Introduction To C++:

The history of C++ dates back to the early 1980s when it was developed by Bjarne Stroustrup, a Danish computer scientist. Here's a brief overview of the key milestones in the history of C++:

- **Origin:** The roots of C++ can be traced back to the C programming language, which was developed by Dennis Ritchie at Bell Labs in the early 1970s. C became widely popular due to its simplicity, efficiency, and portability. Bjarne Stroustrup, while working at Bell Labs in 1979, began experimenting with extending the C language to support object-oriented programming concepts.
- **C with Classes:** In 1983, Stroustrup released "C with Classes," which added support for classes, derived classes, inheritance, and other object-oriented features to the C language. This laid the foundation for what would later become C++.
- **Evolution into C++:** In 1985, Stroustrup further refined and extended "C with Classes," leading to the creation of C++. The name "C++" was chosen to signify an evolutionary step beyond C. C++ retained compatibility with C while introducing additional features such as virtual functions, templates, and exception handling.
- **First Edition of "The C++ Programming Language":** Stroustrup published the first edition of his seminal book, "The C++ Programming Language," in 1985. This book introduced the C++ language to a wider audience and served as a comprehensive reference for developers.
- **Standardization:** As C++ gained popularity, efforts began to standardize the language to ensure its consistency and portability across different platforms and implementations. The first official C++ standard, known as C++98 or ISO/IEC 14882:1998, was published in 1998. Subsequent revisions, including C++03, C++11, C++14, C++17, and C++20, introduced new features and improvements to the language.
- **Adoption and Growth:** C++ saw widespread adoption in various industries, including software development, gaming, finance, telecommunications, and embedded systems. Its efficiency, performance, and flexibility made it a popular choice for building a wide range of applications.
- **Community and Ecosystem:** Over the years, C++ has cultivated a vibrant community of developers, educators, and contributors. The language is supported by a rich ecosystem of libraries, frameworks, tools, and resources, further enhancing its utility and appeal.
- **Ongoing Development:** The evolution of C++ continues with ongoing efforts to modernize the language, improve its usability, and address emerging needs and challenges. The C++ Standards Committee regularly releases new standards that incorporate proposals for language enhancements and refinements.

Overall, the history of C++ reflects its evolution from a simple extension of the C language to a powerful, feature-rich programming language widely used in diverse domains and industries.

Why C++ is still popular?

C++ remains popular for several reasons:

1. **Performance:** C++ provides low-level control over system resources and memory management, allowing developers to write highly efficient code. This makes it well-suited for performance-critical applications such as game development, high-frequency trading systems, operating systems, and embedded systems.
2. **Versatility:** C++ is a versatile language that can be used for a wide range of applications, including system programming, desktop software, game development, scientific computing, and more. Its flexibility allows developers to tackle diverse projects without being constrained by the limitations of higher-level languages.
3. **Legacy codebases:** Many large-scale projects and software systems have been developed in C++ over the years. These legacy codebases often require ongoing maintenance and further development, ensuring that there is a continued demand for developers proficient in C++.
4. **Compatibility and Interoperability:** C++ offers strong compatibility with other programming languages such as C, allowing developers to leverage existing libraries and infrastructure. Additionally, C++ can be integrated with higher-level languages like Python through libraries such as Boost. Python, enabling developers to combine the performance of C++ with the productivity of languages like Python.
5. **Community and Ecosystem:** C++ has a large and active community of developers, contributing to the development of libraries, frameworks, and tools that enhance the language's capabilities. This rich ecosystem includes popular libraries like STL (Standard Template Library), Boost, Qt, and many others, which provide solutions for various programming tasks.
6. **Industry Adoption:** C++ is widely used in industries such as finance, gaming, aerospace, automotive, and telecommunications, where performance, reliability, and control are paramount. The continued adoption of C++ in these sectors ensures its relevance and popularity in the software development landscape.

Overall, the combination of performance, versatility, compatibility, community support, and industry adoption contribute to the enduring popularity of C++ as a programming language.

C++ Application Areas:

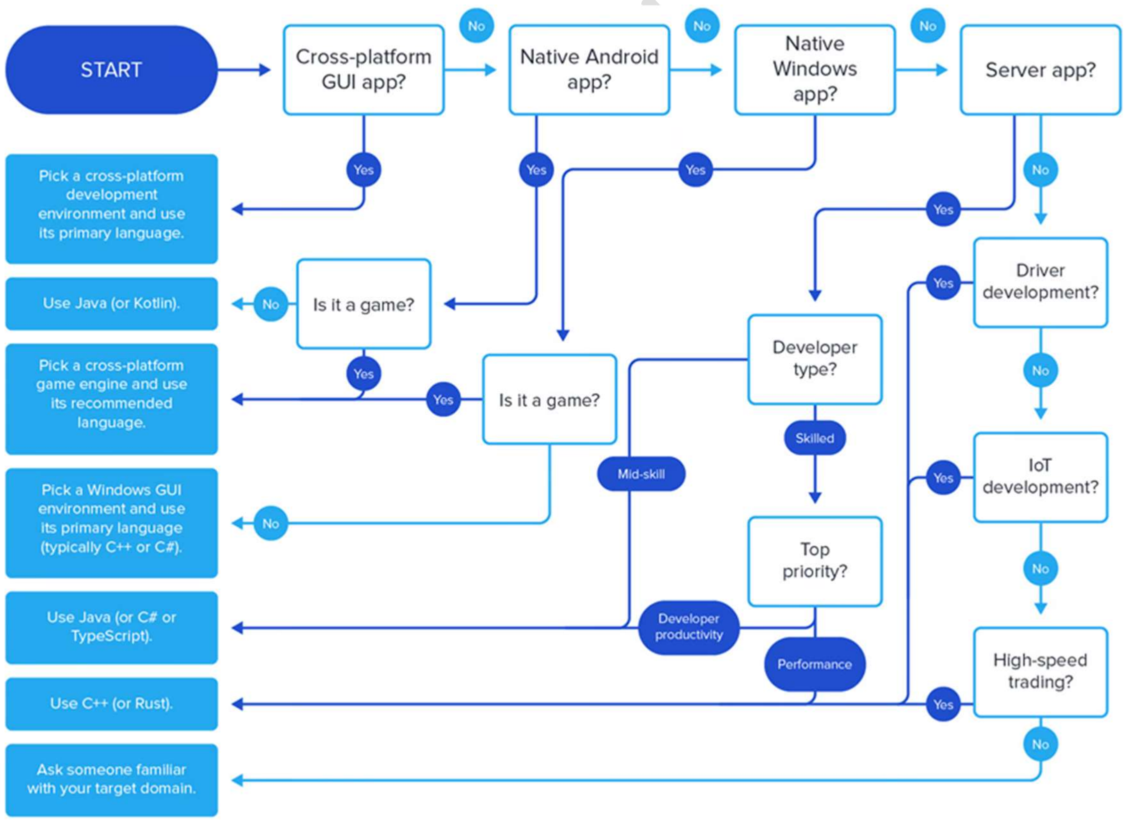
C++ finds application in various domains and roles across different industries due to its versatility, performance, and low-level control over hardware resources. Here's a list of different application domains and roles where C++ is commonly used:

- System Programming:
    - Developing operating systems (e.g., Windows, Linux, macOS).
    - Writing device drivers and firmware for hardware peripherals.
    - Building system utilities and low-level services.
- Game Development:

- Creating high-performance video games for consoles, PCs, and mobile platforms.
- Developing game engines, graphics rendering pipelines, and physics simulations.
- Implementing networking, AI, and multiplayer functionality.

- Embedded Systems:
  - Designing real-time embedded software for microcontrollers, IoT devices, and automotive systems.
  - Developing firmware for consumer electronics, medical devices, and industrial machinery.
  - Implementing communication protocols, sensor interfaces, and control algorithms.
- High-Performance Computing (HPC):
  - Writing scientific simulations, numerical libraries, and computational algorithms.
  - Developing parallel and distributed computing applications for clusters and supercomputers.
  - Optimizing code for performance-critical tasks in scientific research and engineering.
- Finance and Trading:
  - Building algorithmic trading systems, financial modelling software, and risk analysis tools.
  - Developing low-latency trading platforms and market data processing engines.
  - Implementing quantitative analysis algorithms and trading strategies.
- Graphics and Multimedia:
  - Creating graphics-intensive applications such as computer-aided design (CAD) software and 3D modelling tools.
  - Developing multimedia frameworks, audio/video processing libraries, and image manipulation software.
  - Implementing real-time rendering engines, game development tools, and virtual reality (VR) applications.
- Networking and Telecommunications:
  - Developing network protocols, communication stacks, and routing algorithms.
  - Writing network servers, client applications, and middleware for distributed systems.
  - Implementing network infrastructure components such as routers, switches, and gateways.
- Performance Monitoring and Analysis:
  - Building performance profiling tools, debuggers, and code optimization utilities.
  - Developing monitoring agents, data collection systems, and performance visualization tools.

    - Analysing system performance, resource utilization, and bottlenecks in software applications.
- Cross-Platform Development:
    - Writing cross-platform libraries and frameworks for application development.
    - Developing cross-platform GUI applications using toolkits like Qt and wxWidgets.
    - Building portable software components for deployment on multiple operating systems and hardware platforms.
- Legacy Code Maintenance:
    - Maintaining and enhancing existing C++ codebases in industries with long-lived software systems.
    - Modernizing legacy applications by refactoring code, adding new features, and improving performance.
    - Migrating legacy systems to newer platforms, architectures, or programming paradigms.


START
Cross-platform GUI app?
Native Android app?
Native Windows app?
Server app?
No
No
No
Yes
Yes
Yes
Yes
No
Pick a cross-platform development environment and use its primary language.
Use Java (or Kotlin).
Is it a game?
Driver development?
Developer type?
Pick a cross-platform game engine and use its recommended language.
Is it a game?
Skilled
Mid-skill
Pick a Windows GUI environment and use its primary language (typically C++ or C#).
IoT development?
Top priority?
Use Java (or C# or TypeScript).
Developer productivity
Performance
High-speed trading?
Use C++ (or Rust).
Ask someone familiar with your target domain.

These are just a few examples of the diverse roles and applications of C++ across various industries and domains. Its combination of performance, versatility, and low-level control makes it a valuable tool for building a wide range of software solutions.

