Table of Contents

Introduction

This document aims to provide readers with fundamental knowledges needed to develop a console application using C++ programming language that involves data storage and manipulation including all CRUD (Create, Read, Update, Delete) process.

DISCLAIMER: Section 2 and 3 is not step by step development guide but is an explanation on the process and the logic behind the code. Refer to GitHub repository for a complete example.

Resources:

- Visual studio: https://visualstudio.microsoft.com/downloads/
- XAMPP: https://www.apachefriends.org/download.html
- MySQL Connector: https://dev.mysql.com/downloads/connector/cpp/
- Complete example is available on: https://github.com/amnxqid/Workshop1-Demo-

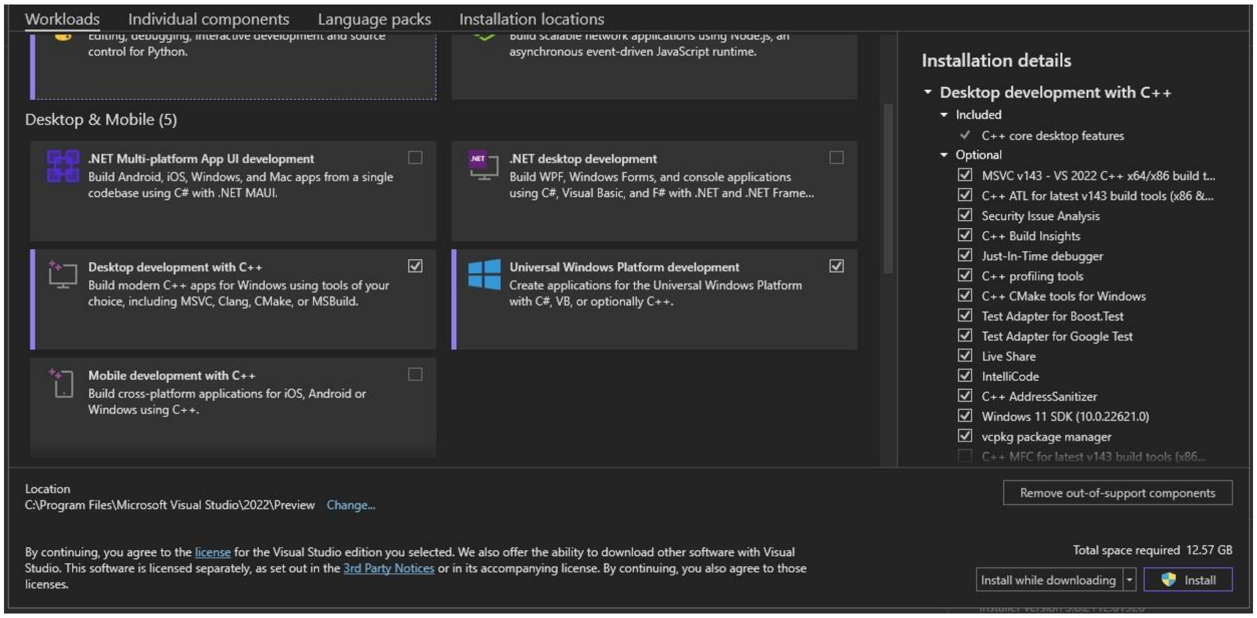
1. Project Setup

This section will explain the installation and configuration of the necessary software which will be required and used to develop the console application.

1.1. Visual Studio

Throughout this guide, the Microsoft Visual Studio IDE will be used to program, compile and run the C++ program. You can install the IDE by downloading the Visual Studio Installer through this link https://visualstudio.microsoft.com/downloads/ (15/10/2023) and select the Community Version which is free for students and non-profit uses.

Open the downloaded visual studio installer and go to the available tab and install Visual Studio Community and **ensure that you have the necessary components (Desktop Development with C++, Universal Windows Platform Development) selected** in the workload tab.



Editing, debugging, interactive development and source control for Python.

Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

.NET Multi-platform App UI development
Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.

.NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...

[✓] Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

[✓] Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.

Installation details

- Desktop development with C++
  - Included
    - ✓ C++ core desktop features
  - Optional
    - ✓ MSVC v143 - VS 2022 C++ x64/x86 build t...
    - ✓ C++ ATL for latest v143 build tools (x86 &...
    - ✓ Security Issue Analysis
    - ✓ C++ Build Insights
    - ✓ Just-In-Time debugger
    - ✓ C++ profiling tools
    - ✓ C++ CMake tools for Windows
    - ✓ Test Adapter for Boost.Test
    - ✓ Test Adapter for Google Test
    - ✓ Live Share
    - ✓ IntelliCode
    - ✓ C++ AddressSanitizer
    - ✓ Windows 11 SDK (10.0.22621.0)
    - ✓ vcpkg package manager
    - C++ MFC for latest v143 build tools (x86...

Location
C:\Program Files\Microsoft Visual Studio\2022\Preview

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 12.57 GB

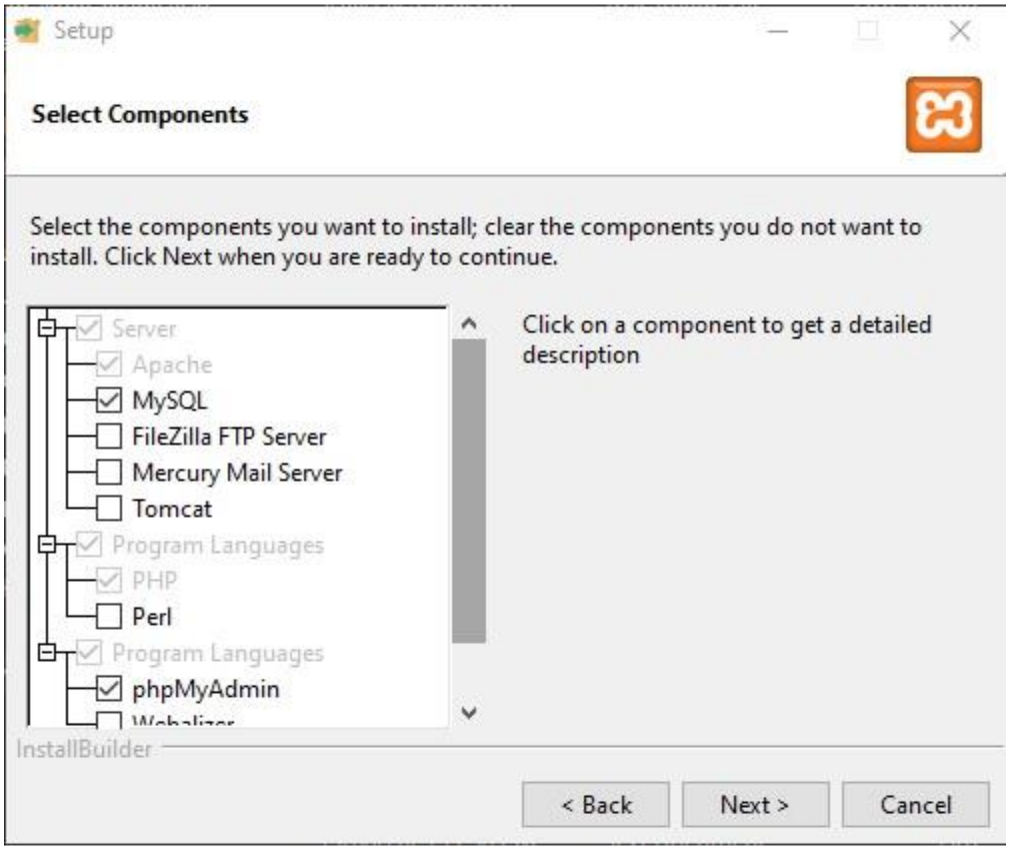
Other installation parameters, path, language etc. is up to your personal preferences. Proceed with the installation and wait until it finishes.

