Faculty of Computing

CS110: Fundamentals of Computer Programming

Class: BESE-16B

Lab 01: Getting Started with C++ and IDE

CLO 1	**Understand** the syntax and semantics of different programming constructs
CLO 3	**Build** a program and associated documentation using appropriate IDE and

Date: 9th September, 2025

Time: 2:00pm-5:00pm

Instructor: Dr. Momina Moetesum
Lab Engineer: Mr. Nadeem Nawaz

Introduction:

This lab marks the starting point of hands-on programming in C++. Students will get familiar with the basic structure of a C++ program and explore the features of an Integrated Development Environment (IDE) already set up for them. The exercises are designed to build confidence in writing, running, and debugging simple programs while developing an understanding of syntax and semantics. Students will also practice documenting their code and observations, laying the groundwork for systematic programming practices throughout the course.

Objectives:

By the end of this lab, students will be able to:

1. Recognize the basic structure of a C++ program.
2. Navigate and utilize an IDE for creating, running, and debugging C++ programs.
3. Demonstrate understanding of fundamental C++ syntax and semantics through simple programs.
4. Identify and correct common syntax errors by interpreting compiler messages.
5. Document programs effectively using comments, error logs, and reflective notes.

Tools/Software Requirement:

Microsoft Visual Studio 2013 or above

Lab Tasks:

Visual Studio is a powerful developer tool that you can use to complete the entire development cycle in one place. It's a comprehensive integrated development environment (IDE) that you can use to write, edit, debug, and build code, and then deploy your app. Visual Studio includes compilers, code completion tools, source control, extensions, and many other features to enhance every stage of the software development process. This lab provides an overview of Visual Studio.

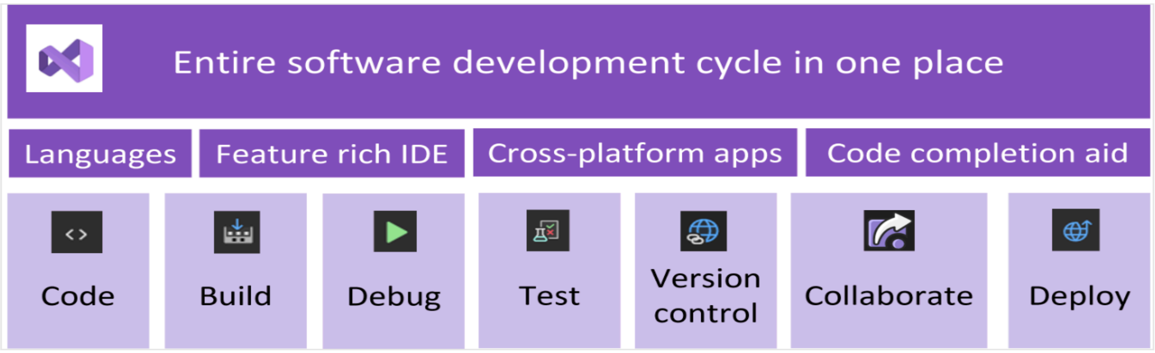


Image Source: https://learn.microsoft.com/en-gb/visualstudio/get-started/visual-studio-ide?view=vs-2022

Task 1 [CLO 3]: Practice creating a blank project in Visual Studio by following the given instructions.

