Practical # 01

Introduction to DEV C++ IDE

Objective: ***To understand the DEV C++ IDE(Integrated Development Environment) and implement a simple C Program.***

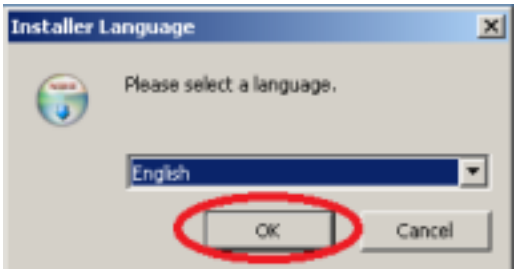
Theory:

The Integrated Development Environment (IDE)

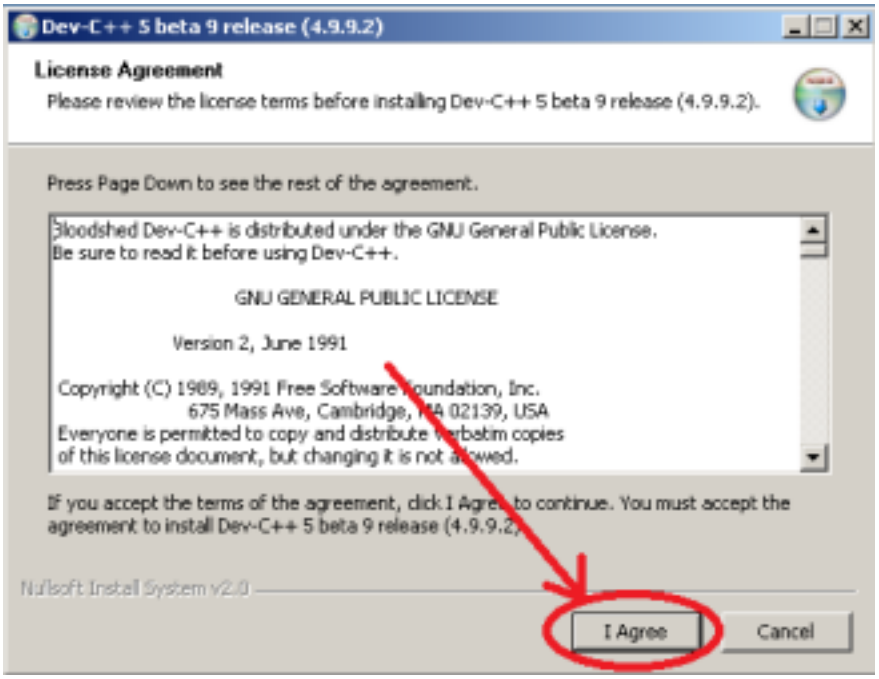
Dev-C++, developed by Bloodshed Software, is a fully featured graphical IDE (Integrated Development Environment), which is able to create Windows or console-based C/C++ programs using the MinGW compiler system. MinGW (Minimalist GNU* for Windows) uses GCC (the GNU g++ compiler collection), which is essentially the same compiler system that is in Cygwin (the unix environment program for Windows) and most versions of Linux.

Installation Steps:

1. Download the installer from the internet. Follow the instructions and install the program. The following screenshots will help you install and run the product:

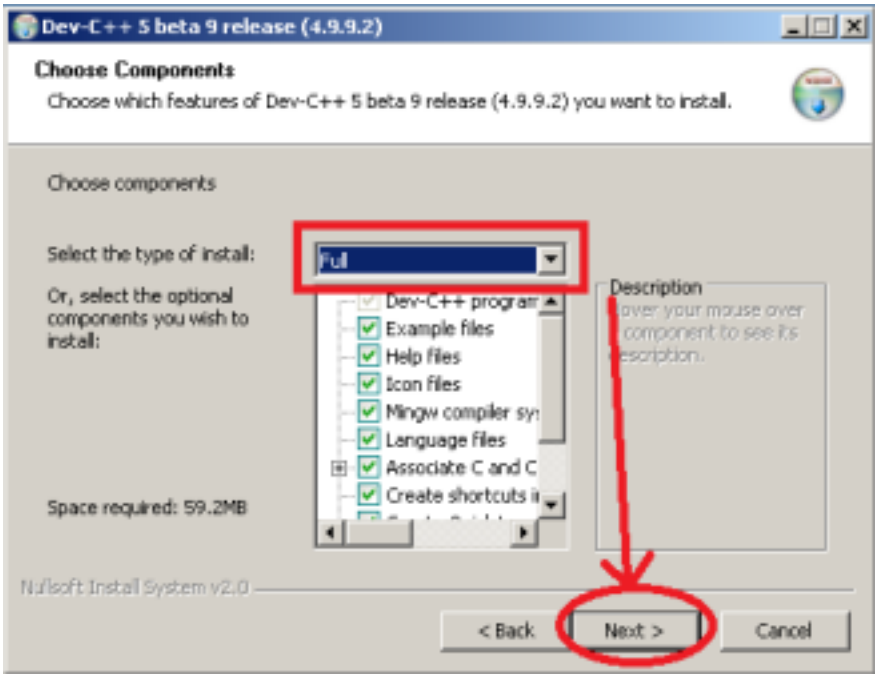


2. License Agreement
   Click on the "I Agree" button to continue

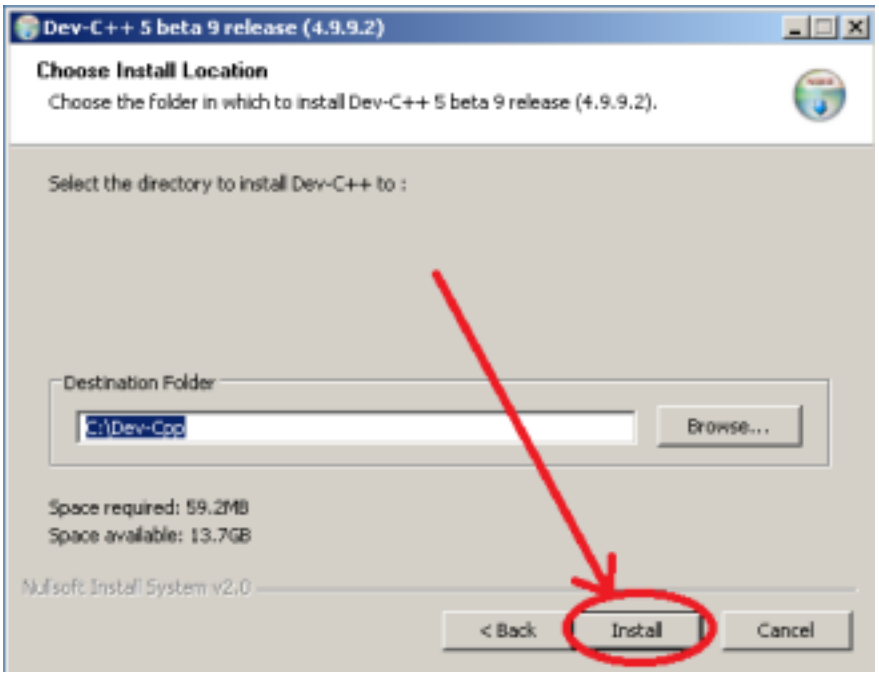


3. Choose Components

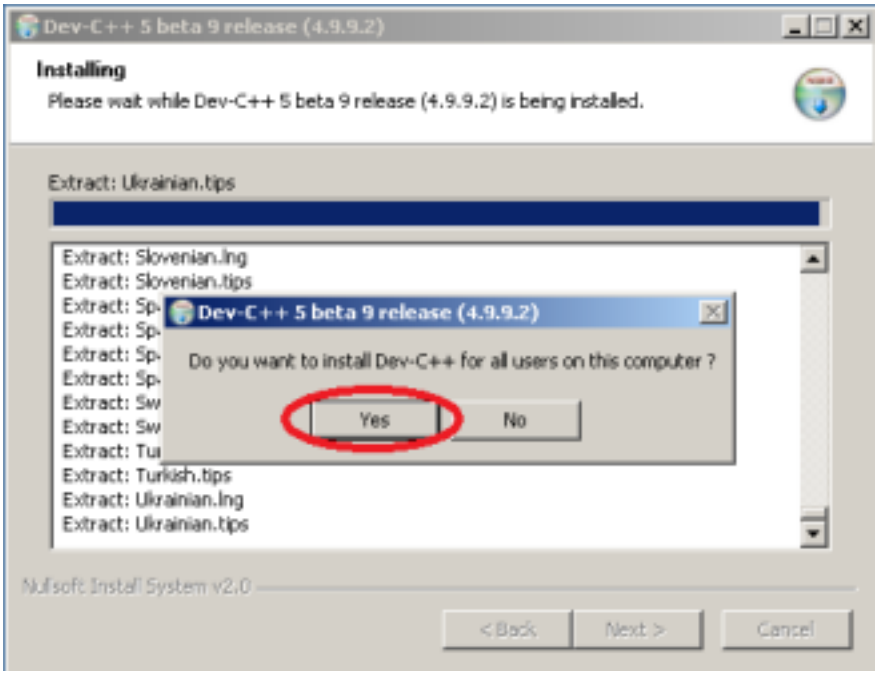
Make sure that the type of install is Full and click the Next button to continue