1.2. MySQL Database

As mentioned previously, the system to be developed throughout this guide will store and manipulate data. These data will have to be stored in a RDBMS (Relational Database Management System). There are variety of database option that can be used, in this case the MySQL or more specifically its variation MariaDB will be used.

1.2.1. XAMPP

XAMPP stands for X-Operating System, Apache, MySQL, PHP and Perl. It is an open-source web server solution package which consists of multiple available components. In short it includes multiple services. However, to develop a simple C++ console application, we will be only using two of the services which is the Apache and MySQL. You can get the installer for XAMPP from Apache official page https://www.apachefriends.org/download.html (15/10/2023). Download the installer and follow the instructions to install it. **You may install all component if you want but ensure that the following component (MySQL and phpMyAdmin) is selected** since they are compulsory to have for this project. Refer to https://www.youtube.com/watch?v=f8N4FEQWyY&ab_channel=edureka%21 for a video guide on XAMPP installation if necessary.

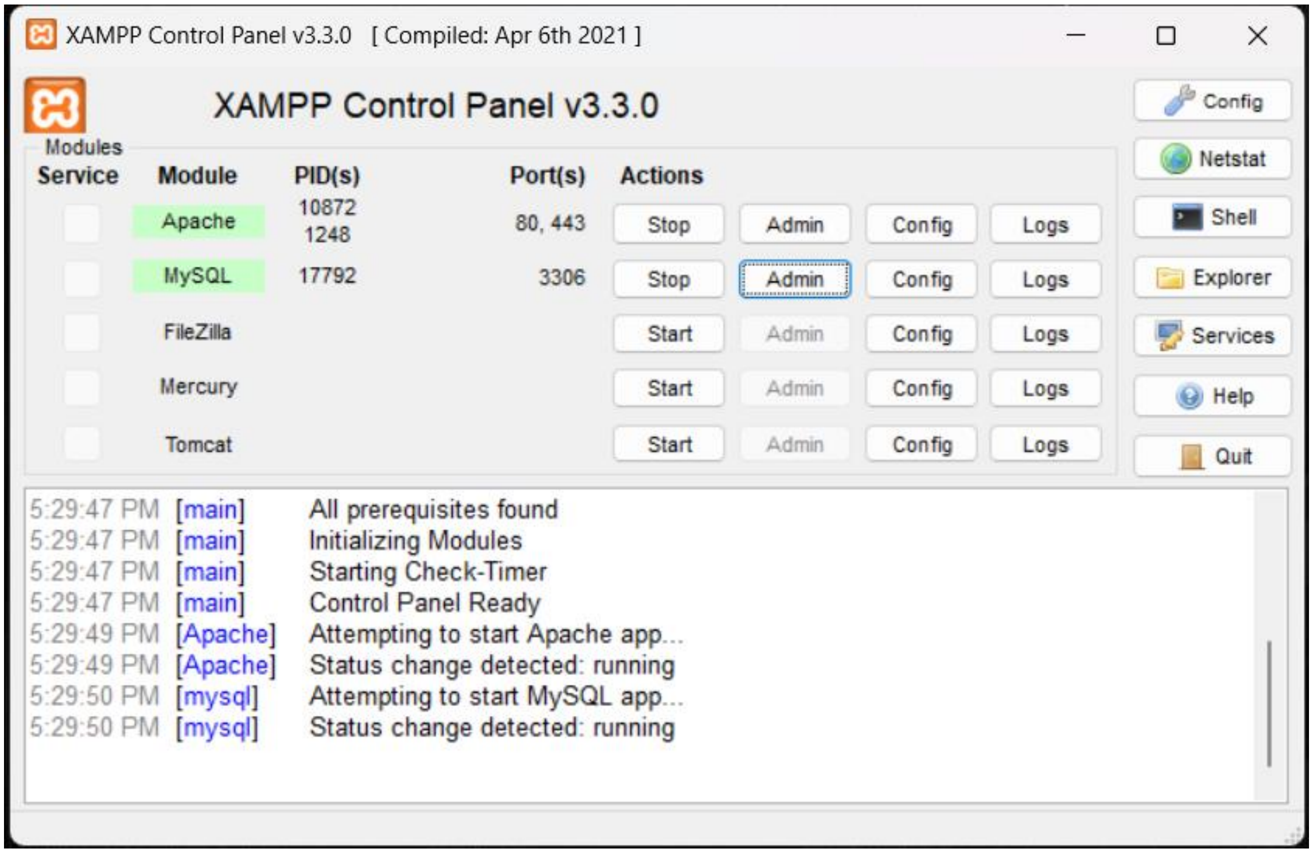


Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Server
  - [✓] Apache
  - [✓] MySQL
  - [ ] FileZilla FTP Server
  - [ ] Mercury Mail Server
  - [ ] Tomcat
- [✓] Program Languages
  - [✓] PHP
  - [ ] Perl
- [✓] Program Languages
  - [✓] phpMyAdmin
  - [ ] Webalizer

1.2.2. MySQL

Now that you have installed XAMPP and the necessary component in the previous section, you can start your database through the XAMPP control panel.


XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	10872 1248	80, 443	Stop	Admin	Config	Logs
	MySQL	17792	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

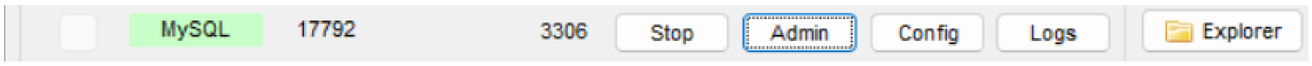
```
5:29:47 PM  [main]     All prerequisites found
5:29:47 PM  [main]     Initializing Modules
5:29:47 PM  [main]     Starting Check-Timer
5:29:47 PM  [main]     Control Panel Ready
5:29:49 PM  [Apache]   Attempting to start Apache app...
5:29:49 PM  [Apache]   Status change detected: running
5:29:50 PM  [mysql]    Attempting to start MySQL app...
5:29:50 PM  [mysql]    Status change detected: running
```

Again, there are multiple services provided by XAMPP. However, the most important service is MySQL which will be your database. Currently your database server will be running in your local server (localhost). You need to ensure that your MySQL is running whenever you want to run your C++ application later since it will need to connect to the MySQL server to perform data operations. Take note of the Port number being used by your MySQL, by default it should be 3306 unless you changed it yourself which you must use later in your C++ codes.