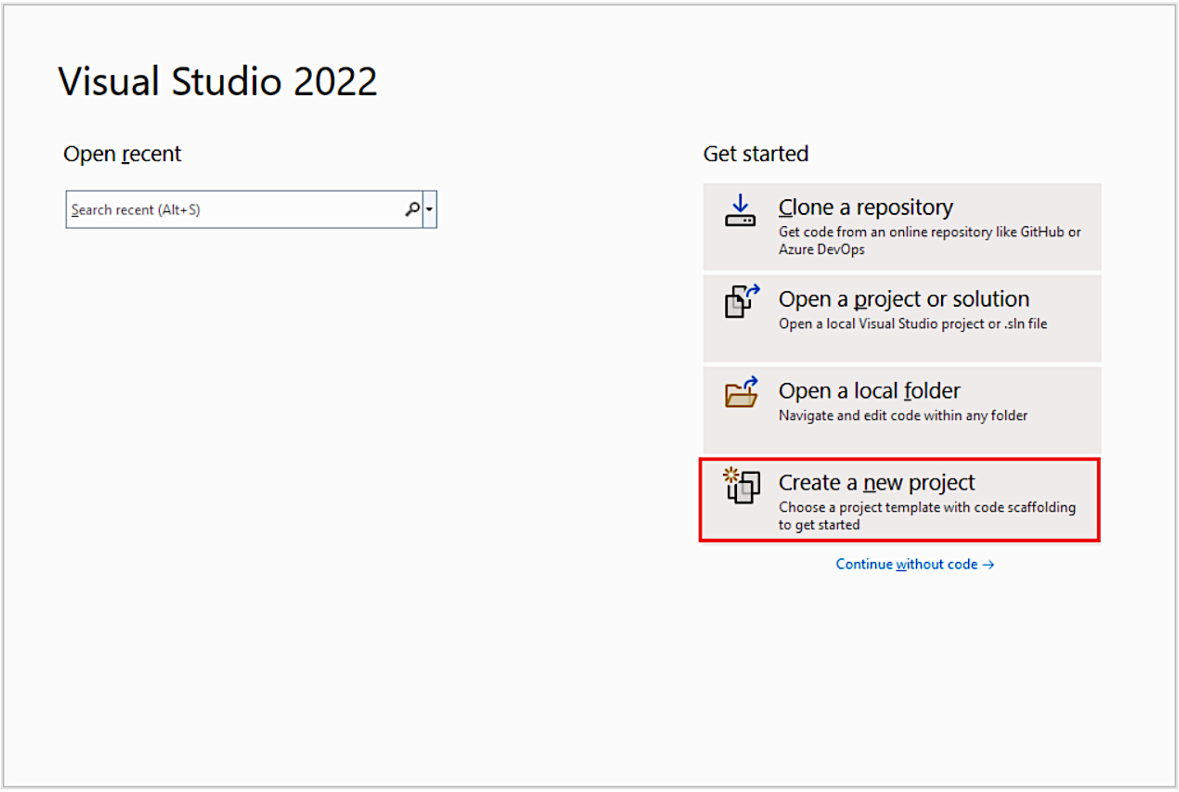
What is a "Project" and a "Solution" in Visual Studio?

Visual Studio uses ***projects*** to organize the code for an app, and ***solutions*** to organize one or more projects. A project contains all the options, configurations, and rules used to build an app. It also manages the relationship between all the project's files and any external files. To create your app, first, create a new project and solution.

In Visual Studio, a solution isn't an *answer*. A solution is simply a container that Visual Studio uses to organize one or more related projects. When you open a solution, Visual Studio automatically loads all the projects that the solution contains.

How to create a Project in Visual Studio (Version 2022):

1. Start Visual Studio--the Visual Studio Start dialog box appears. Select **Create a new project** to get started (Fig. 1.1).



Visual Studio 2022

Open recent

Search recent (Alt+S)