4. Choose Install Location
   Click the Install button to continue

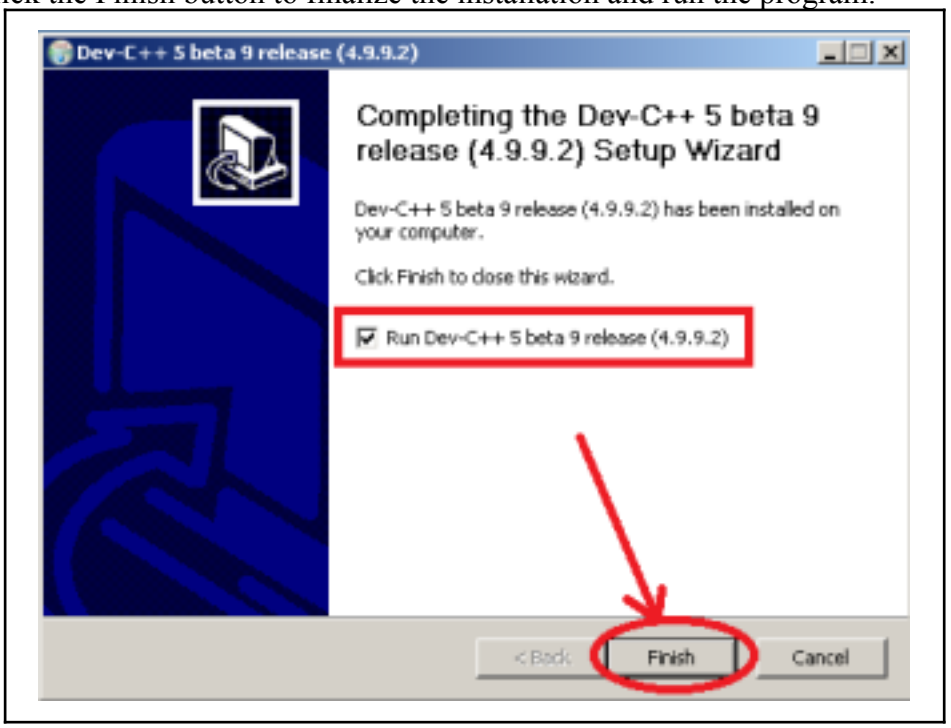


5. Installing
   Click the Yes button



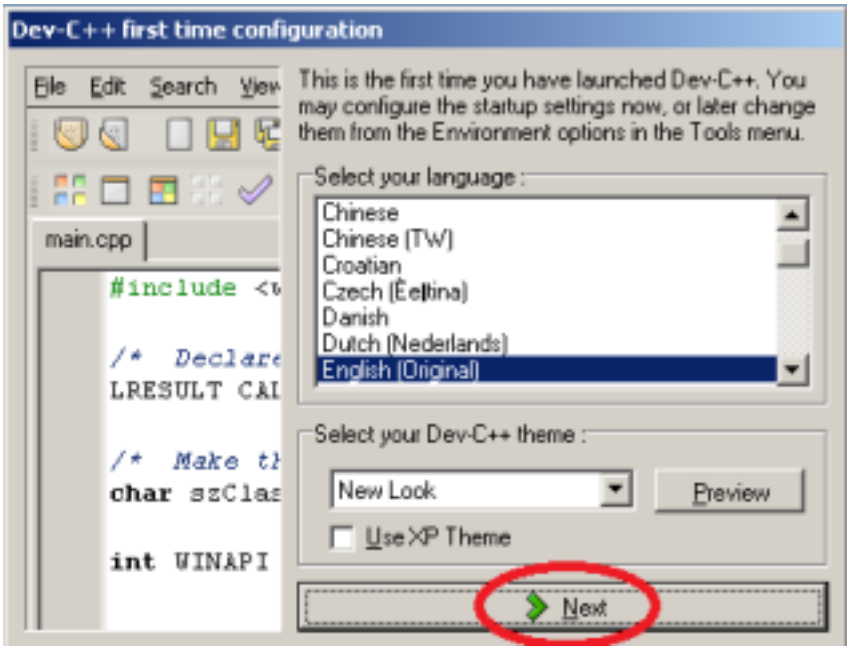
6. Finished

Click the Finish button to finalize the installation and run the program.

