

C++



LEARN IN 1 DAY

KRISHNA RUNGTA

Learn C++ in 1 Day

By Krishna Rungta

Copyright 2022 - All Rights Reserved – Krishna Rungta

ALL RIGHTS RESERVED. No part of this publication may be reproduced or transmitted in any form whatsoever, electronic, or mechanical, including photocopying, recording, or by any informational storage or retrieval system without express written, dated and signed permission from the author.

Table Of Content

[Chapter 1: C++ Programming: What is C++ | Learn Basic Concepts of C++](#)

1. [What is C++?](#)
2. [Popular programming languages in use?](#)
3. [Is C++ best programming language?](#)
4. [Who uses C++?](#)
5. [Five Basic Concepts of C++](#)
6. [Use of C++ Programming Language](#)

[Chapter 2: How to Download and Install C++ IDE on Windows](#)

1. [What is Dev-C++?](#)
2. [How to Download and Install Dev C++ on Windows](#)
3. [Features Of Dev-C++ IDE](#)

[Chapter 3: C++ Hello World Program with Code Explanation](#)

1. [Hello World Program in C++](#)
2. [Your First Program: C++ “Hello World!”](#)
[Explanation](#)
3. [Explanation of C++ Hello World Program Code](#)

[Chapter 4: C++ Variables and Types: int, double, char, string, bool](#)

1. [Variables in C++](#)

2. [Basic types of Variables in C++](#)
3. [Rules of Declaring Variables in C++](#)
4. [C++ Variable Data Types](#)
5. [Variable Name or Identifiers](#)
6. [Const Qualifier in C++](#)
7. [Scope of Variables in C++](#)
8. [Variable Type Conversion](#)
9. [Register Variables](#)
10. [Escape Sequences](#)

Chapter 5: Arrays in C++ | Declare | Initialize | Pointer to Array Examples

1. [What is an Array?](#)
2. [Why do we need arrays?](#)
3. [Declare an array in C++](#)
4. [Array Initialization](#)
5. [Types of Arrays](#)
6. [Accessing the values of an Array](#)
7. [Advantages of an Array in C++](#)
8. [Disadvantages of an Array in C++](#)

Chapter 6: C++ Operators with Examples

1. [What are Operators?](#)
2. [Arithmetic Operators](#)
3. [Relational Operators](#)
4. [Logical Operators](#)
5. [Bitwise Operators](#)
6. [Assignment Operators](#)
7. [Misc operator](#)
8. [Operators Precedence](#)

Chapter 7: C++ For Loop with EXAMPLE

1. [What is a For Loop?](#)
2. [How for loop works?](#)
3. [When to use a for loop?](#)
4. [Syntax of for loop](#)
5. [Example 1](#)
6. [Example 2](#)

Chapter 8: C++ do...while loop with Examples

1. [What is do-while loop?](#)
2. [When to use a do-while loop?](#)
3. [Syntax](#)
4. [How do-while loop works?](#)
5. [Example 1](#)
6. [Example 2](#)
7. [Nested Do While Loop](#)

Chapter 9: C++ Switch Case Statement with EXAMPLE

1. [What is a switch?](#)
2. [When to use a switch?](#)
3. [The break Keyword](#)
4. [Syntax](#)
5. [Example 1](#)
6. [Example 2](#)

Chapter 10: C++ Strings: strcpy(), strcat(), strlen(), strcmp() EXAMPLES

1. [What is a String?](#)
2. [Declaring Strings](#)
3. [Accessing string Values](#)
4. [String Functions:](#)

Chapter 11: C++ Exception Handling: Try, Catch, throw Example

1. [What is Exception Handling in C++?](#)
2. [Why Exception Handling?](#)
3. [Exception Handling Keywords](#)
4. [Syntax:](#)
5. [C++ Standard Exceptions](#)
6. [User-Defined Exceptions](#)

Chapter 12: C++ Dynamic Allocation of Arrays with Example

1. [What is a Dynamic Array?](#)
2. [Factors impacting performance of Dynamic Arrays](#)
3. [The new Keyword](#)
4. [Initializing dynamically allocated arrays](#)
5. [Resizing Arrays](#)
6. [Dynamically Deleting Arrays](#)

