**CPP**

**Day-1**

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++ tutorial adopts a simple and practical approach to describe the concepts of C++ for beginners to advanded software engineers.

**Advantages of C:**

##### 1.1. Building block for many other programming languages

##### C is considered to be the most fundamental language that needs to be studied if you are beginning with any programming language. Many programming languages such as Python, C++, Java, etc are built with the base of the C language.

##### 1.2. Powerful and efficient language

##### C is a robust language as it contains many data types and operators to give you a vast platform to perform all kinds of operations.

##### 1.3. Portable language

##### C is very flexible, or we can say machine independent that helps you to run your code on any machine without making any change or just a few changes in the code.

##### 1.4. Built-in functions

##### There are only 32 keywords in ANSI C, having many built-in functions. These functions are helpful when building a program in C.

##### 1.5. Quality to extend itself

##### Another crucial ability of C is to extend itself. We have already studied that the C language has its own set of functions in the C library. So, it becomes easy to use these functions. We can add our own functions to the C Standard Library and make code simpler.

##### 1.6. Structured programming language

##### C is structure-based. It means that the issues or complex problems are divided into smaller blocks or functions. This modular structure helps in easier and simpler testing and maintenance.

##### 1.7. Middle-level language

##### C is a middle-level programming language that means it supports high-level programming as well as low-level programming. It supports the use of kernels and drivers in low-level programming and also supports system software applications in the high-level programming language.

##### 1.8. Implementation of algorithms and data structures

##### The use of algorithms and data structures in C has made program computations very fast and smooth. Thus, the C language can be used in complex calculations and operations such as MATLAB.

##### 1.9. Procedural programming language

##### C follows a proper procedure for its functions and subroutines. As it uses procedural programming, it becomes easier for C to identify code structure and to solve any problem in a specific series of code. In procedural programming C variables and functions are declared before use.

##### 1.10. Dynamic memory allocation

##### C provides dynamic memory allocation that means you are free to allocate memory at run time. For example, if you don’t know how much memory is required by objects in your program, you can still run a program in C and assign the memory at the same time.

##### 1.11. System-programming

##### C follows a system based programming system. It means the programming is done for the hardware devices.

##### Disadvantages of C Programming language:

##### Concept of OOPs

##### C is a very vast language, but it does not support the concept of OOPs (Inheritance, Polymorphism, Encapsulation, Abstraction, Data Hiding). C simply follows the procedural programming approach.

##### 2. Run-time checking

##### In the C programming language, the errors or the bugs aren’t detected after each line of code. Instead, the compiler shows all the errors after writing the program. It makes the checking of code very complex in large programs.

##### 3. Concept of namespace

##### C does not implement the concept of namespaces. A namespace is structured as a chain of commands to allow the reuse of names in different contexts. Without namespaces, we cannot declare two variables of the same name.

##### But, C programming lacks in this feature, and hence you cannot define a variable with the same name in C.

##### 4. Lack of Exception Handling

##### Exception Handling is one of the most important features of programming languages. While compiling the code, various anomalies and bugs can occur. Exception Handling allows you to catch the error and take appropriate responses. However, C does not exhibit this important feature.

##### 5. Constructor or destructor

##### C does not have any constructor or destructor.

##### Constructors & Destructors support basic functionality of Object Oriented Programming. Both are member functions that are created as soon as an object of the class is created. You will be studying constructor and destructor in detail later on.

##### 6. Low level of abstraction

##### C is a small and core machine language that has minimum data hiding and exclusive visibility that affects the security of this language.

**Advantages of 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.

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 every green 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.

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.

**Advantages and Disadvantages of C++:**

C++ is a multi-purpose programming language used widely across the world. There is no doubt that even after being a very old language, it is one of the most efficient programming languages.

Advantages of C++

1. Portability

C++ provides this feature of portability allowing us to develop codes without caring about the hardware. This lets us move the development of a program from one platform to another.

For example, you’re working on Windows OS and for some reason, you have to switch to LINUX, the codes from Windows OS will also run in the LINUX OS without any error.

2. Mid-level programming language

Being a mid-level programming language, we can treat it as both a low-level and high-level language. Features of high-level language help to develop games and desktop applications, whereas features of low-level language help make kernels and drivers.

3. Object-Oriented

The OOP concepts like polymorphism, encapsulation, inheritance, and abstraction give C++ the biggest advantage over other programming languages. It proved to be of great significance since this feature was not in C, this helped users to treat data as objects and classes.

