University of Wah
Department of Computer Science,
Faculty of Computing

Lab-0 1	
Contents Covered:	**Introduction to Programming** • Different Programming Languages, Introduction to C++ Programming Environment • Role of compiler and linker **Overview of C++** • Basic data types and variables • **Input/output constructs**

➢ Computer Languages:

1-Machine Language
- Uses binary code
- Machine-dependent
- Not portable

2-Assembly Language
- Uses mnemonics
- Machine-dependent
- Not usually portable

3- High-Level Language (HLL)
- Uses English-like language
- Machine independent
- Portable (but must be compiled for different platforms)
- Examples: C, C++, Java
-

Overview of C++

➢ Introduction to C++

In simple terms, C++ is a sophisticated, efficient and a general-purpose programming language and is middle level programming language based on C.

- It was developed by Bjarne Stroustrup in 1979.
- Many of today's OS, system Drivers, browsers and games used C++.
- C++ gives programmers a high level of control over system resources and memory.
- Graphical User Interfaces and embedded systems.
- C++ is used in many applications like

- Adobe Products like Photo-shop, Illustrator, In Design
- Amazon - one of the biggest e-commerce sites
- Auto-desk products for Computer Aided Design
- Facebook - social networking site are heavy C++ centric products.

➢ **Introduction to DEV C++:**

- DEV C++ is developed by Bloodshed software.
- Dev-C++ is a free full-featured integrated development environment (IDE) which is free, portable, fast and simple in use.
- Distributed under the GNU(General Public License) for programming in C and C++.

➢ **Steps to download and install Dev C++**

1. Download Dev C++
   from: https://sourceforge.net/projects/orwelldevcpp/files/latest/download
2. This is a 9 MB file approx.
3. Double click the executable file
4. Start the installation by clicking Next button.
5. Choose the destination folder and install it.
6. Once the installation is complete, go to My Computer > Properties > Advanced System Settings > Advanced Tab.
7. Now click on "Environment variables" button > new.
8. Change the system variable name as: PATH.
9. Change the variable value as: C:\Dev-Cpp\bin;.
10. Click OK and start Dev C++ to write the program.

➢ **Video link to download and install Dev C++:**

Link: https://youtu.be/NTkwZsUasXU?si=de9fiD76RhUXqJhy

➢ **Why using C++**

C++ is a cross-platform language that can be used to create high-performance applications. It gives programmers a high level of control over system resources and memory.

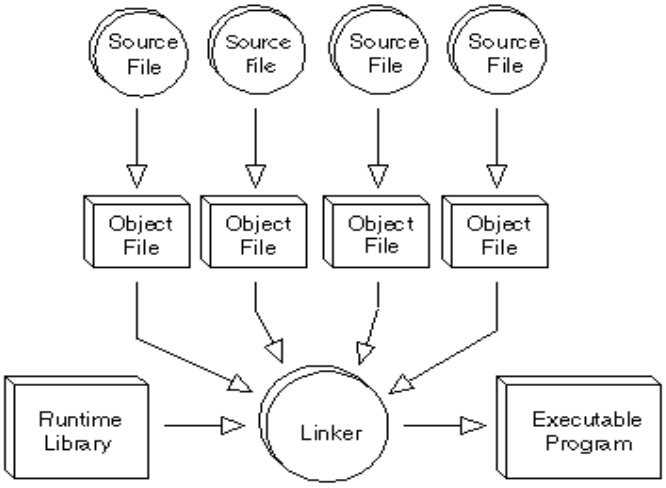
- C++ is one of the world's most popular programming languages.

