من تحليل المشكلات إلى تصميم البرنامج ، الإصدار الخامس :++ Cبرمجة

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

Chapter 6: User-Defined Functions I

الفصل ٦: وظائف معرّفة من قبل المستخدم/

مراجعة البرمجة-I
Revision on Programming-I

- **TASK#1 (review on Decision structure)**

Draw a flowchart, and then write a C++ program that asks the user for his choice: 1 or 2. If the user enters 1, then the program calculates the area of a circle. If the user enters 2, then the program calculates the perimeter of a circle. In the end, the program should display the area and the perimeter to the user. (hint: Pi=3.14)

المهمة رقم ١ (مراجعة هيكل القرار)
يسأل المستخدم عن اختياره: ١ أو ٢. إذا قام المستخدم بإدخال ١ ++ Cارسم مخطط انسيابي ، ثم اكتب برنامج ، فسيحسب البرنامج مساحة الدائرة. إذا قام المستخدم بإدخال ٢ ، فسيقوم البرنامج بحساب محيط الدائرة. في النهايةPi = 3.14) ، يجب أن يعرض البرنامج المنطقة والمحيط للمستخدم. (تلميح:

Revision on Programming-I

- **TASK#2 (review on Decision and Loop structures)**
  - Design a C++ program to get sum of all odd numbers in a given range: minimum range & maximum range.
  - Draw the corresponding flowchart
  - Write the C++ code
  - Show a sample-run given that the minimum range is 25 and the maximum range is 30

المهمة رقم ٢ (مراجعة هياكل القرار والحلقة)
للحصول على مجموع كل الأرقام الفردية في نطاق معين: النطاق الأدنى والنطاق الأقصى. ++ Cصمم برنامج
ارسم المخطط الانسيابي المقابل
++ Cاكتب كود
اعرض نموذج تشغيل إذا كان النطاق الأدنى هو ٢٥ والنطاق الأقصى هو ٣٠

Revision on Programming-I

- **TASK#3 (review on Arrays)**
  - Write a program to find the largest element in a list (array of elements). Your program should ask the user for the numbers, find largest element, and then print it on the screen.
  - Write the C++ code
  - Show a sample-run suing you own input data

المهمة رقم ٣ (مراجعة على المصفوفات)
اكتب برنامجًا للعثور على أكبر عنصر في القائمة (مصفوفة من العناصر). يجب أن يطلب برنامجك من المستخدم الأرقام ، والعثور على أكبر عنصر ، ثم طباعته على الشاشة.
C ++اكتب كود
اعرض عينة قيد التشغيل تقاضي بيانات الإدخال الخاصة بك

Introduction

- Functions are like building blocks.
- They allow complicated programs to be divided into manageable pieces.

- Some advantages of functions:
  - A programmer can focus on just that part of the program and construct it, debug it, and perfect it.
  - Different people can work on different functions simultaneously.
  - Can be re-used (even in different programs)
  - Enhance program readability

الوظائف مثل اللبنات الأساسية.
إنها تسمح بتقسيم البرامج المعقدة إلى أجزاء يمكن التحكم فيها.

بعض مزايا الوظائف:
يمكن للمبرمج التركيز على هذا الجزء فقط من البرنامج وإنشائه وتصحيحه وإتقانه.
يمكن لأشخاص مختلفين العمل في وظائف مختلفة في وقت واحد.
يمكن إعادة استخدامها (حتى في البرامج المختلفة)
تعزيز إمكانية قراءة البرنامج

Introduction (cont'd.)

- **Functions**
  - Called modules
  - Like miniature programs
  - Can be put together to form a larger program

المهام
وحدات تسمى
مثل البرامج المصغرة
يمكن وضعها معًا لتشكيل برنامج أكبر

وظائف محددة مسبقا

Predefined Functions

في الجبر ، يتم تعريف الوظيفة على أنها قاعدة أو مراسلات بين القيم ، تسمى وسيطات الوظيفة ، والقيمة الفريدة للوظيفة المرتبطة بالوسيطات

- In algebra, a function is defined as a rule or correspondence between values, called the function's arguments, and the unique value of the function associated with the arguments
  - If `f(x) = 2x + 5`, then `f(1) = 7`, `f(2) = 9`, and `f(3) = 11`
    - `1`, `2`, and `3` are arguments هي الحجج
    - `7`, `9`, and `11` are the corresponding values هي القيم المقابلة

Predefined Functions (cont'd.)

