C++ Programming Language

Presented By:

T.A. Asmaa Hamad El-saied

E-mail: eng.asmaa134@gmail.com

Contents

Introduction

- **program**

  is a set of instructions a computer follows in order to perform a task.

- **Programming Language**:

  - Is a language between the problem that we want to solve and the computer that execute commands written in that language by Programmers
  - So it is a series of separate parts which fitted together to construct a solution to the problem in hand.

Introduction(cont.)

- **A low-level language** consists of machine-oriented programming. (machine code is the only language with which the CPU understands)
  - For example. the Assembly language.

    The Assembly Language, in other words, has a major control of the computer hardware
  - It isn't a Programmer-Friendly (Syntax and Structure)
- **A high-level languages** are problem-oriented
  - Example, Pascal.
  - It is a Programmer-Friendly

Introduction(cont.)

- Every **high-level language** has its own source which needs to be translated into the CPU's language - machine code
- **An Interpreter** is more slower than a compiler, the fact that it has to be loaded into memory till the end of the translation of the program. It is less practical than the compiler. However, the interpreter can be helpful for correcting out errors in programs i.e. it is ideal for debugging
- **A compiler** is faster due to its single, complete translation of the program into object code. A compiler, compiles the source code (with no syntax errors) and makes a copy of it. This copy is called the 'object code'. After it has been compiled, it does not need a recompilation (unless the source code is changed and a recompilation is required to affect the change). The object code used by the OS in order to execute the compiled program.

Introduction(cont.)

- **C** is a programming language developed in the 1970's alongside the UNIX operating system.

- C provides a comprehensive set of features for handling a wide variety of applications, such as systems development and scientific computation.

- **C++** is an “extension” of the C language, in that most C programs are also C++ programs.

Basics of C++

- **Structure of a program**

```
#include directives
int main()
{


 return 0;
}
```

Basics of C++(cont.)

C++ compiler directives

- Compiler directives appear in blue in Visual C++.
- The #include directive tells the compiler to include some already existing C++ code in your program.
- The included file is then linked with the program.
- There are two forms of #include statements:

```
#include <iostream>   //for pre-defined files

#include "my_lib.h"   //for user-defined files
```

Basics of C++(cont.)

// A simple C++ program

```
#include <iostream>
using namespace std;
int main()
{
cout << "Programming is great fun!";
return 0;
}
```

Basics of C++(cont.)

- #include <iostream>
  - This line must be included in a C++ program in order to get input from the keyboard or print output to the screen
  - The iostream file contains code that allows a C++ program to display output on the screen and read input from the keyboard. Because this program uses cout to display screen output, the iostream file must be included. Its contents are included in the program at the point the #include statement appears. The iostream file is called a *header file* , so it should be included at the head, or top, of the program.

Basics of C++(cont.)

- using namespace std;
  - declares that the program will be accessing entities whose names are part of the namespace called std
  - The reason the program needs access to the std namespace is because every name created by the iostream file is part of that namespace
  - In order for a program to use the entities in iostream , it must have access to the std namespace

Basics of C++(cont.)

- int main()
  - This marks the beginning of a function.
  - A function can be thought of as a group of one or more programming statements that collectively has a name.
  - The name of this function is main, and the set of parentheses that follows the name indicates that it is a function.
  - The word int stands for “integer.” It indicates that the function sends an integer value back to the operating system when it is finished executing.
  - Although most C++ programs have more than one function, every C++ program must have a function called main

Basics of C++(cont.)

❖ Note:

- C++ is a case-sensitive language. That means it regards uppercase letters as being entirely different characters than their lowercase counterparts. In C++, the name of the function Main must be written in all lowercase letters. C++ doesn’t see “main” the same as “Main” or “MAIN.”

Basics of C++(cont.)

- {
  - the beginning of the function main
- cout << "Programming is great fun!";
  - this line displays a message on the screen.
- return 0;
  - This sends the integer value 0 back to the operating system upon the program's completion.
  - The value 0 usually indicates that a program executed successfully.
- The last line of the program contains the closing brace: }

Basics of C++(cont.)

- **Comments**
  - Comments appear in green in Visual C++.
  - Comments are explanatory notes; they are ignored by the compiler.
  - There are two ways to include comments in a program:

```
// A double slash marks the start of a //single line comment.

/* A slash followed by an asterisk marks the start of a multiple line
comment. It ends with an asterisk followed by a slash. */
```

Basics of C++(cont.)

C++ identifiers

- Identifiers appear in black in Visual C++.
  - An identifier is a name for a variable, constant, function, etc.
  - It consists of a letter followed by any sequence of letters, digits, and underscores.
  - Examples of valid identifiers: First_name, age, y2000, y2k
  - Examples of invalid identifiers: 2000y
  - Identifiers cannot have special characters in them. For example: X=Y, J-20, ~Ricky,*Michael are invalid identifiers.
  - Identifiers are case-sensitive. For example: Hello, hello are unique identifiers.

