University of Mosul

College of Computer Sciences and Mathematics

Department of Artificial Intelligence

Algorithms and Structured Programming (2)

First stage

Lecture 1

ا.م. بيداء سليمان بهنام

Assist.Prof.Baydaa Sulaiman Bahnam

Lecture 1 Outline

1. Introduction
2. Functions
3. User-defined Functions
4. General form of Function
5. Types of C++ User-defined Function According to Parameters (or Arguments) and Return Values

Introduction

A function is a set of statements designed to accomplish a particular task. Experience has shown that the best way to develop and maintain a large program is to construct it from smaller pieces or (modules). Modules in C++ are called functions.

Functions are very useful to read, write, debug and modify complex programs. They can also be easily incorporated in the main program. In C++, the main() itself is a function that means the main function is invoking the other functions to perform various tasks. The main advantages of using a function are:

- Easy to write a correct small function.
- Easy to read, write, and debug a function.
- Easier to maintain or modify such a function.
- Small functions tend to be self documenting and highly readable.
- It can be called any number of times in any place with different parameters.

Functions

A function in C++ is a block of code written by the programmer to perform a specific task. E.g. function to calculate the area of a square. A function executes only when it is called. It is an important aspect of code reusability and understandability. You can divide your program into as many functions as you want to solve a difficult task.

There are two types of functions in C++:

1. **Standard Library Functions: built-In functions in C++ (Predefined in C++).**
2. **User-defined Functions: Created by users.**

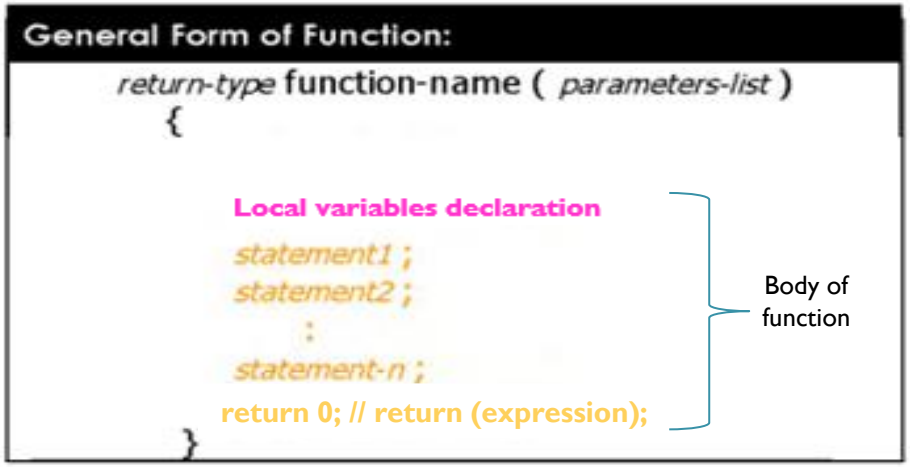
2. User-defined Functions: Created by users.

C++ allows the programmer to define their own function.
A user-defined function groups code to perform a specific task and that group of code is given a name (identifier).
When the function is invoked from any part of the program, it all executes the codes defined in the body of the function

General form of Function

Defining a Function

A function definition has a name, parentheses pair containing zero or more parameters and a body. For each parameter, there should be a corresponding declaration that occurs before the body. Any parameter not declared is taken to be an integer by default. The general format of the function definition is:



return-type signifies the type of value returned by the function. e.g. void, int, float, char, string ,struct...

function-name specifies the name of the function. You must give a function name relevant to the task the function performs. It increases understandability. e.g. area()

parentheses () are placed after the function name. Parentheses () can be empty or contain parameters.