Chapter 13: C++ Pointers with Examples

1. [What are Pointers?](#)

2. [Addresses in C++](#)
3. [Pointer Declaration Syntax](#)
4. [Reference operator \(&\) and Deference operator \(*\)](#)
5. [Pointers and Arrays](#)
6. [NULL Pointer](#)
7. [Pointers of Variables](#)
8. [Application of Pointers](#)
9. [Advantages of using Pointers](#)

Chapter 14: C++ Char Data Type with Examples

1. [What is Char?](#)
2. [What is ASCII?](#)
3. [Char Declaration](#)
4. [Syntax:](#)
5. [Printing ASCII Value](#)
6. [Printing Char Value](#)
7. [Inputting Chars](#)
8. [Converting Character to String](#)

Chapter 15: C++ File Handling: How to Open, Write, Read, Close Files in C++

1. [What is file handling in C++?](#)
2. [The fstream Library](#)
3. [How to Open Files](#)
4. [How to Close Files](#)
5. [How to Write to Files](#)
6. [How to Read from Files](#)

Chapter 16: C++ Operator Overloading with Examples

1. [What is Operator Overloading?](#)
2. [Syntax](#)
3. [Different Approaches to Operator Overloading in C++](#)
4. [Can all C++ Operators be Overloaded?](#)
5. [Things to Remember:](#)
6. [Rules for Operator Overloading:](#)
7. [How to Overload Operator:](#)

Chapter 17: C++ Basic Input/Output: Cout, Cin, Cerr Example

1. [What are Streams in C++?](#)
2. [How do streams work?](#)
3. [Function Table](#)
4. [C++ Header files for Input/ Output](#)
5. [std::cout](#)
6. [std::cin](#)
7. [std::cerr](#)
8. [std::clog](#)
9. [Error handling with IO streams:](#)

Chapter 18: Stack in C++ STL with Example

1. [What is std::stack?](#)
2. [Stack Syntax](#)
3. [Member Types](#)
4. [Operations in Stack](#)
5. [Stack Implementation](#)
6. [push\(\) and pop\(\)](#)
7. [empty\(\), size\(\), top\(\)](#)
8. [emplace\(\) and swap\(\)](#)

9. [Stack in STL](#)

[Chapter 19: C++ Struct With Example](#)

1. [What is a Struct in C++?](#)
2. [When to use a Structure?](#)
3. [C++ Struct Initialization](#)
4. [Creating Struct Instances](#)
5. [Accessing Struct Members](#)
6. [Pointers to Structure](#)
7. [Struct as Function Argument](#)
8. [Limitation of a C++ structure](#)

[Chapter 20: Vector in C++ Standard Template Library \(STL\) with Example](#)

1. [What is a C++ Vector?](#)
2. [When to Use a Vector?](#)
3. [How to Initialize Vectors in C++](#)
4. [Iterators](#)
5. [Modifiers](#)
6. [Example 1](#)
7. [Example 2](#)
8. [Capacity](#)
9. [Example 3](#)

[Chapter 21: Map in C++ Standard Template Library \(STL\) with Example](#)

1. [What is std::map?](#)
2. [Why use std::map?](#)
3. [Syntax:](#)
4. [Member types:](#)

5. [Built-in Functions](#)
6. [Iterating over Map Elements](#)
7. [Inserting data in std::map](#)
8. [Searching in a Map](#)
9. [Deleting Data from a Map](#)

Chapter 22: C++ Class and Object with Example

1. [What is a Class?](#)
2. [Class Declaration](#)
3. [Private and Public Keywords](#)
4. [Object Definition](#)
5. [Accessing Data Members](#)
6. [Class Member Functions](#)
7. [Constructors and Destructors](#)

Chapter 23: C++ Polymorphism with Example

1. [What is Polymorphism in C++?](#)
2. [Types of Polymorphism](#)
3. [Compile Time Polymorphism](#)
4. [Function Overloading](#)
5. [Operator Overloading](#)
6. [Runtime Polymorphism](#)
7. [Function Overriding](#)
8. [C++ Virtual Function](#)
9. [Compile-Time Polymorphism Vs. Run-Time Polymorphism](#)