Industry Adoption and Realtime Applications:

Many top companies across various industries rely on C++ for different purposes. Here's a list of some prominent companies and how they use C++:

Google:

- Google uses C++ extensively in many of its products and services, including:
- Google Chrome: The web browser is built using C++ for its rendering engine (Blink), networking stack, and other core components.
- Google Search: C++ is used in backend systems for search indexing, data processing, and infrastructure.
- Google Earth: C++ powers the 3D mapping and visualization capabilities of Google Earth.
- TensorFlow: Google's machine learning framework utilizes C++ for performance-critical operations.

Microsoft:

- Microsoft utilizes C++ in a variety of products and technologies, such as:
- Windows Operating System: Large portions of the Windows kernel and system libraries are written in C++.
- Visual Studio: Microsoft's integrated development environment (IDE) is developed using C++.
- Microsoft Office Suite: C++ is used in various components of Microsoft Office, including Excel and PowerPoint.
- Xbox: C++ is used for game development on Xbox consoles and the Xbox Live platform.

Apple:

- Apple incorporates C++ into its ecosystem for several purposes, including:
- macOS and iOS Development: Many system-level components and frameworks on Apple's platforms are written in C++.
- Core Animation and Core Graphics: These frameworks, used for user interface rendering and animation, are implemented in C++.
- Safari Browser: C++ is used in the rendering engine (WebKit) and other components of the Safari web browser.

Facebook:

- Facebook relies on C++ for various aspects of its infrastructure and products, such as:
- Facebook Core: C++ is used in backend services, infrastructure components, and data processing systems.
- Instagram: The backend systems of Instagram, including image processing and data storage, utilize C++.
- Oculus VR: C++ is used for developing virtual reality experiences and applications for Oculus VR headsets.

Amazon:

- Amazon employs C++ in many areas of its operations, including:

- Amazon Web Services (AWS): C++ is used in backend services, networking infrastructure, and distributed systems.
- Amazon Prime Video: C++ is used for video encoding, streaming, and playback in the Prime Video platform.
- Kindle: C++ is used in the software stack of Kindle e-readers for rendering, user interface, and system management.

Adobe:

- Adobe utilizes C++ in various products and technologies, including:
- Adobe Photoshop: C++ is used for image processing, rendering, and performance-critical algorithms.
- Adobe Illustrator: C++ powers the vector graphics rendering engine and user interface of Illustrator.
- Adobe Premiere Pro: C++ is used for video editing, encoding, and playback in Premiere Pro.

Electronic Arts (EA):

- EA employs C++ extensively in the game development industry, including:
- Frostbite Engine: EA's proprietary game engine, used in titles such as Battlefield and FIFA, is developed in C++.
- Game Development: C++ is used for developing game logic, rendering, physics simulations, and networking in EA's games.

These are just a few examples of top companies that rely on C++ for various purposes, including software development, system programming, game development, and infrastructure management. C++'s performance, efficiency, and versatility make it a popular choice for building complex and high-performance software systems in a wide range of industries.

Different C++ Versions with list of features added:

Here's an overview of the different versions of C++ along with their key features:

C++98 (ISO/IEC 14882:1998):

- Standardized the core features of C++.
- Introduced classes, inheritance, polymorphism, templates, and exception handling.
- Provided the foundation for modern C++ development.
- Marked the first official standardization of the language.

C++03 (ISO/IEC 14882:2003):

- A minor revision aimed at fixing defects and clarifying ambiguities in the C++98 standard.
- Introduced no significant new language features but focused on improving the consistency and correctness of the language specification.

C++11 (ISO/IEC 14882:2011):

A major update introducing numerous new features and improvements:

- **Auto keyword:** Allows type inference for variables, reducing verbosity.
- **Lambda expressions:** Enable the definition of anonymous functions inline.
- **Range-based for loops:** Simplify iteration over elements of a container.
- **Uniform initialization syntax:** Provides a consistent syntax for initializing objects.
- **Smart pointers (std::unique_ptr, std::shared_ptr):** Manage resources automatically, improving memory safety.
- **Move semantics:** Enables efficient transfer of resources between objects, reducing unnecessary copying.
- **Concurrency support (std::thread, std::mutex):** Introduces standard library support for multithreading and concurrent programming.
- Introduced significant improvements in readability, performance, and expressiveness.

C++14 (ISO/IEC 14882:2014):

A minor update focused on refining and extending features introduced in C++11:

- **Relaxation of constexpr restrictions:** Allows more functions to be evaluated at compile-time.
- **Binary literals and generic lambdas:** Enhances expressiveness and flexibility.
- **Return type deduction for functions:** Simplifies function declarations.
- **Variable templates:** Extends the concept of templates to variables.
- Maintained compatibility with C++11 while providing additional features for developers.

C++17 (ISO/IEC 14882:2017):

Introduced several new language features and library improvements:

- **Structured bindings:** Simplify working with tuples and other structured types.
- **Inline variables:** Allow variables to be defined inline within function scope.
- **Fold expressions:** Enable concise expression of variadic templates.
- **std::optional and std::variant:** Enhance the standard library with optional and variant types.
- **Filesystem library:** Provides standard support for file system operations.
- **Parallel algorithms:** Introduce standard library support for parallel execution of algorithms.
- Focused on improving code readability, maintainability, and performance.