Parameters Information can be passed to functions as a parameter. Parameters act as variables inside the function. Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma;

function body in the {} contains the code to perform the given task.

return statement: the keyword return is used to terminate function and return a value to its caller. The return statement may also be used to exit a function without returning a value. The return statement may or may not include an expression. Its general syntax is:

```
return;
return (expression);
```

Return-type	
void function_name (---);	دوال لا ترجع أي قيمة
int function_name (---);	دوال ترجع قيمة صحيحة
float function_name (---);	دوال ترجع قيمة حقيقية
char function_name (---);	دوال ترجع حرف واحد
string function_name (---);	دوال ترجع عبارة حرفية
struct function_name (---);	دوال ترجع قيمة من نوع **structure**

parentheses pair containing zero or more parameters

Functions

return type *parameters*

```
type functionName(type name, type name, ..., type name) {
    statement;
    statement;
    ...
    statement;
    return expression; // if return type is not void
}
```

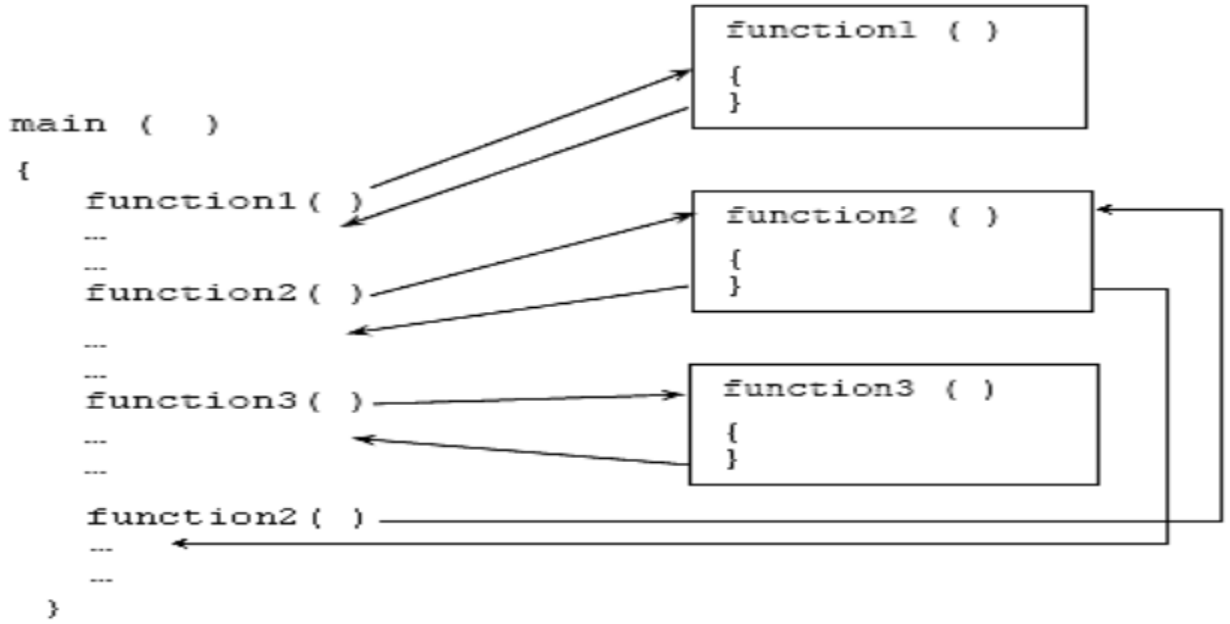
*arguments (called in the **same order** as the parameters)*

Calling a function:

```
functionName(value, value, ..., value);
```

The main program calls the function by declaring a variable followed by the function name and its arguments, as follows:

type variable_name=function_name(arguments)



Functions

Types of C++ User-defined Function According to Parameters (or Arguments) and Return Values

A parameter (or arguments) is a value that is passed when declaring a function. For better understanding of parameters (or arguments) and return values in functions, user-defined functions can be in various forms like:

- ✓ **Function with no parameters and no return value**
- ✓ **Function with parameters and no return value**
- ✓ **Function with no parameters and with return value**
- ✓ **Function with parameters and return value**

Function with no parameters and no return value

Function Declaration and Definition, Call a function

Syntax

```
void myFunction()
{
  // code to be executed
}
```

Example Explained

myFunction() is the name of the function.

void means that the function does not have a return value.

inside the function (the body), add code that defines what the function should do.