Basics of C++(cont.)

Variables

- Variables represent storage locations in the computer's memory.
- Every variable must have a declaration.
  - ✓ type variable-name;

Basics of C++(cont.)

Data Types

Type	Size	range
bool	1 byte	Values: true or false
char	1 byte	
short unsigned short	2 bytes	–32,768 to +32,767 0 to 4,294,967,295
int unsigned int	4 bytes	–2,147,483,648 to +2,147,483,647 0 to 4,294,967,295
float	4 bytes	±3.4E-38 and ±3.4E38
double	8 bytes	±1.7E-308 and ±1.7E308

Basics of C++(cont.)

- The sizeof operator may be used to determine the size of a data type on any system.
- **Example**

```
#include <iostream>
using namespace std;
int main()
 {
      cout << "Size of char : " << sizeof(char) <<
      endl;
      cout << "Size of int : " <<sizeof(int) << endl;
      return 0;
 }
```

- **Output**

Size of char : 1 Size of int : 4

Basics of C++(cont.)

- Variable Definition

```
int a, b;
int c;
float f;
```

- Variable initialization

```
a = 10;
b = 20;
c = a + b;
```

Basics of C++(cont.)

Variable Scope

- Local Variables:
  - ✓ Variables that are declared inside a function or block are local variables. They can be used only by statements that are inside that function or block of code. Local variables are not known to functions outside their own.
- Global Variables:
  - ✓ Global variables are defined outside of all the functions, usually on top of the program. The global variables will hold their value throughout the life-time of your program.
  - ✓ A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration.

Basics of C++(cont.)

- **Example**

```
#include <iostream>
using namespace std;
// Global variable declaration:
 int g;
int main ()
 {
        // Local variable declaration:
        int a, b;
       // actual initialization
        a = 10; b = 20; g = a + b;
        cout << g;
        return 0;
}
```

Basics of C++(cont.)

Constants

- A variable is called a “variable” because its value may be changed. A constant, on the other hand, is a data item whose value does not change during the program’s execution.

Defining Constants:

- There are two simple ways in C++ to define constants:
  - Using #define preprocessor.
  - Using const keyword.

Basics of C++(cont.)

- **The #define Preprocessor:**
  - the form to use #define preprocessor to define a constant:

    #define identifier value

- **Example**

```
#include <iostream>
 using namespace std;
#define length 10
#define width 5
#define newline '\n'
int main()
{     int area;
      area = length *width;
      cout << area;
      cout<<newline;
     return 0;
 }
```

Basics of C++(cont.)

The const Keyword:

- You can use const prefix to declare constants with a specific type as follows:

  const type variable = value;

Example

```
#include <iostream>
 using namespace std;
int main()
{   const int length = 10;
    const int width = 5;
    const char newline = '\n';
     int area;
     area = length *width;
     cout << area;
     cout<<newline;
    return 0;
}
```

Basics of C++(cont.)

- **C++ keywords**
  - Keywords appear in blue in Visual C++.
  - Each keyword has a predefined purpose in the language.
  - Do not use keywords as variable and constant names!!
- **Some Keywords**
  - bool, break, case, char, const, continue, do, default, double, else, extern, false, float, for, if, int, long, namespace, return, short, static, struct, switch, typedef, true, unsigned, void, while

Basics of C++(cont.)

- **Input statements**
  - cin >> variable-name;
  - Meaning: read the value of the variable called <variable-name> from the user

Example:

```
cin >> a;

cin >> b >> c;

cin >> x;

cin >> my-character;
```

Basics of C++(cont.)

- **Output statements**

  cout << variable-name;

  Meaning: print the value of variable <variable-name> to the user

  cout << “any message “;

  Meaning: print the message within quotes to the user

  cout << endl;

  Meaning: print a new line

Example:

```
cout << a;
cout << b << c;
cout << “This is my character: “ << my-character << “ he he he” << endl;
```

some Examples

Examples

- **Example 1:** adding 2 numbers

```
#include <iostream>
using namespace std;
int main()
{
    int first, second, sum;
     cout << “Enter the first number."<< endl;
     cin >> first;
     cout << “Enter the second number."<< endl;
     cin >> second;
     sum = first + second;
     cout << sum << endl;
         return 0;
}
```

Examples(cont.)

- **Example 2:** Write a program which accepts a character and display its ASCII value

```
#include<iostream>
 using namespace std;
int main()
{          char ch;
             cout<<"\nEnter any character : ";
             cin>>ch;
            cout<<"ASCII equivalent is : "<<(int)ch<<endl;
            return 0;
}
```

Thanks