

OBJECT ORIENTED PROGRAMMING LAB



Lab Manual # 01

Introduction to C++

Instructor: Fariba Laiq

Semester Spring 2021

Course Code: CL217

**Fast National University of Computer and Emerging Sciences
Peshawar**

Department of Computer Science

Table of Contents

C++ Introduction	1
C++ History.....	1
Writing a C++ program.....	1
Structure of C++ program	1
1. Preprocessor directives.....	2
2. The main() function.....	3
3. C++ statements	3
C++ Program Syntax.....	3
Example explained	4
C++ Output.....	4
C++ Comments.....	5
Types of comments.....	6
1. Single line Comment	6
2. Multi-line Comment.....	6
C++ Keywords.....	7
Tokens.....	8
Variables	8
Rules for Naming/Writing Variables	8
C++ Data Types.....	9
C++ Input.....	12
Creating a Simple Calculator	13
Character Data Type	13
References	14

C++ Introduction

- ❖ C++ pronounced as “See Plus Plus”. It is a powerful computer programming language. It is the advance version of C language.
- ❖ The C is a procedural programming language while C++ is an Object-Oriented Programming language. Procedural means step by step process or step by step order.

C++ History

- ❖ In 1972 Dennis Ritchie developed C language at Bell Laboratory.
- ❖ In 1980 C++ language was developed by “Bjarne Stroustrup” .
- ❖ C++ was an extension of C.

Writing a C++ program

- ❖ Extension of C++ program is `cpp` i.e. `filename.cpp`
- ❖ Text editor/Dev C++ or any other IDE is required to write and edit C++ source code.
- ❖ C++ Compiler is required for compilation of source code into machine code. This machine code is called object code. It is stored in a new file with extension ***obj***. The object code is then linked to the libraries. After linking the object code to the libraries, an executable file with extension ***exe*** is created. The executable program is then run from operating system command line.
- ❖ For example, a source code of program in C++ is written and stored with `first.cpp`.
- ❖ After compilation, the object code is saved in file `first.obj` and executable code is stored in file `first.exe`.
- ❖ Before compilation code is called **source code**.
- ❖ After compilation code is called **object code**.

Structure of C++ program

- ❖ A C++ program consists of three main parts. These are:
 1. Preprocessor directives
 2. The `main()` function
 3. C++ statements

1. Preprocessor directives

- ❖ Provides instructions to the compiler before the beginning of the actual program. Also called compiler directives.
- ❖ The preprocessor directives consist of instructions for the compiler.
- ❖ The compiler adds special instructions or code from these directives into program at the time of compilation.
- ❖ The preprocessor directives normally start with a number sign (#) and the keyword “include” or “define”. For example preprocessor directives are used to include header files in the program.
- ❖ A program example is given below. The first statement of the program is a preprocessor directive. The preprocessor directive has been written to include the iostream.h header file.

```
#include <iostream>
int main()
{
    cout << "Hello C++ Programming";
    return 0;
}
```

Header File

- ❖ Header file is a C++ source file that contains definitions of library functions/objects.
- ❖ A header files are added into the program at the compilation of the program.
- ❖ A header file is added if the function/object defined in it is to be used the program.
- ❖ The preprocessor directive “**include**” is used to add a header file into the program. The name of the file is written in single brackets (<>) after “**include**” directive. The C++ has a large number of header files in which library functions are defined. The header file must be included in the program before calling its function in the program. A single header file may contain a large number of built-in library functions.
- ❖ For example the header file iostream.h has definitions of different built in input and output objects and functions. It is included in the above program because its object “cout” is used in the program.

Syntax is given by

#include<name of the header file>

Eg: #include<iostream.h> input and output stream

#include<conio.h> console input and output (console means screen)

For example `cout<<` is defined in `iostream.h` and `getch()` is defined in `conio.h`.

2. The `main()` function

- ❖ The `main()` function indicates the beginning of C++ program.
- ❖ When we run a program first of all OS runs main function.
- ❖ Main function is built in function.
- ❖ It is automatically called.
- ❖ Without it programs is not executed.

Syntax

```
int main()
{
    program statements...
}
```

3. C++ statements

The statements of the program are written under the `main()` function between the curly braces {}.

These statements are the body of the program. Each statement in C++ ends with semicolon (;).

C++ is case sensitive language. The C++ statements are normally written in lowercase letters but in some exceptional cases, these can also be written in upper case.

