



Lesson 1 - Intro to C++

CSC/ITS – 101 (PROGRAMMING 1)



What is C++?

C++ is a middle-level programming language developed by Bjarne Stroustrup starting in 1979 at Bell Labs. C++ runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. This C++ lessons adopts a simple and practical approach to describe the concepts of C++ for beginners to advanced software engineers.

Why should we learn C++?

C++ is a MUST for students and working professionals to become a great Software Engineer. I will list down some of the key advantages of learning C++:

- C++ is very close to hardware, so you get a chance to work at a low level which gives you lot of control in terms of memory management, better performance and finally a robust software development.



Why should we learn C++?

- C++ programming gives you a clear understanding about Object Oriented Programming. You will understand low level implementation of polymorphism when you will implement virtual tables and virtual table pointers, or dynamic type identification.
- C++ is one of the evergreen programming languages and loved by millions of software developers. If you are a great C++ programmer then you will never sit without work and more importantly you will get highly paid for your work.




Why should we learn C++?

- C++ is the most widely used programming languages in application and system programming. So you can choose your area of interest of software development.
- C++ really teaches you the difference between compiler, linker and loader, different data types, storage classes, variable types their scopes etc.



Why should we learn C++?

There are 1000s of good reasons to learn C++ Programming. But one thing for sure, to learn any programming language, not only C++, you just need to code, and code and finally code until you become expert.



Hello World using C++

This is the part where you become a new C++ programmer (for newbies).

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

// main() is where program execution begins.
int main() {
    cout << "Hello World"; // prints Hello World
    return 0;
}
```

Applications of C++ Programming

As mentioned before, C++ is one of the most widely used programming languages. It has its presence in almost every area of software development. I'm going to list few of them here:

- Application Software Development - C++ programming has been used in developing almost all the major Operating Systems like Windows, Mac OSX and Linux. Apart from the operating systems, the core part of many browsers like Mozilla Firefox and Chrome have been written using C++. C++ also has been used in developing the most popular database system called MySQL.



Applications of C++ Programming

- Games Development - C++ is extremely fast which allows programmers to do procedural programming for CPU intensive functions and provides greater control over hardware, because of which it has been widely used in development of gaming engines.
- Embedded System - C++ is being heavily used in developing Medical and Engineering Applications like softwares for MRI machines, high-end CAD/CAM systems etc.



Applications of C++ Programming

- Programming Languages Development - C++ has been used extensively in developing new programming languages like C#, Java, JavaScript, Perl, UNIX's C Shell, PHP and Python, and Verilog etc.
- Computation Programming - C++ is the best friends of scientists because of fast speed and computational efficiencies.



Lesson 2 - Basic C++ Syntax

CSC/ITS – 101 (PROGRAMMING 1)

Basic Syntax

When we consider a C++ program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what a class, object, methods, and instant variables mean.

- Object – Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating. An object is an instance of a class.
- Class – A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.
- Methods – A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- Instance Variables – Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

C++ Program Structure

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

// main() is where program execution begins.
int main() {
    cout << "Hello World"; // prints Hello World
    return 0;
}
```

C++ Program Structure

Looking at the various parts of the previous slide –

- The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header `<iostream>` is needed.
- The line using namespace `std`; tells the compiler to use the `std` namespace. Namespaces are a relatively recent addition to C++.
- The next line `// main()` is where program execution begins.' is a single-line comment available in C++. Single-line comments begin with `//` and stop at the end of the line.



C++ Program Structure

Looking at the various parts of the previous slide –

- The line `int main()` is the main function where program execution begins.
- The next line `cout << "Hello World";` causes the message "Hello World" to be displayed on the screen.
- The next line `return 0;` terminates `main()` function and causes it to return the value 0 to the calling process.

