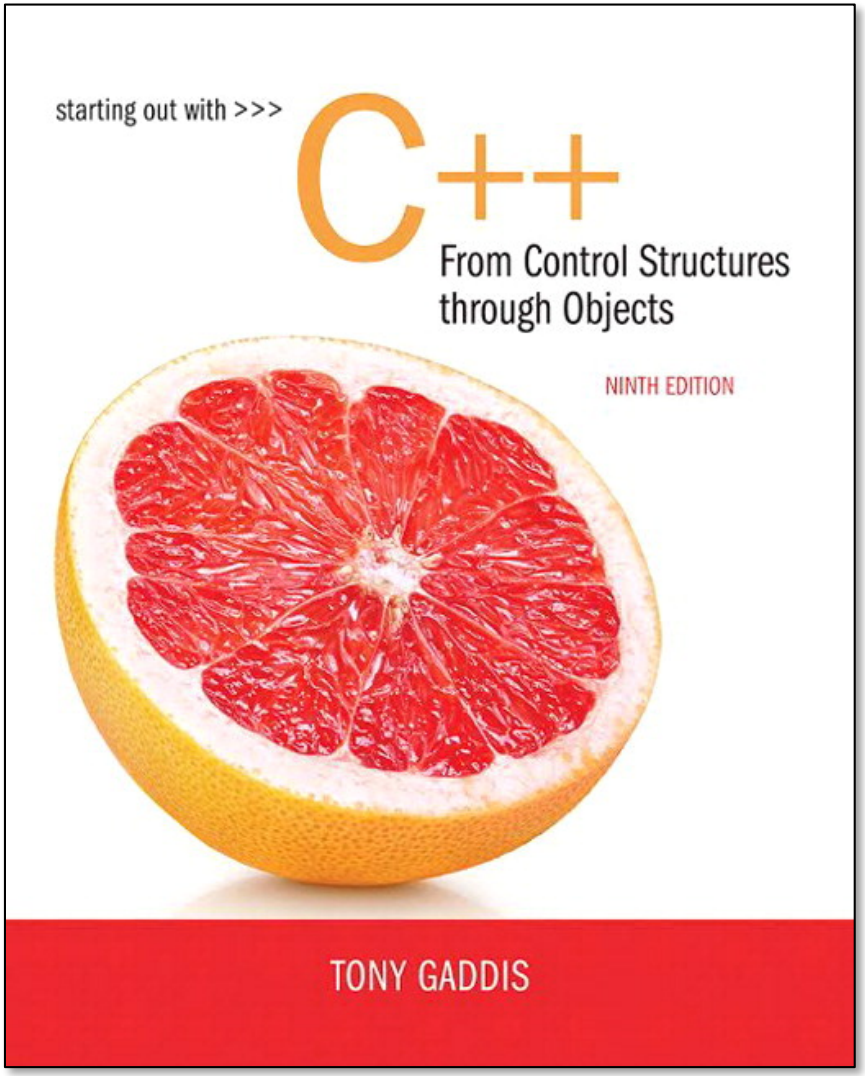
STARTING OUT WITH C++

9th Edition



Chapter 6 :الفصل السادس

Functions :الدوال

6.1

البرمجة النموذجية (التركيبية): Modular Programming

Modular Programming

- البرمجة النموذجية: هي عملية تجزئة البرنامج إلى دوال أو وحدات أصغر قابلة للإدارة والتحكم بها.
- الدالة: function هي مجموعة من العبارات تنفذ مهمة ما.
- ما هي مبررات استخدام البرمجة النموذجية:
  - تحسين عملية صيانة البرامج
  - تبسيط عملية كتابة البرامج

This program has one long, complex function containing all of the statements necessary to solve a problem.

```
int main()
{
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
   statement;
}
```

In this program, the problem has been divided into smaller problems, each of which is handled by a separate function.

```
int main()
{
   statement;
   statement;
   statement;
}
```

main function

```
void function2()
{
   statement;
   statement;
   statement;
}
```

function 2

```
void function3()
{
   statement;
   statement;
   statement;
}
```

function 3

```
void function4()
{
   statement;
   statement;
   statement;
}
```

function 4

6.2

Defining and Calling Functions: تعريف واستدعاء الدوال

Defining and Calling Functions

- استدعاء دالة: هي عبارة تسبب تنفيذ الدالة.
- تعريف دالة: هي العبارات التي تشكل الدالة.

Function Definition

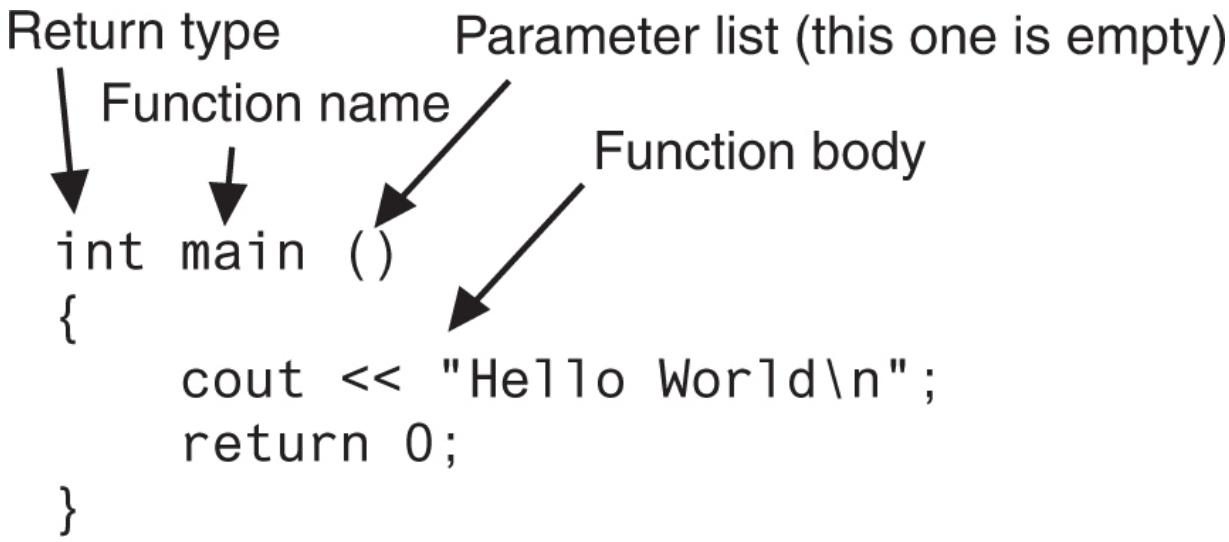
- تضم عملية التعريف:
  - نوع العودة (الرجوع): هو نوع البيانات للقيمة التي سترجعها الدالة لجزء البرنامج الذي قام بطلبها.

✓اسم: اسم الدالة. تتبع أسماء الدوال نفس قواعد تسمية المتغيرات.

✓قائمة الوسيطات: هي متغيرات تحتوي على القيم الممررة للدالة.

✓جسم: هي العبارات التي تنفذ مهمة الدالة وتكون محاطة بـ { }.

Function Definition



Note: The line that reads `int main()` is the *function header*.

