جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الأمن السيبراني

Lecture: (1)

The function

Subject: Structured Programming
First Stage
Lecturer: Asst. Prof. Dr. Ali Kadhum Al-Quraby

The function

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:

1. Easy to write a correct small function.
2. Easy to read, write, and debug a function.
3. Easier to maintain or modify such a function.
4. It can be called any number of times in any place with different parameters.

Depending on whether a function is predefined or created by programmer; there are two types of function:

1. **Library function**
2. **User-defined function**

1. Library function

Library functions are the built-in function in C++ programming. Programmer can use library function by invoking (استدعاء) function directly; they don't need to write it themselves.

Ex 1:

```
#include <iostream.h>
#include <cmath.h>

int main ()

{

    double number, squareRoot;

    cout << "Enter a number: ";

    cin >> number;
    // sqrt() is a library function to calculate square root
    squareRoot = sqrt(number);
    cout << "Square root of "<<number<<"="<< squareRoot<<endl;

    return (0);

}
```

Output

```
Enter a number: 26

Square root of 26 = 5.09902
```

In the example above, **sqrt()** library function is invoked to calculate the square root of a number. Notice code #include <cmath> in the above program. Here, cmath is a header file. The function definition of sqrt()(body of that function) is present in the cmath header file. You can use all functions defined in cmath when you include the content of file cmath in this program using #include <cmath>. **Every valid C++ program has at least one function, that is, main () function.**

2. User-defined function

C++ allows programmer to define his or her 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 program, it all executes the codes defined in the body of function.

- **How user-defined function works?**

In order to understand how user-defined function works in C++ Programming, let us see the following C++ program.

Ex 2:

-Write a C++ program to add two integers using function.

```
#include <iostream.h>
// Function prototype (declaration)
int add_two_numbers(int, int);

int main()

{
        int num1, num2, result;
        cout<<"Enter two numbers: ";

        cin >> num1 >> num2;
        // Function call
        result = add_two_numbers(num1, num2);
        cout << "The result is = " << result<<"\n";

        return 0;

}
// Function definition
```

Output

```
Enter two numbers: 12 4
```

```
int add_two_numbers(int a, int b)

{
        int sum=0;

        sum = a + b;
        // Return statement
        return sum;

}
```

1. **Function prototype (declaration)**

If a user-defined function is defined after **main()** function, compiler will show error. It is because compiler is unaware of user-defined function, types of parameters passed to function and return type. In C++, function prototype is a declaration of function without its body to give compiler information about user-defined function. A function declaration has the following form: -

return-type **function_name** (parameter1, parameter2, ...**);**

For example, in the above example the declaration of function is:

int **add_two_numbers**(int, int);

You can see that, there is no body of function in prototype. Also, there are only return type of parameter but no parameter.

2. **Function definition**

A function definition has the following form:-

return-type **function_name** (parameter1, parameter2, ...**)**

```
{
    // function-body
}
```

- return_type- Some functions return value. This parameter denotes the data type of the return value. Some functions don't return value. In that case, the value of this parameter becomes void.
- **function_name**- This is the name of the function. The function name and parameters form the function signature.
- Parameters- This is the type, the order, and the number of function parameters. Some functions don't have parameters.
- function body- these are statements defining what the function will do.

For example, in the above example the definition of function is:

```
// Function definition
int add_two_numbers(int a, int b)

{
    // function-body

    int sum=0;

    sum = a + b;
    // Return statement
    return sum;

}
```

3. **Function calling**

When a program calls a function, program control is transferred to

the called function. A called function performs defined task and when it's return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program. To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value. In the above example, two variables, num1 and num2 are passed to function during function call. These arguments are known as actual arguments.

The value of **num1** and **num2** are initialized to variables **a** and **b** respectively. These arguments a and b are called formal arguments.

- **Types of functions**

The user defined functions may be classified in the following three ways based on the formal arguments passed and the usage of the return statement, and based on that, there are three of user defined functions.

1. **Functions with no parameters and no return value**

Function with no parameter and with no return type which does not return value because its return type is void. To declare a function that can only be called without any parameter, we should use void **function_name**().

Ex 3:

- Write a C++ program to print **Hello student** !!! using function.

```
#include <iostream.h>

#include<conio.h>

// Function prototype (declaration)
```

```
void printmessage ();

void main ()

{

// Function call

        printmessage ();

        cout<<"\n";

        getch();

}

// Function definition

void printmessage ()

{

        cout << "Hello student";

}
```

Output

```
Hello student !!!
```

2. **Functions with no parameters and with return value**

This type of function, there is no parameter passed through the calling function to called function but the function returns value.

Ex 4:

- Write a C++ program to read number and increment it by 10, using function.

```
#include <iostream>

// Function prototype (declaration)

int enter_number();
int main()
{
        int result,number;
        cout << "Enter Value: ";

        // Function call

        number=enter_number();
        result=number+10;
        cout<<"The result is:"<<result;
        return (0);
}

// Function definition

int enter_number()
{
        int x;
        cin>>x;
        return x;
}
```

Output

```
Enter Value: 12
```

3. Functions with parameters and with no return value

Function with parameter and with no return type which does not return value because its return type is **void**.

Ex 5:

- Write a C++ program to display two integer numbers after adding values for each one.

```
#include <iostream.h>
// Function prototype (declaration)
void displayNum(int, int);
int main()
{
        int num1,num2;
        cout<<"Enter the first no.:";
        cin>>num1;
        cout<<"Enter the second no.:";
        cin>>num2;
        // Function call
        displayNum(num1, num2);
        return (0);
}
// Function definition
void displayNum(int n1, int n2)
{
        cout <<n1<< "  added by 3 and the result is: "<<n1+3<<endl;
        cout <<n2<< "  added by 10 and the result is: "<<n2+10;
}
```

Output

```
Enter the first no.:10
Enter the second no.:5
10 added by 3 and the result is: 13
```