Chapter 24: std::list in C++ with Example

1. [What is an std::list?](#)
2. [Why use std::list?](#)

3. [List Syntax](#)
4. [C++ List Functions](#)
5. [<list> Constructors](#)
6. [Container properties](#)
7. [Inserting into a List](#)
8. [Deleting from a List](#)

Chapter 25: C++ Functions with Examples

1. [What is a Function in C++?](#)
2. [Why use functions?](#)
3. [Built-in Functions](#)
4. [User-Defined Functions](#)
5. [Function Declaration/Prototype](#)
6. [Function Definition](#)
7. [Syntax:](#)
8. [Function Call](#)
9. [Passing Arguments](#)

Chapter 26: Difference between Structure and Class: Explained with C++ Example

1. [What is Structure?](#)
2. [What is Class?](#)
3. [Syntax of Class in C++](#)
4. [Syntax of Structure in C++](#)
5. [Difference between Structure and Class](#)
6. [Which One Should You Choose?](#)

Chapter 1: C++ Programming: What is C++ | Learn Basic Concepts of C++

What is C++?

C++ is a general-purpose, object-oriented programming language. It was created by Bjarne Stroustrup at Bell Labs circa 1980. C++ is very similar to C (invented by Dennis Ritchie in the early 1970s). C++ is so compatible with C that it will probably compile over 99% of C programs without changing a line of source code. Though C++ is a lot of well-structured and safer language than C as it OOPs based.

Some computer languages are written for a specific purpose. Like, Java was initially devised to control toasters and some other electronics. C was developed for programming OS. Pascal was conceptualized to teach proper programming techniques. But C++ is a general-purpose language. It well deserves the widely acknowledged nickname “Swiss Pocket Knife of Languages.” In this introduction to C++ tutorial, you will learn C++ basic concepts-

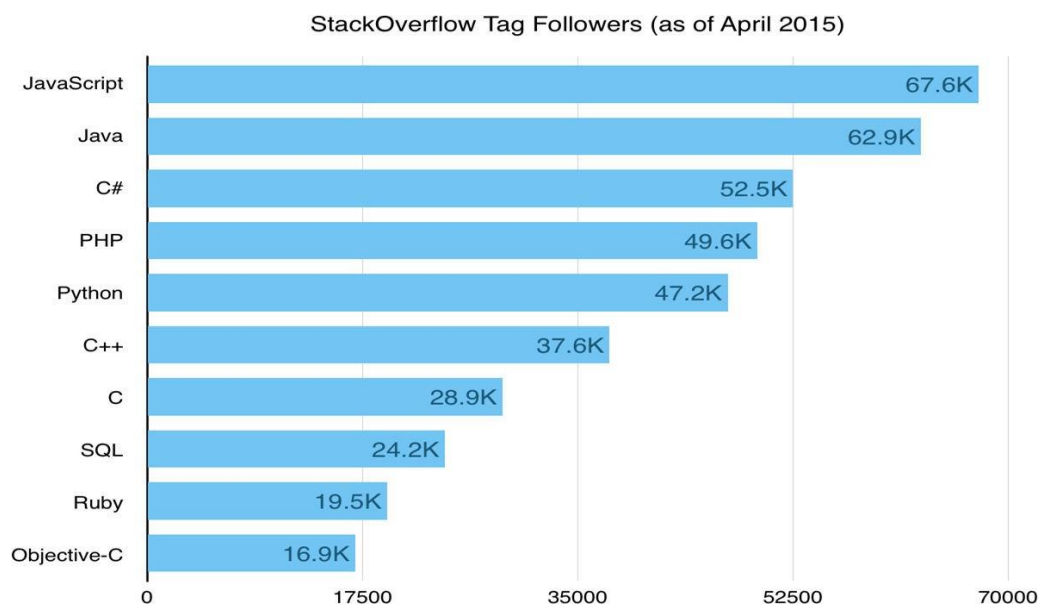
- What is C++?
- Popular programming languages in use?
- Is C++ best programming language?
- Who uses C++?
- Five Basic Concepts of C++
- Use of C++ Programming Language

Popular programming languages in use?

Popular languages that are mainly in use are Java, C++, Python, and C. Lower level languages like

- Assembly Language
- C
- C++

These languages force the programmer to think more about the problem in computer programming terms and its implementations, instead of the business logic.