- Some of the predefined mathematical functions are:

```
sqrt(x)
pow(x, y)
floor(x)
```

- **Predefined** functions are organized into **separate libraries**
- I/O functions are in `iostream` header
- Math functions are in `cmath` header

بعض الوظائف الرياضية المحددة مسبقًا هي:
الجذر التربيعي (x)
الأسرى (س ، ص)
الطابق (x)
يتم تنظيم الوظائف المحددة مسبقًا في مكتبات منفصلة
وظائف الإدخال / الإخراج موجودة في رأس iostream
وظائف الرياضيات في رأس cmath

Predefined Functions (cont'd.)

- `pow(x,y)` calculates x^y
  - `pow(2, 3) = 8.0`
  - Returns a value of type `double`
  - `x` and `y` are the parameters (or arguments)
    - The function has two parameters
- `sqrt(x)` calculates the nonnegative square root of `x`, for `x>=0.0`
  - `sqrt(2.25)` is `1.5`
  - Type `double`

- ترجع قيمة من النوع مزدوج
- xو yهما المعلمات (أو الوسائط)
- الوظيفة لها معلمتان

يحسب الجذر التربيعي غير السالب لـ x

Predefined Functions (cont'd.)

- The `floor` function `floor(x)` calculates largest whole number not greater than `x`
  - `floor(48.79)` is `48.0`
  - Type `double`
  - Has only one parameter

تحسب وظيفة floor (x) أكبر عدد صحيح لا يزيد عن x
floor(٤٨,٧٩) هي ٤٨,٠
نوعها مزدوج
لها معلمة واحدة فقط

Predefined Functions (cont'd.)

TABLE 6-1 Predefined Functions

Function	Header File	Purpose	Parameter(s) Type	Result
`abs(x)`	`<cstdlib>`	Returns the absolute value of its argument: `abs(-7) = 7`	`int`	`int`
`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`

Predefined Functions (cont'd.)

TABLE 6-1 Predefined Functions (continued)

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 `true` if x is a lowercase letter; otherwise, it returns `false`; `islower('h')` is `true`	`int`	`int`
`isupper(x)`	`<cctype>`	Returns `true` if x is a uppercase letter; otherwise, it returns `false`; `isupper('K')` is `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

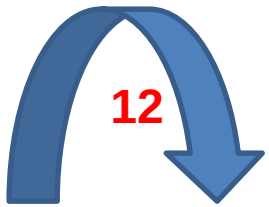
EXAMPLE 6-1

```
//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;
}
```

Example 6-1 sample run:

تشغيل العينة

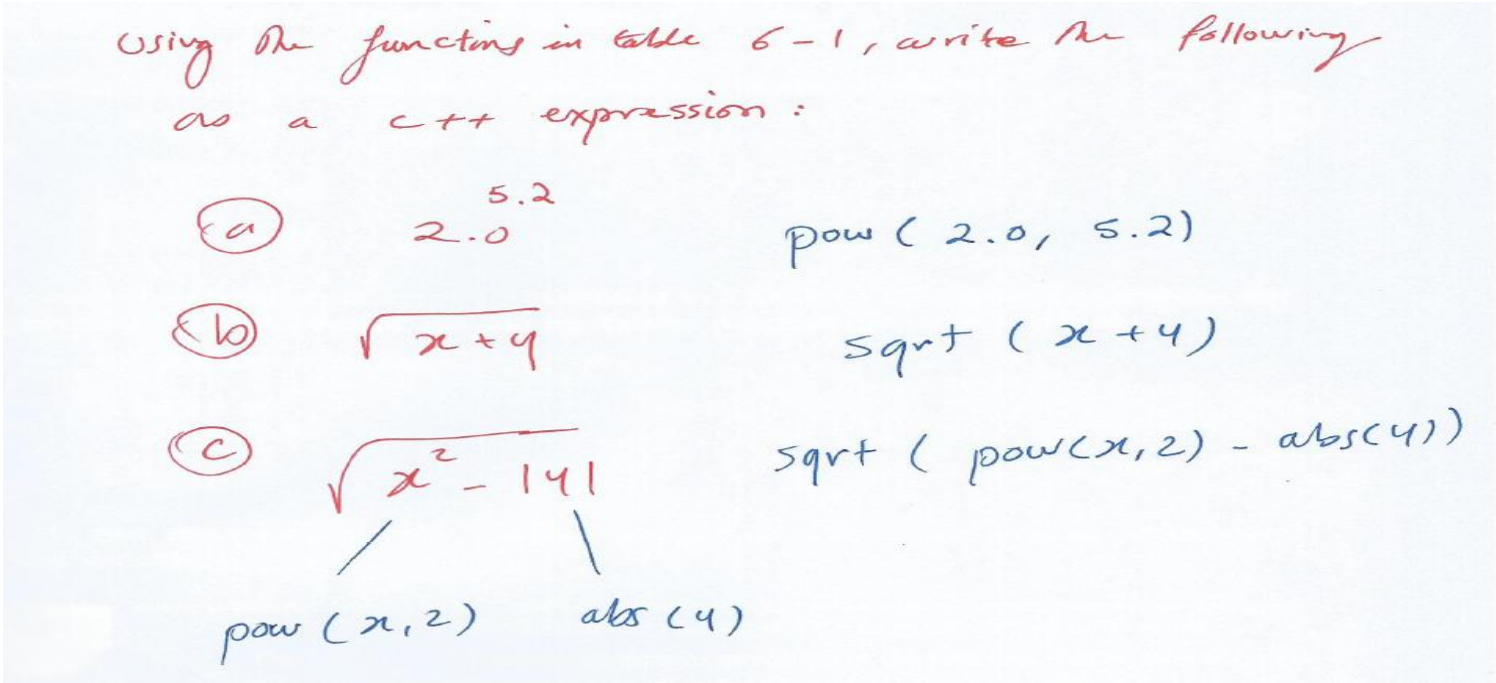
```
Line 1: Uppercase a is A
Line 4: 4.2 to the power of 3 = 74.088
Line 5: 5.0 to the power of 4 = 625
Line 7: u = 31.2
Line 9: Absolute value of -15 = 15
```



Examples

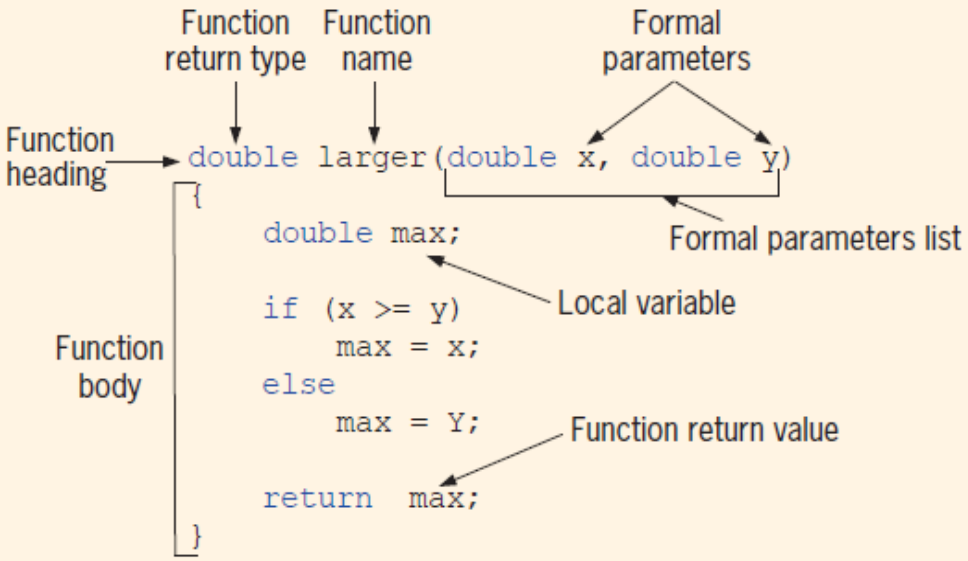
Using the predefined functions in Table 6-1, write the following as C++ expressions:

باستخدام الوظائف المحددة مسبقًا في الجدول ٦-١ ، اكتب ما يلي على هيئة تعبيرات :C ++



Various Parts of a Function

أجزاء مختلفة من الدوال



User-Defined Functions

وظائف من تحديد المستخدم

- **Value-returning functions**: have a return type
  - Return a value of a specific data type using the `return` statement
- **Void functions:** do not have a return type
  - *Do not* use a `return` statement to return a value

وظائف إرجاع القيمة: لها نوع إرجاع
إرجاع قيمة نوع بيانات معين باستخدام بيان الإرجاع

دوال باطلة: ليس لها نوع إرجاع
لا تستخدم بيان الإرجاع لإرجاع قيمة

Value-Returning Functions
وظائف إرجاع القيمة

- To use these functions you must:
  - Include the appropriate header file in your program using the include statement
  - Know the following items:
    - Name of the function
    - Number of parameters, if any
    - Data type of each parameter
    - Data type of the value returned: called the type of the function

لاستخدام هذه الوظائف ، يجب عليك:
قم بتضمين ملف الرأس المناسب في برنامجك باستخدام العبارة include
تعرف على العناصر التالية:
اسم الوظيفة
عدد المعلمات ، إن وجدت
نوع البيانات لكل معلمة
نوع بيانات القيمة التي تم إرجاعها: يسمى نوع الوظيفة

Value-Returning Functions (cont'd.)

- Because the value returned by a value-returning function is unique, must:
  - Save the value for further calculation
  - Use the value in some calculation
  - Print the value
- A value-returning function is used in an assignment or in an output statement

نظرًا لأن القيمة التي تُرجعها دالة إرجاع القيمة فريدة ، يجب:
احفظ القيمة لمزيد من الحساب
استخدم القيمة في بعض العمليات الحسابية
اطبع القيمة
يتم استخدام دالة إرجاع القيمة في مهمة أو في بيان الإخراج

Value-Returning Functions (cont'd.)

```
int abs(int number)
{
    if (number < 0)
        number = -number;

    return number;
}
```

Value-Returning Functions (cont'd.)

- **Heading**: first four properties above
  - Example: `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)`

العنوان: أول أربع خصائص أعلاه
مثال
المعامل الرسمي: المتغير المعلن في العنوان
مثال: رقم
المعلمة الفعلية: متغير أو تعبير مدرج في استدعاء دالة
مثال

Syntax: Value-Returning Function

- Syntax:

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

- **`functionType`** is also called the **data type** or **return type**

تسمى functionType أيضًا بنوع البيانات أو نوع الإرجاع

Syntax: **Formal Parameter List**

بناء الجملة: قائمة المعاملات الرسمية

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

Function Call

استدعاء الداله

```
functionName(actual parameter list)
```

Syntax: Actual Parameter List

بناء الجملة: قائمة المعلمات الفعلية

- The syntax of the **actual parameter** list is:

صيغة قائمة المعلمات الفعلية هي:

```
expression or variable, expression or variable, ...
```

- **Formal parameter list** can be empty:

يمكن أن تكون قائمة المعلمات الرسمية فارغة:

```
functionType functionName()
```

استدعاء دالة إرجاع القيمة بقائمة معلمات رسمية فارغة هو:

- **A call to a value-returning function** with an empty formal parameter list is:

```
functionName()
```

Return Statement
بيان العودة

- Once a value-returning function computes the value, the function **returns** this value via the **`return`** statement
  - It **passes** this value outside the function via the `return` statement

بمجرد أن تقوم دالة إرجاع القيمة بحساب القيمة ، تقوم الدالة بإرجاع هذه القيمة عبر
بيان الإرجاع
يمرر هذه القيمة خارج الوظيفة عبر بيان الإرجاع

Syntax: `return` Statement

- The **`return` statement** has the following syntax: بيان الإرجاع له الصيغة التالية:

```
return expr;
```