C++20 (ISO/IEC 14882:2020):

Introduced significant language and library enhancements:

- **Concepts:** A major new feature enabling the definition of concepts to constrain template parameters.

- **Ranges:** A comprehensive library for working with ranges of elements, providing a unified approach to sequence manipulation.
- **Coroutines:** Enable the definition of lightweight asynchronous tasks and generators.
- **Modules:** Introduce a new module system for organizing code and improving build times.
- **std::format:** A modern formatting library based on Python's str.format() syntax.
- Calendar and timezone support: Enhancements to the standard library for working with dates, times, and timezones.
- Addressed long-standing deficiencies and introduced modern features to improve developer productivity and code quality.

Each version of C++ builds upon its predecessors, introducing new features, refining existing ones, and addressing community feedback and evolving programming paradigms. These updates enable developers to write more expressive, efficient, and maintainable code while ensuring compatibility and interoperability with existing codebases.

Best C++ Compilers and IDEs to Use:

- A C++ compiler is a computer application that translates C/C++ source code into machine code (binary) that a computer system executes directly.
- An IDE, or Integrated Development Environment, is a software application that provides comprehensive facilities to programmers for developing software applications.

What are the common uses of an IDE?

- Writing, editing, and debugging source code in different programming languages.
- Building and managing software applications and projects.
- Automating the software development process.
- Integrating with version control systems (such as Git and Subversion).
- Providing set of tools and resources to help programmers be more productive and efficient.

C++ Compilation Process and steps involved?

C++ Compiler transforms source code into machine readable code to execute directly by the computer. The following are the steps involved in the C++ compilation process:

1. **Pre-processing:** First, the pre-processor reads the source code and performs macro expansions, inclusion of header files, and other operations as specified by pre-processor directives (#include, #define, #ifndef, etc.).
2. **Compilation:** Second step performs the actual translation of the source code into object code. The object code is a machine-readable representation of the source code, but it is not yet executable.

```
                  Instructions              Prog_nm
                       |____________________|
                                  |
                                  |            Debug
source code                 *.c/*.cpp <---------------------|
                                  |                          ^
                                  |                          |
                             compile ---------------------->| compile-time error
                                  |                          |  (syntax Errors)
                                  |                          |
backup file                    *.bak                         |
                                  |                          |
                  Linker --------->|------------------------>| Linker errors
                                  |                          |
object code file               *.obj                         |
                                  |                          |
                                  |                          ^
executable file                *.exe                         |
                                  |                          |
                               Run ----------------------->| Runtime errors
                                  |
                              Output
```

3. **Assembly:** The compiler then passes the object code to an assembler, which converts the object code into assembly code.
4. **Linking:** The linker then combines assembly code with any library functions that are required by the program and resolves any references to external symbols/libraries. The linker produces an executable file to run on the target platform.
5. **Execution:** Finally, the compiler produces and executable file that runs on the computer system.

Note that the exact steps involved in the C++ compilation process may vary slightly depending on the specific compiler and development environment. Some compilers, for example, may perform the pre-processing, compilation, and linking stages in a single step.

What are the main features of C++ Compiler?

- C++ Compiler translates C or C++ source code into machine readable code to execute directly by the computer.
- It optimizes the machine code to improve its performance by dead code elimination, register allocation, and instruction scheduling.
- Checks the source code for syntax and semantic errors before execution, and providing error/warning messages that helps programmer fix the issues.
- C++ Compiler proves a bridge between the high-level abstractions of C++ code and the underlying hardware.

Here is a list of the top ten compilers to use, in no particular order:

1) GCC (GNU Compiler Collection)

GCC (GNU Compiler Collection) is a free and open-source compiler for various programming languages including C, C++, Objective-C, FORTRAN, Ada, and others. It is widely used for various platforms including Linux, UNIX, and macOS.

Main features of GCC:

- Cross-platform Compatibility: GCC supports multiple platforms and operating systems, including Windows, Linux, macOS, and various other Unix-like systems.
- High Performance: GCC uses advanced optimization techniques to produce highly optimized code, making it a popular choice for high-performance computing and embedded systems.
- Standard C++ Compliance: GCC implements a large number of language standards, including ISO C11, ISO C++17, and others, ensuring compatibility with the latest language specifications.
- Debugging and Profiling: GCC includes a powerful debugger, GDB, and a profiler, which can be used to analyse and optimize code performance.
- Inter-language Support: GCC provides support for compiling and linking code written in multiple programming languages, making it a popular choice for projects that involve multiple languages.
- Active Community: GCC is an active open-source project with a large and dedicated community of developers, ensuring that it continues to improve and evolve over time.

Homepage: https://gcc.gnu.org/

2) LLVM Clang

Clang is a C, C++, and Objective-C compiler that is developed by the LLVM Project. It is known for its high performance, good diagnostics, and compatibility with the latest language standards.