As you can see despite being old, C++ is relatively popular still today which is a feat in and its own. The graph is from stackoverflow.com

Is C++ best programming language?

The answer depends on perspective and requirements. Some tasks can be done in C++, though not very quickly. For

example, designing GUI screens for applications. Other languages like Visual Basic, Python have GUI design elements built into them. Therefore, they are better suited for GUI type of task. Some of the scripting languages that provide extra programmability to applications. Such as MS Word and even photoshop tend to be variants of Basic, not C++. C++ is still used widely, and the most famous software have their backbone in C++. This tutorial will help you learn C++ basic and the advanced concepts.

Who uses C++?

Some of today's most visible used systems have their critical parts written in C++. Examples are Amadeus (airline ticketing)

- Bloomberg (financial formation),
- Amazon (Web commerce), Google (Web search)
- Facebook (social media)

Many programming languages depend on C++'s performance and reliability in their implementation. Examples include:

- Java Virtual Machines
- JavaScript interpreters (e.g., Google's V8)
- Browsers (e.g., Internet Explorer, Mozilla's Firefox, Apple's Safari, and Google's Chrome)
- Application and Web frameworks (e.g., Microsoft's .NET Web services framework).

Applications that involve local and wide area networks, user interaction, numeric, graphics, and database access highly depend on C++ language.

Five Basic Concepts of C++

Here are five basic C++ concepts:

C++ Variables

- Variables are the backbone of any programming language.
- A variable is merely a way to store some information for later use. We can retrieve this value or data by referring to a “word” that will describe this information.
- Once declared and defined they may be used many times within the scope in which they were declared.

C++ Control Structures

- When a program runs, the code is read by the compiler line by line (from top to bottom, and for the most part left to right). This is known as “**code flow**.”
- When the code is being read from top to bottom, it may encounter a point where it **needs to make a decision**. Based on the decision, the program may jump to a different part of the code. It may even make the compiler re-run a specific piece again, or just skip a bunch of code.
- You could think of this process like if you were to choose from different courses from Guru99. You decide, click a link and skip a few pages. In the same way, a computer program has a set of strict rules to decide the flow of program execution.

C++ Data Structures

Let’s use a list of courses on “guru99” as the example! You probably have a list of courses in front of you. But how do you think they stored that. There can be a lot of courses, and different users may register for different courses. Do they generate a different variable for each user? For example, let’s say we need to keep track of 10 courses. First, the **WRONG**

WAY: If we need to store 10 courses, we would probably define 10 variables, right? Wrong. In the world of programming, this is just a horrible way of trying to store 10 different variables. This is because of two main reasons:

- The huge amount of text that you'll need to write in your program. Sure, right now we only have 10 courses, so it's not too bad, but what if we had 1,000 courses! Imagine typing that out a thousand times! Forget about it!
- The flexibility. Adding another course would need manual edits to the code. We would have created variable COURSE11. This is just crazy!

So, what is the **RIGHT WAY**?

Storing them in data structures.

A **data structure** is a great **way to get around having to create thousands of variables**. C++ contains many types of inbuilt data structures. Most often used is arrays which will be taught later.

C++ Syntax

The syntax is a layout of words, expression, and symbols. Well, it's because an email address has its well-defined syntax. You need some combination of letters, numbers, potentially with underscores (_) or periods (.) in between, followed by an at the rate (@) symbol, followed by some website domain (company.com). So, syntax in a programming language is much the same. They are some well-defined set of rules that allow you to create some piece of well-functioning software. But, if you don't abide by the rules of a programming language or syntax, you'll get errors.

C++ Tools

In the real world, a tool is something (usually a physical object) that helps you to get a certain job done promptly. Well, this holds true with the programming world too. A tool in programming is some piece of software which when used with the code allows you to program faster. There are probably tens of thousands, if not millions of different tools across all the programming languages. Most crucial tool, considered by many, is an IDE, an **Integrated Development Environment**. An IDE is a software which will make your coding life so much easier. IDEs ensure that your files and folders are organized and give you a nice and clean way to view them.