Semicolons and Blocks in C++

In C++, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

For example, following are three different statements –

```
x = y;  
y = y + 1;  
add(x, y);
```


Semicolons and Blocks in C++

A block is a set of logically connected statements that are surrounded by opening and closing braces. For example –

```
{  
    cout << "Hello World"; // prints Hello World  
    return 0;  
}
```

Semicolons and Blocks in C++

C++ does not recognize the end of the line as a terminator. For this reason, it does not matter where you put a statement in a line. For example –

```
x = y;  
y = y + 1;  
add(x, y);
```

is the same as

```
x = y; y = y + 1; add(x, y);
```



C++ Identifiers

A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

C++ does not allow punctuation characters such as @, \$, and % within identifiers. C++ is a case-sensitive programming language. Thus, Manpower and manpower are two different identifiers in C++.



C++ Identifiers

A C++ identifier is a name used to identify a variable, function, class, module, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

C++ does not allow punctuation characters such as @, \$, and % within identifiers. C++ is a case-sensitive programming language. Thus, Manpower and manpower are two different identifiers in C++.

C++ Identifiers

Here are some examples of acceptable identifiers –

```
mohd      zara      abc      move_name  a_123
myname50  _temp     j       a23b9     retVal
```

C++ Identifiers

Here are some examples of acceptable identifiers –

```
mohd      zara      abc      move_name  a_123
myname50  _temp     j       a23b9     retVal
```

C++ Keywords

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	



Whitespace in C++

A line containing only whitespace, possibly with a comment, is known as a blank line, and C++ compiler totally ignores it.

Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as `int`, ends and the next element begins.

Whitespace in C++

Statement 1

```
int age;
```

In the above statement there must be at least one whitespace character (usually a space) between `int` and `age` for the compiler to be able to distinguish them.

Statement 2

```
fruit = apples + oranges;    // Get the total fruit
```

In the above statement 2, no whitespace characters are necessary between `fruit` and `=`, or between `=` and `apples`, although you are free to include some if you wish for readability purpose.

Comments in C++

A comment can also start with `//`, extending to the end of the line. For example –

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

main() {
    cout << "Hello World"; // prints Hello World

    return 0;
}
```


Comments in C++

Within a `/*` and `*/` comment, `//` characters have no special meaning. Within a `//` comment, `/*` and `*/` have no special meaning. Thus, you can "nest" one kind of comment within the other kind. For example –

```
/* Comment out printing of Hello World:
```

```
cout << "Hello World"; // prints Hello World
```

```
*/
```



Lesson 3 - C++ Data Types

CSC/ITS – 101 (PROGRAMMING 1)



C++ Data Types

While writing program in any language, you need to use various variables to store various information. Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

You may like to store information of various data types like character, wide character, integer, floating point, double floating point, boolean etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

Primitive Built-in Types

C++ offers the programmer a rich assortment of built-in as well as user defined data types. Following table lists down seven basic C++ data types –

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t



Primitive Built-in Types


Several of the basic types can be modified using one or more of these type modifiers –

- signed
- unsigned
- short
- long



Primitive Built-in Types

The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value which can be stored in such type of variables.



Primitive Built-in Types

The following table shows the variable type, how much memory it takes to store the value in memory, and what is maximum and minimum value which can be stored in such type of variables.

Type	Typical Bit Width	Typical Range
char	1byte (1111 1101)	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-9223372036854775808 to 9223372036854775807
signed long int	8bytes	same as long int
unsigned long int	8bytes	0 to 18446744073709551615
long long int	8bytes	$-(2^{63})$ to $(2^{63})-1$
unsigned long long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	
double	8bytes	
long double	12bytes	
wchar_t	2 or 4 bytes	1 wide character



Primitive Built-in Types

The size of variables might be different from those shown in the above table, depending on the compiler and the computer you are using.

Following is the example, which will produce correct size of various data types on your computer.

Primitive Built-in Types

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

int main() {
    cout << "Size of char : " << sizeof(char) << endl;
    cout << "Size of int : " << sizeof(int) << endl;
    cout << "Size of short int : " << sizeof(short int) << endl;
    cout << "Size of long int : " << sizeof(long int) << endl;
    cout << "Size of float : " << sizeof(float) << endl;
    cout << "Size of double : " << sizeof(double) << endl;
    cout << "Size of wchar_t : " << sizeof(wchar_t) << endl;

    return 0;
}
```



QUESTIONS?