Chapter 1

Introduction

Outlines

- ★ Introduction to Programming Language
- ★ Basic Terminologies of Programming Language
- ★ Overview of c++ Programming Language
- ★ Basic Syntax
- ★ Data Types and Variables

1. What is Programming Language

➔ A vocabulary and set of grammatical rules (syntax) for instructing a computer to perform specific tasks.

➔ Can be used to create computer programs.

➔ E.g. C++, Java, PHP etc.

➔ High vs Low programming level

- ◆ High level: Closer to human languages (English, French, etc.)
- ◆ Low Level: Machine-friendly

2. Basic Terminologies of Programming Language

➔ IDE

➔ Data and Data Type

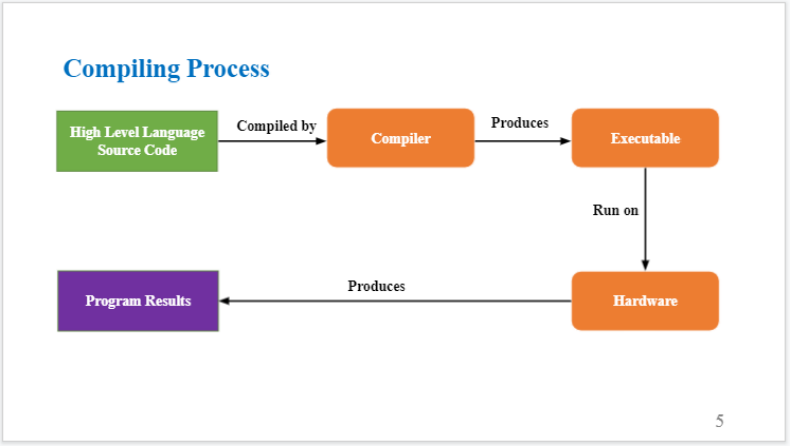
➔ Variable

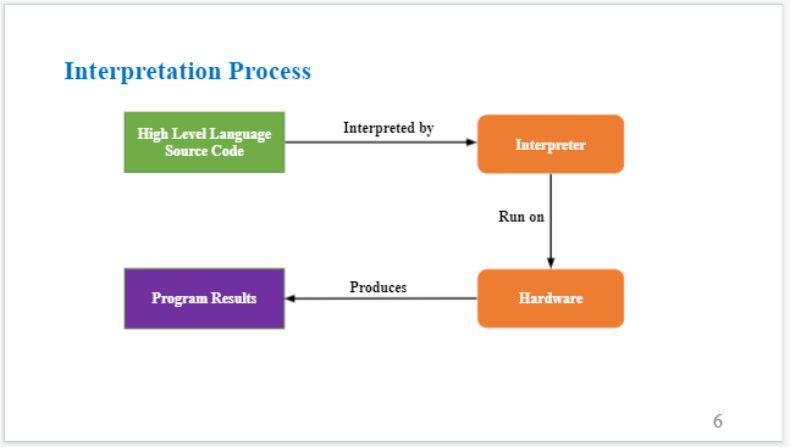
➔ Identifier

➔ What is Syntax

➔ What is Statement

➔ Keyword/Reserved Words





3. Overview of C++ Programming

➔ C++ a high-level programming language.

➔ C++ adds object-oriented features to its predecessor, C.

➔ The syntax is largely inherited from the C language.

➔ a combination of Procedural and Object-Oriented programming language, unlike other programming languages

Sample of C++ Program

```
#include <iostream>

int main()
{
  /* my first program in C++ */
   std::cout << "Hello world!" ;
   return 0;
}
```

```
Output

Hello World!
```

Let us take a look at the various parts of the above program −

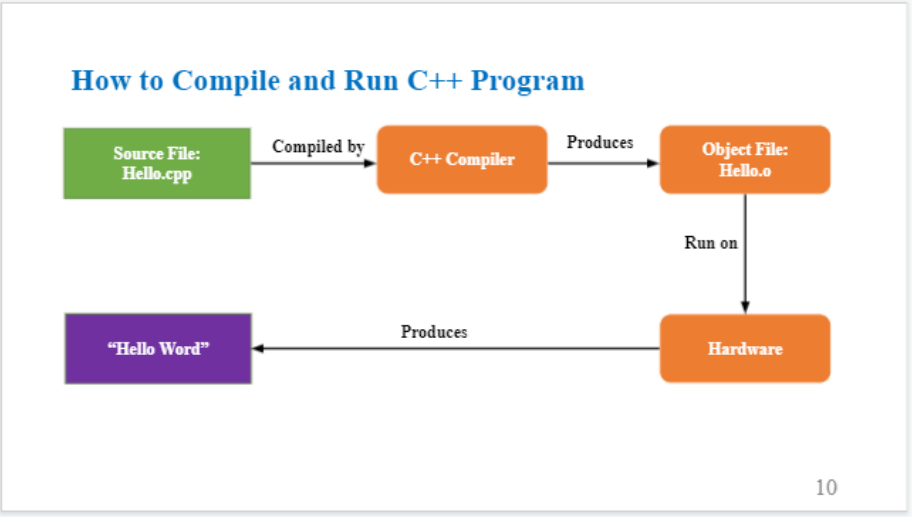
➔ The first line of the program **#include<iostream>** tells a C++ compiler to include iostream.h file before going to actual compilation.

➔ The next line **int main ()** is the main function where the program execution begins.

➔ The next line **/*...*/** will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.

➔ The next line **std::cout<< "Hello World!"** is another function available in C ++which causes the message "Hello, World!" to be displayed on the screen.

➔ The next line **return 0;** terminates the main () function and returns the value 0.