Use of C++ Programming Language

Here are some prime uses of C++ Programming Language:

Operating Systems:

Whether it is Microsoft Windows or Mac OSX or Linux - all of the operating systems have some parts which are programmed in C++. It is the backbone of all the well-known OSs as C++ is a strongly typed and quick programming language, that makes it an ideal choice for developing an operating system.

Games:

Because of the fact that it is one of the fastest programming languages, C++ is widely used in programming of game development engines. C++ can easily manipulate hardware resources and it can also provide procedural programming for CPU intensive functions.

Browsers:

The rendering engines of various web browsers are programmed in C++ because of the speed it offers.

Libraries:

Many high-level libraries use C++ as the core programming language. For example, several Machine Learning libraries use C++ in the backend because of its speed.

Graphics:

C++ is widely used in almost all graphics applications that require fast rendering, image processing, real-time physics and mobile sensors.

Banking Applications:

One of the most popularly used core-banking systems - Infosys Finacle, uses C++ as the backend programming language. Banking applications need to process millions of transactions on a daily basis and require high concurrency and low latency support.

Cloud/Distributed Systems:

Cloud storage systems use scalable file-systems that work close to the hardware. That's why C++ becomes a preferred choice for Cloud systems.

Embedded Systems:

Various embedded systems like medical machines, smartwatches, etc., use C++ as the primary programming language.

Compilers:

Compilers of various programming languages use C++ as the backend programming language.

Chapter 2: How to Download and Install C++ IDE on Windows

What is Dev-C++?

Dev-C++, developed by Bloodshed Software, is a fully-featured graphical IDE (Integrated Development Environment) for C and C++ programming. It is distributed under the GNU General Public License for programming in C and C++.

How to Download and Install Dev C++ on Windows

There are many compilers available for C++ programming. You can download anyone. Here, we are going to use Dev C++. It will work for both C++ and C programming languages.

To install Dev C++ software, you need to follow the following steps.

Step 1) First you must download the Dev C++ on your Windows machine. Visit to Download Dev C++:

<http://www.bloodshed.net/>

Step 2) There are packages for different Operating Systems.

Get Dev-C++

Download original Dev-C++ 5
Dev-C++ 5.0 (4.9.9.2) with Mingw/GCC 3.4.2 compiler and GDB 5.2.1 debugger (9.0 MB)

Download Embarcadero Dev-C++ 6
This is a new fork of Dev-C++ sponsored by [Embarcadero](#) (developers of Delphi, C++ Builder and RAD Studio). Includes the TDM-GCC compiler

Order original Dev-C++ CD
Includes Dev-C++ 4 & 5, extra Dev-C++ Packages, Dev-Pascal, documentation and other software

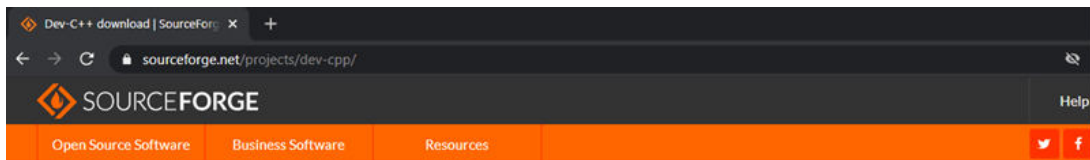
Download original Dev-C++ 5 IDE only
Dev-C++ 5.0 (4.9.9.2), IDE only - no compiler included (2.4 MB)

Step 3) Under package Dev-C++ 5.0 (4.9.9.2) with Mingw/GCC 3.4.2 compiler and GDB 5.2.1 debugger (9.0 MB) Click on the link “Download from SourceForge”.

Download original Dev-C++ 5

Dev-C++ 5.0 (4.9.9.2) with Mingw/GCC 3.4.2 compiler and GDB 5.2.1 debugger (9.0 MB)

Step 4) This package will download C++ .exe file for Windows that can be used to install on Windows 7/8/XP/VISTA/10.



Home / Browse / Development / Integrated Development Environments (IDE) / Dev-C++

Dev-C++
Open Source C & C++ IDE for Windows
Brought to you by: [claplace](#)

As of 2020-10-31, this project can be found here.