Main features of Clang:

- Fast Compile Times: Clang has a reputation for fast compile times, making it a popular choice for large, complex projects.
- High-quality Diagnostics: Provides clear and concise error messages, making it easier to identify and fix problems in your code.
- Standard C++ Compliance: It implements the latest language standards, including ISO C11, ISO C++20, and Objective-C, ensuring compatibility with the latest language specifications.
- Inter-language Support: Clang provides support for compiling and linking code written in multiple programming languages. This makes it a popular choice for projects that involve multiple languages.

- Active Community: It has an active open-source project with a large and dedicated community of developers, ensuring that it continues to improve and evolve over time.
- Clang Static Analyzer: It includes a static code analyser tool, which can be used to analyse your code and identify potential problems such as memory leaks and undefined behaviour.
- Auto Code Format: It includes a tool called Clang-Tidy, which can be used to automatically format and clean up code, making it easier to maintain and improve.
- Integration with other Tools: Clang integrates well with other development tools, such as IDEs and build systems, making it a popular choice for many developers.
- Open-source and Cross-platform: Makes it accessible and usable on a wide range of systems and platforms.
- Optimizing Code Generation: By using advanced code generation techniques of the tool, it is possible to produce highly optimized code.

Homepage: https://clang.llvm.org/

3) Microsoft Visual C++

Microsoft Visual C++ is a C and C++ compiler that is part of Microsoft's Visual Studio Integrated Development Environment (IDE). Some of its key features related to C++ compilation are listed below.

Main features of Microsoft Visual C++:

- Code Editing: Visual Studio provides a rich code editor that supports syntax highlighting, code navigation, and IntelliSense, making it easier to write, debug, and maintain C++ code.
- Build System: Build system automates the process of compiling, linking, and building C and C++ applications.
- Debugging: Integrated debugger makes it easy to find and fix bugs in your code. The debugger supports breakpoints, watches, and other advanced debugging features.
- Standard Library Support: Microsoft Visual C++ includes Standard Library, which provides a comprehensive set of classes and functions for C and C++ development. Some are containers, algorithms, strings, and more.
- Standard Compliance: Visual Studio implements the latest C standards, including ISO C11 and C++17, ensuring compatibility with the latest language specifications.
- Cross-platform Development: Programmers use it to develop code for a variety of platforms. For example, Windows, iOS, and Android.
- Advanced Code Generation: Visual C++ uses advanced code generation techniques to produce highly optimized code, ensuring that your code runs as fast as possible.
- Support for Parallel Programming: VC++ includes support for parallel programming, making it easier to write code that takes advantage of multi-core processors and hardware acceleration features.
- Dynamic Memory Management: Visual C++ includes support for dynamic memory management, making it easier to write code that is both efficient and robust.

Homepage: https://visualstudio.microsoft.com/vs/features/cplusplus/

4) C++ Builder

C++ Builder is a rapid application development (RAD) environment for developing applications in the C++ programming language. It is developed by Embarcadero Technologies and is part of the RAD Studio suite. Some of its main features include:

Main features of C++ Builder:

- Modern C++ language Support: Supports the latest version of C++, including features such as C++11, C++14, and C++17.
- Visual Component Library (VCL): C++ Builder has a collection of visual and non-visual components for building Windows applications.
- Cross-platform development: Allows you to develop applications for Windows, macOS, iOS, and Android.
- High-performance UI: Includes visual design tools for creating high-performance and visually appealing user interfaces.
- Database Support: Integrates with popular database systems such as SQL Server, Oracle, and InterBase.
- Debugging and Profiling Tools: Includes integrated debugging and profiling tools for optimizing application performance.
- Cloud and Web development: Includes support for developing cloud and web-based applications, including RESTful services and web-based user interfaces.

Homepage: https://www.embarcadero.com/products/cbuilder

5) Dev-C++

Dev-C++ is a free, open-source integrated development environment (IDE) for the C++ programming language. It is developed by Bloodshed Software and runs on Windows operating systems. Some of its main features are listed here.

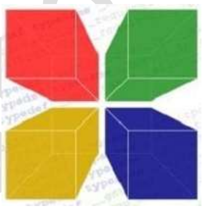
Main features of Dev C++:

- Syntax Highlighting: Automatically highlights C++ keywords, comments, and other language elements for improved code readability.
- Code Completion: Dev C++ suggests code snippets based on the context of what you are typing, making it easier to write code quickly and efficiently.
- Integrated Debugger: A built-in debugger that allows you to step through your code and inspect variables, making it easier to find and fix bugs.
- Project Management: Lets you organize and manage multiple projects within the same IDE, making it easy to switch between different projects.
- Compiler Support: Includes support for the GCC compiler, which is widely used for compiling C++ code.

- Plug-in Support: Dev C++ supports plug-ins, which are programs that extend the IDE to provide additional functionality.
- Source Code Control: Includes support for version control systems such as Git, making it easier to manage and track changes to your code.
- Active Community: Has a large and active community of users and developers who provide support and resources for using Dev C++.

Homepage: https://www.bloodshed.net/

6) Code::Blocks IDE for C/C++:



Code::Blocks is a free C/C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable. Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, event compiling and debugging functionality is provided by plugins!