- C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.
- C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
- C++ is portable and can be used to develop applications that can be adapted to multiple platforms.
- C++ is fun and easy to learn!
- As C++ is close to C# and Java, it makes it easy for programmers to switch to C++ or vice versa

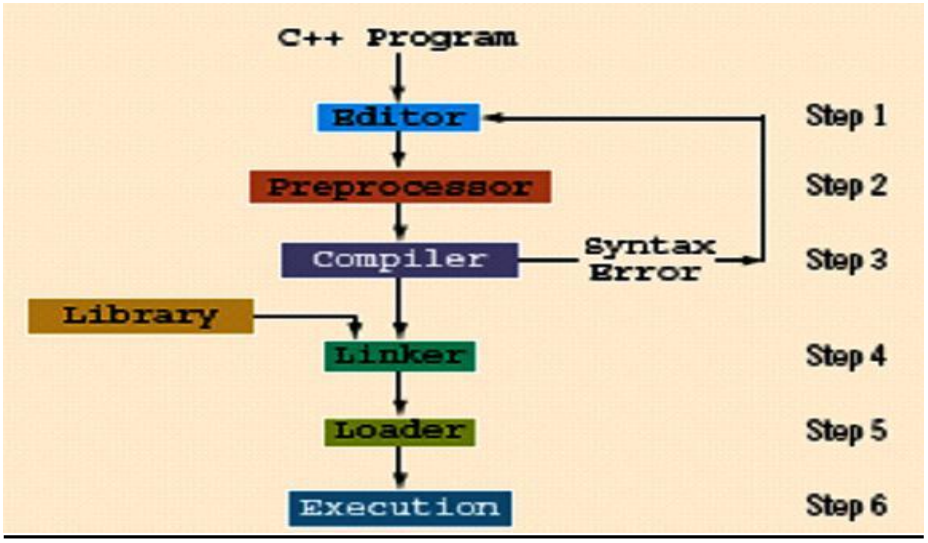
➢ **Processing a C++ Program**

- **Source program**
  - A program written in a high-level language.
- **Pre-processor directives**
  - In a C++ program, statements that begin with the symbol # are called pre-processor directives.
- **Object program**
  - The machine language version of the high-level language program.
- **Linker**
  - A program that combines the object program with other programs
- **Loader**
  - A program that loads an executable program into main memory.

University of Wah
Department of Computer Science,
Faculty of Computing



Processing a C++ Program

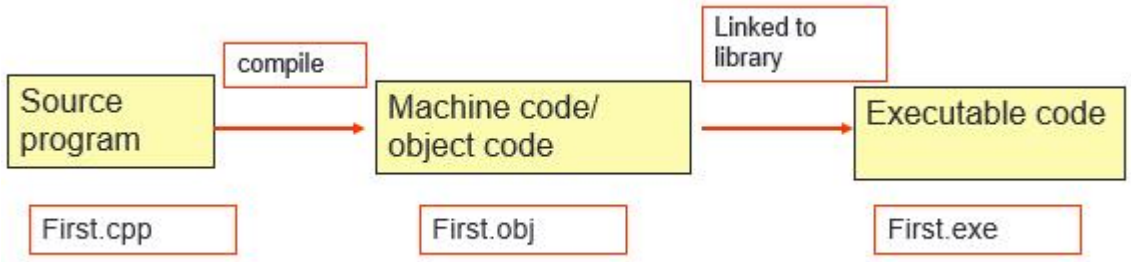


Writing C++ program

- C++ program is stored on disk with **.cpp** extension

Code is written in the editor which is then compiled

University of Wah
Department of Computer Science,
Faculty of Computing



C++ Program Structure

A C++ program consists of three parts:

- Preprocessor Directives
- Main() function
- C++ statements

Preprocessor directives in C++:

Preprocessor directives in C++ are not libraries; they are commands that are processed by the preprocessor before the actual compilation of the code begins. They are used to control the compilation process and include additional code or information in the source file.