★★★★★ 75 Reviews

Downloads: 9,242 This Week

Last Update: 2020-11-19

Download | Get Updates | Share This

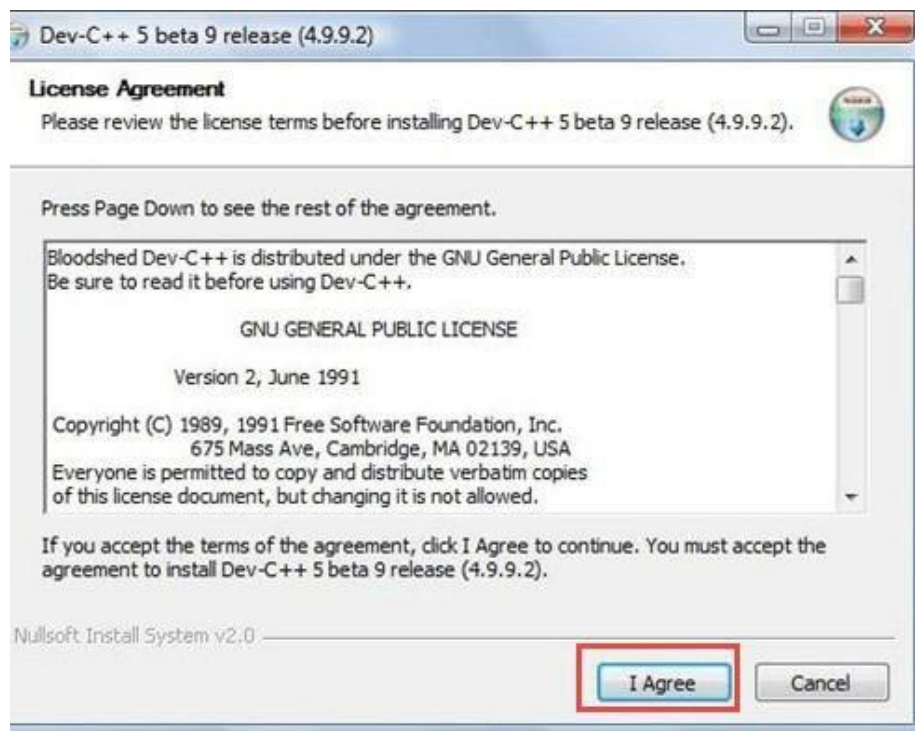
Windows | BSD | Linux

Step 5) You will direct to SourceForge website, and your C++ download will start automatically.

- Click on save button to save. By default, it is saved in “Downloads” folder.
- After the download completes, go to the saved .exe file and click on it to Run.
- The installer will ask you a language to select. Select “English” and click on “OK”.