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

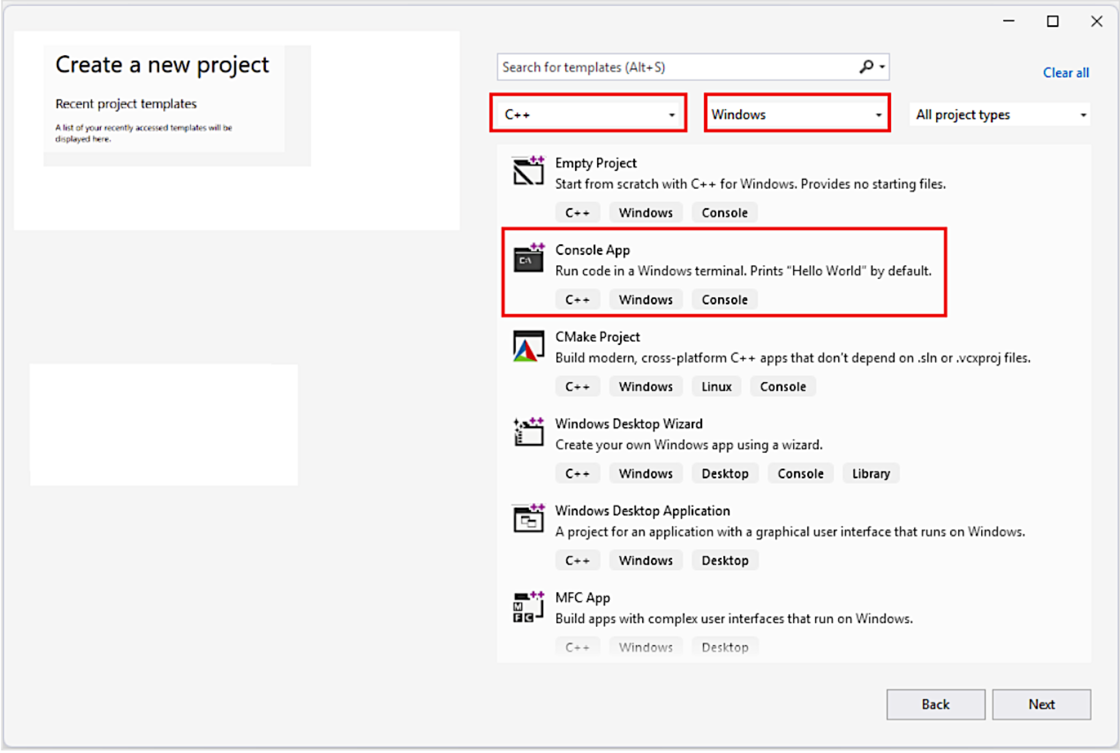
Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Continue without code →

Figure 1.1: Step 1: Select Create a new project

2. In the **Create a new project** dialog, set the language dropdown to **C++**, set the platform dropdown to **Windows**, select **Console App** from the list of project types, then select **Next** (Fig. 1.2).



Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S)

Clear all

C++ | Windows | All project types

Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ Windows Console

Console App
Run code in a Windows terminal. Prints "Hello World" by default.
C++ Windows Console

CMake Project
Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.
C++ Windows Linux Console

Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library

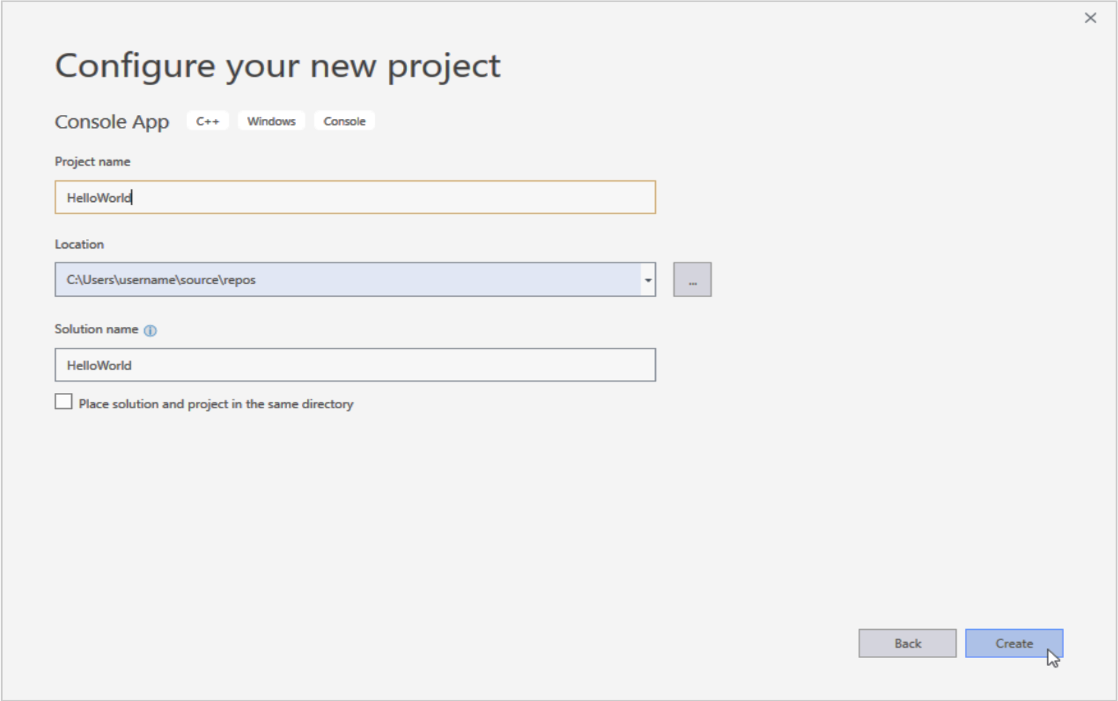
Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.
C++ Windows Desktop