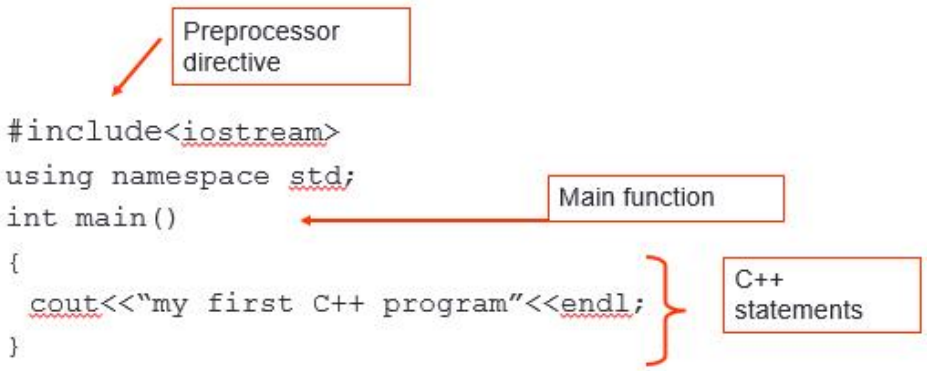
Key Points About Preprocessor Directives:

- **Purpose**: They instruct the compiler to perform specific tasks before the actual compilation starts, such as including header files, defining constants, or conditional compilation.
- **Syntax**: They begin with a # symbol. For example, **#include, #define, and #ifdef.**
- **Example**:
  - **#include <iostream>**: Includes the standard input-output stream library.

Differences Between Preprocessor Directives and Libraries:

- **Libraries**: These are collections of precompiled code that you can link against and use in your programs. They provide functions and classes for various tasks, such as input/output operations, mathematical computations, etc.
- **Preprocessor Directives**: These are instructions for the preprocessor to handle certain aspects of the code before it is compiled. They do not provide functionality directly but control how the code is processed.

University of Wah
Department of Computer Science,
Faculty of Computing



Getting started with C++ with DEV C++:

Important Header files in C++

Some library Files and where to use them:

<iostream>

Provides C++ input and output fundamentals.

<iomanip>

Provides facilities to manipulate output formatting, such as the base used when formatting integers and the precision of floating point values.

<math.h>

math.h header file supports all the mathematical related functions in C language.

<string.h>

Use for all string functions

Task 1:

Write a program to write "Hello World!"

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

int main()
{
 cout << "Hello World!";
 return 0;
}
```

Explanation:

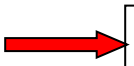
Line 1: #include <iostream> is a **header file library** that lets us work with input and output objects, such as cout (used in line 5). Header files add functionality to C++ programs.

Line 2: using namespace std means that we can use names for objects and variables from the standard library.

Line 3: A blank line. C++ ignores white space.

Line 4: Another thing that always appear in a C++ program, is int main(). This is called a **function**. Any code inside its curly brackets {} will be executed.

Line 5: cout (pronounced "see-out") is an **object** used to output/print text. In our example it will output "Hello World!".



Note: Every C++ statement ends with a semicolon ;

Note: The body of int main() could also been written as:
int main () { cout << " Hello World !"; return 0; }

Remember: The compiler ignores white spaces. However, multiple lines makes the code more readable.

Line 6: return 0 ends the main function.
Omitting Namespace

You might see some C++ programs that run without the standard namespace library. The using namespace std line can be omitted and replaced with the std keyword, followed by the :: operator for some objects:

```
#include <iostream>

int main()

{
 std::cout << " Hello World!";
 return 0;
}
```

Task 2:
Write a program to print your "name"," country", and "gender".

```
#include<iostream>
using namespace std;
int main()
{
cout<<"Hamza"<<endl;
cout<<"Pakistan"<<endl;
cout<<"Female"<<endl;