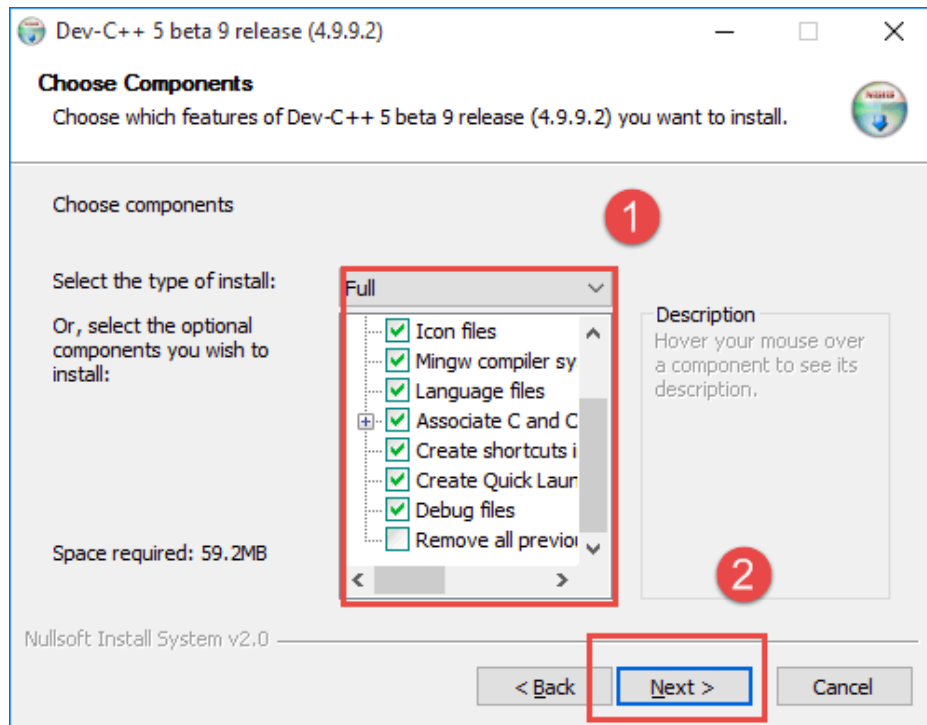


- Then screen for license agreement will appear. Click on “I agree” to proceed further.



