C++ Programming Language

C++ is the most used and most popular programming language developed by Bjarne Stroustrup. C++ is a high-level and object-oriented programming language. This language allows developers to write clean and efficient code for large applications and software development, game development, and operating system programming. It is an expansion of the C programming language to include Object Oriented Programming(OOPs) and is used to develop programs for computers. This C++ Tutorial will cover all the basic to advanced topics of C++ like C++ basics, C++ functions, C++ classes, OOPs and STL concepts.

What is C++?

C++ is a most popular cross-platform programming language which is used to create high-performance applications and software like OS, Games, E-commerce software, etc. It was developed by Bjarne Stroustrup, as an extension of C language. C++ give a high level of control over system resources and memory.

Why Learn C++?

- C++ is one of the most used and popular programming languages.
- C++ is used in making operating systems, embedded systems, and Graphical User Interfaces.
- It is an object-oriented programming language that implements all the OOPs concepts such as Abstraction, Encapsulation, and Inheritance, which gives a clear structure to programs and allows code to be reused, lowering development costs and providing security.
- It is portable and can be used to create applications that can be adapted to multiple platforms.
- C++ is easy to learn so that you can choose it as your first programming language.
- It makes programming easy for programmers to switch to C++ because its syntax is similar to C, Java, and C#.

C++ Tutorial – Table of Content

- Overview of C++
- Basics of C++
- Variables and Constants in C++
- Data Types and Literals in C++
- Operators in C++
- Input/Output in C++

- Control Statements in C++
- Functions in C++
- Pointers and References in C++
- Arrays in C++
- Strings in C++
- Structures and Union in C++
- Dynamic Memory Management in C++
- Object-Oriented Programming in C++
- Encapsulation and Abstraction in C++
- Polymorphism in C++
- Function Overloading in C++
- Operator Overloading in C++
- Inheritance in C++
- Virtual Functions in C++
- Exception Handling in C++
- Files and Streams in C++
- Templates in C++
- STL in C++
- Iterators in C++
- Preprocessors in C++
- Namespace in C++
- Advanced C++
- C vs C++
- C++ vs Java
- Competitive Programming in C++
- Interview Questions in C++

C++ Overview

- Introduction to C++
- Features of C++
- History of C++
- Interesting Facts about C++
- Setting up C++ Development Environment
- Similarities and Differences between C++ and C

C++ Basics

- First C++ Program
- C++ Basic Syntax
- C++ Comments
- C++ Tokens
- C++ Identifiers

- C++ Keywords
- Difference between Keyword and Identifier

C++ Variables and Constants

- C++ Variables
- C++ Constants
- Scope of C++ Variables
- C++ Storage Classes
- C++ Static Variables

C++ Data Types and Literals

- C++ Data Types
- C++ Literals
- C++ Derived Data Types
- C++ User-Defined Data Types
- C++ Data Type Ranges and Their Macros
- C++ Type Modifiers
- C++ Data Type Conversion
- C++ Typecasting Operators

C++ Operators

- C++ Operators
- C++ Arithmetic Operators
- C++ Unary Operators
- C++ Bitwise Operators
- C++ Relational Operators
- C++ Logical Operators
- C++ Assignment Operators
- C++ Ternary/Conditional Operators
- C++ Sizeof Operator
- C++ Scope Resolution Operator

C++ Input/Output

- C++ Basic Input / Output
- C++ Standard Input Stream (cin)
- C++ Standard Output Stream (cout)
- C++ Standard Error Stream (cerr)
- C++ Input / Output Manipulator

C++ Control Statements

- Decision Making in C++
- C++ if Statement
- C++ if-else Statement
- C++ if-else-if Ladder
- C++ Nested if-else Statement
- C++ Switch Statement
- C++ Jump Statements
- C++ Loops
- C++ for Loop
- C++ Range-Based for Loop
- C++ while Loop
- C++ do…while Loop

C++ Functions

- C++ Functions
- C++ return
- C++ Parameter Passing Techniques
- Difference between Call by Value and Call by Reference
- C++ Default Arguments
- C++ Recursion
- C++ Inline Functions
- C++ Lambda Expression

C++ Pointers and References

- C++ Pointers and References
- C++ Pointers
- C++ Pointer Arithmetic
- Dangling, Void, Null, and Wild Pointers
- Applications of Pointers
- C++ nullptr
- C++ References
- Can references refer to an invalid location in C++?
- Difference Between Pointers and References in C++
- Passing by pointer Vs Passing by Reference in C++
- When do we pass arguments by reference or pointer?