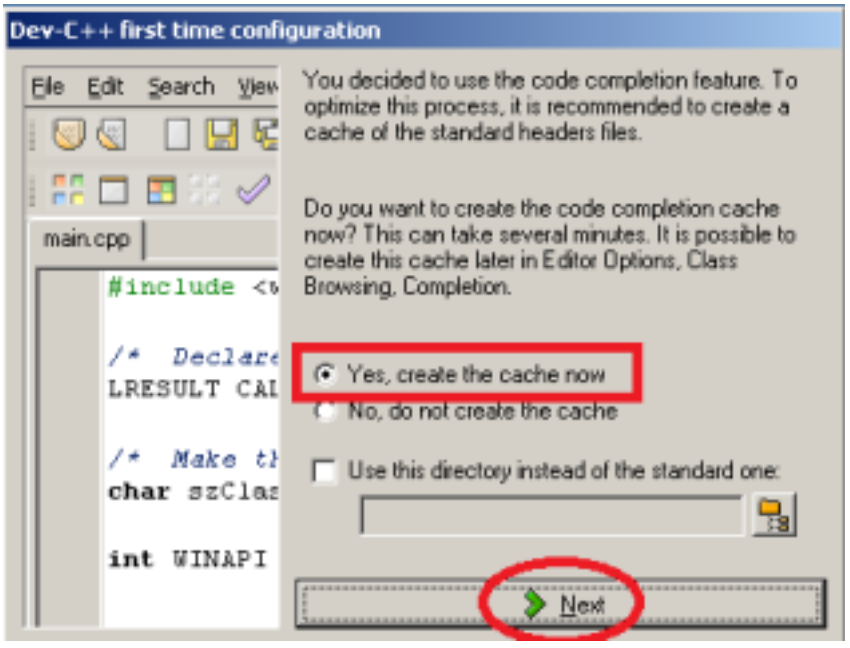


7. First Time Configuration

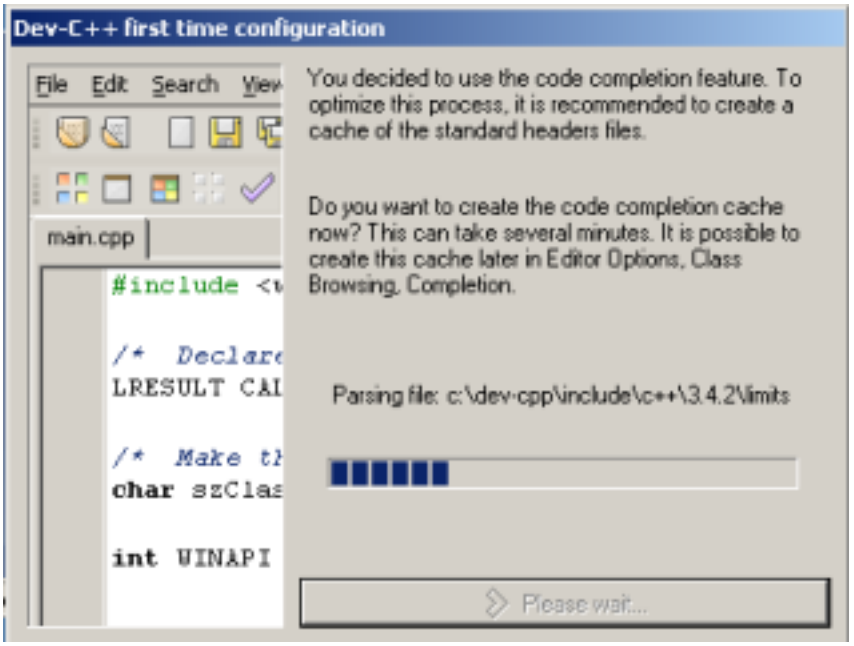
Click the Next button to continue



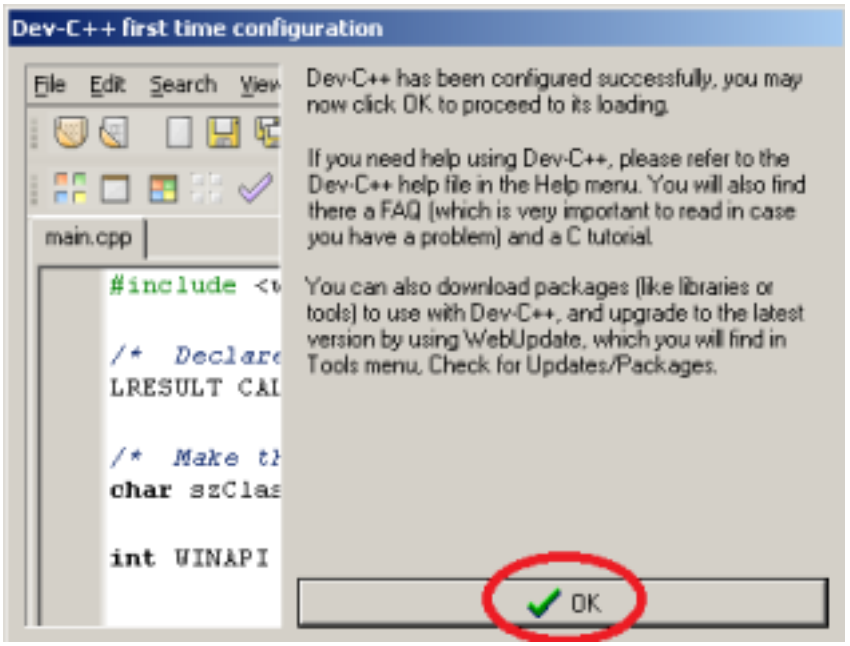
8. First Time Configuration
   Click the Next button to continue