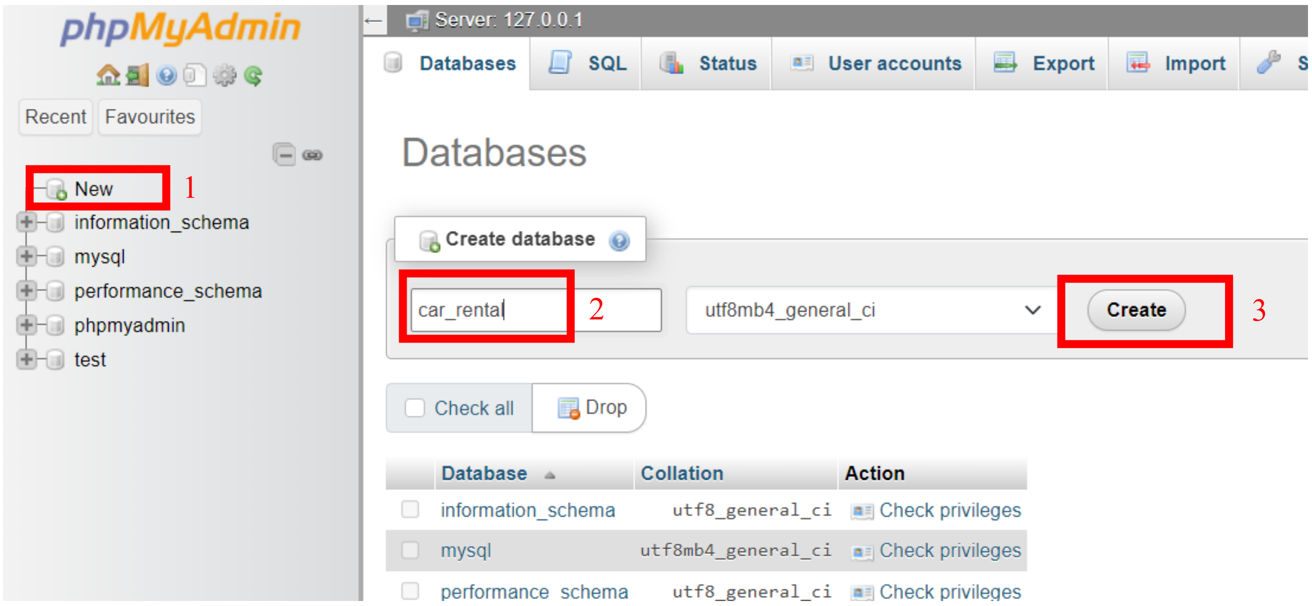
1.2.3. PHPMyAdmin

PHPMyAdmin is part of the XAMPP which provide user with GUI to manage their database easily instead of using CLI (Command Line Interface). In this guide we will only show basic usage of PHPMyAdmin to import the demo database used in the example project. You are advised to not to rely too much on the convenience of PHPMyAdmin which omits lot of manual queries. Try to understand the process of manipulating and managing database via queries.

To run PHPMyAdmin you must ensure that Apache service is running in your XAMPP control panel since PHPMyAdmin is basically a php web application that runs on the Apache server locally. Then, click the admin button in the MySQL row in the control panel to start it.

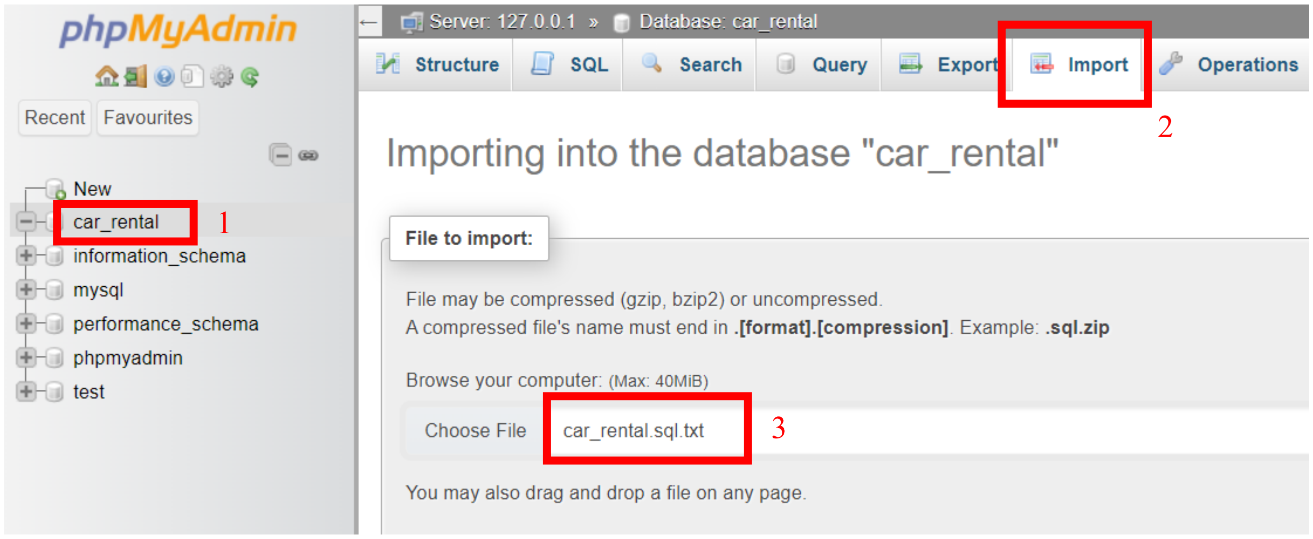


The following diagram shows the interface of PHPMyAdmin. On the left-hand side, you can see the list of databases already created on your MySQL. To create a new database simply click new (1) and then fill in the name (2) and click create (3).



Database	Collation	Action
information_schema	utf8_general_ci	Check privileges
mysql	utf8mb4_general_ci	Check privileges
performance_schema	utf8_general_ci	Check privileges

Then import the SQL file provided, by selecting your database, go to import and locate the .sql file. Finally scroll down and find the import button.



Importing into the database "car_rental"

File to import:

File may be compressed (gzip, bzip2) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: (Max: 40MiB)

You may also drag and drop a file on any page.

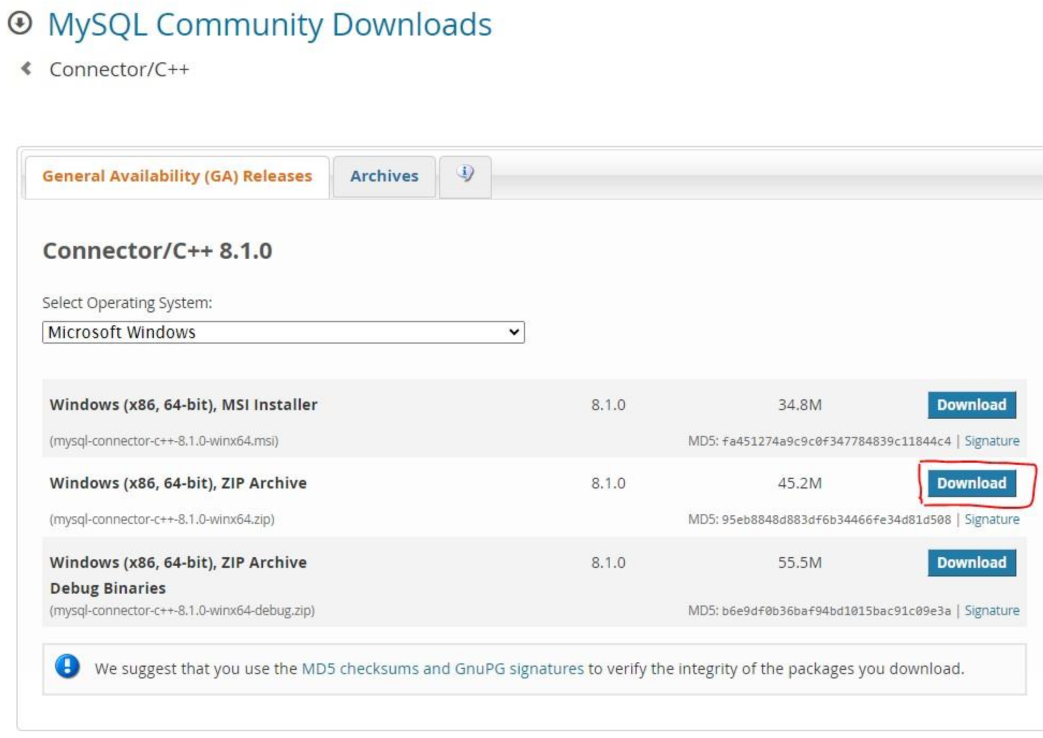
Scroll down and click the import button. After you have imported the database, you can now proceed to follow through this guide.