C++ Arrays

- C++ Arrays
- C++ Multidimensional Arrays
- C++ Variable Length Arrays
- C++ Pointer to an Array
- Size of Array parameter
- Passing Arrays to Functions in C++
- What is Array Decay in C++? How can it be prevented?

C++ Strings

- C++ Strings
- C++ std::string Class
- C++ Array of Strings
- C++ String Functions
- C++ String Concatenation
- Tokenizing a String in C++
- C++ Substring

C++ Structures and Unions

- C++ Structures, Unions, and Enumerations
- C++ Structures
- C++ Pointer to Structure
- C++ Self-Referential Structures
- Difference Between C Structures and C++ Structures
- C++ Unions
- C++ Bit Fields
- C++ Enumeration
- C++ typedef
- Array of Structures vs Array within a Structure in C/C++

C++ Dynamic Memory Management

- C++ Dynamic Memory Management
- C++ new and delete Operators
- new vs malloc() and free() vs delete in C++
- Memory leak in C++
- Difference between Static and Dynamic Memory Allocation in C++

C++ Object-Oriented Programming

- C++ Object Oriented Programming (OOPs)
- C++ Classes and Objects
- C++ Access Modifiers
- C++ Friend Class and Function
- C++ Constructors
- C++ Default Constructors
- C++ Copy Constructor
- C++ Destructors
- C++ Private Destructor
- When is the Copy Constructor Called?
- Shallow Copy and Deep Copy in C++
- When Should We Write Our Own Copy Constructor?
- Does the Compiler Create a Default Constructor When We Write Our Own?
- C++ Static Data Members

- C++ Static Member Functions
- C++ this pointer
- C++ Scope Resolution Operator vs this pointer
- C++ Local Class
- C++ Nested Classes
- C++ enum Class
- Difference between Structure and Class in C++
- Why C++ is a partially Object Oriented Language?

C++ Encapsulation and Abstraction

- C++ Encapsulation
- C++ Abstraction
- Difference between Abstraction and Encapsulation in C++

C++ Polymorphism

- C++ Polymorphism
- C++ Function Overriding
- C++ Virtual Functions and Runtime Polymorphism
- Difference between Compile-time and Run-time Polymorphism in C++
- Difference between Inheritance and Polymorphism in C++

C++ Function Overloading

- C++ Function Overloading
- C++ Constructor Overloading
- C++ Functions that Cannot be Overloaded
- C++ Function Overloading and const Keyword
- C++ Function Overloading and Return Type
- C++ Function Overloading and float Data Type
- C++ Function Overloading and Default Arguments
- Can main() be overloaded?
- C++ Function Overloading Vs Function Overriding
- Advantages and Disadvantages of C++ Function Overloading

C++ Operator Overloading

- C++ Operator Overloading
- Types of C++ Operator Overloading
- C++ Functors
- C++ Operators that Cannot be Overloaded

C++ Inheritance

- C++ Inheritance
- C++ Inheritance Access
- C++ Multiple Inheritance
- C++ Hierarchical Inheritance
- C++ Multilevel Inheritance
- C++ Constructor in Multiple Inheritance
- C++ Inheritance and Friendship
- Does Function Overloading Work with Inheritance in C++?
- Difference Between Inheritance and Polymorphism in C++

C++ Virtual Functions

- C++ Virtual Functions
- C++ Virtual Functions in Derived Classes
- C++ Default Arguments and Virtual Function
- C++ Inline Virtual Functions
- C++ Virtual Destructor
- C++ Virtual Constructor
- C++ Virtual Copy Constructor
- C++ Pure Virtual Functions and Abstract Class
- C++ Pure Virtual Destructor in C++
- Can Static Functions be Virtual in C++?
- C++ RTTI (Run-Time Type Information)
- Can C++ Virtual Functions be Private?