- In C++, `return` is a **reserved word**
- When a return statement executes
  - **Function immediately terminates**
  - **Control goes back to the caller**
- When a `return` statement executes in the function **`main`**, the **program terminates**

في C ++، العودة هي كلمة محجوزة
عندما يتم تنفيذ بيان العودة
تنتهي الوظيفة على الفور
يعود التحكم إلى المتصل
عندما يتم تنفيذ تعليمة العودة في الوظيفة main، ينتهي البرنامج

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

```
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.

Function Prototype
نموذج الوظيفة

- **Function prototype:** function heading without the body of the function
- Syntax:

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

- It is not necessary to specify the variable name in the parameter list
- The data type of each parameter must be specified

نموذج الوظيفة: عنوان الوظيفة بدون جسم الوظيفة
بناء الجملة:

ليس من الضروري تحديد اسم المتغير في قائمة المعلمات
يجب تحديد نوع البيانات لكل معلمة

Example 6-2

Write a program that uses functions: ***larger***, ***compareThree***, and main to determine the larger/largest of two or three numbers

اكتب برنامجًا يستخدم الدوال: أكبر ، قارن ثلاثة ، وأساسي لتحديد أكبر / أكبر رقمين أو ثلاثة

Function Prototype (cont'd.)

```
//Program: Largest of three numbers

#include <iostream>

using namespace std;

double larger(double x, double y);
double compareThree(double x, double y, double z);

int main()
{
    double one, two;                                    //Line 1

    cout << "Line 2: The larger of 5 and 10 is "
         << larger(5, 10) << endl;                      //Line 2

    cout << "Line 3: Enter two numbers: ";              //Line 3
    cin >> one >> two;                                  //Line 4
    cout << endl;                                       //Line 5

    cout << "Line 6: The larger of " << one
         << " and " << two << " is "
         << larger(one, two) << endl;                   //Line 6
    cout << "Line 7: The largest of 23, 34, and "
         << "12 is " << compareThree(23, 34, 12)
         << endl;                                       //Line 7

    return 0;
}
```

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

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

    return max;
}

double compareThree (double x, double y, double z)
{
    return larger(x, larger(y, z));
}
```

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

```
Line 2: The larger of 5 and 10 is 10
Line 3: Enter two numbers: 25.6 73.85

Line 6: The larger of 25.6 and 73.85 is 73.85
Line 7: The largest of 43.48, 34.00, and 12.65 is 43.48
```

Function Prototype (cont'd.)

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

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

    return max;
}

double compareThree (double x, double y, double z)
{
    return larger(x, larger(y, z));
}
```

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

```
Line 2: The larger of 5 and 10 is 10
Line 3: Enter two numbers: 25.6 73.85

Line 6: The larger of 25.6 and 73.85 is 73.85
Line 7: The largest of 23, 34, and 12 is 34.0
```

الإخراج المصحح

Corrected output

Value-Returning Functions: Some Peculiarity

وظائف إرجاع القيمة: بعض الخصوصية

```
int secret(int x)
{
    if (x > 5)                  //Line 1
        return 2 * x;           //Line 2
}
```

A correct definition of the function secret is:

```
int secret(int x)
{
    if (x > 5)                  //Line 1
        return 2 * x;           //Line 2

    return x;                   //Line 3
}
```

Value-Returning Functions: Some Peculiarity (cont'd.)

```
return x, y;   //only the value of y will be returned


int funcRet1()
{
     int x = 45;

     return 23, x;   //only the value of x is returned
}

int funcRet2(int z)
{
     int a = 2;
     int b = 3;

     return 2 * a + b, z + b; //only the value of z + b is returned
}
```

Question Challenge تحدي السؤال

Write a program that reads two numbers to find their sum. Re-write the program using a function called *Add*.

اكتب برنامجًا يقرأ عددين لإيجاد مجموعهما. أعد كتابة البرنامج باستخدام وظيفة تسمى Add.

Example 6-3

Write a function ***courseGrade*** that reads a course ***score*** to find the ***grade***.

اكتب دورة وظيفية: الدرجة التي تقرأ درجة مقرر دراسي للعثور على الدرجة.

Value-Returning Functions:
More Examples

EXAMPLE 6-3

In this example, we write the definition of function `courseGrade`. This function takes as a parameter an `int` value specifying the score for a course and returns the grade, a value of type `char`, for the course. (We assume that the test score is a value between `0` and `100` inclusive.)