MFC App
Build apps with complex user interfaces that run on Windows.
C++ Windows Desktop

Back Next

Figure 2.2: Step 2: Select Console APP

In the **Configure your new project** dialog box, select the **Project name** text box, name your new project *"HelloWorld"*, then select **Create** (Fig. 1.3). Visual Studio creates a new project. It's ready for you to add and edit your source code.



Configure your new project

Console App C++ Windows Console

Project name

HelloWorld

Location

C:\Users\username\source\repos

Solution name

HelloWorld

□ Place solution and project in the same directory

Back Create

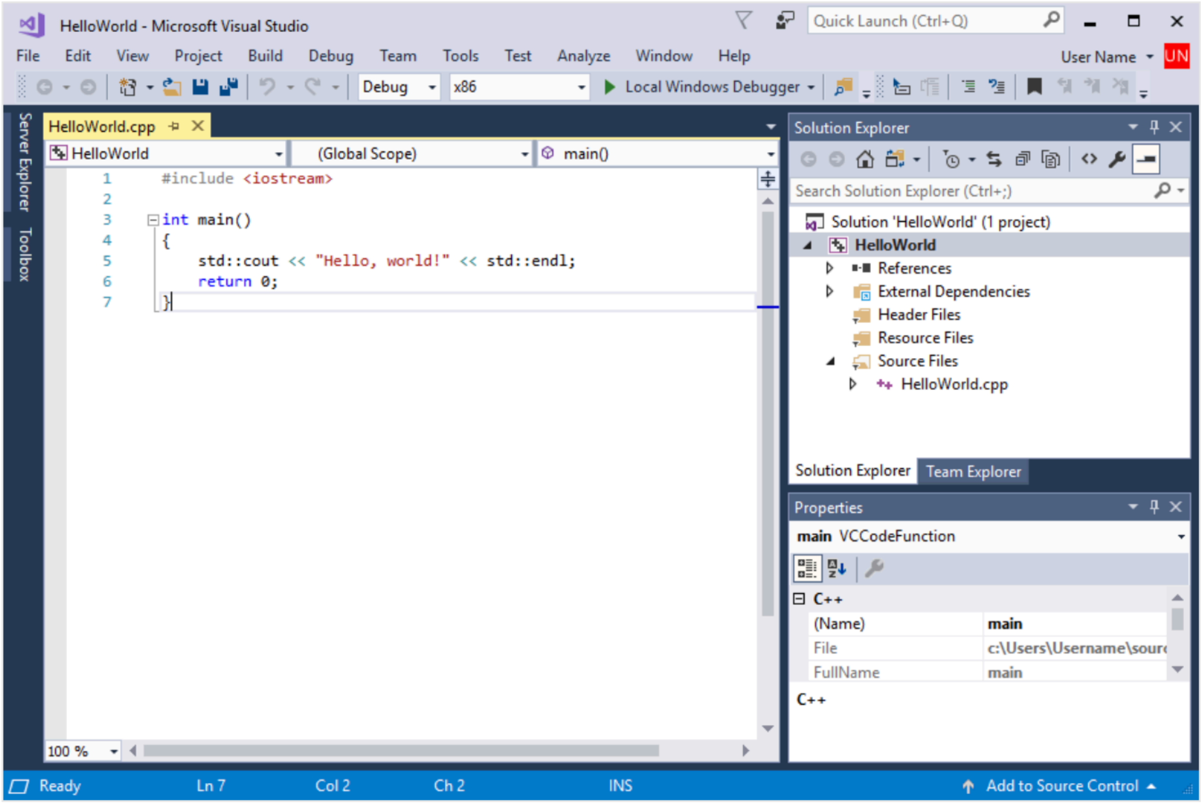
Figure 3.3: Step 3: Select Project Name

3. When you create a new project, a solution file (*.sln*) is also created. You can add another project to the solution by right-clicking the solution name in **Solution Explorer** > **Add** > **New project**.