C++ Exception Handling

- C++ Exception Handling
- C++ Exception Handling using Classes
- C++ Stack Unwinding
- C++ User-Defined Exceptions

C++ Files and Streams

- C++ Files and Streams
- C++ I/O Redirection

C++ Templates

- C++ Templates
- C++ Template Specialization
- C++ using Keyword

C++ Standard Template Library (STL)

- The C++ Standard Template Library (STL)
- STL Algorithms
- STL Containers
- STL Vector
- STL Pair
- STL Set
- STL Multiset
- STL Stack
- STL Queue
- STL Priority Queue
- STL Deque
- STL List
- STL Forward List
- STL Map
- STL Multimap
- STL Bitset
- STL Unordered Sets
- STL Unordered Multiset
- STL Unordered Map
- STL Unordered Multimap

C++ Iterators

- Introduction to C++ Iterators
- C++ Input Iterators
- C++ Output Iterators
- C++ Forward Iterators
- C++ Bidirectional Iterators
- C++ Random Access Iterators
- C++ istream_iterator and ostream Iterator
- Difference between C++ Iterators and Pointers

C++ Preprocessors

- C++ Preprocessor
- C++ Preprocessor Directives
- C++ #include Directive
- C++ #define Directive
- C++ Conditional Preprocessors
- Difference between C++ Preprocessor Directives and Function Templates

C++ Namespace

- C++ Namespaces
- Extending C++ Namespace and Unnamed Namespace
- Accessing, Creating Header, Nesting, and Aliasing Namespace
- C++ Inline Namespaces

Advanced C++

- C++ Multithreading
- C++ Smart Pointers
- Differences between Different C++ Smart Pointers
- Type of ‘this’ Pointer in C++
- Delete ‘this’ Pointer in C++
- Passing C++ Function as a Parameter
- C++ Signal Handling
- C++ Generics

C vs C++

- Differences and Similarities between C++ and C
- Difference between C++ and Objective C
- C programs that won’t compile in C++
- Program that produces different results in C and C++
- Void * in C vs C++
- Type Difference of Character Literals in C vs C++
- Difference between Structures in C and Structures in C++
- Cin-Cout vs Scanf-Printf

C++ vs Java

- Differences and Similarities between C++ and Java
- Inheritance in C++ vs Java
- Static keyword in C++ vs Java
- Default Virtual Behavior in C++ vs Java
- Exception Handling in C++ vs Java
- Foreach loop in C++ vs Java
- Templates in C++ vs Generics in Java
- Floating Point Operations & Associativity in C, C++, and Java

Competitive Programming in C++

- Competitive Programming – A Complete Guide
- C++ Tricks for Competitive Programming
- Writing C/C++ code efficiently in Competitive Programming
- Why C++ is Best for Competitive Programming?
- Generating Test Cases in C++
- Fast I/O for Competitive Programming in C++
- Setting up Sublime Text for C++ Competitive Programming Environment
- Setting up VS Code for C++ Competitive Programming Environment
- Which C++ libraries are useful for competitive programming?
- Common mistakes to be avoided in Competitive Programming in C++

C++ Interview Questions

- Top 50 C++ Interview Questions and Answers
- Top C++ STL Interview Questions and Answers
- 30 OOPs Interview Questions and Answers
- Top C++ Exception Handling Interview Questions and Answers

Applications of C++

Here are the uses of C++ with real-world applications:

1. Operating Systems

C++ is most widely used programming language and become an ideal choice for developing operating systems. Mac OS X has majority of parts written in C++ and Most of Microsoft's software like Windows, Microsoft Office, IDE Visual Studio, and Internet Explorer are also written in C++.

2. Games

C++ used for game development and companies use it as their first choice to develop gaming systems because C++ is very close to the machine so It can easily manipulate resources and able to built complex 3D games, multiplayer game, etc. Unreal game engine make games using C++.

3. Web Browsers

Most of the browsers in Computers are developed in C++ for effecting goals and Mozilla Firefox is totally developed by C++ and Google Applications and software like Chrome and Google File System are partly written in C++.

4. Compilers

Compilers of many programming languages are designed in C and C++ and this is because they are moderately lower-level when compared to other higher-level programming languages and C/C++ are closer to the hardware.

5. Embedded Systems

Embedded systems that need the program closer to the hardware such as smartwatches, medical equipment systems, mobile phones etc., are developed in C++ and It can perform a lot of low-level function calls, unlike different high-level programming languages.

- ***Recent Articles on C++***

- ***C++ Programs***
- ***C++ Interview Questions***