```
char courseGrade(int score)
{
    switch (score / 10)
    {
    case 0:
    case 1:
    case 2:
    case 3:

    case 4:
    case 5:
        return 'F';
    case 6:
        return 'D';
    case 7:
        return 'C';
    case 8:
        return 'B';
    case 9:
    case 10:
        return 'A';
    }
}
```

You can also write an equivalent definition of the function `courseGrade` that uses an `if...else` structure to determine the course grade.

Example 6-3, Continued

Think of the full C++ program that asks the student for his course *Score*, and then uses the function *courseGrade* to find the *grade*.

فكر في برنامج C ++ الكامل الذي يطلب من الطالب الحصول على درجة المقرر ، ثم يستخدم الدورة التدريبية الوظيفية ابحث عن الدرجة.

Value-Returning Functions:
More Examples

EXAMPLE 6-3

In this example, we write the definition of function `courseGrade`. This function takes as a parameter an `int` value specifying the score for a course and returns the grade, a value of type `char`, for the course. (We assume that the test score is a value between 0 and 100 inclusive.)

```
char courseGrade(int score)
{
    switch (score / 10)
    {
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        return 'F';
    case 6:
        return 'D';
    case 7:
        return 'C';
    case 8:
        return 'B';
    case 9:
    case 10:
        return 'A';
    }
}
```

1- Prototype,
Char courseGrade(int score);

2- In main(),
Cout<<"enter your score: ";
Cin>>score;

Then
Cout<<"your grade is: "<<courseGrade(score) ;

Or
Char Grade;
Grade=courseGrade(score)
Cout<<"your grade is "<<Grade;

3-Write the body of the function

You can also write an equivalent definition of the function `courseGrade` that uses an `if...else` structure to determine the course grade.

تدفق التنفيذ

Flow of Execution

- Execution always **begins** at the first statement in the function `main`
- Other functions are executed **only when they are called**