9. First Time Configuration
   Wait for the Progress Bar to Complete

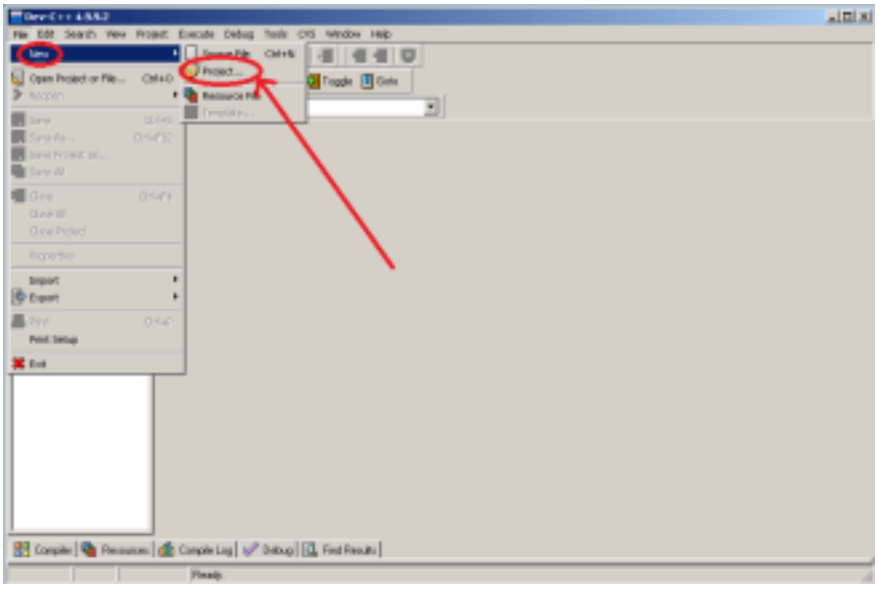


10. First Time Configuration
Click the OK button to Finalize



New Project Menu

Click the File menu, then select the New menu item and click the Project menu item.