4. Multi-paradigm programming language

Paradigm refers to the planning involved in programming. It concerns the logic, the style, and the way how we proceed with the program. C++ is a multi-paradigm programming language as it follows three paradigms:

a. Generic – Using a single idea that serves multiple purposes.

b. Imperative – Using steps that change the state of the program.

c. Object-Oriented – Using methods and classes for reusability and modularity.

5. Memory Management

C++ supports DMA (Dynamic Memory Allocation), which helps to free and allocate memory. Since there is no garbage collection, C++ gives the programmer total control over memory management.

6. Fast and Powerful

As C++ is a compiler-based programming language; we do not require to install a special runtime while running the program. Hence, they are pre-interpreted and it makes the code faster and more powerful.

Even the compilation and execution are faster allowing it to create several kinds of programs from games to drivers to complicated GUIs.

7. Similar to other languages

C++ syntax is similar to C#, C, and Java. It makes learning C++ easier if you already know one of them. It also makes switching to and from other languages easier.

This can be treated as an added benefit that C++ is compatible with C programs i.e. every running C program can be run as a C++ program. Most of the time we just need to run the program on a file .cpp extension.

8. Standard Library

C++ provides a good range of built-in libraries. They help in making the software development faster and allows the user to do more with less.

9. Wide Range of Applications

C++ is useful to make GUIs as well as games. C++ is also useful to develop graphics and real-time algebraic simulation. Hence, C++ is beneficial in every stream.

10. Huge Community

C++ has a vast community around it. Community size is very important if you want to get supported every now and then. The larger the community size, more the help you’ll get to solve your problem

A huge number of paid/free online courses and lectures are available, which shows how community support works.

11. Scalability

One of the greatest advantages of C++ is its scalability, i.e. its program can be scaled to another level. Hence, resource-intensive applications can be built using C++, as the programs can be low-scale and high-scale.

12. Big Job Market

As we know that C++ has benefits in various departments from finance to app developments, GUI to Games, C++ has a very big job market. Knowledge of C++ can help you secure a job at such departments where C++ comes in handy.

**Disadvantages of C++**

1. Pointers

When it comes to pointers in C++, it is a very tough conception compared to other topics. Uninitialized pointers might result in system failure.

Memory corruption can also take place if one puts wrong values in the same. To sum up, debugging pointer bugs is very difficult and hence one of the major disadvantages of C++.

2. No garbage collection

C++ doesn’t support garbage collectors, this means that the entire power to manage the data memory goes in the hands of the user. Absence of the same results in redundant data being stored in turn increasing the memory.

3. Unsafe

C++ is unsafe in a strong sense. The presence of pointers, global variables, etc. is the main reason behind these security issues. It means it is possible to corrupt the entire program just by using a part of the memory as an incorrect type.

4. Complex

C++ is a Multi-Paradigm language, i.e. object-oriented programming with runtime polymorphism, templates, and static polymorphism, some support for functional programming. C++ is not useful for platform-dependent apps and hence is complex in a very huge high-level program.

5. Less flexible

C++ is very strict regarding the syntax, a little mishap gives a series of errors. It generally takes more time to excel in C++ than any other programming language. It is not easy to write a C++ code in a readable way, making the language less user-friendly and less flexible for the user.

6. No custom operators

In many programming languages like Java, we can define operators for specific operations. But, in C++ it is not quite possible. We can redefine existing operators using operator overloading, but nothing more.

7. No built-in threads

There’s no support for built-in threads in C++. Even though it is a relatively newer concept it was later added to the newest standard of C++. But it is still far-fetched compared to programming languages like Java.

8. Lack of algebraic data types

Algebraic data types like tuples and structs are not supported in C++. Because of this, we need to use libraries or our own executions if we need to use it.

9. Functions are not first-class type

First-class type functions are those where:

a. Values can be passed and return, without restrictions.

b. Functions can be created and constructed anywhere, without any restrictions.

c. The function can be typed in such a way that an entity can be assigned to it.

While points a and c hold true for functions in C++, point b fails.

Why choose C++ over C?

To date, C is a dominant language and doesn’t seem to be retiring anytime soon. But, given the advances in microcontroller hardware and optimization, it might be time that developers and students start considering C++ for their perusal.

Listed below are some of the reasons:

1. Modern OOP Techniques: C language invented 50 years ago, lacks everything that newer programming language has, such as OOP concepts like Encapsulation, Inheritance, and Polymorphism.

In C++, we can use these techniques and make the code much more efficient and increase its readability as well as re-usability.

**2. Microcontrollers**: Compiler and toolchain support for microcontrollers has been frequently updated by the tool providers. Beyond this, microcontrollers have begun to add hooks to the tools, such that it is easier to develop C++ applications.

**3. Exception Handling:** In C, error/exception handling has been lacking all the while. Whereas, in C++ exception handling makes the language even superior.