Main features of Code::Blocks:

- Multi-platform Support: Code::Blocks is available on multiple platforms, including Windows, macOS, and Linux, ensuring developers can work in their preferred environment.
- Customizable Interface: The IDE provides a customizable user interface, allowing developers to adjust layouts, toolbars, and themes to suit their preferences and workflow.
- Built-in Compiler Support: Code::Blocks comes bundled with multiple compilers, including GCC (GNU Compiler Collection), Clang, and Microsoft Visual C++, enabling developers to compile and build C++ projects without the need for additional setup.
- Integrated Debugger: The IDE features an integrated debugger for debugging C++ code, allowing developers to set breakpoints, inspect variables, and step through code execution to identify and fix issues efficiently.
- Syntax Highlighting and Code Completion: Code::Blocks provides syntax highlighting and code completion features to enhance code readability and productivity. It helps developers write code faster and with fewer errors by suggesting completions and highlighting syntax elements.
- Project Management: Code::Blocks offers project management capabilities, allowing developers to organize source files, headers, and resources into projects and manage dependencies effectively.
- Resource Management: The IDE provides tools for managing project resources, including files, images, libraries, and external dependencies, simplifying the process of including and linking resources in C++ projects.
- Plugin Support: Code::Blocks supports plugins, extending its functionality with additional features and tools. Developers can install plugins to enhance the IDE's capabilities, such as version control integration, additional compilers, and code generation tools.

- Build System Integration: Code::Blocks integrates with popular build systems like Makefiles, allowing developers to build and execute complex C++ projects with ease. It supports customizable build configurations and options to tailor the build process to specific requirements.
- Project Templates and Wizards: The IDE provides project templates and wizards for creating various types of C++ projects, such as console applications, GUI applications, libraries, and more. This accelerates project setup and configuration, especially for new developers.
- Version Control Integration: Code::Blocks offers integration with version control systems such as Git, Subversion, and CVS, allowing developers to manage source code repositories directly from within the IDE.
- Documentation Integration: The IDE supports integration with documentation systems, enabling developers to access API documentation, tutorials, and reference materials while coding, improving productivity and code comprehension.

Homepage: https://www.codeblocks.org/

7) Eclipse IDE for C++

Eclipse IDE is a free, open-source integrated development environment (IDE) for the C++ programming language. It is part of the Eclipse platform and is developed by the Eclipse Foundation.

Main features of Eclipse IDE for C++

- Cross-platform Compatibility: Eclipse runs on multiple platforms, including Windows, Linux, and macOS, making it a popular choice for cross-platform development.
- Code Editing: Eclipse IDE has an advanced code editing features, such as syntax highlighting, code folding, and code completion, to help you write code more efficiently.
- Integrated Debugger: Includes an integrated debugger that allows you to step through your code, inspect variables, and find and fix bugs.
- Project Management: This IDE lets you manage multiple projects at once, making it easy to switch between different projects.
- Plug-in Support: Supports plug-ins, which are small programs that can be added to the IDE to provide additional functionality.
- Version Control: Includes support for version control systems, such as Git and Subversion, making it easier to manage and track changes to your code.
- Build and Deployment: Includes a powerful build system that can handle complex build and deployment processes.
- Active Community: Has a large and active community of users and developers who provide support and resources for using Eclipse IDE for C++.

Homepage: https://github.com/eclipse-cdt/

8) Qt Creator

Qt Creator is a free, open-source, and cross-platform integrated development environment (IDE) designed specifically for the development of applications using the Qt framework. It is written in C++ and provides a wide range of features to help developers create applications quickly and efficiently.

Main features of Qt Creator include:

- Code Editor: Qt Creator provides a code editor with syntax highlighting, code completion, and refactoring support.
- Debugging: The IDE has an integrated debugger that allows you to debug your application as you develop it.
- Project Management: Qt Creator provides a project management system that makes it easy to organize and manage your code.
- UI Design: The IDE includes a drag-and-drop interface designer for creating user interfaces for your applications.
- Efficient Build System: Qt Creator provides a flexible build system that supports multiple compilers and build configurations.
- Version Control: Qt Creator integrates with popular version control systems, such as Git, Subversion, and Mercurial, to provide an integrated development workflow.
- Cross Platform Support: Qt Creator supports multiple platforms, including Windows, Linux, macOS, and others.

Homepage: https://www.qt.io/product/development-tools

9) Intel C++ Compiler

The Intel C++ Compiler, also known as the Intel oneAPI DPC++/C++ Compiler, is a commercial-grade compiler that is designed to optimize C++ code for Intel architecture. It is part of the Intel open oneAPI industry standards initiative and is available for Windows, Linux, and macOS platforms.

The main features of the Intel C++ Compiler include:

- OpenMP support: The compiler supports the OpenMP programming model for shared memory parallel programming.
- High-level optimizations: The compiler includes optimizations for performance-critical libraries, such as the Intel Math Kernel Library and Threading Building Blocks.
- Floating-point precision: The compiler provides flexible floating-point precision options, allowing developers to choose the level of precision for their application.
- Code optimization: The compiler uses the latest Intel processors, vectorization, and auto-parallelization capabilities to improve code performance.
- Compatibility: The Intel C++ Compiler is compatible with the latest C++ standard and supports both Microsoft Visual Studio and GCC development environments.