4. Create a word document and add step-wise screenshots of your project.

Task 2 [CLO 1]: Understand the syntax and structure of a basic C++ program by building and running a simple 'Hello World' program.

When you create a project, an empty C++ Windows console application 'Hello World' app is created by default (Fig. 1.4). By default, it contains code to print Hello World!



```
#include <iostream>

int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Figure 4.4: Hello World App Code

1. To build your project, from the main menu choose **Build** > **Build Solution** (Fig. 1.5).

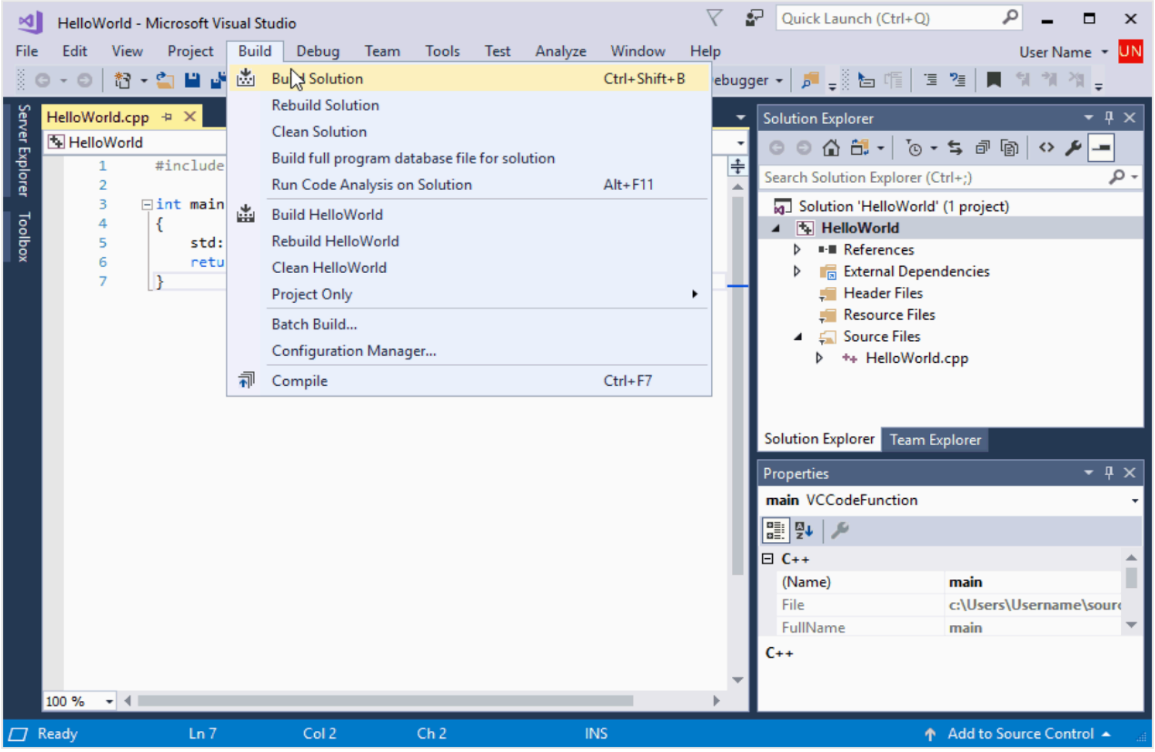
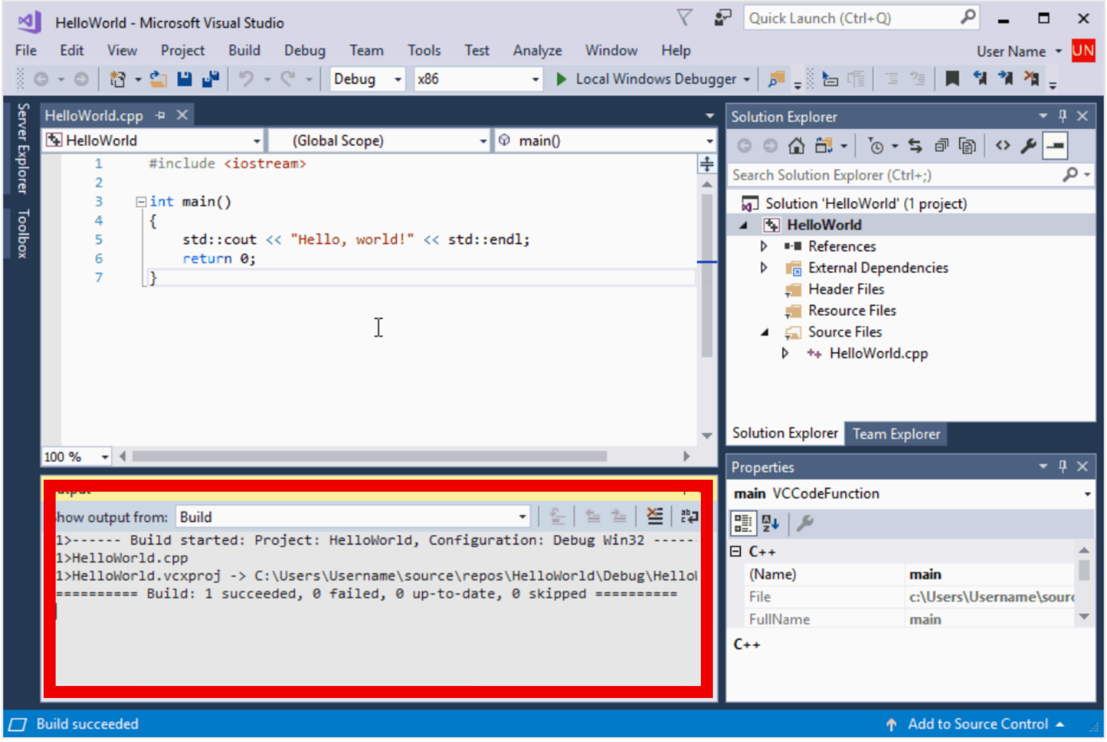


Figure 5.5: To Build Solution

2. The **Output** window shows the results of the build process (Fig. 1.6).



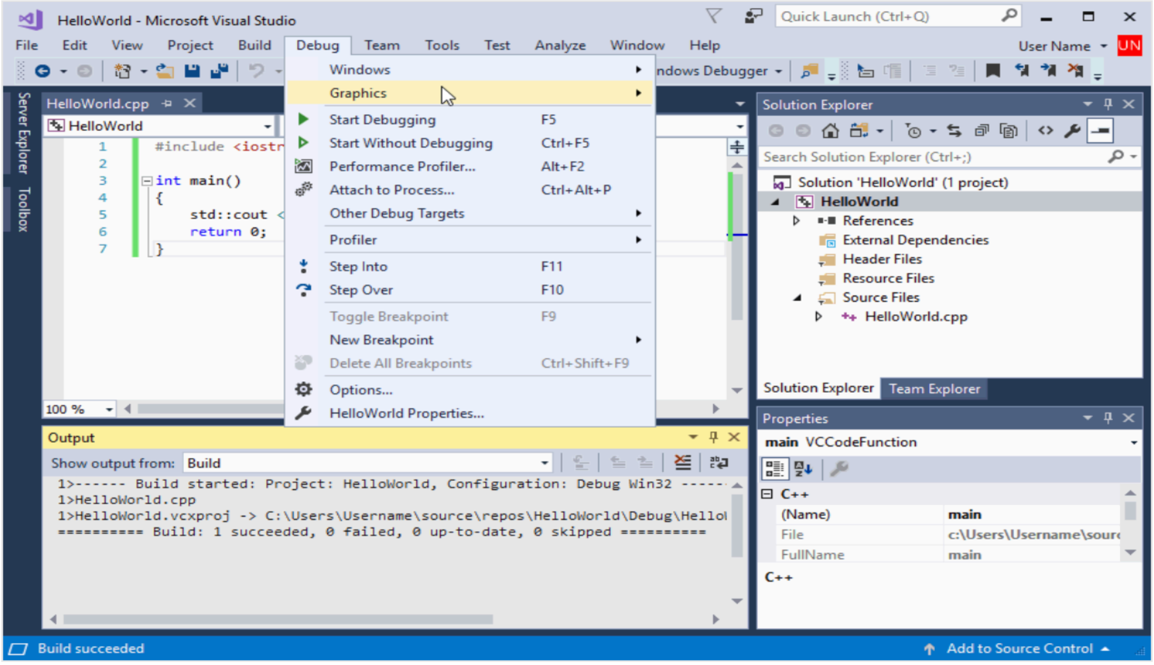
```
#include <iostream>