1.3. MySQL Connector

After following the steps in 1.1 and 1.2, you should now have a Visual Studio Installed and XAMPP with its necessary components. This section will explain how to connect a C++ application with your MySQL database. The content of this section is similar to the explanation in https://www.youtube.com/watch?v=QsKnRk1gzxM&ab_channel=AmirulAsraf. Thus, you may refer to the video if necessary but do note that the video was recorded, and the version shown might not be same with what you will see.

C++ by default does not support interacting with MySQL database. Thus, we need to get an additional library which can fulfil this purpose. There is multiple option of libraries for C++ which allows database interaction which is mysql.h and others but in this guide, we will be using mysql/jdbc.h. The following are the step by steps explanation on how to download and import the library into your project.

1. Go to https://dev.mysql.com/downloads/connector/cpp/ or you can google MySQL C++ connector. Choose your operating system and download the connector file in ZIP archive.


MySQL Community Downloads

Connector/C++

General Availability (GA) Releases | Archives

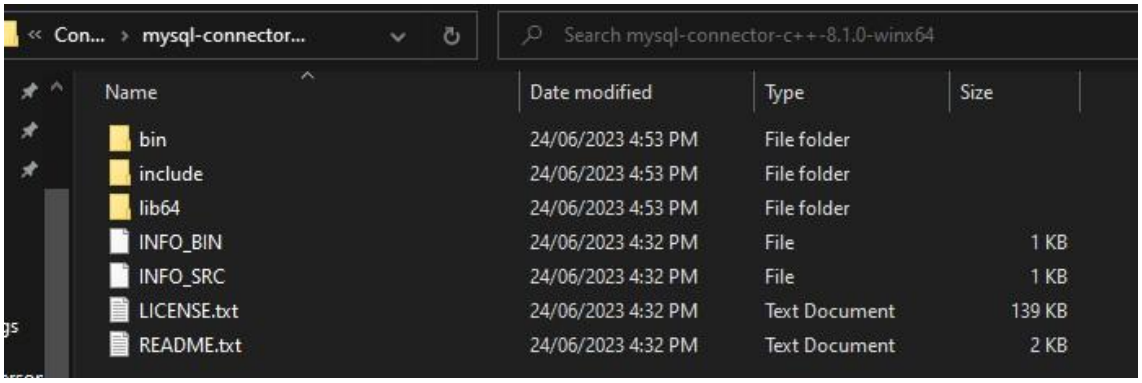
Connector/C++ 8.1.0

Select Operating System:
Microsoft Windows

Windows (x86, 64-bit), MSI Installer (mysql-connector-c++-8.1.0-winx64.msi)	8.1.0	34.8M MD5: fa451274a9c9c0f347784839c11844c4 | Signature	Download
Windows (x86, 64-bit), ZIP Archive (mysql-connector-c++-8.1.0-winx64.zip)	8.1.0	45.2M MD5: 95eb8848d883df6b34466fe34d81d508 | Signature	Download
Windows (x86, 64-bit), ZIP Archive Debug Binaries (mysql-connector-c++-8.1.0-winx64-debug.zip)	8.1.0	55.5M MD5: b6e9df0b36baf94bd1015bac91c09e3a | Signature	Download

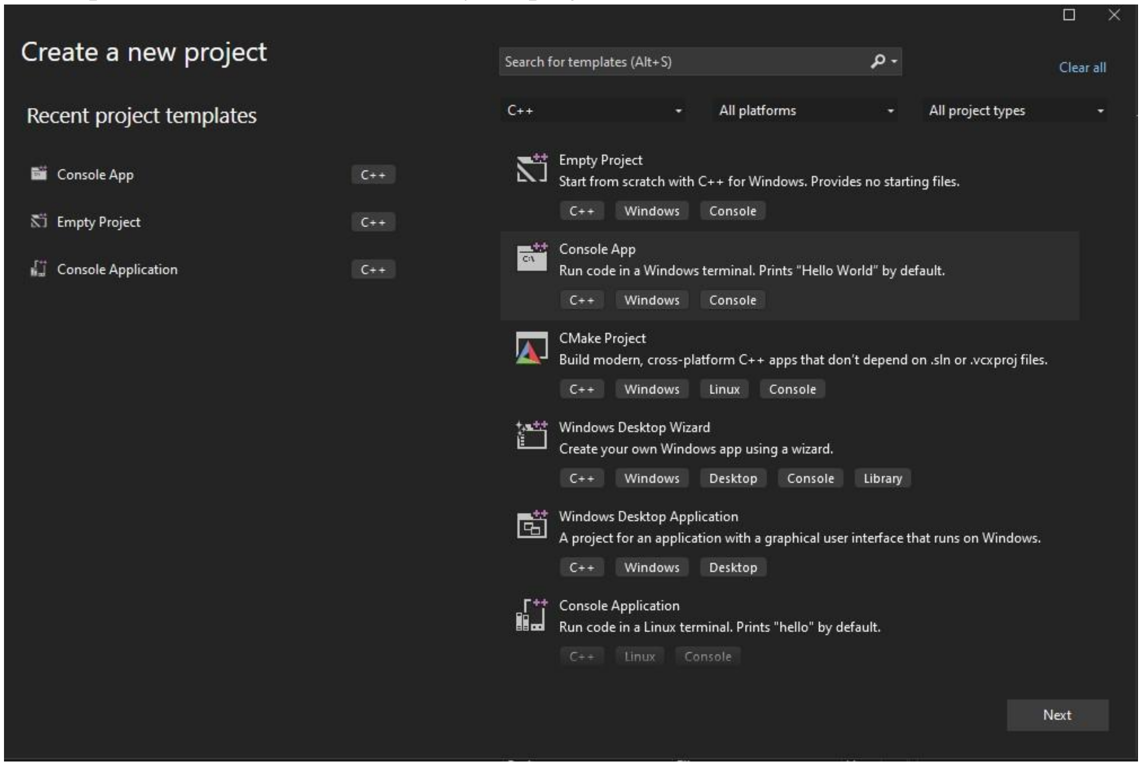
We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

2. Extract the downloaded file and make sure you remember where you extracted it since we need to locate it later. Basically, the file will consist of bin, include, and lib64 folder along with other files.


« Con... › mysql-connector...
Search mysql-connector-c++-8.1.0-winx64

Name	Date modified	Type	Size
bin	24/06/2023 4:53 PM	File folder	
include	24/06/2023 4:53 PM	File folder	
lib64	24/06/2023 4:53 PM	File folder	
INFO_BIN	24/06/2023 4:32 PM	File	1 KB
INFO_SRC	24/06/2023 4:32 PM	File	1 KB
LICENSE.txt	24/06/2023 4:32 PM	Text Document	139 KB
README.txt	24/06/2023 4:32 PM	Text Document	2 KB