Call a Function

To call a function, write the function's name followed by two parentheses ()and a semicolon;

In the following example, myFunction() is used to print a text (the action), when it is called:

```
// Create a function
void myFunction() {
cout << "I just got executed!";
}
int main() {
myFunction(); // call the function inside the main
return 0;
}
// Outputs "I just got executed!"
```

Function Declaration and Definition, Call a function

Either this declaration :
Example

```
// Function declaration and definition
void myFunction()
{
  cout << "I just got executed!";
}
// The main
int main()
{
  myFunction();  // call the function
   return 0;
}
```

Note: If a user-defined function, such as myFunction() is declared after the main() function, **an error will occur**. It is because C++ works from **top to bottom;** which means that if the function is not declared above main(), the program is unaware of it:

ملاحظة لجميع انواع الدوال التي يتم تعريفها من قبل المستخدم : إذا تم إعلان دالة محددة من قبل المستخدم، مثل function-name() بعد البرنامج الرئيسي main()، فسيحدث خطأ. وذلك لأن C++ تعمل من أعلى إلى أسفل؛ مما يعني أنه إذا لم يتم إعلان الدالة أعلى main()، فلن يكون البرنامج على علم بذلك. لذلك :

اما يتم الاعلان عن الدالة وتعريفها قبل البرنامج الرئيسي

Or this declaration
Example

او يتم ألاعلان عن الدالة ثم يتم تعريفها بعد البرنامج الرئيسي نستفاد من هذه الطريقة عند كتابة البرامج الطويلة ولغرض الترتيب

```
// Function declaration
void myFunction();

// The main
int main()
{
  myFunction();  // call the function
  return 0;
}

// Function definition
void myFunction()
 {
  cout << "I just got executed!";
}
```

عند الاعلان عن الدالة نضع فارزة منقوطة

عند استدعاء الدالة نضع فارزة منقوطة

عند تعريف الدالة (اي كتابة الدالة) لانضع فارزة منقوطة

Function with no parameters and no return value

Write a c++ function named (info) to print University, College, Department then call it in the main program.

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

// Define the function info (This function with no arguments and no return value)

void info()
{
    // Print the required statements
    cout << "University of Mosul" << endl;
    cout << "   ************   " << endl;
    cout << "College of Computer Science and Mathematics" << endl;
    cout << "   ************   " << endl;
    cout << "Department of Artificial Intelligence" << endl;
    cout << "   ************   " << endl;
}

int main()
{
    // Call the info function
    info();
    return 0;
}
```

لا نضع ; عند كتابة الدالة

```
University of Mosul
   ************
College of Computer Science and Mathematics
   ************
Department of Artificial Intelligence
   ************
```

يجب وضع ; عند استدعاء الدالة

هذا النوع من الدوال الذي لايرجع اي قيمة (اي من نوع void) يسمى ايضا بالـ procedure

Here,

- We haven't passed any argument to info().
- The return type of info() is **void** so it doesn't return any value.

Function with parameters (arguments) and no return value

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

// Function that takes two integers as arguments and calculates their sum (but does not return it)
void sum(int num1, int num2)
{
    int result = num1 + num2;  // Calculate the sum
    cout << "The sum of the two numbers is: " << result << endl;  // Print the result
// cout << "The sum of the  "<< num1<< " and " <<num2<<" is : " << result << endl;  // Print the result
}
int main()
{
    int a, b;

    // Ask the user to enter two integers
    cout << "Enter the first number: ";
    cin >> a;
    cout << "Enter the second number: ";
    cin >> b;

    // Call the sum function with user inputs
    sum(a, b);

    // Call the sum function with predefined values
    sum(10, 90);

    return 0;
}
```

ممكن ايضا بهذه الطريقة

```
Enter the first number: 70
Enter the second number: 60
The sum of the two numbers is: 130
The sum of the two numbers is: 100
```

Here,

- We have passed two parameters of type int to sum function.
- The return type of sum function is **void** **so it doesn't return any value**.
- The main() function calls sum function with two parameters a and b.