int main()
{
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Show output from: Build

```
1>------ Build started: Project: HelloWorld, Configuration: Debug Win32 -----
1>HelloWorld.cpp
1>HelloWorld.vcxproj -> C:\Users\Username\source\repos\HelloWorld\Debug\HelloW
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Figure 6.6: Build Results

3. To run the code, on the menu bar, choose **Debug**, **Start without debugging (Fig. 1.7)**.

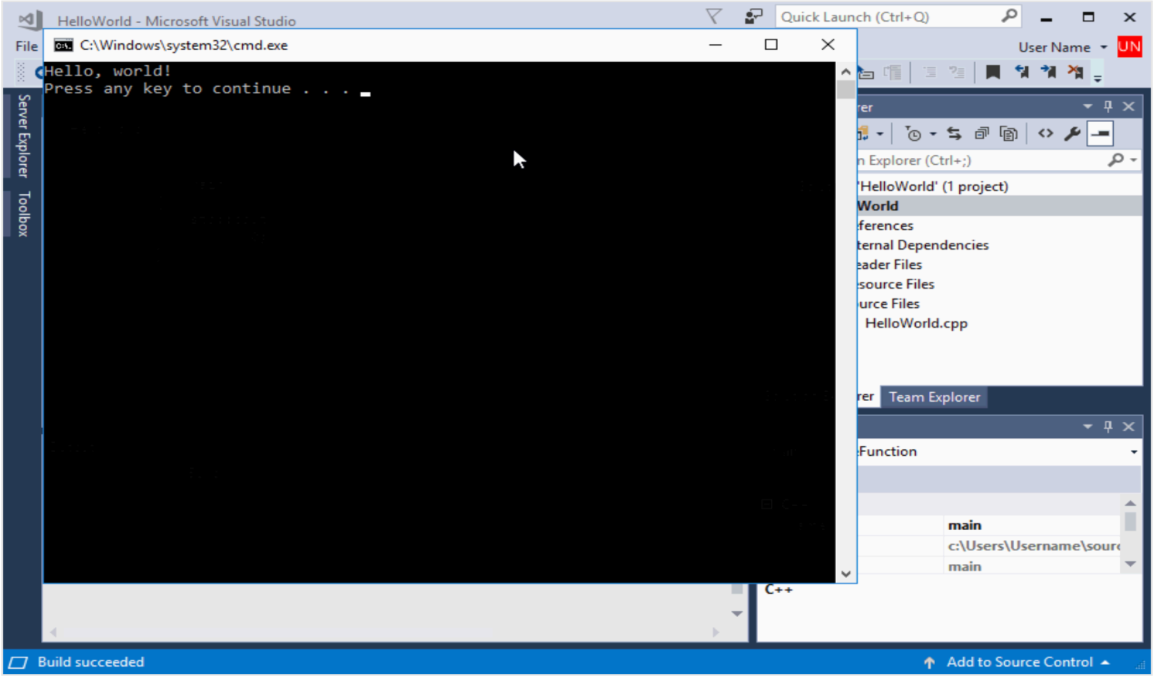


```
#include <iostr
int main()
{
    std::cout <
    return 0;
}
```

```
1>------ Build started: Project: HelloWorld, Configuration: Debug Win32 -----
1>HelloWorld.cpp
1>HelloWorld.vcxproj -> C:\Users\Username\source\repos\HelloWorld\Debug\HelloW
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Figure 7.7: Build Results

4. Console applications use a Windows console window to display output and accept user input (Fig. 1.8).



```
Hello, world!
Press any key to continue . . .
```

Figure 8.8: Console Output

5. Take a screenshot of the IDE output after successful execution.

Task 3 [CLO 1]: Understand the syntax errors and compiler error handling.