- Debugging and Profiling: The Intel C++ Compiler includes tools for debugging and profiling, including Intel VTune Amplifier, Intel Advisor, and Intel Inspector.
- Vectorization Advisor: The Vectorization Advisor helps developers identify and optimize loops in their code to take advantage of vector processing capabilities.

Homepage:https://www.intel.com/content/www/us/en/developer/tools/oneapi/dpc-compiler.html#gs.4qlujl

10) NetBeans IDE:

NetBeans IDE is a free, open-source, and cross-platform integrated development environment (IDE) designed for C++ development, as well as other programming languages such as Java, PHP, and HTML5. It is written in Java and provides a range of features to help developers create applications quickly and efficiently.

Main features of NetBeans IDE for C++ include:

- Code Editor: NetBeans provides a code editor with syntax highlighting, code completion, and code folding.
- Debugging: The IDE has an integrated debugger that allows you to debug your application as you develop it.
- Project Management: NetBeans provides a project management system that makes it easy to organize and manage your code.
- Refactoring: NetBeans provides code refactoring support, which makes it easier to restructure your code and make it more maintainable.
- C++11 Support: NetBeans supports the latest C++11 standard, as well as earlier versions of the language.
- Cross-platform Compatibility: NetBeans supports multiple platforms, including Windows, Linux, macOS, and others.
- Version Control: NetBeans integrates with popular version control systems, such as Git, Subversion, Mercurial, and others, to provide an integrated development workflow.
- Plug-in Support: NetBeans has a plug-in architecture that allows you to extend its functionality by installing additional plug-ins from the NetBeans plug-in portal.

Homepage: https://netbeans.apache.org/front/main/

Above listed compilers are the best C++ Compilers that are available in the market. When choosing a C++ compiler, it is essential to consider factors such as the speed of code compilation and the efficiency in handling larger projects and cross-platform support.

Code::Blocks Installation:

Here in the further practise we are using code::Blocks, so download and install it as shown in the link given below.

Click to see: https://www.youtube.com/watch?v=UytHPb_xvh4

Advantages of C++ over C:

C++ offers several advantages over C, building upon the strengths of the C language while introducing additional features and capabilities. Here are some of the key advantages of C++ compared to C:

Object-Oriented Programming (OOP):

- C++ supports object-oriented programming paradigms, including classes, objects, inheritance, polymorphism, and encapsulation.
- OOP facilitates modular and reusable code, leading to improved code organization, maintenance, and scalability.

Classes and Encapsulation:

- C++ allows the definition of classes, which encapsulate data and related functions into a single unit.
- Encapsulation hides the internal implementation details of objects, promoting information hiding and reducing complexity.

Inheritance and Polymorphism:

- C++ supports inheritance, enabling the creation of hierarchical class structures where derived classes inherit properties and behaviour from base classes.
- Polymorphism allows objects of different classes to be treated uniformly through function overloading, virtual functions, and dynamic binding.

Function Overloading:

- C++ allows multiple functions with the same name but different parameter lists (function overloading), enhancing code expressiveness and readability.
- Function overloading enables the creation of more intuitive and self-documenting APIs.

Operator Overloading:

- C++ supports operator overloading, allowing operators to be redefined for user-defined types.
- Operator overloading enables natural syntax and intuitive usage for user-defined data types, enhancing code clarity and expressiveness.

Templates and Generics:

- C++ templates enable the creation of generic functions and classes, allowing algorithms to be implemented independently of specific data types.
- Templates facilitate code reuse and improve performance by enabling compile-time type checking and optimization.

Exception Handling:

- C++ provides built-in support for exception handling, allowing code to gracefully handle errors and exceptional conditions.
- Exception handling promotes robustness and reliability by separating error-handling logic from regular program flow.

Standard Template Library (STL):

- C++ includes the Standard Template Library (STL), a powerful collection of generic algorithms and data structures.
- STL components such as containers, iterators, algorithms, and function objects simplify common programming tasks and promote code reuse.

Improved Type Safety:

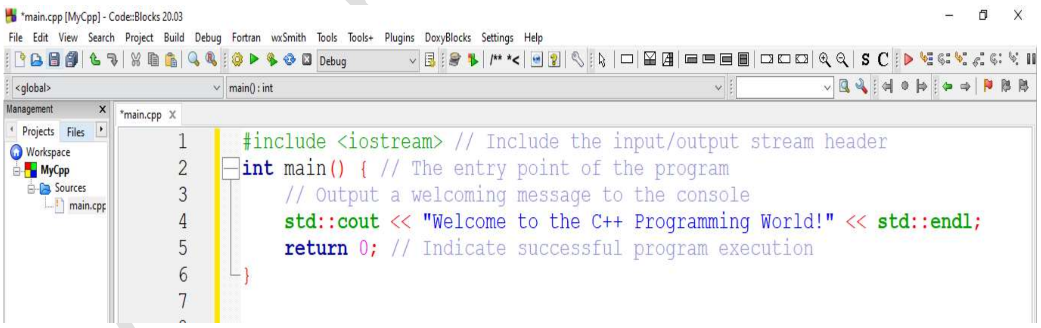
- C++ offers stronger type checking compared to C, reducing the likelihood of type-related errors and vulnerabilities.
- Stronger type safety enhances code reliability, security, and maintainability.