**4. Special Methods:**The concept of constructors (and their different types) and destructors for objects makes C++ worth switching.

**5. Better Macro:** In C, we use Macros and in C++ we use Inline Functions, which makes the entire function body act like a Macro.

**6. Advanced Features:** In C++ there is no restriction for variable declaration i.e. we can declare them anywhere as long as the declaration is before the actual use. Type Checking is stronger in C++.

Applications of C++ Programming

As mentioned before, C++ is one of the most widely used programming languages. It has it's 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.

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.

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.

**Comments in C++**

There are 2 types of comments :

1. Single-line comments (//)

A single-line comment starts with two front slashes and continues to the end of the line.

2. Multi-line comments (/\* ….. \*/)

Multi-line comments or block comments are used when the explanation isn’t short and writing it on one continued line makes it unreadable. They begin with “ /\* ” and end with “ \*/ “.

Just to be clear, comments are not a part of the program and the compiler ignores them, so even if the comments have errors they won’t be detected.

Escape Sequences in C++

Escape sequences are non-printing characters used to control the printing behavior of the display stream objects, hence are also known as control characters. These characters are not printed in the output. They are prefixed with a backslash and a character to control the printing behavior. They can be inserted in any position of the string.

Some of the escape sequences along with their purpose are mentioned below:

\a – Bell

\n – New line

\r – Carriage return

\b – Backspace

\f – Formfeed

\t – Horizontal tab

\“ – Quotation mark

\v – Vertical tab

\’ – Apostrophe

\\ – Backslash

\? – Question mark

\0 – Null

Ex: first.cpp

#include<iostream>

//using namespace std;

int main()

{

    std::cout<<"Baba!Please bless me and my family";

    return 0;

}

The line using namespace std; tells the compiler to use the std namespace. Namespaces are a relatively recent addition to C++.

Execution:

g++ first.cpp

C:\Users\hai\CPP\Module3>first

Baba!Please bless me and my family

Make sure that g++ is in your path and that you are running it in the directory containing file hello.cpp.

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++ Keywords

The following list shows the reserved words in C++. These reserved words may not be used as constant or variable or any other identifier names.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |

C++ New Lines:

#include<iostream>

using namespace std;

int main()

{

    cout<<"Baba!Please bless me and my family\n";

    cout<<"Have a nice day";

    return 0;

}

(or)

#include<iostream>

using namespace std;

int main()

{

    cout<<"Baba!Please bless me and my family"<<endl;

    cout<<"Have a nice day";

    return 0;

}

endl is called as “manipulator”.

# C++ Variables

Variables are containers for storing data values.

In C++, there are different types of variables (defined with different keywords), for example:

int - stores integers (whole numbers), without decimals, such as 123 or -123

double - stores floating point numbers, with decimals, such as 19.99 or -19.99

char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes

string - stores text, such as "Hello World". String values are surrounded by double quotes

bool - stores values with two states: true or false

**Datatypes in C++:**

**Primitive data types:**

|  |  |
| --- | --- |
| **Type** | **Keyword** |
| Boolean | bool |
| Character | char |
| Integer | int |
| Floating point | float |
| Double floating point | double |
| Valueless | void |

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

**signed**

**unsigned**

**short**

**long**

**Ex:**

**Numeric Types**

**Use int when you need to store a whole number without decimals, like 35 or 1000, and float or double when you need a floating point number (with decimals), like 9.99 or 3.14515.**

**int**

**int myNum = 1000;**

**cout << myNum;**

**float**

**float myNum = 5.75;**

**cout << myNum;**

**double**

**double myNum = 19.99;**

**cout << myNum;**

**float vs. double**

**The precision of a floating point value indicates how many digits the value can have after the decimal point. The precision of float is only six or seven decimal digits, while double variables have a precision of about 15 digits. Therefore it is safer to use double for most calculations.**

# C++ Boolean Data Types

**Boolean Types**

**A boolean data type is declared with the bool keyword and can only take the values true or false. When the value is returned, true = 1 and false = 0.**

**Example**

**bool isCodingFun = true;**

**bool isFishTasty = false;**

**cout << isCodingFun; // Outputs 1 (true)**

**cout << isFishTasty; // Outputs 0 (false)**

# C++ Character Data Types

**The char data type is used to store a single character. The character must be surrounded by single quotes, like 'A' or 'c':**

**Example**

**char myGrade = 'B';**

**cout << myGrade;**

**Alternatively, you can use ASCII values to display certain characters:**

**Example**

**char a = 65, b = 66, c = 67;**

**cout << a;**

**cout << b;**

**cout << c;**

**String Types**

**The string type is used to store a sequence of characters (text). This is not a built-in type, but it behaves like one in its most basic usage. String values must be surrounded by double quotes:**