}
```

- **Data Types:**

  A variable provides us with named storage that our programs can manipulate. Each variable in C++ has a specific type, which determines the size and layout of the variable's memory, this type is known as data type of that variable.

  The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C++ is case-sensitive.

Type	Description
char	Typically a single octet (one byte). This is an integer type.
int	The most natural size of integer for the machine.
float	A single-precision floating point value.
double	A double-precision floating point value.
void	Represents the absence of type.

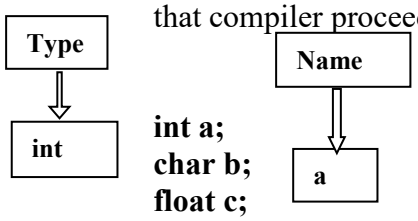
Figure 1: Basic Data Types

The table below shows the fundamental data types, their meaning, and their sizes (in bytes):

Data Type	Meaning	Size (in Bytes)
Int	Integer	2 or 4
float	Floating-point	4
double	Double Floating-point	8
char	Character	1
wchar_t	Wide Character	2
bool	Boolean	1
void	Empty	0

Variable Declaration in C++

Preprocessor directives in C++A variable declaration provides assurance to the compiler that there is one variable existing with the given type and name so that compiler proceed for further compilation.



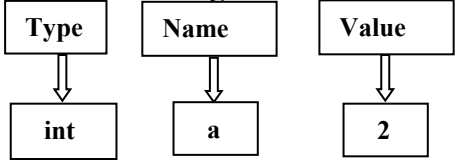
int a;
char b;
float c;

Preprocessor directives in C++

Variable Initialization in C++

A variable can be initialized at the time of declaration or even after that. Basically initialization means storing some actual meaningful data inside the variable.

int a=2;
char b='x';
float c=2.907;



Constants

Constants refer to fixed values that the program may not alter during its execution. These fixed values are also called literals.

Defining Constants

There are two simple ways in C++ to define constants:

Using **#define** preprocessor
Using **const** keyword
#define LENGTH 10
#define WIDTH 5
const int LENGTH = 10;
const int WIDTH = 5;

C++ Basic Input/Output

In C++, cout sends formatted output to standard output devices, such as the screen. We use the cout object along with the << operator for displaying output.
In C++, cin takes formatted input from standard input devices such as the keyboard. We use the cin object along with the >> operator for taking input.

Try the following codes by yourself:

Example 1:

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

int main() {
   int var1;                          // variable declaration
   cout << "Enter a number: ";              // output statement
   cin >> var1;                        // input statement
   cout << "You have entered: " << var1;       // output statement

   return 0;
}
```

Example 2:

```
// this program is about comments
#include<iostream> // allows program to output data to screen

// function main begins program execution
int main () {
   std::cout << "This is about comments"; // display message
   return 0; // indicates that program ended successfully
}

        // end of main function
```

Example 3:

```
/* this program is about multiple-line comments  its another type of comment
this program has been written by programmer X  and it prints some text on screen
*/
#include<iostream>

using namespace std;

int main () {

   cout << "This is about comments"; // display message

   return 0;

}
```

Escape Sequences

Sometimes, it is necessary to use characters that cannot be typed or has special meaning in C++ programming. For example, newline (enter), tab, question mark, etc. In order to use these characters, escape sequences are used.

- ## Escape Sequences Characters

\b	Backspace
\f	Form feed
\n	Newline
\r	Return

\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\'	Single quotation mark
\"	Double quotation mark
\?	Question mark
\0	Null Character

Try the following escape sequences:

```
\n
cout <<“welcome \n to \n C++”;              // new line
\t
cout <<“welcome \t to \t C++”;
\r
cout<<“welcome \r to \r C++”;
\a
cout<<“program end \a”;
\\
cout<<“Select D: \\ drive”;
\“
cout<<“welcome to \“ programming\“ ”;
\”
cout<<“welcome to \” programming \” ”;
```

Example 5:

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

int main() {
   int num1, num2, sum;

   cout << "Let's add two numbers, enter first number: \n";
   cin >> num1;

   cout << "Enter second number: \n";
   cin >> num2;
```

```
    cout << "Sum of \t" << num1 << " and \t" << num2 << " is: ";
    sum = num1 + num2;

    cout << sum;

    return 0;
}
```

Lab 1 TASKS:

Task 1:

Write a program to calculate area of square and circle.

Task 2:

Write a program to add two floating point numbers.

Task 3:

Write a program to take four marks from user and calculate average. Also show the result.

Task 4:

Write a program that displays value of an integer and character.