
Inside test x = 8 and y = 11
Inside test x = 10 and y = 11
```

التحميل الزائد للدوال
Functions overload

- التحميل الزائد للدوال (Functions overload) يشير الى استخدام نفس أسم الدالة لأغراض مختلفة.

- توفر C++ للمبرمج وسيلة جيدة في أستعمال أسماء الدوال والأدوات لأغراض متعددة كل منها يؤدي دورا معينا.

بهذه الطريقة سوف يكون لدينا عددة دوال تنفذ عددة وظائف مختلفة بنفس الأسم

التحميل الزائد للدوال
Functions overload

- مثال:
- الدالة cout تستخدم لطباعة المخرجات على الشاشة, ولكن نوع هذه المخرجات قد يكون مختلف مثل

```
cout<<“welcome”;
cout<<12345;
cout<<11.234;
cout<<‘N’;
```

- نلاحظ انه تم استخدام الدالة cout لطباعة أنواع مختلفة من البيانات على الشاشة

التحميل الزائد للدوال
Functions overload

- مثال:
- اذا اردنا ان نصرح على دالة sum بحيث تتعامل مع انواع بيانات مختلفة, سيكون ذلك كالتالي:

1. int sum(int a ،int b) ;
2. int sum(int a ،int b ،int c) ;
3. double sum(double x ،double y) ;
4. double sum(int p ،double q) ;
5. double sum(double p ،int q) ;

التحميل الزائد للدوال
Functions overload

التصريح عن الدالة	استدعاء الدالة
1. int sum(int a ،int b) ;	cout<<sum(5, 5);
2. int sum(int a ،int b ،int c) ;	cout<<sum(2, 4, 6);
3. double sum(double x ،double y) ;	cout<<sum(2.2, 2.4);
4. double sum(int p ،double q) ;	cout<<sum(3, 6.5);
5. double sum(double p ،int q) ;	cout<<(3.3, 5);

الاستدعاء الذ اتي Recursion

- في C++ الدالة ممكن ان تستدعي نفسها، مثل هذه الدالة تسمى دالة الاستدعاء الذاتي، بحيث يتم استدعاء الدالة من داخل جسم الدالة أي الدالة تستدعي نفسها

- الاستدعاء الذاتي هي عملية تعريف شيء بدلالة نفسة وفي بعرض الاحيان يسمى التعريف الدائري.

- مثلاً اذا اردنا كتابة برنامج لحساب مجموع ارقام موجبة بطريقة الاستدعاء الذاتي

1*2*3*4*5*.....*n

مثال 7: برنامج لحساب مضروب العدد5

```
#include <iostream.h>
using namespace std;
int main()
{
  int fact = 1;

  for(int i=1; i<=5; i++)
  {
     fact = fact*i;

  }
    cout<< "5! = "<<fact;
return 0;
}
```

الناتج = 120

مثال8: برنامج لحساب مضروب أي عدد صحيح بأستخدام مفهوم الأستدعاء الذاتي للدوال

```
#include <iostream.h>
using namespace std;
long factorial (long a)
{
   if (a > 1)
   return (a * factorial (a-1));
   else
      return (1);
}

int main ()
{
   long number;
   cout << "Please type a number: ";
   cin >> number;
   cout << number << "! = " << factorial(number);
 return 0;
}
```

Class Exercise 1

Write a program that has the following functions:

اكتب برنامج باستخدام الدوال لدية الوظائف التالية :

- A function that returns the largest number among two numbers (called compareTwo)

1) دالة تستخدم لمقارنة رقمين وترجـــع اكبر قيمة

- A function that returns the largest number among three numbers (called compareThree)?

2) دالة مقارنة ثلاثة ارقام وارجاع اكبر قيمة باستخدام مبدأ الاستدعاء الذاتي

- A function that returns the largest number among three numbers (called compareThree)?

3) دالة مقارنة اربعة ارقام وارجاع اكبر قيمة باستخدام مبدأ الاستدعاء الذاتي

Class Exercise 2

4) اكتب برنامج يحتوي على دالتين الاولى تعيد مجموع مربع الاعداد من واحد الى عشرة والدالة الثانية تطبع الناتج.

- باستخدام تمرير **قيمة** في معاملات الدوال call by value
- باستخدام تمرير **مرجع** في معاملا ت الدوال call by reference

5) أكتب برنامج يحتوي على دالة لحساب مساحة المكعب cube area .

- ملاحظة: مساحة المكعب = الطول * العرض * الأرتفاع

Textbook

- D.S. MALIK, C++ Programming: From Problem Analysis to Program Design, Eighth Edition, 2017, Course Technology.