Function Return Type

- إذا كانت الدالة ترجع قيمة يجب تبيان نوع القيمة:

```
int main()
```

- إذا لم ترجع الدالة قيمة ما يكون نوع رجوعها void:

```
void printHeading()
{
     cout << "Monthly Sales\n";
}
```

Calling a Function

- لاستدعاء دالة ما استخدم الاسم متبوعا بالقوسين ( ) والفاصلة المنقوطة ;

```
printHeading();
```

- عند استدعاء الدالة يقوم البرنامج بتنفيذ جسم الدالة المستدعاة.
- بعد انتهاء الدالة يستأنف التنفيذ عند نقطة الاستدعاء.

- After the function terminates, execution resumes in the calling function at point of call.

Functions in Program 6-1

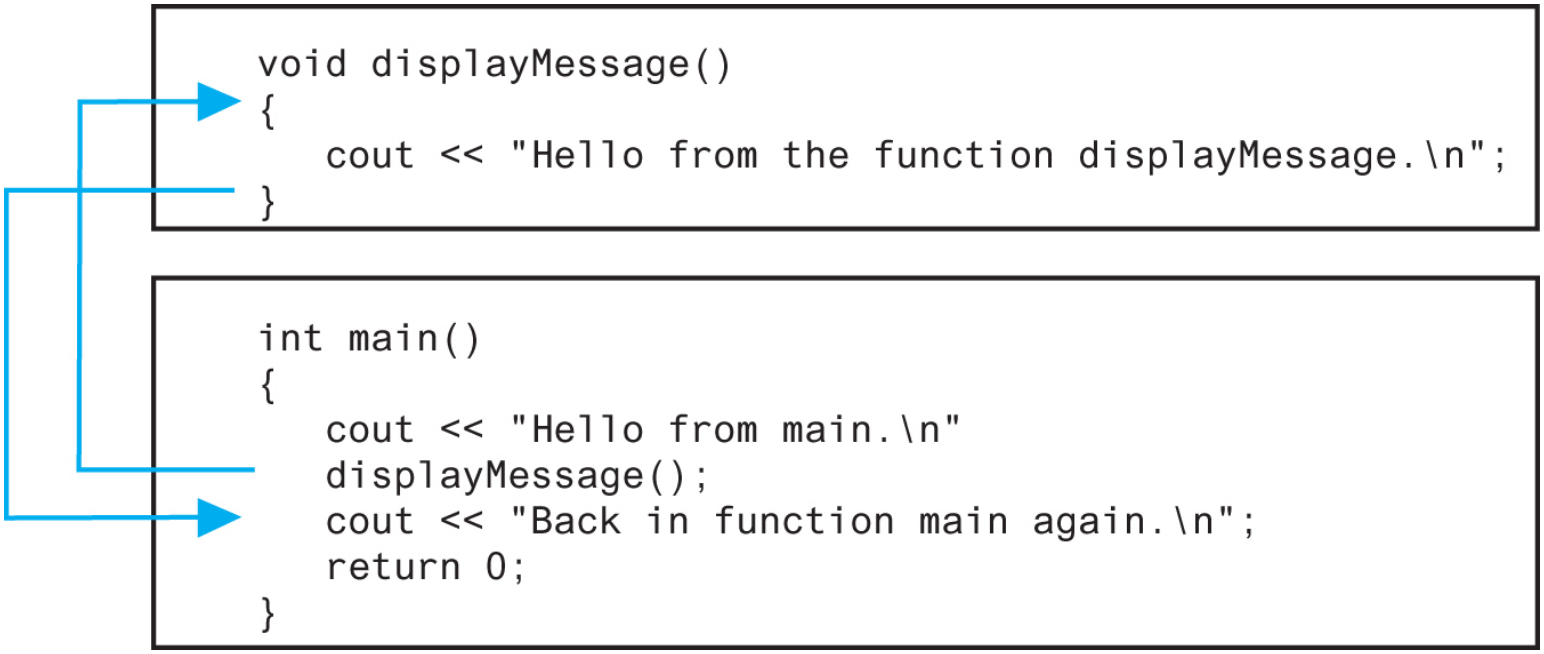
Program 6-1

```
1  // This program has two functions: main and displayMessage
2  #include <iostream>
3  using namespace std;
4
5  //****************************************
6  // Definition of function displayMessage  *
7  // This function displays a greeting.     *
8  //****************************************
9
10 void displayMessage()
11 {
12    cout << "Hello from the function displayMessage.\n";
13 }
14
15 //****************************************
16 // Function main                          *
17 //****************************************
18
19 int main()
20 {
21    cout << "Hello from main.\n";
22    displayMessage();
23    cout << "Back in function main again.\n";
24    return 0;
25 }
```

Program Output

```
Hello from main.
Hello from the function displayMessage.
Back in function main again.
```

Flow of Control in Program 6-1 : تدفق التحكم في البرنامج ٦-١



```
void displayMessage()
{
   cout << "Hello from the function displayMessage.\n";
}

int main()
{
   cout << "Hello from main.\n"
   displayMessage();
   cout << "Back in function main again.\n";
   return 0;
}
```

استدعاء الدوال: Calling Functions

- يمكن للدالة main استدعاء أي عدد من الدوال.
- يمكن للدوال أن تستدعي دوال أخرى.
- يجب أن يعرف المترجم ما يلي حول الدالة قبل استدعائها:

  - name
  - return type
  - number of parameters
  - data type of each parameter

6.3

Function Prototypes :النماذج الأولية للدالة

Function Prototypes

- هي طرق تستخدم لإخطار المترجم حول دالة ما قبل استدعائها:
  - ضع تعريف الدالة قبل تعريف دالة الاستدعاء
  - استخدم نموذج (عينة) الدالة (التصريح عن الدالة)- مثل تعريف الدالة بدون الجسم
    - الترويسة `void printHeading()`
    - النموذج (العينة) `void printHeading();`

Function Prototypes in Program 6-5

Program 6-5

```
1  // This program has three functions: main, First, and Second.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototypes
6  void first();
7  void second();
8
9  int main()
10 {
11    cout << "I am starting in function main.\n";
12    first();    // Call function first
13    second();   // Call function second
14    cout << "Back in function main again.\n";
15    return 0;
16 }
17
```

(Program Continues)

Function Prototypes in Program 6-5

```
18  //***************************************
19  // Definition of function first.        *
20  // This function displays a message.    *
21  //***************************************
22
23  void first()
24  {
25     cout << "I am now inside the function first.\n";
26  }
27
28  //***************************************
29  // Definition of function second.       *
30  // This function displays a message.    *
31  //***************************************
32
33  void second()
34  {
35     cout << "I am now inside the function second.\n";
36  }
```

ملاحظات حول العينة (النموذج): Prototype Notes

- ضع العينات (النماذج) قريبا من قمة البرنامج
- يجب أن يضم البرنامج إما العينة أو تعريف الدالة بالكامل قبل أي عملية استدعاء لها وإلا ستحدث أخطاء بعملية التجميع
- عند استخدام العينات (النماذج الأولية) يمكننا وضع التعاريف بأي ترتيب في البرنامج.