3. Now open Visual Studio and create your project.



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

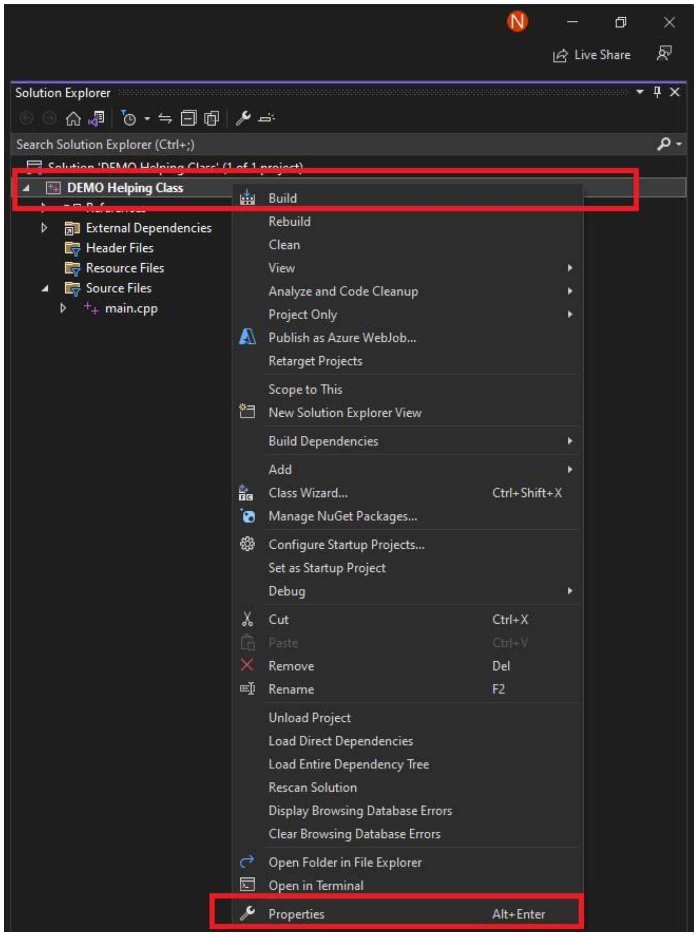
Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ Windows Desktop Console Library

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

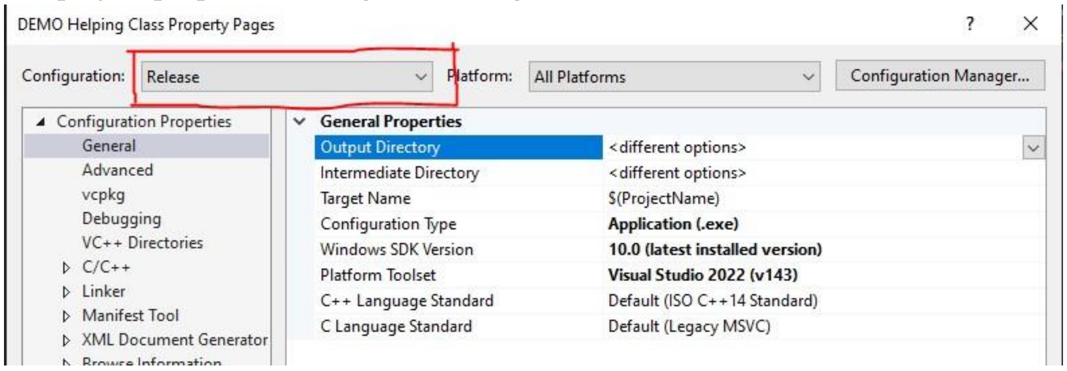
Console Application
Run code in a Linux terminal. Prints "hello" by default.
C++ Linux Console

4. If you selected empty project, you might not see C/C++ option in your project property. In that case you need to create 1 .cpp file first in your project.

5. Go to your project properties via the solution explorer. Make sure you are right clicking the project name, not the solution since it will be different.



6. In your project properties, change the configuration to "release"

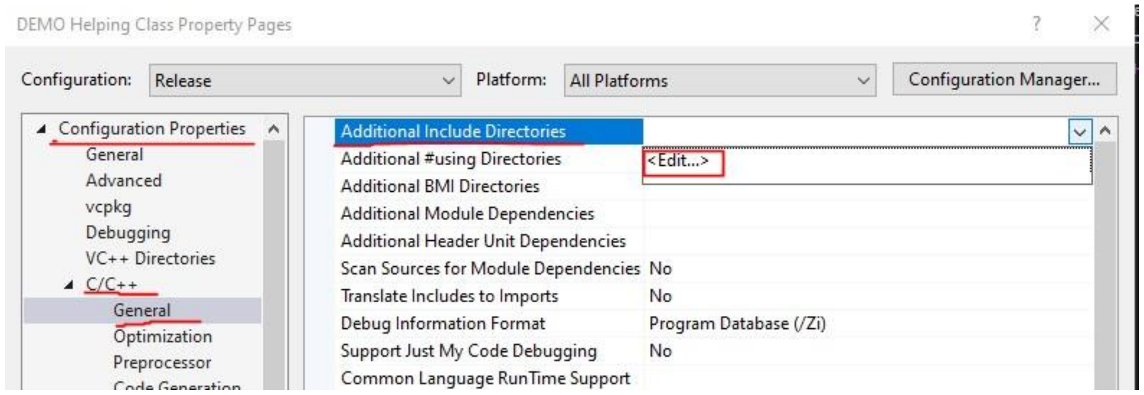


Configuration: Release
Platform: All Platforms

General Properties

Output Directory	<different options>
Intermediate Directory	<different options>
Target Name	$(ProjectName)
Configuration Type	**Application (.exe)**
Windows SDK Version	**10.0 (latest installed version)**
Platform Toolset	**Visual Studio 2022 (v143)**
C++ Language Standard	Default (ISO C++14 Standard)
C Language Standard	Default (Legacy MSVC)

7. In your project properties, go to Configuration Properties > C/C++ > General. Then select the Additional Include Directories and click edit.

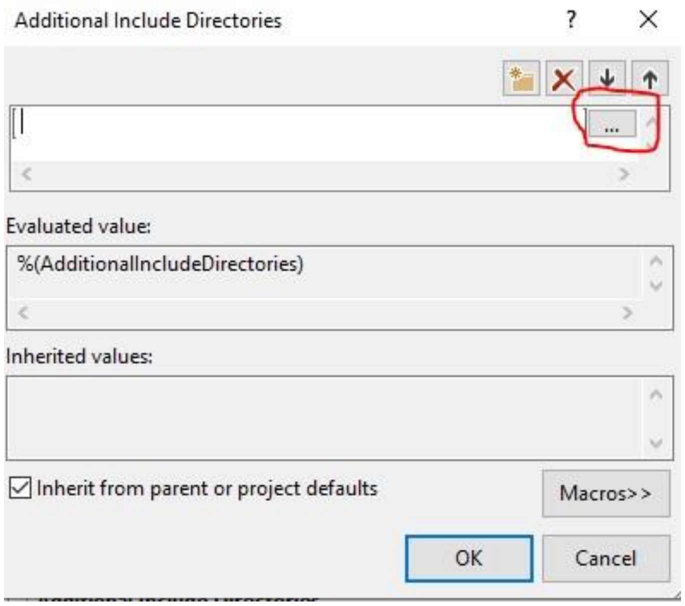


DEMO Helping Class Property Pages

Configuration: Release Platform: All Platforms Configuration Manager...

Additional Include Directories	
Additional #using Directories	<Edit...>
Additional BMI Directories	
Additional Module Dependencies	
Additional Header Unit Dependencies	
Scan Sources for Module Dependencies	No
Translate Includes to Imports	No
Debug Information Format	Program Database (/Zi)
Support Just My Code Debugging	No
Common Language RunTime Support	

8. Then click on the empty row and lick the button at the right of it. Locate the folder which you extracted the ZIP files in step 2 previously.