**Example**

**string greeting = "Hello";**

**cout << greeting;**

**To use strings, you must include an additional header file in the source code, the <string> library:**

**Example**

**// Include the string library**

**#include <string>**

**// Create a string variable**

**string greeting = "Hello";**

**// Output string value**

**cout << greeting;**

#include<iostream>

#include<string>

using namespace std;

int main()

{

    bool find=false;

    string name="siri";

    cout<<find<<endl;

    cout<<name<<endl;

}

**Declaring (Creating) Variables**

To create a variable, specify the type and assign it a value:

**Syntax**

type variableName = value;

Where type is one of C++ types (such as int), and variableName is the name of the variable (such as x or myName). The equal sign is used to assign values to the variable.

To create a variable that should store a number, look at the following example:

int myNum = 5; // Integer (whole number without decimals)

double myFloatNum = 5.99; // Floating point number (with decimals)

char myLetter = 'D'; // Character

string myText = "Hello"; // String (text)

bool myBoolean = true; // Boolean (true or false)

#include<iostream>

using namespace std;

int main()

{

    int a=10;

    printf("%d\n",a);

    int b=20;

    int c=a+b;

    cout<<c;

}

**sizeof ():**

#include<iostream>

using namespace std;

int main()

{

    unsigned short int a=10;

    float b=12.34;

    cout<<"short int size="<<sizeof(a)<<endl;

    cout<<"sizeof float="<<sizeof(b)<<endl;

}

Declare Many Variables

To declare more than one variable of the same type, use a comma-separated list:

Example

int x = 5, y = 6, z = 50;

cout << x + y + z;

# C++ Identifiers

All C++ variables must be identified with unique names.

These unique names are called identifiers.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

Example

// Good

int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is

int m = 60;

The general rules for naming variables are:

Names can contain letters, digits and underscores

Names must begin with a letter or an underscore (\_)

Names are case sensitive (myVar and myvar are different variables)

Names cannot contain whitespaces or special characters like !, #, %, etc.

Reserved words (like C++ keywords, such as int) cannot be used as names

# C++ Constants

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'

**Operators**

C++ divides the operators into the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Logical operators

Bitwise operators

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** | **Try it** |
| + | Addition | Adds together two values | x + y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_add) |
| - | Subtraction | Subtracts one value from another | x - y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_sub) |
| \* | Multiplication | Multiplies two values | x \* y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_mult) |
| / | Division | Divides one value by another | x / y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_div) |
| % | Modulus | Returns the division remainder | x % y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_mod) |
| ++ | Increment | Increases the value of a variable by 1 | ++x | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_inc) |
| -- | Decrement | Decreases the value of a variable by 1 |  |  |

A list of all assignment operators:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Example** | **Same As** | **Try it** |
| = | x = 5 | x = 5 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass1) |
| += | x += 3 | x = x + 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass2) |
| -= | x -= 3 | x = x - 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass3) |
| \*= | x \*= 3 | x = x \* 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass4) |
| /= | x /= 3 | x = x / 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass5) |
| %= | x %= 3 | x = x % 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass6) |
| &= | x &= 3 | x = x & 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass7) |
| |= | x |= 3 | x = x | 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass8) |
| ^= | x ^= 3 | x = x ^ 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass9) |
| >>= | x >>= 3 | x = x >> 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass10) |
| <<= | x <<= 3 | x = x << 3 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_ass11) |

[❮ Previous](https://www.w3schools.com/CPP/cpp_operators.asp)[Next ❯](https://www.w3schools.com/CPP/cpp_operators_comparison.asp)

A list of all comparison operators:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Try it** |
| == | Equal to | x == y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_compare1) |
| != | Not equal | x != y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_compare2) |
| > | Greater than | x > y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_compare3) |
| < | Less than | x < y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_compare4) |
| >= | Greater than or equal to | x >= y | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_compare5) |
| <= | Less than or equal to | x <= y |  |

## Logical Operators

Logical operators are used to determine the logic between variables or values:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Name** | **Description** | **Example** | **Try it** |
| && | Logical and | Returns true if both statements are true | x < 5 &&  x < 10 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_logical1) |
| || | Logical or | Returns true if one of the statements is true | x < 5 || x < 4 | [Try it »](https://www.w3schools.com/CPP/trycpp.asp?filename=demo_oper_logical2) |
| ! | Logical not | Reverse the result, returns false if the result is true | !(x < 5 && x < 10) |  |