6.4

إرسال البيانات إلى دالة ما

Sending Data into a Function

Sending Data into a Function

- يمكن تمرير القيم إلى دالة ما عند استدعائها:

```
c = pow(a, b);
```

تعتبر القيم الممررة إلى دالة ما هي الوسيطات `arguments`.

المتغيرات في دالة ما والتي تحتوي القيم الممررة كوسيطات هي الوسيطات (المتغيرات) `parameters`

A Function with a Parameter Variable

```
void displayValue(int num)
{
   cout << "The value is " << num << endl;
}
```

المتغير الصحيح num هو وسيط parameter وهو يقبل أي قيمة صحيحة ممرة إلى الدالة.

Function with a Parameter in Program 6-6

Program 6-6

```
1  // This program demonstrates a function with a parameter.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototype
6  void displayValue(int);
7
8  int main()
9  {
10    cout << "I am passing 5 to displayValue.\n";
11    displayValue(5);  // Call displayValue with argument 5
12    cout << "Now I am back in main.\n";
13    return 0;
14 }
15
```

(Program Continues)

Function with a Parameter in Program 6-6

Program 6-6 *(continued)*

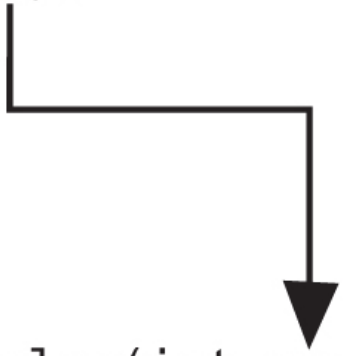
```
16  //*********************************************************
17  // Definition of function displayValue.                   *
18  // It uses an integer parameter whose value is displayed. *
19  //*********************************************************
20
21  void displayValue(int num)
22  {
23     cout << "The value is " << num << endl;
24  }
```

Program Output

```
I am passing 5 to displayValue.
The value is 5
Now I am back in main.
```

Function with a Parameter in Program 6-6

```
displayValue(5);
```



```
void displayValue(int num)
{
  cout << "The value is " << num << endl;
}
```

استعاء الدالة في السطر ١١ يمرر القيمة ٥ كـ argument إلى الدالة

Other Parameter Terminology

- يمكن تسمية الوسيط parameter أيضا بالوسيط الرسمي formal parameter (النظامي) أو بـ formal argument
- أيضا يمكن تسمية الـ argument باسم actual parameter أو actual argument

Parameters, Prototypes, and Function Headers

- بالنسبة لكل argument للدالة:

  - يجب أن يضم النموذج (العينة) نوع البيانات لكل وسيط داخل قوسيه.

  - يجب أن تضم الترويسة تصريحا بالنسبة لكل وسيط بين قوسيها ()

```
void evenOrOdd(int);  //prototype
void evenOrOdd(int num) //head
evenOrOdd(val);       //call
```

Function Call Notes

- Value of argument is copied into parameter when the function is called
- A parameter's scope is the function which uses it
- Function can have multiple parameters
- There must be a data type listed in the prototype ( ) and an argument declaration in the function header ( ) for each parameter
- Arguments will be promoted/demoted as necessary to match parameters

تمرير عدة وسيطات
Passing Multiple Arguments

عند استدعاء دالة ما وتمرير عدة وسيطات:

- يجب أن يوافق عدد الوسيطات في الاستدعاء العينة (النموذج) والتعريف.
- سيستخدم الـ argument الأول لتهيئة المعامل (الوسيط) parameter الأول والـ argument الثاني لتهيئة المعامل (الوسيط) الثاني وهكذا.

Passing Multiple Arguments in Program 6-8

Program 6-8

```
1  // This program demonstrates a function with three parameters.
2  #include <iostream>
3  using namespace std;
4
5  // Function Prototype
6  void showSum(int, int, int);
7
8  int main()
9  {
10    int value1, value2, value3;
11
12    // Get three integers.
13    cout << "Enter three integers and I will display ";
14    cout << "their sum: ";
15    cin >> value1 >> value2 >> value3;
16
17    // Call showSum passing three arguments.
18    showSum(value1, value2, value3);
19    return 0;
20 }
21
```

(Program Continues)

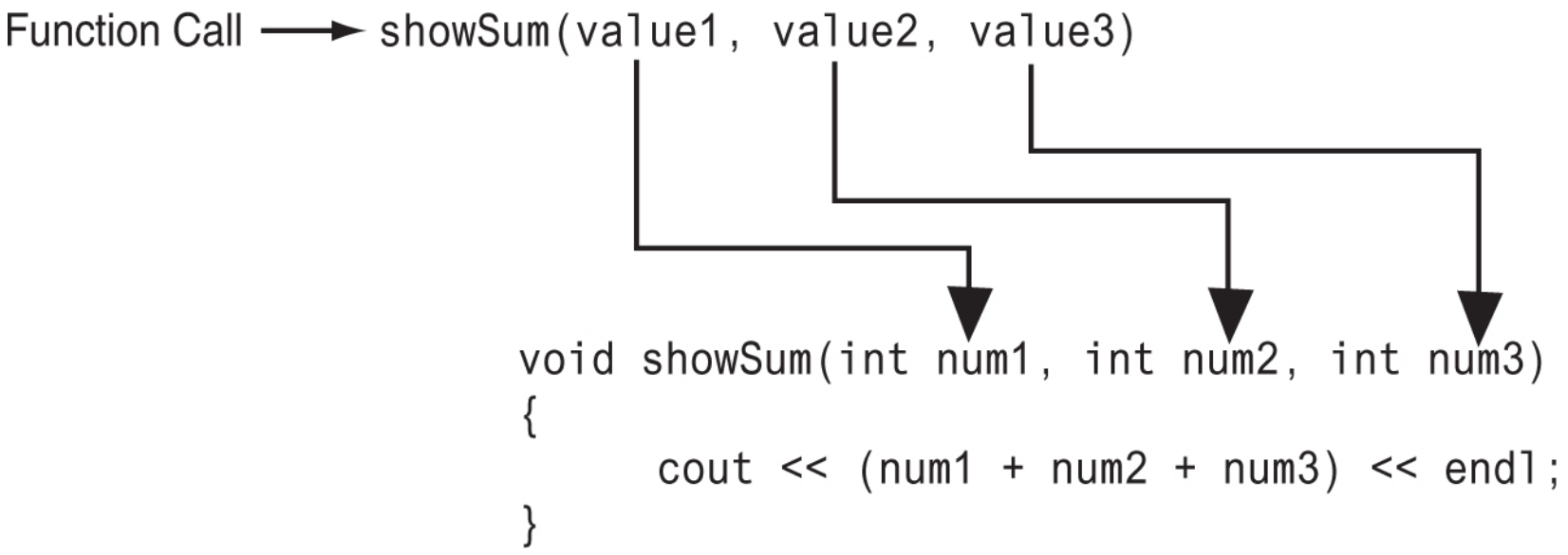
Passing Multiple Arguments in Program 6-8

```
22  //*********************************************************
23  // Definition of function showSum.                        *
24  // It uses three integer parameters. Their sum is displayed. *
25  //*********************************************************
26
27  void showSum(int num1, int num2, int num3)
28  {
29     cout << (num1 + num2 + num3) << endl;
30  }
```

Program Output with Example Input Shown in Bold

```
Enter three integers and I will display their sum: 4 8 7 [Enter]
19
```

Passing Multiple Arguments in Program 6-8



The function call in line 18 passes value1, value2, and value3 as a arguments to the function.