Additional Include Directories

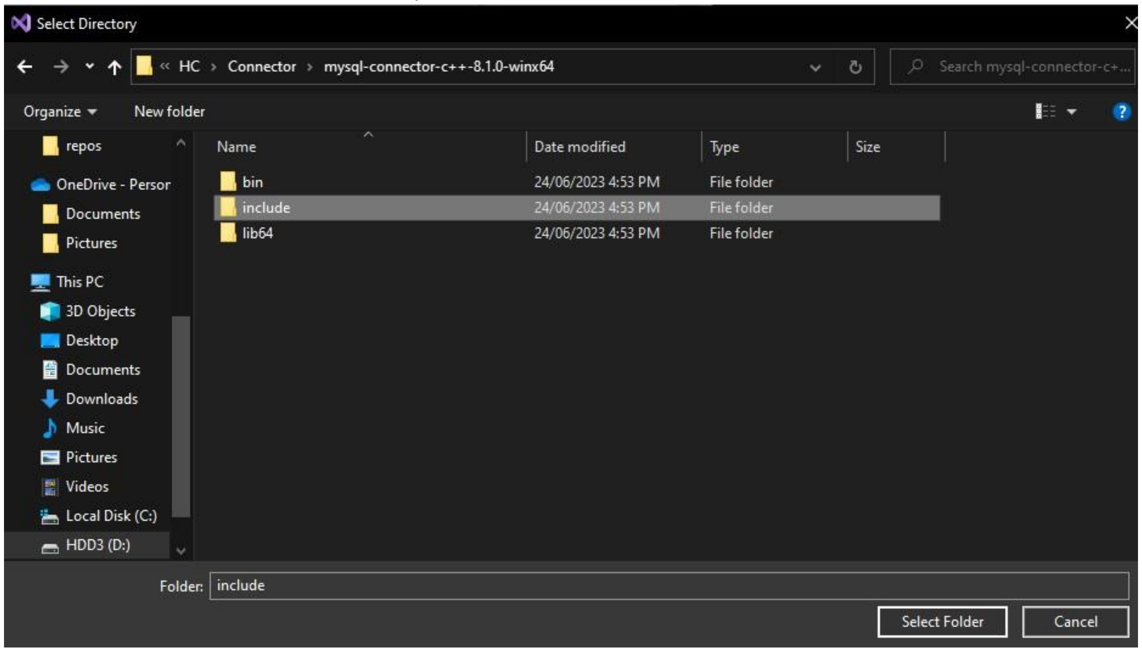
Evaluated value:

%(AdditionalIncludeDirectories)

Inherited values:

☑ Inherit from parent or project defaults

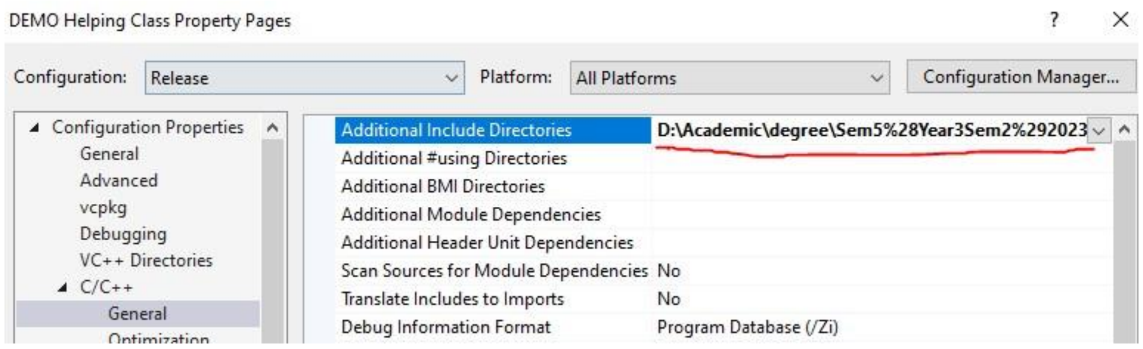
9. Select the “include” folder from your connector folder.



Name	Date modified	Type	Size
bin	24/06/2023 4:53 PM	File folder	
include	24/06/2023 4:53 PM	File folder	
lib64	24/06/2023 4:53 PM	File folder	

Folder: include

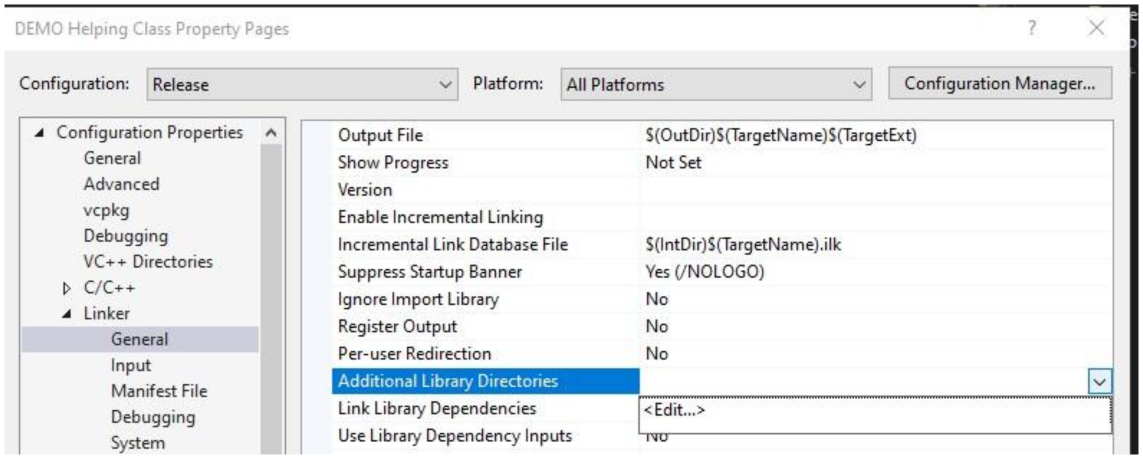
10. Press Ok. Now your project properties should look like this. Ensure that there is a value in the additional include directories.



Configuration: Release   Platform: All Platforms

Additional Include Directories	D:\Academic\degree\Sem5%28Year3Sem2%292023
Additional #using Directories	
Additional BMI Directories	
Additional Module Dependencies	
Additional Header Unit Dependencies	
Scan Sources for Module Dependencies	No
Translate Includes to Imports	No
Debug Information Format	Program Database (/Zi)

11. In your project properties go to Configuration Properties > Linker > General. Then select the Additional Library Directories to edit it.

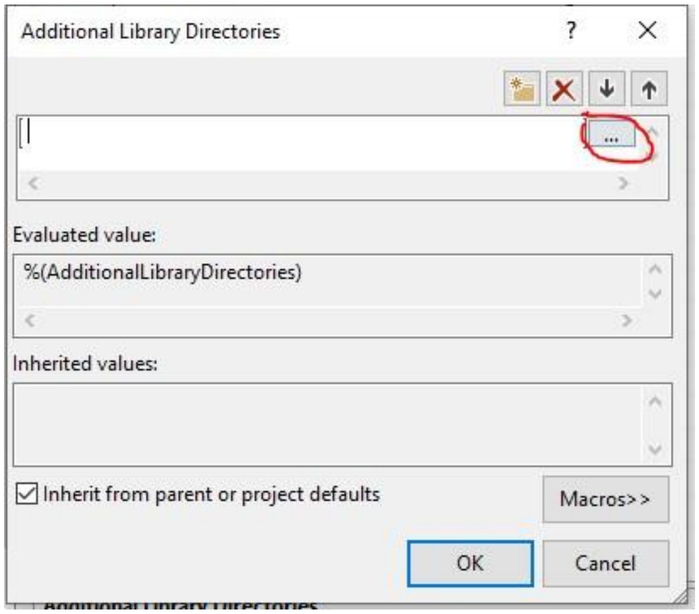

DEMO Helping Class Property Pages

Configuration: Release   Platform: All Platforms   Configuration Manager...