4. Basic Syntax

4.1 Tokens in C++

➔ A C program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol.

➔ For example, the following C statement consists of six tokens:

```
std::cout<< “Hello World!”;
```

➔ Individual Tokens are -

```
std
::
cout
<<
“Hello World!”
;
```

4.2 Semicolons

➔ In a C++ program, the semicolon is a statement terminator.

➔ That is, each individual statement must be ended with a semicolon.

➔ Given below are two different statements :

```
std:cout << “Hello World!”;
return 0;
```

4.3 Comments

➔ Comments are like helping text in your C++ program and they are ignored by the compiler.

➔ There are two types of comments in c++.

- ◆ Single line comment - starts with //
- ◆ Multi line comment - starts with /* and terminate with */

```
// my first program in C++
/*
   my first program
   in c++
*/
```

4.4 Identifier

➔ Identifier is a name given to any programming element like variables, constants, functions, statements.

➔ An identifier starts with a letter A to Z, a to z, or an underscore '_' followed by zero or more letters, underscores, and digits (0 to 9).

➔ No special character is allowed.

- ➔ Some special characters are ! , * , + , \ , “, <, # , (, =, |,{ , >,%,), ~, ; , }, /
- ➔ C++ is a case-sensitive programming language.
- ➔ Thus, name and Name are two different identifiers in C++.
- ➔ Valid identifiers : rno1, roll_number
- ➔ Invalid identifiers: 25Mark, Student number, #totalsalary

Good Programming Practice

- ➔ You can choose any name for an identifier (excluding keywords).
- ➔ However, if you give a meaningful name to an identifier, it will be easy to understand and work on for you and your fellow programmers.

4.5 Keywords

- ➔ Keywords have standard, predefined meanings in C++.
- ➔ These keywords can be used only for their intended purpose.
- ➔ They cannot be used as programmer-defined identifiers.
- ➔ Some of the keywords are : do, for, case, while, break ,float, static ,if, else etc.

4.6 Data Types and Variable

4.6.1 Variable

- ➔ A variable is an identifier that represents a value.
- ➔ The value represented by the identifier may be changed during the execution of the program.

4.6.2 Data and Data Types

- ➔ Data represents raw facts.
- ➔ Data type indicates the type of value represented or stored.
- ➔ Data types in C++ is mainly divided into three types:
    1. **Primitive or Primary or Basic data types** - These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char , float, bool etc.
    2. **Derived data types** - They are a combination of primitive data types. They are used to represent a collection of data. Example: array, pointer etc.
    3. **User defined data type** - These data types are defined by the user itself. Like defining a class in C++.

4.6.3 Size of Data Types

- ➔ Size of each data types (Machine Dependent)

  sizeof(int) = 4 bytes

  sizeof(char) = 1 byte

  sizeof(long) = 4 bytes

sizeof(short) = 2 bytes

sizeof(float) = 4 bytes

sizeof(double) = 8 bytes

```
// C++ program to sizes of data types
#include<iostream>
using namespace std;
int main(){
   cout<<"Size of int : "<<sizeof(int);
   cout<<"\nSize of short int : "<<sizeof(short int);
   cout<<"\nSize of long int : "<<sizeof(long int);
   cout<<"\nSize of float : "<<sizeof(float);
   cout<<"\nSize of double : "<<sizeof(double);
   cout<<"\nSize of char : "<<sizeof(char);
   cout<<"\nSize of string : "<<sizeof(string);
   cout<<"\nSize of boolean : "<<sizeof(bool);
}
```

Output:

```
Size of int : 4
Size of short : 2
Size of long : 4
Size of float : 4
Size of double : 8
Size of char : 1
Size of string : 24
Size of boolean : 1
```

4.6.4 Declaring and Initializing Variables

Declaring a Variable

- ➔ Syntax: data-type variable-list;
- ➔ Example:

  int i,j,k;

  float x,y,z;

  char ch;

Initializing a Variable

- ➔ You can initialize a variable when you declare it.
- ➔ Example: int total = 0;

Where Variables are Declared

- ➔ Variables can be declared in three places: inside functions, in the definition of function parameters, and outside of all functions.
- ➔ These positions correspond to local variables, formal parameters, and global variables, respectively.

```
#include <iostream>
using namespace std;
// global Variable declaration: (global variable)
int a = 10;
// inside function Var declaration:(local variable)
void display(){
int a=20;
cout<<"value of a inside fun : "<<a<<"\n";
}
```

```
// inside function parameter declaration:(formal parameter)
void displayAnother(int a){
   cout<<"value of a inside function parameter : "<<a<<"\n";
}
int main () {

   cout<<"value of a : "<<a<<"\n";
   display();
   displayAnother(30);
}
```

```
Output:

value of a : 10
value of a inside fun : 20
value of a inside function parameter : 30
```

4.6.5 Constant Variable

- ➔ A constant variable cannot be modified thereafter.
- ➔ The type of value stored in the constant must also be specified in the declaration.
- ➔ In C++, we are using **const** as a keyword to declare constant variables.

Syntax: const datatype variable = value;

For example, an integer and float constants can be declared as follows:

const int MAX_VAL= 100; const float PI=3.14;

```
#include <iostream>
using namespace std;
int main()
{
   const int LENGTH = 10;
   const int WIDTH = 5;
   const string NAME = "Rectangle";
   int area;
   area = LENGTH * WIDTH;
   cout<<"value of length : "<<LENGTH;
   cout<<"\nvalue of width : "<<WIDTH;
   cout<<"\nvalue of area : "<<area;
   cout<<"\nvalue of name : "<<NAME;

}
```

```
Outcome

value of length : 10
value of width : 5
value of area : 50
value of name : Rectangle
```