6.5

تمرير البيانات بالقيمة

Passing Data by Value

Passing Data by Value

- التمرير بالقيمة: عند تمرير argument إلى دالة ما يتم نسخ قيمتها في المعامل (الوسيط) parameter
- لا تؤثر التغييرات على المعامل في الدالة على قيمة الـ argument.

Passing Information to Parameters by Value

- Example: `int val=5;`
  `evenOrOdd(val);`

- `evenOrOdd` can change variable `num`, but it will have no effect on variable `val`

6.6

استخدام الدوال في البرامج المقادة بالقائمة

Using Functions in Menu-Driven Programs

Using Functions in Menu-Driven Programs

- يمكن استخدام الدوال:

- لتنفيذ خيارات المستخدم من القائمة

- لتنفيذ مهام الأغراض العامة:

- يمكن أن تستدعي دوال المستوى الأعلى دوال أغراض عامة مما ينقص عدد الدوال الكلي ويسرع من زمن تطوير البرنامج.

- *See Program 6-10 in the book*

6.7

The `return` Statement

عبارة return
The `return` Statement

- تستخدم لإنهاء تنفيذ دالة ما.
- يمكن وضعها في أي مكان ضمن الدالة، والعبارات التي تليها لن تنفذ.
- يمكن استخدامها لمنع الإنهاء الشاذ للبرنامج.
- في الدالة void بدون عبارة return تنتهي الدالة عند آخر قوس }.

Performing Division in Program 6-11

Program 6-11

```
1  // This program uses a function to perform division. If division
2  // by zero is detected, the function returns.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype.
7  void divide(double, double);
8
9  int main()
10 {
11    double num1, num2;
12
13    cout << "Enter two numbers and I will divide the first\n";
14    cout << "number by the second number: ";
15    cin >> num1 >> num2;
16    divide(num1, num2);
17    return 0;
18 }
```

(Program Continues)

Performing Division in Program 6-11

```
20  //*************************************************************
21  // Definition of function divide.                             *
22  // Uses two parameters: arg1 and arg2. The function divides arg1*
23  // by arg2 and shows the result. If arg2 is zero, however, the  *
24  // function returns.                                          *
25  //*************************************************************
26
27  void divide(double arg1, double arg2)
28  {
29     if (arg2 == 0.0)
30     {
31        cout << "Sorry, I cannot divide by zero.\n";
32        return;
33     }
34     cout << "The quotient is " << (arg1 / arg2) << endl;
35  }
```

Program Output with Example Input Shown in Bold

```
Enter two numbers and I will divide the first
number by the second number: 12 0 [Enter]
Sorry, I cannot divide by zero.
```

6.8

إرجاع قيمة من دالة ما

Returning a Value From a Function

Returning a Value From a Function

- يمكن للدالة أن ترجع قيمة إلى العبارة التي استدعت الدالة.
- رأينا سابقا دالة POW والتي ترجع القيمة:

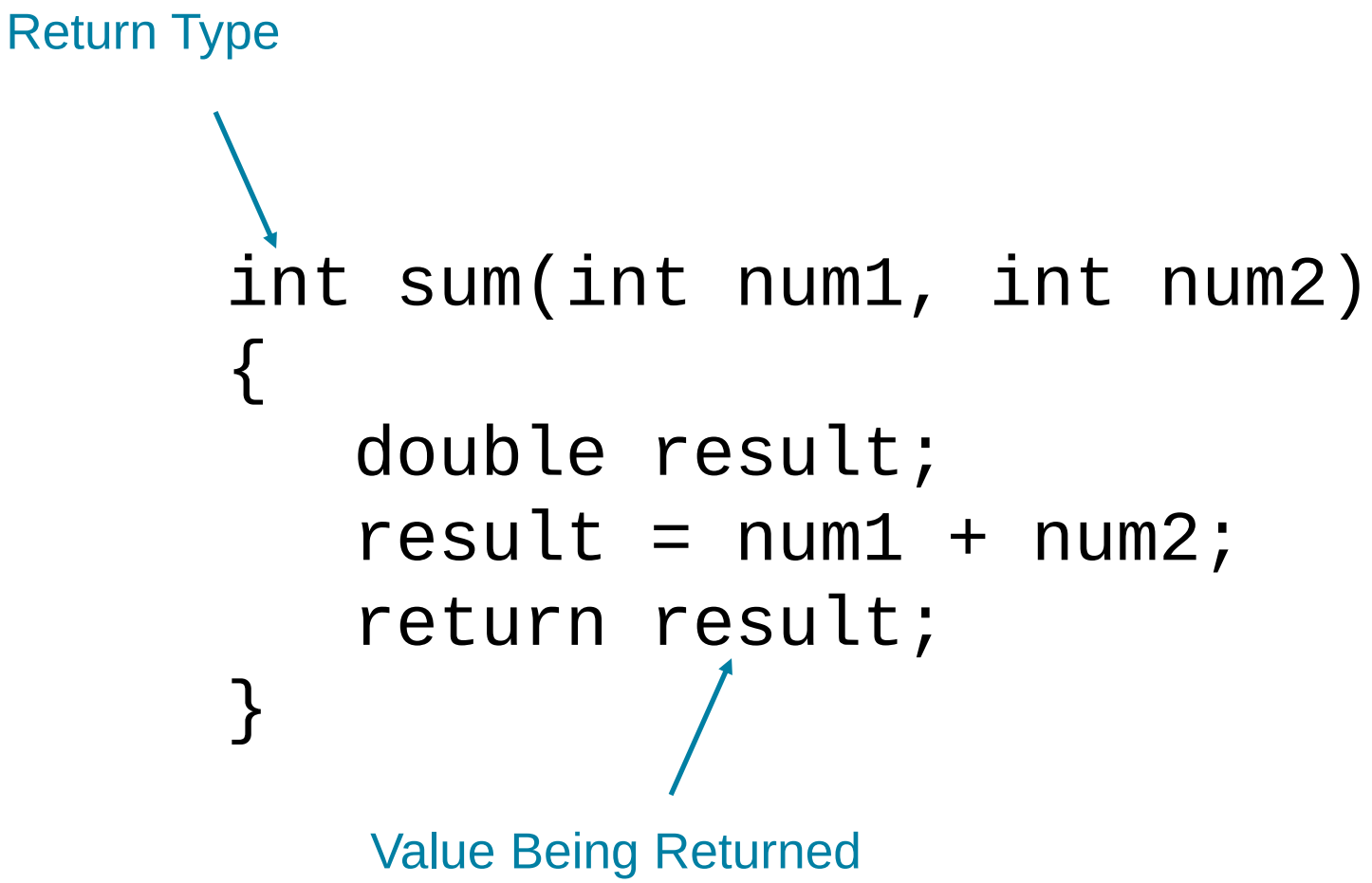
```
double x;
x = pow(2.0, 10.0);
```

Returning a Value From a Function

- في الدالة التي ترجع قيمة ما يمكن استخدام عبارة return لإرجاع قيمة من الدالة إلى نقطة الاستدعاء. مثال:

```
int sum(int num1, int num2)
{
  double result;
  result = num1 + num2;
  return result;
}
```