- Configuration Properties
  - General
  - Advanced
  - vcpkg
  - Debugging
  - VC++ Directories
  - C/C++
  - Linker
    - General
    - Input
    - Manifest File
    - Debugging
    - System

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	
Incremental Link Database File	$(IntDir)$(TargetName).ilk
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	<Edit...>
Link Library Dependencies	
Use Library Dependency Inputs	No

12. Then click on the empty row and lick the button at the right of it. Locate the folder which you extracted the ZIP files in step 2 previously.


Additional Library Directories

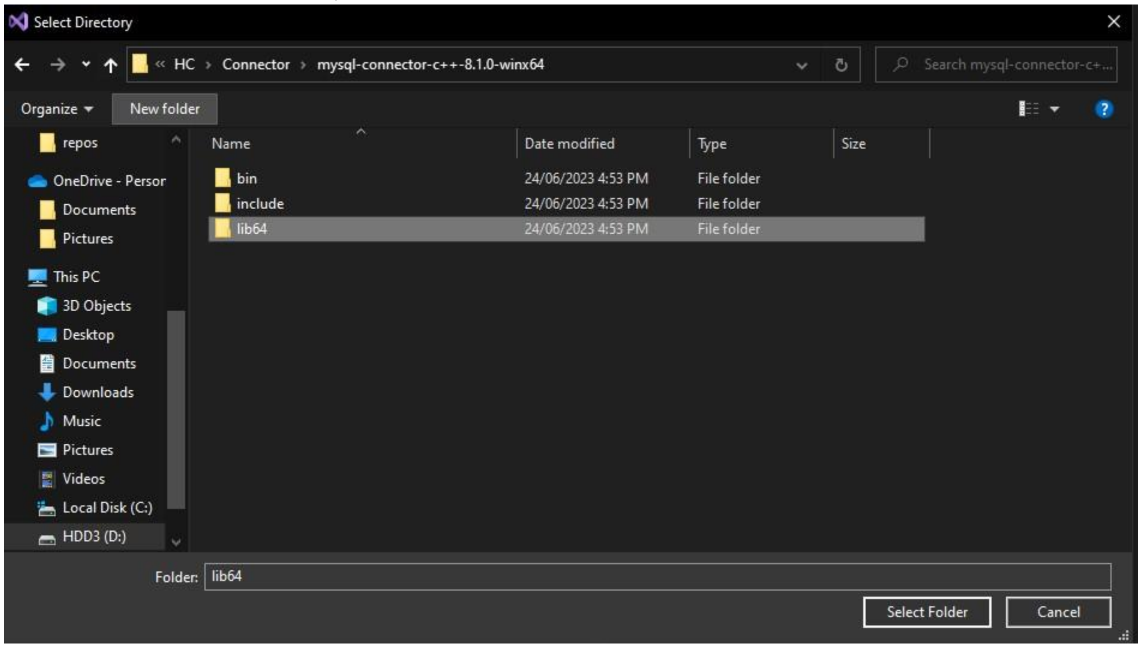
Evaluated value:

%(AdditionalLibraryDirectories)

Inherited values:

☑ Inherit from parent or project defaults

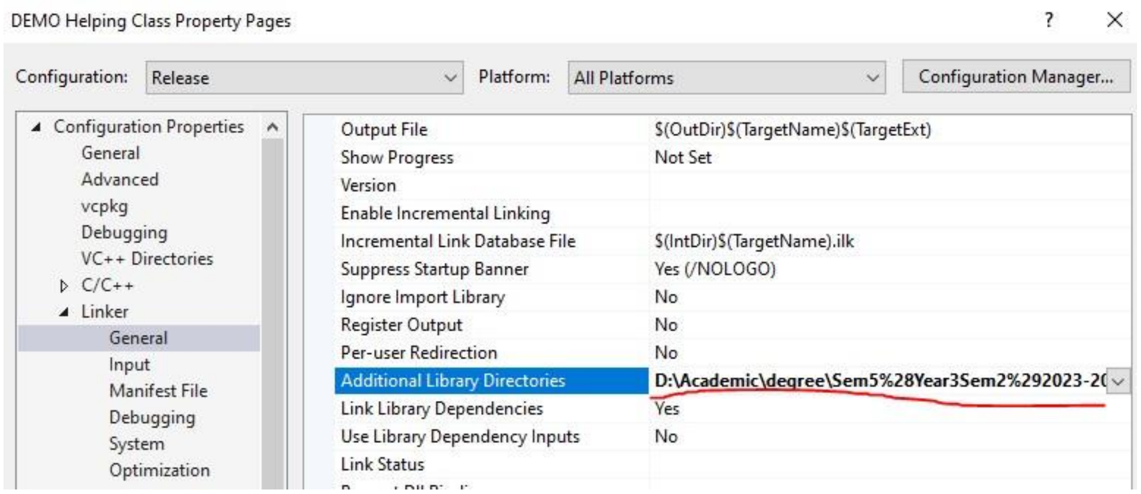
13. Select the lib folder from your connector folder.



Name	Date modified	Type	Size
bin	24/06/2023 4:53 PM	File folder	
include	24/06/2023 4:53 PM	File folder	
lib64	24/06/2023 4:53 PM	File folder	

Folder: lib64

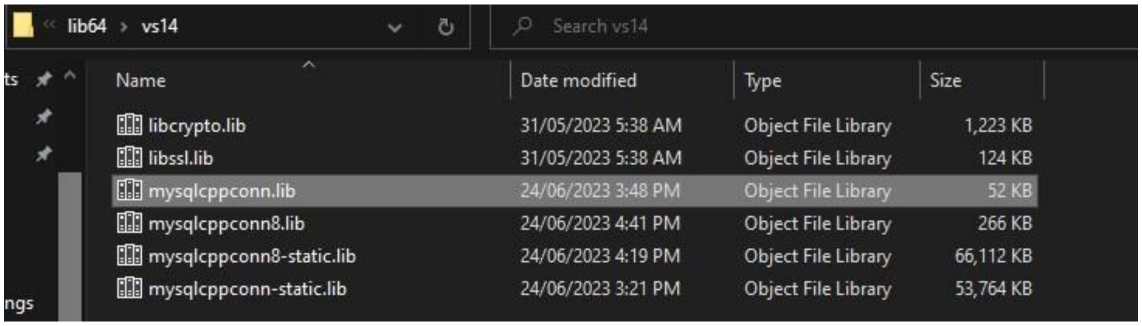
14. Click Oks. Now your project properties should look like this. Ensure that there is value in the Additional Library Directories.