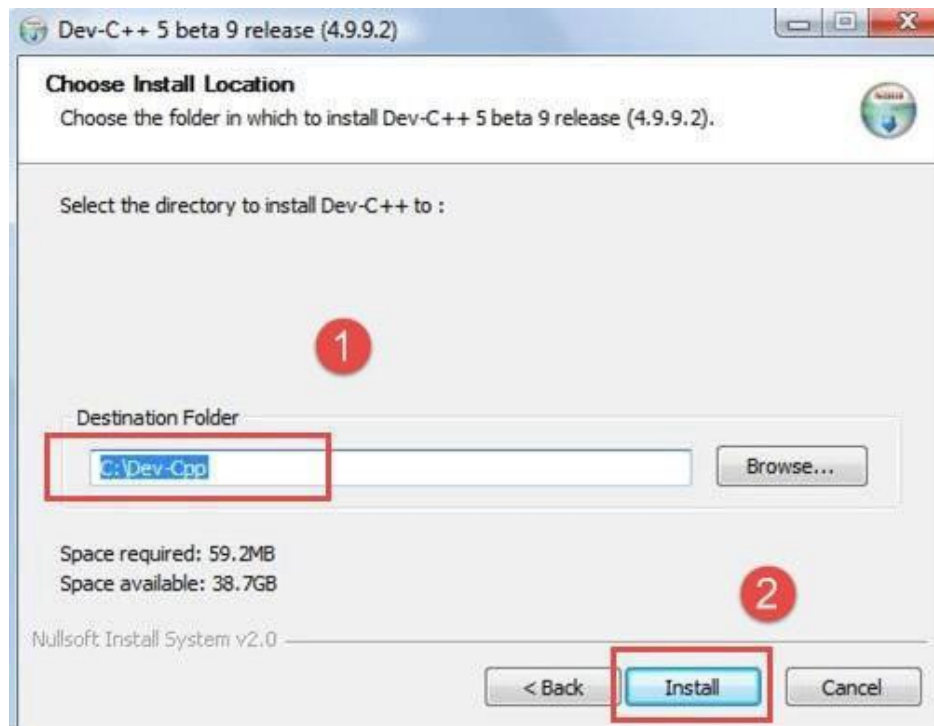
Step 6) In this step,

1. You can see different components of Dev C++ that will be installed with this package.
2. Just click on “next” button.

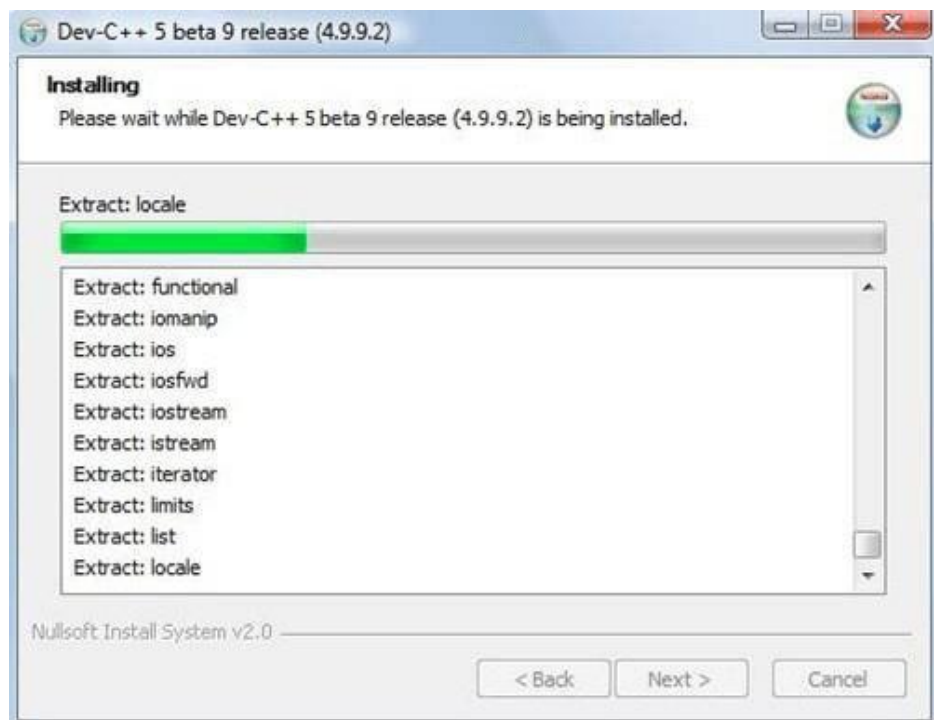


Step 7) In this step,

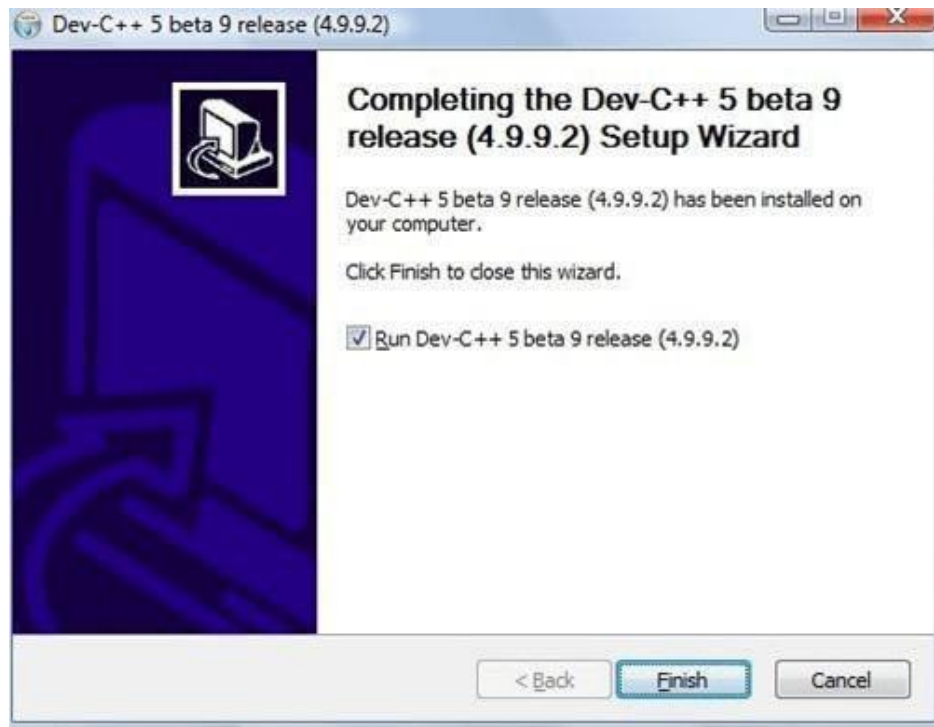
1. By default, the destination folder is in C drive. You are free to change this destination folder but make sure you have enough memory.
2. Click on “Install” button.



In the next screen, installation begins



Now, Dev C++ is installed successfully on your Windows. Select "Run Dev C++" to run it and click on "Finish" button.



That's it! Now you are ready to compile your C or C++ programs with Dev C++ compiler.

Features Of Dev-C++ IDE

Here are some important features of Dev-C++ IDE:

- Dev-C++ IDE allows us to use integrated debugging using GDB.
- Localization feature that provides support for multiple languages.
- Offers editing and compiling the resource files.
- It has a inbuilt find and replace facility.
- Helps you to create your own project templates to create project types.
- It offers support for class browser as well as debug variable browser.
- Provides project manager feature that helps you to manage various projects.
- Provides CVS support for source code management.

Buy Now \$9.99