Lecture-14-

Functions

- A **function** is a group of statements that perform a particular task.
- You may define your own functions in C++.
- Using functions can have many advantages, including the following:
  - You can reuse the code within a function.
  - You can easily test individual functions.
  - If it's necessary to make any code modifications, you can make modifications within a single function, without altering the program structure.
  - You can use the same function for different inputs.
- Every valid C++ program has at least one function - the **main()** function.

The Return Type

- The **main** function takes the following general form:

```
int main()
{
// some code
return 0;
}
```

- A function's **return type** is declared before its name.
- In the example above, the return type is **int**, which indicates that the function returns an integer value.
- Occasionally, a function will perform the desired operations without returning a value.
- Such functions are defined with the keyword **void**.
- **void** is a basic data type that defines a valueless state.

Defining a Function

- Define a C++ function using the following syntax:

```
return_type function_name( parameter list )
{
body of the function
}
```

- **return-type**: Data type of the value returned by the function.
- **function name**: Name of the function.

- **parameters**: When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function.
- **body of the function**: A collection of statements defining what the function does.
- Parameters are **optional**; that is, you can have a function with no parameters.
- As an example, let's define a function that does not return a value, and just prints a line of text to the screen.

```
void printSomething()
{
cout << "Hi there!";
}
```

- Our function, entitled **printSomething**, returns **void**, and has no parameters.
- Now, we can use our function in **main()**.

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

- To **call** a function, you simply need to pass the required parameters along with the function name.

Functions

- You must declare a function prior to calling it.
- Example:

```
#include <iostream>
using namespace std;
void printSomething() {
cout << "Hi there!";
}
```

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

- Putting the declaration after the **main()** function results in an error.
- A function **declaration**, or **function prototype**, tells the compiler about a function name and how to call the function.
- The actual body of the function can be defined separately.
- **Example:**

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

//Function declaration
void printSomething();

int main() {
printSomething();

return 0;
}

//Function definition
void printSomething() {
cout << "Hi there!";
}
```

- Function declaration is required when you define a function in one source file and you call that function in another file.
- In such case, you should declare the function at the top of the file calling the function.

Function Parameters

- For a function to use **arguments**, it must declare formal **parameters**, which are variables that accept the argument's values.
- **Argument:** a piece of data that is passed into a function or a program.
-

- Example:

```
void printSomething(int x)
{
cout << x;
}
```

- This defines a function that takes one **integer** parameter and prints its value.
- Formal parameters behave within the function similarly to other local variables.
- They are created upon entering the function and are destroyed upon exiting the function.
- Once parameters have been defined, you can pass the corresponding arguments when the function is called.
- **Example:**

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

void printSomething(int x) {
cout << x;
}

int main() {
printSomething(42);
}

// Outputs 42
```

- The value 42 is passed to the function as an **argument** and is assigned to the formal **parameter** of the function: **x**.
- Making changes to the parameter within the function does not alter the argument.
- You can pass different arguments to the same function.
- For example:

```
int timesTwo(int x) {
    return x*2;
}
```

- The function defined above takes one integer parameter and returns its value, multiplied by 2.
- We can now use that function with different arguments.

```
int main() {
cout << timesTwo(8);
// Outputs 16

cout <<timesTwo(5);
// Outputs 10

cout <<timesTwo(42);
// Outputs 84
}
```