Function with no parameters (arguments) and with return value

Return Values

The `void` keyword, used in the previous examples, indicates that the function should not return a value. If you want the function to return a value, you can use a data type (such as `int`, float, char, `string`, etc.) instead of `void`, and use the `return` keyword inside the function:

Write a c++ function named (info) which returns the product of two numbers (v1=5 and v2=10) to the main program.

```
The product of the two values is: 50
```

```cpp
#include <iostream>
using namespace std;
// Define the function info which returns the product of two numbers (v1=5 and v2=10)
int mul()
{
    int v1 = 5;
    int v2 = 10;
    // Return the product of the two values v1 and v2
    return v1 * v2;
}

int main()
{
    // Call the mul function and store the returned value
    int res = mul();

    // Print the result   res
    cout << "The product of the two values is: " << res << endl;
    return 0;
}
```

Here,

- We haven't passed any argument to `mul() function`.
- The return type of `mul()` is **int** so it returns the product of the two values v1 and v2
- The `mul()` function takes user input for a name, and returns the name.
- The main() function gives the output based on the value returned by `mul()`.

Function with no parameters (arguments) and with return value

نفس البرنامج السابق لكن الدالة من نوع float

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

// Define the function info which returns the product of two numbers
float mul()
{
    float v1 = 7.5;
    float v2 = 13.7;
    // Return the product of the two values v1 and v2
    return v1 * v2;
}

int main()
{
    // Call the mul function and store the returned value
    float res = mul();

    // Print the result   res
    cout << "The product of the two values is: " << res << endl;
    return 0;
}
```

```
The product of the two values is: 102.75
```

اذا بالاستدعاء عملنا نوع الدالة int بدلا من float سوف يتم ارجع القيمة الصحيحة فقط اي مثلا هنا 102

Here,

- We haven't passed any argument to `mul() function`.
- The return type of `mul()` is **float** so it returns the product of the two values v1 and v2
- The `mul()` function takes user input for a name, and returns the name.
- The main() function gives the output based on the value returned by `mul()`.

Function with parameters (arguments) and with return value

Example

```
int myFunction(int x)
{
  return 5 + x;
}

int main() {
  cout << myFunction(3);
  return 0;
}
// Outputs 8
```

The following example returns the sum of a function with **two parameters**:

Example

```
int myFunction(int x, int y) {
return x + y;}
int main() {
cout << myFunction(5, 3);
 return 0;
}
// Outputs 8
```

You can also store the result in a variable:

Example

```
int myFunction(int x, int y)
 { return x + y;
}
int main() {
int z = myFunction(5, 3);
cout << z;
return 0;
}
// Outputs 8
```

Function with parameters (arguments) and with return value

Write a C++ function that calculate the sum of two integers and returns the result to the main program.

الدالة من هذا النوع تتم في داخلها عملية المعالجة فقط بدون قراءة او طباعة وهذه هي مهمة الدالة الصحيحة.

```
#include <iostream>
using namespace std;
// Function that returns the sum of two integers
int sum(int num1, int num2)
{
    return num1 + num2;  // Return the sum of num1 and num2
}

int main()
{
    int a, b;
    // Ask the user to enter two integers
    cout << "Enter the first number: ";
    cin >> a;
    cout << "Enter the second number: ";
    cin >> b;

    // Call the sum function and store the result
    int result = sum(a, b);
    // Print the result
    cout << "The sum of the two numbers is: " << result << endl;
                int result2 = sum(10, 90);
    cout << "The sum of the two numbers is: " << result2 << endl;
    return 0;
}
```

```
Enter the first number: 66
Enter the second number: 55
The sum of the two numbers is: 121
```

ممكن استدعاء الدالة لقيم اخرى ايضا

Here,

- We have passed two parameters of int type to sum function.
- The return type of sum function is **int** .
- The main() function calls sum function with two parameters.
- The sum function returns the sum of num1 and num2. Then main() gives the output based on the value returned by sum function .