A Value-Returning Function



A Value-Returning Function

```
int sum(int num1, int num2)
{
   return num1 + num2;
}
```

Functions can return the values of expressions, such as `num1 + num2`

Function Returning a Value in Program 6-12

Program 6-12

```
1  // This program uses a function that returns a value.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype
6  int sum(int, int);
7
8  int main()
9  {
10    int value1 = 20,    // The first value
11        value2 = 40,    // The second value
12        total;          // To hold the total
13
14    // Call the sum function, passing the contents of
15    // value1 and value2 as arguments. Assign the return
16    // value to the total variable.
17    total = sum(value1, value2);
18
19    // Display the sum of the values.
20    cout << "The sum of " << value1 << " and "
21         << value2 << " is " << total << endl;
22    return 0;
23 }
```

(Program Continues)

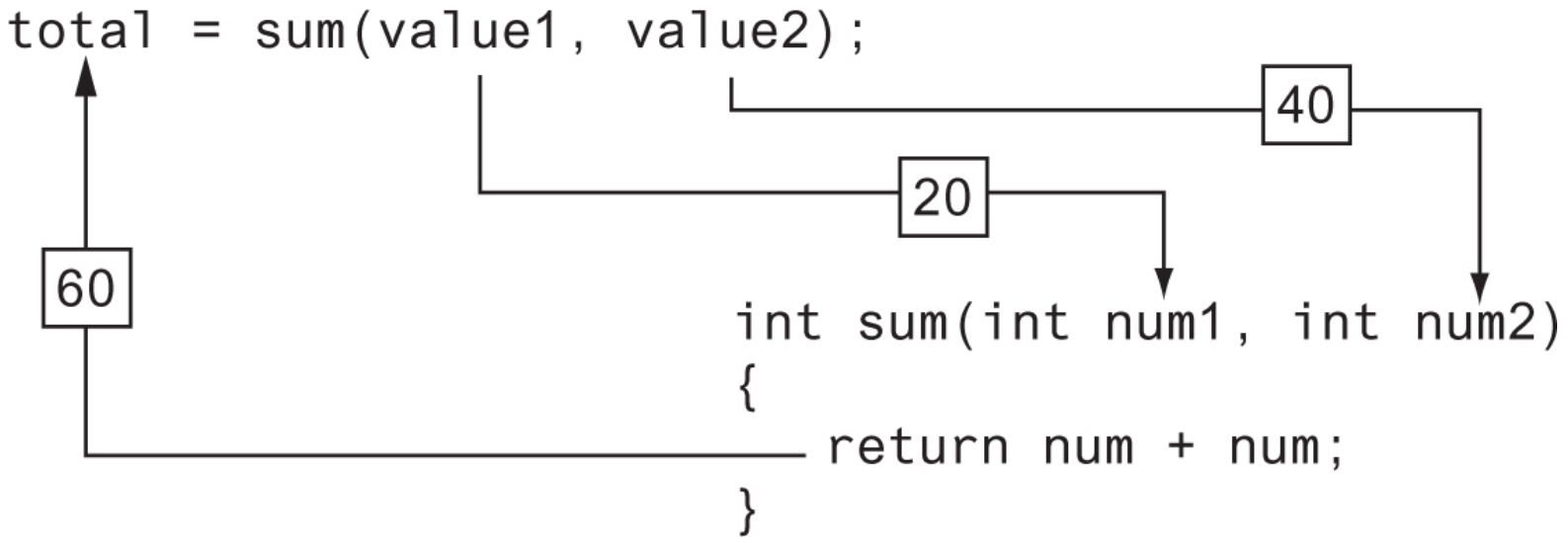
Function Returning a Value in Program 6-12

```
24
25  //*************************************************
26  // Definition of function sum. This function returns  *
27  // the sum of its two parameters.                     *
28  //*************************************************
29
30  int sum(int num1, int num2)
31  {
32     return num1 + num2;
33  }
```

Program Output

```
The sum of 20 and 40 is 60
```

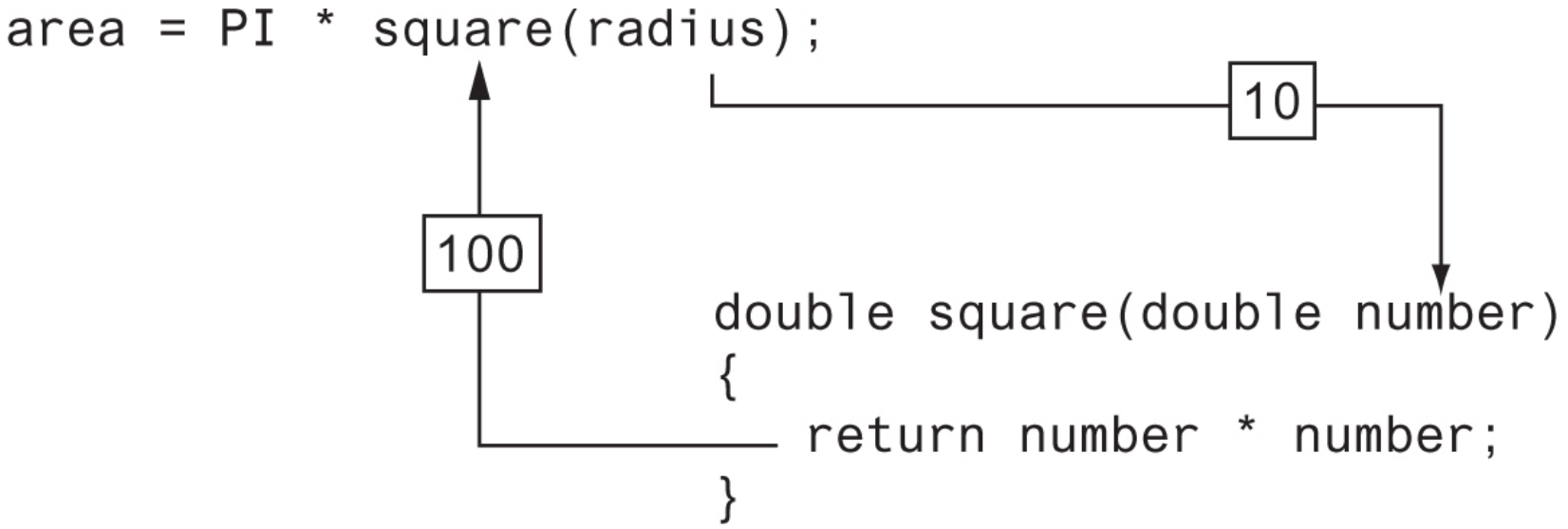
Function Returning a Value in Program 6-12



تستدعي العبارة في السطر ١٧ دالة المجموع وتمرر لها الوسيطتان value1 وvalue2.