- **Function prototypes** appear before any function definition
  - The compiler translates these first
- The compiler can then correctly translate a function call

يبدأ التنفيذ دائمًا عند العبارة الأولى في الوظيفة main
يتم تنفيذ الوظائف الأخرى فقط عندما يتم استدعاؤها
تظهر نماذج الوظائف قبل أي تعريف للدالة
المترجم يترجم هذه أولا
يمكن للمترجم بعد ذلك ترجمة استدعاء دالة بشكل صحيح

Flow of Execution (cont'd.)

- A function call results in transfer of **control** to the first statement in the body of the called function
- After the last statement of a function is executed, **control** is passed back to the point immediately following **the function call**
- A value-returning function **returns** a value
  - After executing the function the returned value replaces the function call statement

ينتج عن استدعاء الوظيفة نقل التحكم إلى العبارة الأولى في جسم الوظيفة التي تم استدعاؤها
بعد تنفيذ الجملة الأخيرة للدالة ، يتم تمرير التحكم مرة أخرى إلى النقطة التي تلي استدعاء الوظيفة مباشرة
تقوم دالة إرجاع القيمة بإرجاع قيمة
بعد تنفيذ الوظيفة ، تحل القيمة المرتجعة محل عبارة استدعاء الوظيفة

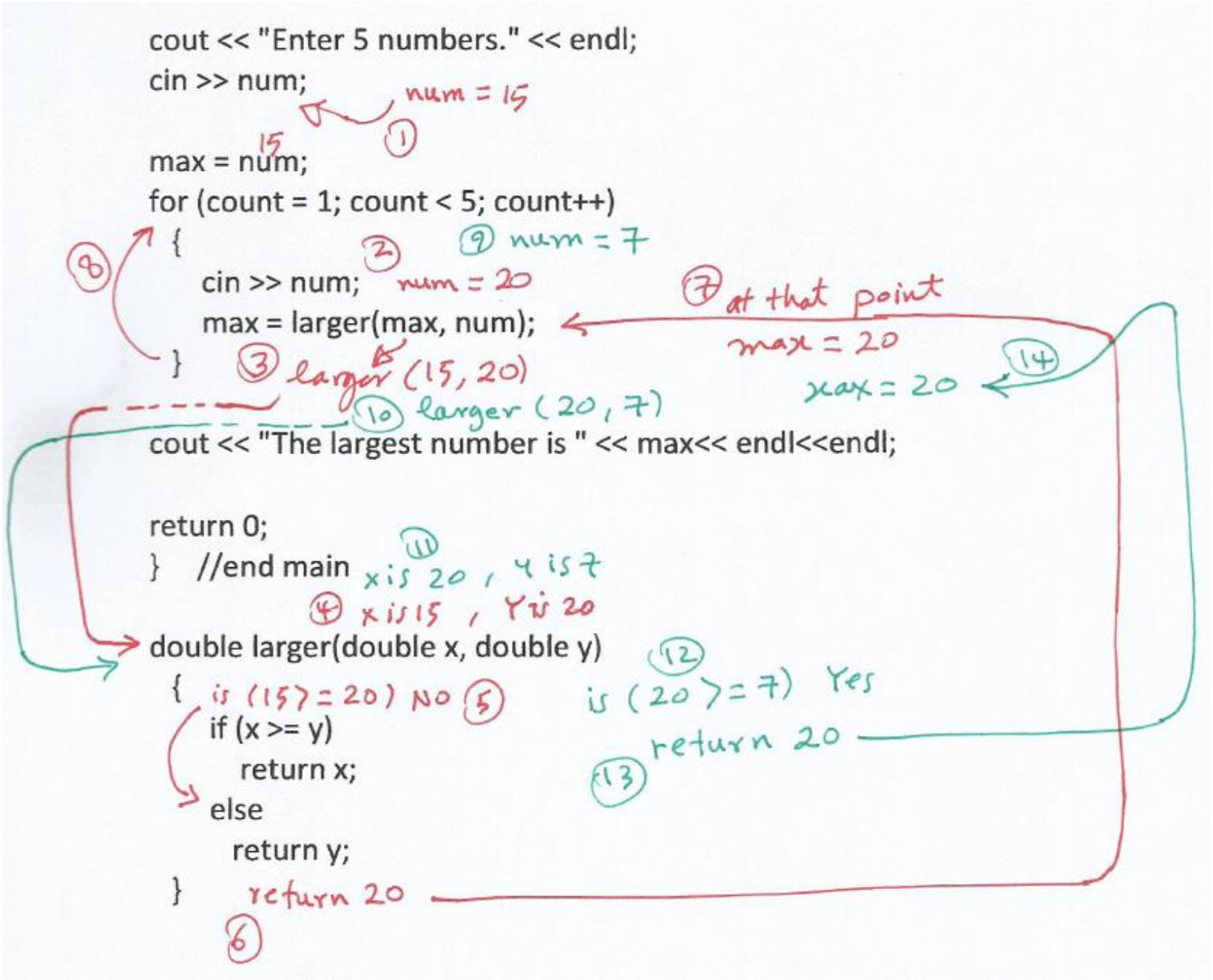
CHAPTER REVIEW

Question Challenge

Think of the full C++ program that uses the function *larger* to determine the largest number from a set of 5 numbers

-Group Work-

فكر في برنامج C ++ الكامل الذي يستخدم دالة أكبر لتحديد أكبر رقم من مجموعة من ٥ أرقام
-مجموعة عمل-


cout << "Enter 5 numbers." << endl;
cin >> num;
num = 15
1
max = num;
15
for (count = 1; count < 5; count++)
{
8
2
9 num = 7
cin >> num;
num = 20
7 at that point
max = 20
max = larger(max, num);
}
3 larger (15, 20)
14
xax = 20
10 larger (20, 7)
cout << "The largest number is " << max<< endl<<endl;
return 0;
} //end main
11
x is 20 , y is 7
4 x is 15 , y is 20
double larger(double x, double y)
{
is (15) = 20) NO 5
12
is (20 >= 7) Yes
if (x >= y)
return x;
return 20
13
else
return y;
}
return 20
6