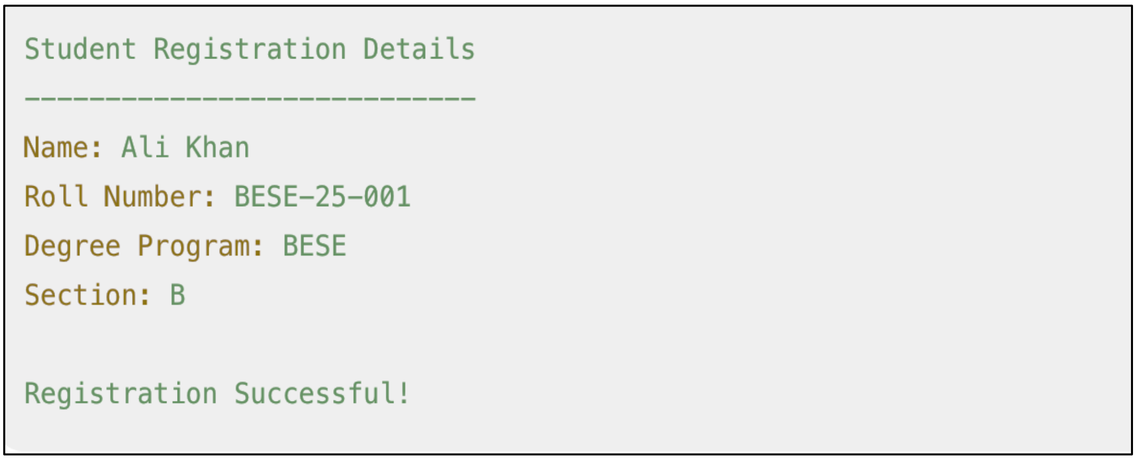
1. Introduce following syntax errors:

    a. Remove a semicolon
    b. Remove a matching {}
    c. Misspell a keyword

2. Observe compiler error messages and record them in an error log table (Error → Reason → Correction).

Task 4 [CLO 3]: Practice using IDE by creating a Student Information Display Program.

1. Using the IDE, create a new C++ project and develop a program that displays student registration details (Name, Qalam ID, Degree Program, Section) using `cout`.
2. The details can be hardcoded in the program.
3. Ensure the program is **well-formatted and commented**.
4. Save both the **source code** and the **execution screenshot** in your lab report.
5. Expected output is as follows:



```
Student Registration Details
----------------------------
Name: Ali Khan
Roll Number: BESE-25-001
Degree Program: BESE
Section: B

Registration Successful!
```

Figure 9.9: Console Output

Lab Rubrics:

Your Lab 1 will be graded out of 5 for each rubric according to the following rubrics.

Lab Rubrics for Lab 1

Sr. No.	Assessment	Unacceptable (0 Marks)	Does Not Meet Expectations (1/2 Marks)	Meets Expectations (3/4 Marks)	Exceeds Expectations (5 Marks)
1	**Illustrating the basic understanding of semantics and syntax (CLO1, PLO1)**	The student did not submit any work. OR The student plagiarized the solution and/or used unfair means.	The student is unable to demonstrate the understanding of syntax of C language and is unable to write an executable code. The student is not able to understand the structure of a program at all.	The student demonstrates some understanding of syntax of C language and is able to write a code with few errors. The student is able to understand the structure but still learning the syntax.	The student demonstrates good understanding of syntax of C language and is able to write executable code without help The student is able to understand the structure and is able to identify problems in the code when introduced
2	**Software Tool Usage (CLO3-PLO5)**		The student demonstrates a lack of understanding of tool usage. Implementation has syntax/semantic/runtime errors, and the student is unable to debug and correct the errors. The code has inadequate comments and variable names and does not adhere to the coding standards. No Error handling has been performed. Documentation is poorly structured.	The student demonstrates some understanding of tool usage. The codes are correct in terms of their syntax, however, the program output is not always correct in all test cases. The code has limited comments and inconsistent variable names and may not adhere to the coding standards. Some Error handling has been performed. Documentation is adequately structured.	The student demonstrates a good understanding of tool usage. Furthermore, his/her coding is complete and functional, and the program output is correct in all test cases. The code has sufficient comments and consistent variable names and reasonably adhere to the coding standards. Adequate Error handling has been performed. Documentation is well structured.