تسند قيمة الرجوع إلى المتغير total

Another Example from Program 6-13



Returning a Value From a Function

- يجب أن يشير كلا من العينة (النموذج) prototype والتعريف definition إلى نوع البيانات لقيمة الرجوع (غير void).
- يجب أن تستخدم دالة الاستدعاء قيمة الرجوع:
  - تخصصها (تسندها) لمتغير ما.
  - ترسلها إلى cout
  - تستخدمها في تعبير ما

6.9

إرجاع قيمة بوليانية

Returning a Boolean Value

Returning a Boolean Value

- يمكن أن ترجع الدالة قيمة `true` أو `false`
- صرح عن نوع الرجوع في عينة الدالة والترويسة على أنها `bool`
- يجب على جسم الدالة أن يضم عبارة (عبارات) ترجع true أو false
- يمكن أن تستخدم دالة الاستدعاء قيمة رجوع في العبارة النسبية (العلائقية).

Returning a Boolean Value in Program 6-15

Program 6-15

```
1  // This program uses a function that returns true or false.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype
6  bool isEven(int);
7
8  int main()
9  {
10    int val;
11
12    // Get a number from the user.
13    cout << "Enter an integer and I will tell you ";
14    cout << "if it is even or odd: ";
15    cin >> val;
16
17    // Indicate whether it is even or odd.
18    if (isEven(val))
19       cout << val << " is even.\n";
20    else
21       cout << val << " is odd.\n";
22    return 0;
23 }
24
```

(Program Continues)

Returning a Boolean Value in Program 6-15

```
25  //*************************************************************
26  // Definition of function isEven. This function accepts an    *
27  // integer argument and tests it to be even or odd. The function  *
28  // returns true if the argument is even or false if the argument  *
29  // is odd. The return value is a bool.                        *
30  //*************************************************************
31
32  bool isEven(int number)
33  {
34     bool status;
35
36     if (number % 2 == 0)
37        status = true;  // The number is even if there is no remainder.
38     else
39        status = false; // Otherwise, the number is odd.
40     return status;
41  }
```

Program Output with Example Input Shown in Bold

```
Enter an integer and I will tell you if it is even or odd: 5 [Enter]
5 is odd.
```

6.10

المتغيرات المحلية والشاملة

Local and Global Variables

Local and Global Variables

- تعتبر المتغيرات المعرفة ضمن دالة ما محلية (موضعية) بالنسبة لهذه الدالة. بمعنى أنها تكون مخبأة عن العبارات في الدوال الأخرى والتي لا يمكن أن تصل إليها.
- لأن المتغيرات المعرفة في دالة ما تكون مخبأة فقد يكون للدوال الأخرى متغيرات خاصة بها وذات نفس الاسم.

Local Variables in Program 6-16

Program 6-16

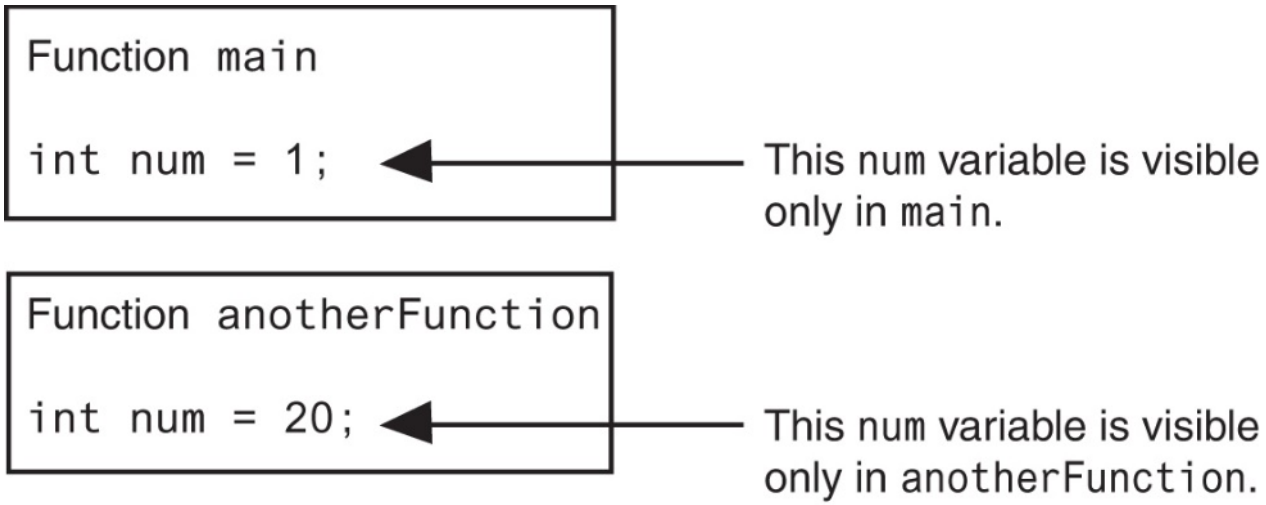
```
1  // This program shows that variables defined in a function
2  // are hidden from other functions.
3  #include <iostream>
4  using namespace std;
5
6  void anotherFunction(); // Function prototype
7
8  int main()
9  {
10    int num = 1;    // Local variable
11
12    cout << "In main, num is " << num << endl;
13    anotherFunction();
14    cout << "Back in main, num is " << num << endl;
15    return 0;
16 }
17
18 //*********************************************************
19 // Definition of anotherFunction                          *
20 // It has a local variable, num, whose initial value      *
21 // is displayed.                                          *
22 //*********************************************************
23
24 void anotherFunction()
25 {
26    int num = 20;  // Local variable
27
28    cout << "In anotherFunction, num is " << num << endl;
29 }
```

Local Variables in Program 6-16

Program Output

```
In main, num is 1
In anotherFunction, num is 20
Back in main, num is 1
```

When the program is executing in `main`, the `num` variable defined in `main` is visible. When `anotherFunction` is called, however, only variables defined inside it are visible, so the `num` variable in `main` is hidden.



Local Variable Lifetime

- تكون المتغيرات المحلية لدالة ما موجودة فقط أثناء تنفيذ البرنامج. يعرف ذلك باسم عمر lifetime المتغير المحلي.
- عند بدء تنفيذ الدالة يتم إنشاء المتغيرات المحلية ومتغيراتها الوسيطة parameter variables في الذاكرة وعند انتهاء الانتهاء من تنفيذ الدالة يتم هدم المتغيرات المحلية ومتغيراتها الوسيطة parameter variables
- هذا يعني أن أي قيمة مخزنة في متغير موضعي تضيع بي تضيع بين استدعاءات الدالة التي تم فيها التصريح عن المتغير.

Global Variables and Global Constants

- المتغير الشامل هو أي متغير معرف خارج كل الدوال في برنامج ما.
- مجال المتغير الشامل هو من بدء تعريف المتغير حتى النهاية في البرنامج
- هذا يعني أن المتغير الشامل يمكن أن يتم الدخول إليه (استخدامه) من قبل كل الدوال التي تم تعريفها بعد تعريف المتغير الشامل.
- يجب تجنب استخدام المتغيرات الشاملة كونها تصعب عملية كشف الخطأ في البرامج.
- أي متغير شامل تنشؤه يجب أن يكون ثابتا شاملا.