C++ Program Syntax

1. `#include <iostream>`
2. `using namespace std;`
- 3.
4. `int main() {`
5. `cout << "Hello World!";`
6. `return 0;`
7. `}`

Example explained

- ❖ **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.
- ❖ Don't worry if you don't understand how `#include <iostream>` and `using namespace std` works. Just think of it as something that (almost) always appears in your program.
- ❖ **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 together with the *insertion operator* (`<<`) 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.
- ❖ **Line 7:** Do not forget to add the closing curly bracket `}` to actually end the main function.

C++ Output

The `cout` object, together with the `<<` operator, is used to output values/print text:

1. `#include <iostream>`
2. `using namespace std;`
3. `int main() {`
4. `cout << "Hello World!";`
5. `return 0;`
6. `}`

- ❖ You can add as many `cout` objects as you want. However, note that it does not insert a new line at the end of the output:

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

```
int main() {  
    cout << "Hello World!";  
    cout << "I am learning C++";  
    return 0;  
}
```

C++ New Lines

To insert a new line, you can use the `\n` character:

```
#include <iostream>  
using namespace std;  
  
int main() {  
    cout << "Hello World! \n";  
    cout << "I am learning C++";  
    return 0;  
}
```

Tip: Two `\n\n` characters after each other will create a blank line:

```
#include <iostream>  
using namespace std;  
int main() {  
    cout << "Hello World! \n\n";  
    cout << "I am learning C++";  
    return 0;  
}
```

Another way to insert a new line, is with the `endl` manipulator:

```
#include <iostream>  
using namespace std;  
  
int main() {  
    cout << "Hello World!" << endl;  
    cout << "I am learning C++";  
    return 0;  
}
```

C++ Comments

A well-documented program is a good practice as a programmer. It makes a program more readable and error finding become easier. One important part of good documentation is Comments.

- In computer programming, a comment is a programmer-readable explanation or annotation in the source code of a computer program
- Comments are statements that are not executed by the compiler and interpreter.

Types of comments

1. Single line comment
2. Multi-line comment

1. Single line Comment

Represented as `//` double forward slash. It is used to denote single line comment. It applies comment to a single line only. It is referred as C++-style comments as it is originally part of C++ programming.

Example

```
#include<iostream>
int main()
{
    // Single line Welcome user comment
    cout<<"Welcome to PFlab";
    return 0;
}

/*
Output
Welcome to PFlab
*/
```

2. Multi-line Comment

Represented as `/* any text */`. Start with forward slash and asterisk (`/*`) and end with asterisk and forward slash (`*/`). It is used to denote multi-line comment. It can apply comment to more than a single line. It is referred as C-Style comment as it was introduced in C programming.

```
#include<iostream>
int main()
{
    /* Multi-line Welcome user comment
    written to demonstrate comments
    in C/C++ */
    cout<<"Welcome to PF lab";
    return 0;
}

/*
Output
```



```
Welcome to PFlab
*/
```

Single or multi-line comments?

It is up to you which you want to use. Normally, we use `//` for short comments, and `/* */` for longer.

C++ Keywords

The words that are used by the language for special purpose are called keywords. These are also called reserved words. For example, in a C++ the word `main` is used to indicate the starting of program, `include` is used to add header files, `int` is used to declare integer type variable. All these words are keywords of C++. The reserved words cannot be used as variable names in a program.

A list of 32 Keywords in C++ Language which are also available in C language are given below.

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

asm	dynamic_cast	namespace	reinterpret_cast	bool
-----	--------------	-----------	------------------	------

explicit	new	static_cast	false	catch
operator	template	friend	private	class
this	inline	public	throw	const_cast
delete	mutable	protected	true	try
typeid	typename	using	virtual	wchar_t

Tokens

A program statement consists of variable names, keywords, constants, punctuations marks, operators etc. In C++ these elements of a statement are called tokens. In the following program segment:

```
main()
{
    int a, b;
}
```

main, {, }, int, a, b and punctuation marks (,) and (;) are tokens of the program.

Variables

- ❖ A named memory location where data is stored is called variable.
- ❖ A quantity whose value may change during execution of the program is called variable. It is represented by a symbol or name.
- ❖ **Variable** is name of *reserved area allocated in memory*. In other words, it is a *name of memory location*.
- ❖ It is a combination of "vary + able" that means its value can be changed.
- ❖ `int data=10; // Here data is variable`

Rules for Naming/Writing Variables

1. The first character of variable must be an alphabetic character.
2. Underscore can be used as first character of variable name.
3. Blank spaces are not allowed in a variable name.
4. Special character such as arithmetic operators, #, ^, cannot be used in a variable name.
5. The maximum length of a variable name depends upon the compiler of C++.
6. A variable name declared for one data type cannot be used to declare another data type.