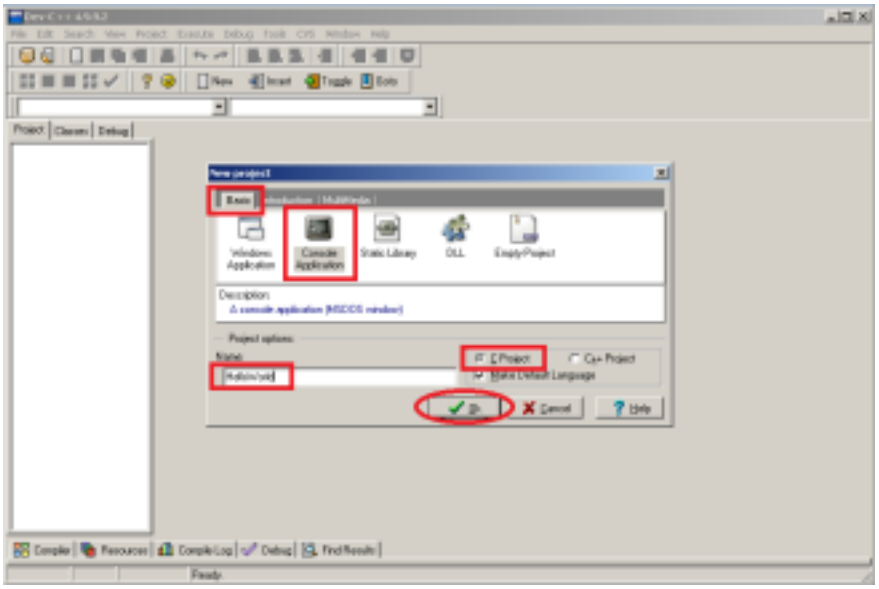


New Project

On the top, make sure that Basic tab is selected and under the Basic tab, select "Console Application"

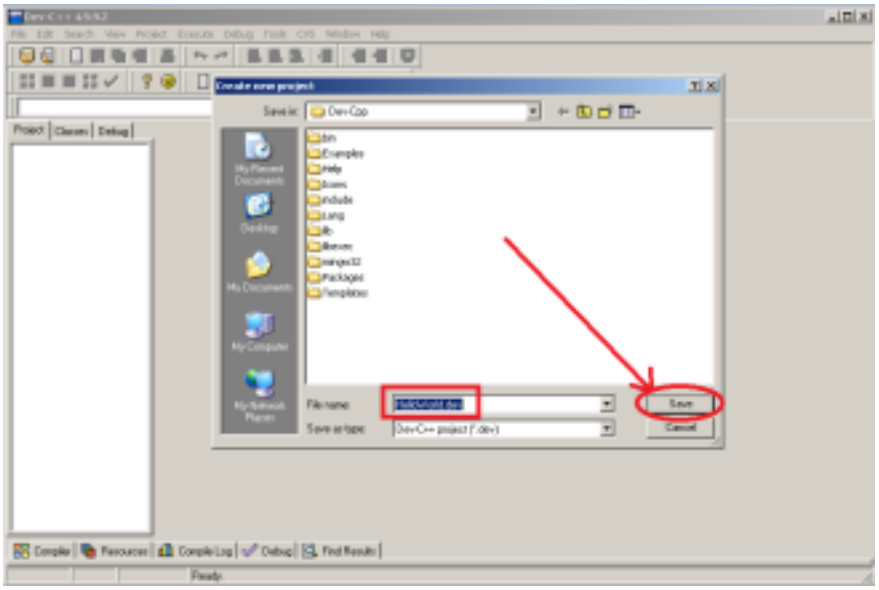
Give a name to your project using the Name text box, For instance, "Hello World". Important: Choose "C Project" under "Project Options", on the left