Global Constants in Program 6-19

Program 6-19

```
1  // This program calculates gross pay.
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  // Global constants
7  const double PAY_RATE = 22.55;     // Hourly pay rate
8  const double BASE_HOURS = 40.0;    // Max non-overtime hours
9  const double OT_MULTIPLIER = 1.5; // Overtime multiplier
10
11 // Function prototypes
12 double getBasePay(double);
13 double getOvertimePay(double);
14
15 int main()
16 {
17    double hours,              // Hours worked
18           basePay,            // Base pay
19           overtime = 0.0,     // Overtime pay
20           totalPay;           // Total pay
```

Global constants defined for values that do not change throughout the program's execution.

Global Constants in Program 6-19

The constants are then used for those values throughout the program.

```
29      // Get overtime pay, if any.
30      if (hours > BASE_HOURS)
31         overtime = getOvertimePay(hours);
```

```
56      // Determine base pay.
57      if (hoursWorked > BASE_HOURS)
58         basePay = BASE_HOURS * PAY_RATE;
59      else
60         basePay = hoursWorked * PAY_RATE;
```

```
75      // Determine overtime pay.
76      if (hoursWorked > BASE_HOURS)
77      {
78         overtimePay = (hoursWorked - BASE_HOURS) *
79                  PAY_RATE * OT_MULTIPLIER;
```

تهيئة المتغيرات المحلية والشاملة
Initializing Local and Global Variables

- لا تتم تهيئة المتغيرات المحلية بشكل آلي ويجب تهيئتها من قبل المبرمج.
- تتم تهيئة المتغيرات الشاملة (وليس الثوابت الشاملة) بشكل آلي بقيمة الصفر (عددي) أو Null (محرف) عند تعريف المتغير.

6.11

المتغيرات المحلية الساكنة

Static Local Variables

Static Local Variables

- توجد المتغيرات المحلية أثناء تنفيذ الدالة فقط. عند انتهاء الدالة تضيع محتويات هذه المتغيرات.
- تحتفظ المتغيرات المحلية الساكنة static بمحتوياتها بين عمليات استدعاء الدوال.
- يتم تعريف المتغيرات المحلية الساكنة وتهيئتها عند أول تنفيذ للدالة فقط. قيمة التهيئة الافتراضية هي الصفر.

Local Variables Do Not Retain Values Between Function calls in Program 6-21

Program 6-21

```
1  // This program shows that local variables do not retain
2  // their values between function calls.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype
7  void showLocal();
8
9  int main()
10 {
11    showLocal();
12    showLocal();
13    return 0;
14 }
15
```

(Program Continues)

Local Variables Do Not Retain Values Between Function calls in Program 6-21

Program 6-21 *(continued)*

```
16  //*************************************************************
17  // Definition of function showLocal.                          *
18  // The initial value of localNum, which is 5, is displayed.   *
19  // The value of localNum is then changed to 99 before the     *
20  // function returns.                                          *
21  //*************************************************************
22
23  void showLocal()
24  {
25     int localNum = 5; // Local variable
26
27     cout << "localNum is " << localNum << endl;
28     localNum = 99;
29  }
```

Program Output

```
localNum is 5
localNum is 5
```

In this program, each time `showLocal` is called, the `localNum` variable is re-created and initialized with the value 5.

A Different Approach, Using a Static Variable in Program 6-22

Program 6-22

```
1  // This program uses a static local variable.
2  #include <iostream>
3  using namespace std;
4
5  void showStatic(); // Function prototype
6
7  int main()
8  {
9     // Call the showStatic function five times.
10    for (int count = 0; count < 5; count++)
11       showStatic();
12    return 0;
13 }
14
```

(Program Continues)

A Different Approach, Using a Static Variable in Program 6-22

Program 6-22 *(continued)*

```
15  //*************************************************************
16  // Definition of function showStatic.                         *
17  // statNum is a static local variable. Its value is displayed *
18  // and then incremented just before the function returns.     *
19  //*************************************************************
20
21  void showStatic()
22  {
23     static int statNum;
24
25     cout << "statNum is " << statNum << endl;
26     statNum++;
27  }
```

Program Output

```
statNum is 0
statNum is 1
statNum is 2
statNum is 3
statNum is 4
```

`statNum` is automatically initialized to 0. Notice that it retains its value between function calls.

If you do initialize a local static variable, the initialization only happens once. See Program 6-23.

```
16  //*************************************************************
17  // Definition of function showStatic.                         *
18  // statNum is a static local variable. Its value is displayed *
19  // and then incremented just before the function returns.     *
20  //*************************************************************
21
22  void showStatic()
23  {
24     static int statNum = 5;
25
26     cout << "statNum is " << statNum << endl;
27     statNum++;
28  }
```

Program Output

```
statNum is 5
statNum is 6
statNum is 7
statNum is 8
statNum is 9
```

6.12

الوسيطات (القيم) الافتراضية

Default Arguments

Default Arguments

الوسيطة (القيمة) الافتراضية: هي الوسيطة الممرة بشكل آلي إلى متغير (parameter) إذا كانت الوسيطة ناقصة عند استدعاء الدالة.

- يجب أن يكون مصرح عنها على أنها ثابت في العينة:

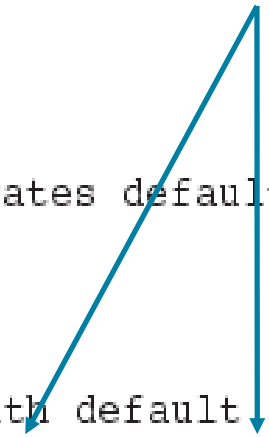
```
void evenOrOdd(int = 0);
```

- يمكن التصريح عنها في الترويسة إذا لم يتم ذلك في العينة.
- قد يكون للدوال متعددة المتغيرات وسيطات افتراضية بالنسبة لبعضها أو كلها:

```
int getSum(int, int=0, int=0);
```

Default Arguments in Program 6-24

Default arguments specified in the prototype



Program 6-24

```
1  // This program demonstrates default function arguments.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype with default arguments
6  void displayStars(int = 10, int = 1);
7
8  int main()
9  {
10    displayStars();      // Use default values for cols and rows.
11    cout << endl;
12    displayStars(5);     // Use default value for rows.
13    cout << endl;
14    displayStars(7, 3);  // Use 7 for cols and 3 for rows.
15    return 0;
16 }
```

(Program Continues)

Default Arguments in Program 6-24

```
18  //*********************************************************
19  // Definition of function displayStars.                   *
20  // The default argument for cols is 10 and for rows is 1.*
21  // This function displays a square made of asterisks.     *
22  //*********************************************************
23
24  void displayStars(int cols, int rows)
25  {
26     // Nested loop. The outer loop controls the rows
27     // and the inner loop controls the columns.
28     for (int down = 0; down < rows; down++)
29     {
30        for (int across = 0; across < cols; across++)
31           cout << "*";
32        cout << endl;
33     }
34  }
```