C++ is a case sensitive language. Thus, variable name with the same spellings but different cases are treated as different variable names. For example, variable "Pay" And "pay" are two different variables.

❖ C++ is a case sensitive language: number is different from Number or nUmber

Valid identifiers:

❖ Int abc, char aBc, int first_var, float first

Invalid identifiers:

Int 3bc, int a*b, int #a, int void

C++ Data Types

Data types specify the different sizes and values that can be stored in the variable.

Data Types tell us:

- i) Type of data.
- ii) Memory reserved (amount of memory).
- iii) Range of value.

Keyword	Range		Bytes of Memory
	Low	High	
char	-128	127	1
short	-32,768	32768	2
int	-2,147,483,648	2,147,483,647	4
long	-2,147,483,648	2,147,483,647	4
float	3.4×10^{-38}	3.4×10^{38}	4
double	1.7×10^{-308}	1.7×10^{308}	8
long double	3.4×10^{-4932}	3.4×10^{4932}	10

sizeof() Data Type

sizeof() function is used to find size (bytes) of data type

e.g: sizeof(char)

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

    cout << "Size of short int : " << sizeof(short int) << " bytes"
    << endl;
    cout << "Size of long int : " << sizeof(long int) << " bytes" << endl;
    cout << "Size of signed long int : " << sizeof(signed long int)
    << " bytes" << endl;
    cout << "Size of unsigned long int : " << sizeof(unsigned long int)
    << " bytes" << endl;

    cout << "Size of float : " << sizeof(float) << " bytes" << endl;
    cout << "Size of double : " << sizeof(double) << " bytes" << endl;
    return 0;
}

/*
```

Output

Size of char : 1 byte

Size of int : 4 bytes

Size of short int : 2 bytes

Size of long int : 8 bytes

Size of signed long int : 8 bytes

Size of unsigned long int : 8 bytes

Size of float : 4 bytes

Size of double : 8 bytes

*/

C++ Constant

When you do not want others (or yourself) to override existing variable values, use the const keyword (this will declare the variable as "constant", which means unchangeable and read-only)

Example

```
const int myNum = 15; // myNum will always be 15
myNum = 10; // error: assignment of read-only variable 'myNum'
```

You should always declare the variable as constant when you have values that are unlikely to change:

```
#include <iostream>
using namespace std;
int main() {
    const int minutesPerHour = 60;
    const float PI = 3.14;
    cout << minutesPerHour << "\n";
    cout << PI;
    return 0;
}
```

C++ Input

- ❖ You have already learned that cout is used to output (print) values. Now we will use cin to get user input.
- ❖ cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).
- ❖ In the following example, the user can input a number, which is stored in the variable x. Then we print the value of x:

```
#include <iostream>
using namespace std;
int main() {
    int x;
    cout << "Type a number: "; // Type a number and press enter
    cin >> x; // Get user input from the keyboard
    cout << "Your number is: " << x;
    return 0;
}
```

Good To Know

- ❖ cout is pronounced "see-out". Used for output, and uses the insertion operator (<<)
- ❖ cin is pronounced "see-in". Used for input, and uses the extraction operator (>>)

Creating a Simple Calculator

```
#include <iostream>
using namespace std;
int main() {
    int x, y;
    int sum;
    cout << "Type a number: ";
    cin >> x;
    cout << "Type another number: ";
    cin >> y;
    sum = x + y;
    cout << "Sum is: " << sum;
    return 0;
}
```

/*

Output

```
Type a number:2
Type another number: 4
Sum is: 6
*/
```

Character Data Type

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

int main() {
    char x;
    x='a';
    cout<<"The character is :"<<x<<endl;
}
```

Output:

```
The character is :a
```

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

```
int main() {
    char x;
    cout<<"Enter any character: ";
    cin>>x;
    cout<<"The character you entered is :"<<x<<endl;
}
```

Output:

```
Enter any character: A
The character you entered is :A
```

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

```
int main() {
    char x;
    cout<<"Enter any character: ";
    cin>>x;
    cout<<"ASCII Value is :"<<int(x)<<endl;
}
```

Output:

```
Enter any character: F
ASCII Value is :70
```

References

<https://beginnersbook.com/2017/08/cpp-data-types/>

http://www.cplusplus.com/doc/tutorial/basic_io/

<https://www.w3schools.com/cpp/default.asp>

<https://www.javatpoint.com/cpp-tutorial>

<https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/?ref=lbp>

<https://www.programiz.com/>

<https://ecomputernotes.com/cpp/>