Click the OK button to create your project



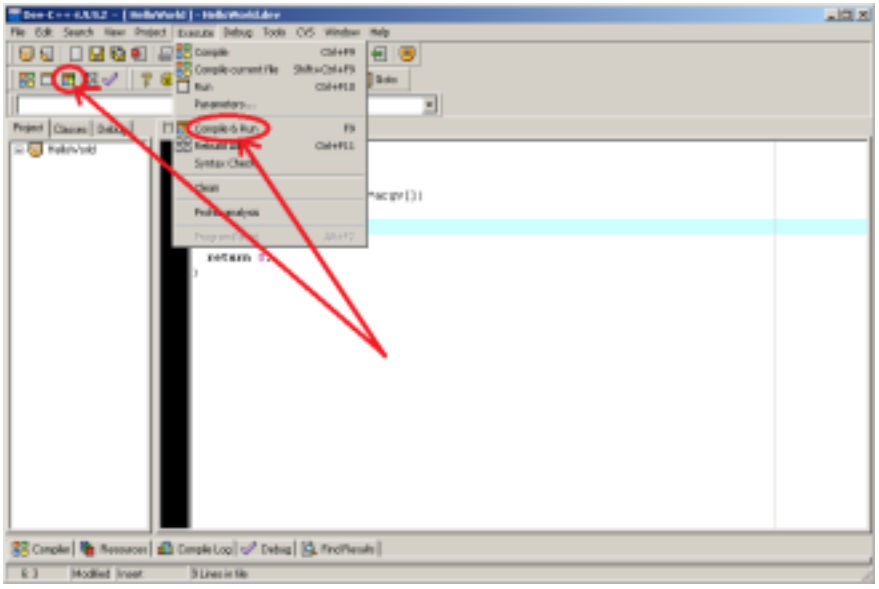
Create New Project

Give a name to your project file and click the Save button to continue



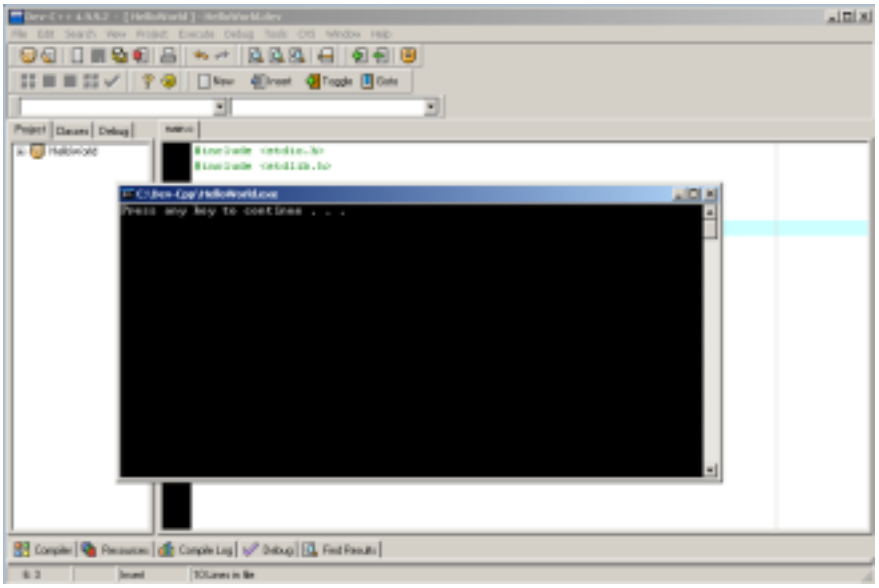
Compile & Run:

Click “Compile & Run” menu item or the icon displayed in the below screenshot or just Press F9 to compile and run your program.



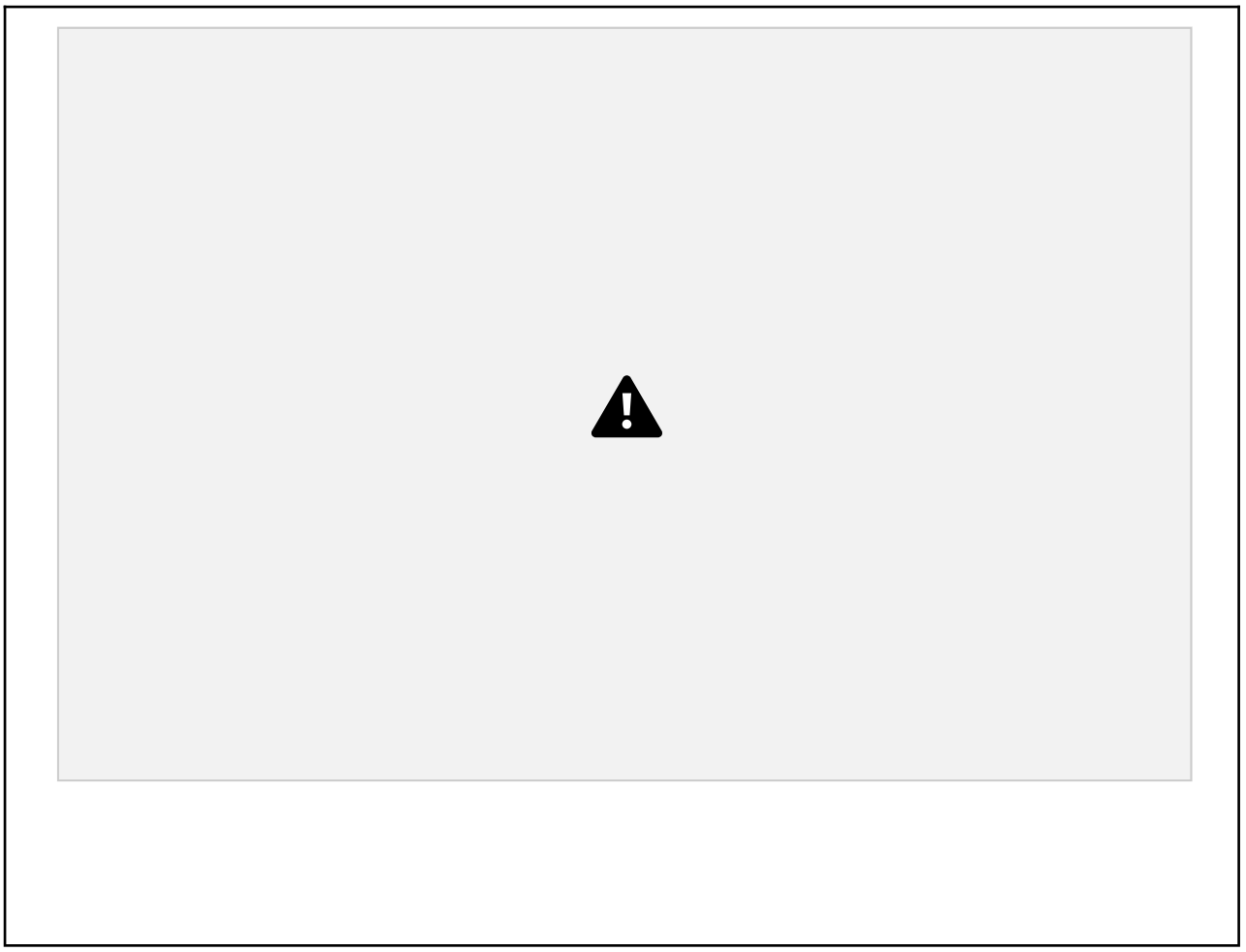
Running:

Assuming you did not make any syntax errors on your code, you should see a similar output window running your program.



Compile Failed

If you try to compile a code which has syntax errors, Compiler window lists the errors with their line numbers. You can double click the error and see the error highlighted in the code.



Review Questions/ Exercise:

1. Discuss the steps necessary to produce executable file?

Ans: Download DEV-C++/Visual studio code with MinGW (Minimalist GNU* for Windows) **as a compiler,after the installation create a file with "C" extension containing the C code.For example, hello.c**

Run the C preprocessor to handle directives(e.g., #include, #define).This step produces an intermediate file where marcos are expanded,and all headers are included.

The preprocessed output is typically not saved in a separate file unless specified._

2. Discuss the purpose of Compiler & the file needed by compiler?

Ans: A compiler is a specialized program that translates source code written in a high-level programming language (like C) into machine code.

Purpose of Compiler:

.Converts high-level code into low-level code.

.Improves the performance of the code by optimizing resource use (CPU, memory, etc.).

.Identifies syntax and semantic errors in the code during compilation.

.Produces an executable file or intermediate code that can be run on a computer.

.Allows programs to be compiled on different hardware architectures by generating machine-specific code.

File needed by Compiler:

1. Source Code File:

.Typically has a `.c` extension (e.g., `program.c`).

2.Header Files:

.Commonly have a `.h` extension (e.g., `stdio.h`, `stdlib.h`).

3.Makefile (optional):

.A file that defines how to compile and link a program, managing dependencies and build rules.

.Useful for larger projects with multiple source files.

4.Libraries:

.Precompiled code files (often with `.lib`, `.a`, or `.so` extensions) that the compiler links against to provide additional functionality.

---

3. Discuss the linker & the file needed by the linker?

Ans:In the C compilation process, the **linker** is a crucial tool that combines various pieces of code and libraries into a single executable file.

The Linkers primary tasks are:

1.Combining object files:

Links multiple object files (usually generated by the compiler) into one executable file.

2.Resolving symbols:

Resolves symbols (e.g., functions and variables) that are referenced in one object file but defined in another.

3.Address Binding: Assigns final memory addresses to the various program segments (code, data, etc.).

4.Library Linking:

Integrates external libraries, both static and dynamic, into the executable.

5.Creating Executable: Produces a final executable file that can be run on the target system.

Files Needed by the Linker:

1. **Object Files**:
    - Usually have a `.o` extension (e.g., `file1.o`, `file2.o`).
    - These are the compiled versions of the source code files (e.g., `program.c`).

2. **Library Files:**
    - Static libraries (e.g., `.a` files) or dynamic/shared libraries (e.g., `.so` on Linux, `.dll` on Windows).
    - Provide reusable code that can be linked into the program (e.g., standard libraries like `libc.a`).
3. **Linker Script (optional):**
    - A text file that specifies the memory layout and organization of the final executable.
    - Used for embedded systems or when specific memory management is needed.
4. **Executable File**:
    - The output of the linking process, usually with no specific extension on some systems (e.g., `program` or `program.exe` on Windows).
5. **Debugging Symbols (optional)**:
    - Files that may be generated during compilation that help with debugging (e.g., `.dSYM` on macOS, `.pdb` on Windows).

.