Program Output

```
**********

*****

*******
*******
*******
```

Default Arguments

- إذا لم يكن لكل متغيرات parameters دالة ما قيما افتراضية فإنه يتم التصريح في البداية عن القيم بدون الافتراضية (defaultless) في البداية في قائمة المتغيرات

```
int getSum(int, int=0, int=0);// OK
int getSum(int, int=0, int);  // NO
```

- عند إغفال قيمة وسيطة ما من استدعاء الدالة يجب أيضا إغفال كل الوسيطات التالية لها:

```
sum = getSum(num1, num2);     // OK
sum = getSum(num1, , num3);   // NO
```

6.13

استخدام المتغيرات المرجعية كمتغيرات

Using Reference Variables as Parameters

Using Reference Variables as Parameters

- هي آلية تسمح لدالة ما بالعمل مع الوسيطة الأصلية من استدعاء الدالة وليس نسخة من الوسيطة argument.
- تسمح للدالة بتعديل القيم المخزنة في بيئة الاستدعاء.
- تقدم طريقة للدالة كي تعيد أكثر من قيمة واحدة.

Passing by Reference

- المتغير المرجعي هو كناية (استعارة) alias عن متغير آخر.
- يعرف بالرمز (&)

```
void getDimensions(int&, int&);
```

- التغيرات على المتغير المرجعي تنفذ على المتغير الذي يشير إليه.
- استخدم المتغيرات المرجعية لتنفيذ عملية تمرير الوسيطات بالمرجع by reference

Passing a Variable By Reference in Program 6-25

Program 6-25

The & here in the prototype indicates that the parameter is a reference variable.

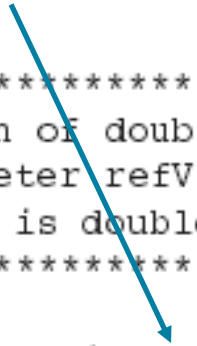
```
1  // This program uses a reference variable as a function
2  // parameter.
3  #include <iostream>
4  using namespace std;
5
6  // Function prototype. The parameter is a reference variable.
7  void doubleNum(int &);
8
9  int main()
10 {
11    int value = 4;
12
13    cout << "In main, value is " << value << endl;
14    cout << "Now calling doubleNum..." << endl;
15    doubleNum(value);
16    cout << "Now back in main. value is " << value << endl;
17    return 0;
18 }
19
```

Here we are passing value by reference.

(Program Continues)

Passing a Variable By Reference in Program 6-25

The & also appears here in the function header.



```
20  //*********************************************************
21  // Definition of doubleNum.                               *
22  // The parameter refVar is a reference variable. The value *
23  // in refVar is doubled.                                  *
24  //*********************************************************
25
26  void doubleNum (int &refVar)
27  {
28     refVar *= 2;
29  }
```

Program Output

```
In main, value is 4
Now calling doubleNum...
Now back in main. value is 8
```

Reference Variable Notes

- يجب أن يحتوي كل متغير مرجعي على الرمز &
- لا يكون الفراغ بين النوع والرمز & مهما.
- يجب استخدام & في كل من العينة والترويسة
- يجب أن تكون الوسيطة argument الممرة إلى المتغير المرجعي reference parameter متغيرا variable- لا يمكن أن تكون تعبيرا أو ثابتا.

- Use when appropriate – don't use when argument should not be changed by function, or if function needs to return only 1 value

6.14

تحميل الدوال

Overloading Functions

Overloading Functions

- للدوال المحملة بشكل زائد نفس الاسم لكن ذات قوائم متغيرات مختلفة (parameter lists)
- يمكن استخدامها لتوليد الدوال التي تنفذ نفس المهمة لكنها تأخذ أنواع متغيرات مختلفة أو عدد مختلف من المتغيرات parameters
- يحدد المترجم أي نوع من الدوال يجب استدعاؤه وذلك عن طريق الوسيط argument وقوائم المتغيرات parameter lists.

Function Overloading Examples

Using these overloaded functions,

```
void getDimensions(int);              // 1
void getDimensions(int, int);         // 2
void getDimensions(int, double);      // 3
void getDimensions(double, double);// 4
```

the compiler will use them as follows:

```
int length, width;
double base, height;
getDimensions(length);              // 1
getDimensions(length, width);       // 2
getDimensions(length, height);      // 3
getDimensions(height, base);        // 4
```

Function Overloading in Program 6-27

Program 6-27

```
1  // This program uses overloaded functions.
2  #include <iostream>
3  #include <iomanip>
4  using namespace std;
5
6  // Function prototypes
7  int square(int);
8  double square(double);
9
10 int main()
11 {
12    int userInt;
13    double userFloat;
14
15    // Get an int and a double.
16    cout << fixed << showpoint << setprecision(2);
17    cout << "Enter an integer and a floating-point value: ";
18    cin >> userInt >> userFloat;
19
20    // Display their squares.
21    cout << "Here are their squares: ";
22    cout << square(userInt) << " and " << square(userFloat);
23    return 0;
24 }
```

The overloaded functions have different parameter lists

Passing a **`double`**

Passing an **`int`**

(Program Continues)

Function Overloading in Program 6-27

```
26  //*************************************************************
27  // Definition of overloaded function square.                  *
28  // This function uses an int parameter, number. It returns the *
29  // square of number as an int.                                *
30  //*************************************************************
31
32  int square(int number)
33  {
34     return number * number;
35  }
36
37  //*************************************************************
38  // Definition of overloaded function square.                  *
39  // This function uses a double parameter, number. It returns  *
40  // the square of number as a double.                          *
41  //*************************************************************
42
43  double square(double number)
44  {
45     return number * number;
46  }
```

Program Output with Example Input Shown in Bold

```
Enter an integer and a floating-point value: 12 4.2 [Enter]
Here are their squares: 144 and 17.64
```

6.15

exit ( ) الدالة

The `exit( )` Function

The `exit()` Function

- تنهي تنفيذ برنامجا ما
- يمكن أن تستدعى من أي دالة
- يمكن أن تمرر قيمة int لنظام التشغيل كي تبين حالة إنهاء البرنامج.
- عادة تستخدم للإنهاء غير النظامي للبرنامج.
- تتطلب ملف ترويسة cstdlib.

The `exit()` Function

- Example:

```
exit(0);
```

- تحدد ترويسة cstdlib ثابتين يمرران بشكل عام لبيان النجاح أو الفشل:

```
exit(EXIT_SUCCESS);
exit(EXIT_FAILURE);
```

6.16

Stubs والسواقات

Stubs and Drivers

Stubs and Drivers

- مفيدة في اختبار وتفحص أخطاء برنامج ما وكذلك في تصميم ومنطق دالة ما.
- الـ Stub: هي دالة مزيفة تستخدم مكان دالة فعلية:
  - تظهر عادة رسالة تبين أنها قد استدعيت، وقد تظهر متغيرات parameters أيضا.
- السواقة: Driver هي دالة تختبر دالة أخرى عبر استدعائها.
  - تمرر الوسيطات المختلفة وتختبر قيم العودة.