Compatibility with C:

- C++ is largely compatible with C, allowing existing C code to be seamlessly integrated and reused within C++ projects.
- Developers can leverage existing C libraries and infrastructure while gradually adopting C++ features and modernization practices.

C++ Welcome Program:

A "welcome program" in C++ is a simple program that prints a welcoming message to the user when executed. It serves as a basic example to demonstrate how to write, compile, and run a C++ program. Here's a simple welcome program in C++:


*main.cpp [MyCpp] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
main() : int
Management
Projects Files
Workspace
MyCpp
Sources
main.cpp
*main.cpp

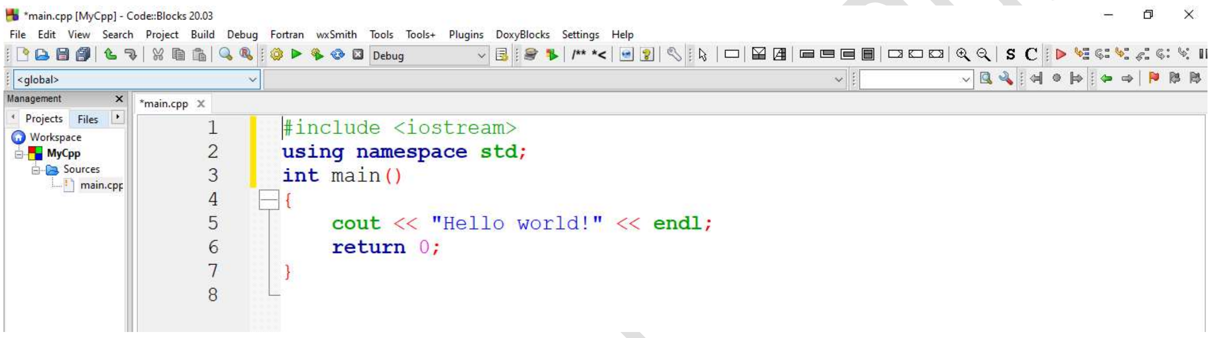
```cpp
#include <iostream> // Include the input/output stream header
int main() { // The entry point of the program
    // Output a welcoming message to the console
    std::cout << "Welcome to the C++ Programming World!" << std::endl;
    return 0; // Indicate successful program execution
}
```

Let's break down the components of this program:

- #include <iostream>: This line includes the header file iostream, which stands for "input/output stream." This header provides functionality for input and output operations in C++.
- int main() { ... }: This is the main function of the program, which serves as the entry point for execution. It returns an integer (int) value to indicate the status of the program to the operating system. In this case, 0 is returned to indicate successful execution.

- std::cout << "Welcome to the C++ Programming World!" << std::endl;: This line prints the welcoming message to the console. std::cout is the standard output stream, and << is the stream insertion operator used to output data. "Welcome to the C++ Programming World!" is the message to be printed, and std::endl is used to insert a newline character and flush the output stream.
- std:: is the namespace qualifier that indicates that cout is part of the std namespace. std is a namespace that encapsulates many standard C++ library components. In C++, the Standard Template Library (STL) provides a set of classes and functions for various common tasks, such as input/output operations.
- return 0;: This statement exits the main function and returns 0 to the operating system, indicating that the program executed successfully.

Instead of using std:: every time you can refer it by "using namespace" as:


*main.cpp [MyCpp] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
Management
Projects Files
Workspace
MyCpp
Sources
main.cpp
*main.cpp

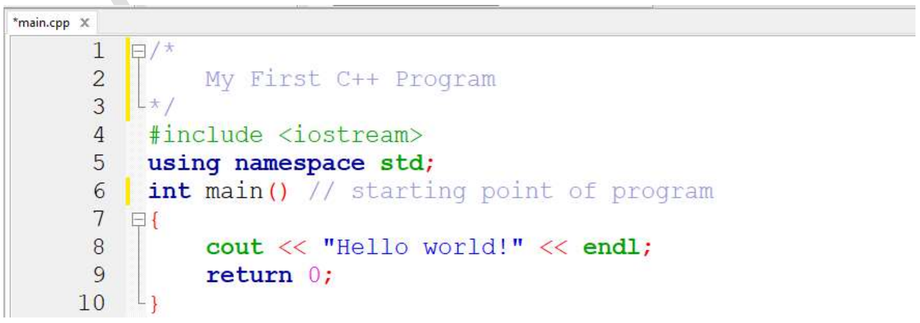
```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Comments in C++:

Comments in C++ are non-executable statements used to provide explanations or annotations within the code. They are ignored by the compiler during the compilation process and serve as documentation for the programmers or other readers of the code. There are two main types of comments in C++:

Single-Line Comments: Single-line comments begin with two forward slashes (//) and continue until the end of the line They are commonly used for brief explanations of code or for temporarily disabling code segments.

Multi-Line Comments: Multi-line comments, also known as block comments, start with /* and end with */. They can span multiple lines and are often used for longer explanations or comments that encompass multiple lines of code.



```
/*
    My First C++ Program
*/
#include <iostream>
using namespace std;
int main() // starting point of program
{
    cout << "Hello world!" << endl;
    return 0;
}
```