Configuration: Release
Platform: All Platforms

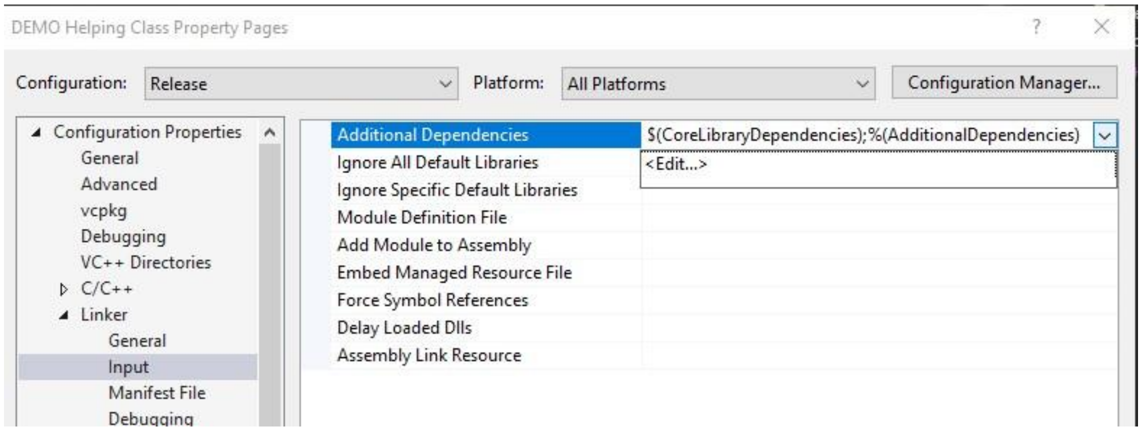
Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	
Incremental Link Database File	$(IntDir)$(TargetName).ilk
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	D:\Academic\degree\Sem5%28Year3Sem2%292023-2(
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	

15. (optional) In this part we want to check the library dependency file name. By default, it will most likely be “mysqlcppconn.lib”. But in case of MySQL themselves make changes you can confirm the file name by going to your connector folder/lib64/vs14



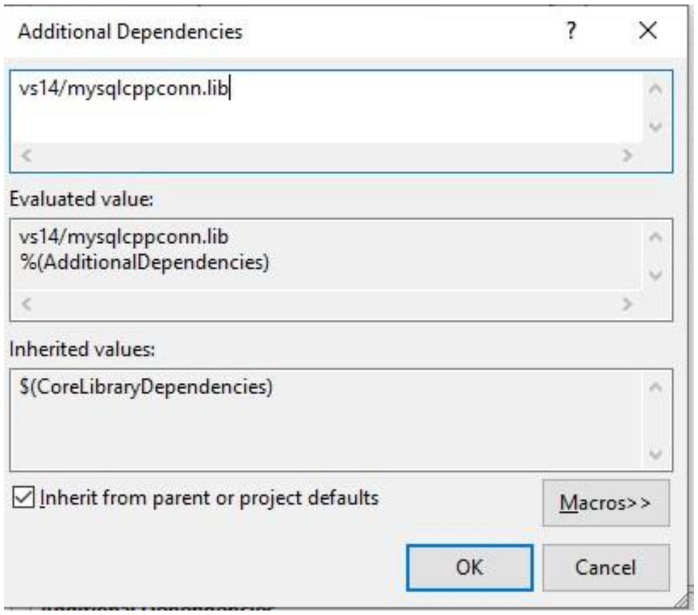
Name	Date modified	Type	Size
libcrypto.lib	31/05/2023 5:38 AM	Object File Library	1,223 KB
libssl.lib	31/05/2023 5:38 AM	Object File Library	124 KB
mysqlcppconn.lib	24/06/2023 3:48 PM	Object File Library	52 KB
mysqlcppconn8.lib	24/06/2023 4:41 PM	Object File Library	266 KB
mysqlcppconn8-static.lib	24/06/2023 4:19 PM	Object File Library	66,112 KB
mysqlcppconn-static.lib	24/06/2023 3:21 PM	Object File Library	53,764 KB

16. In your project properties. Go to Configuration Properties > Linker > Input. Select the Additional Dependencies to edit it.



Configuration: Release
Platform: All Platforms
Additional Dependencies: $(CoreLibraryDependencies);%(AdditionalDependencies)

17. Then put “vs14/mysqlcppconn.lib”. This basically tells your IDE which component of the library that you added into the linker just now you would like to use in your project which in this case is the mysqlcppconn.lib



Additional Dependencies

vs14/mysqlcppconn.lib

Evaluated value:

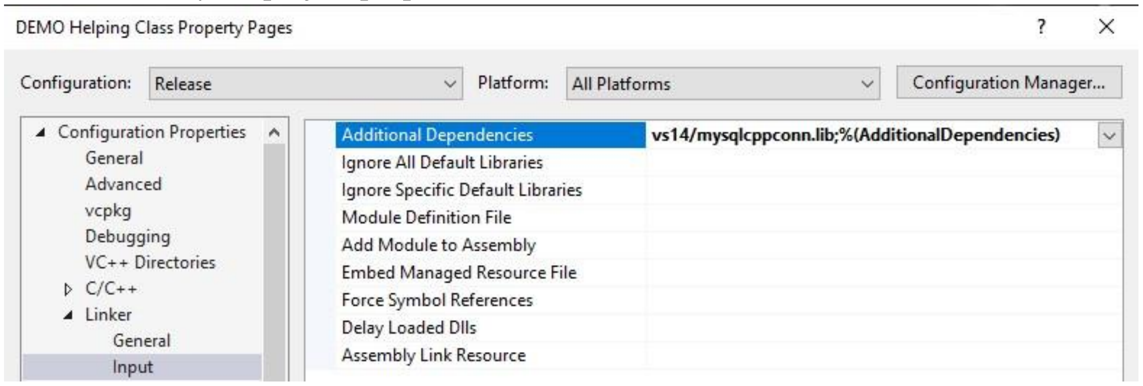
vs14/mysqlcppconn.lib
%(AdditionalDependencies)

Inherited values:

$(CoreLibraryDependencies)

☑ Inherit from parent or project defaults

18. Press ok. Now your project properties should look like this.



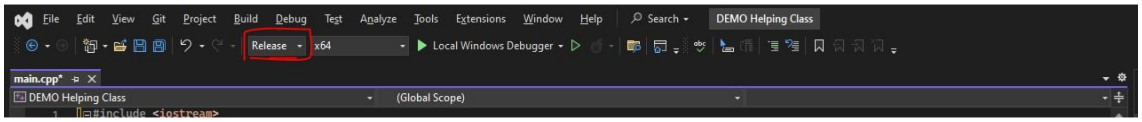
DEMO Helping Class Property Pages

Configuration: Release   Platform: All Platforms

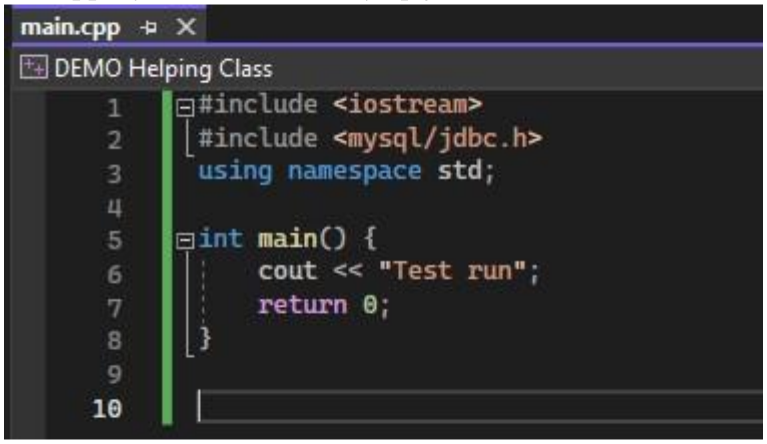
Additional Dependencies	vs14/mysqlcppconn.lib;%(AdditionalDependencies)
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

19. Click apply and then the configuration on project property part is done.
20. (optional) If you want to double check basically there is 3 important config to check which is the
    a. C/C++ > General > Additional Include Directories
    b. Linker > General > Additional Library Directories
    c. Linker > Input > Additional Dependencies

21. Now go to your main.cpp. Firstly, ensure that the development config at top of the screen is changed to release since your project properties is configured for release mode.



22. Then, in your main.cpp try to include the mysql/jdbc.h



```cpp
#include <iostream>
#include <mysql/jdbc.h>
using namespace std;

int main() {
    cout << "Test run";
    return 0;
}

```

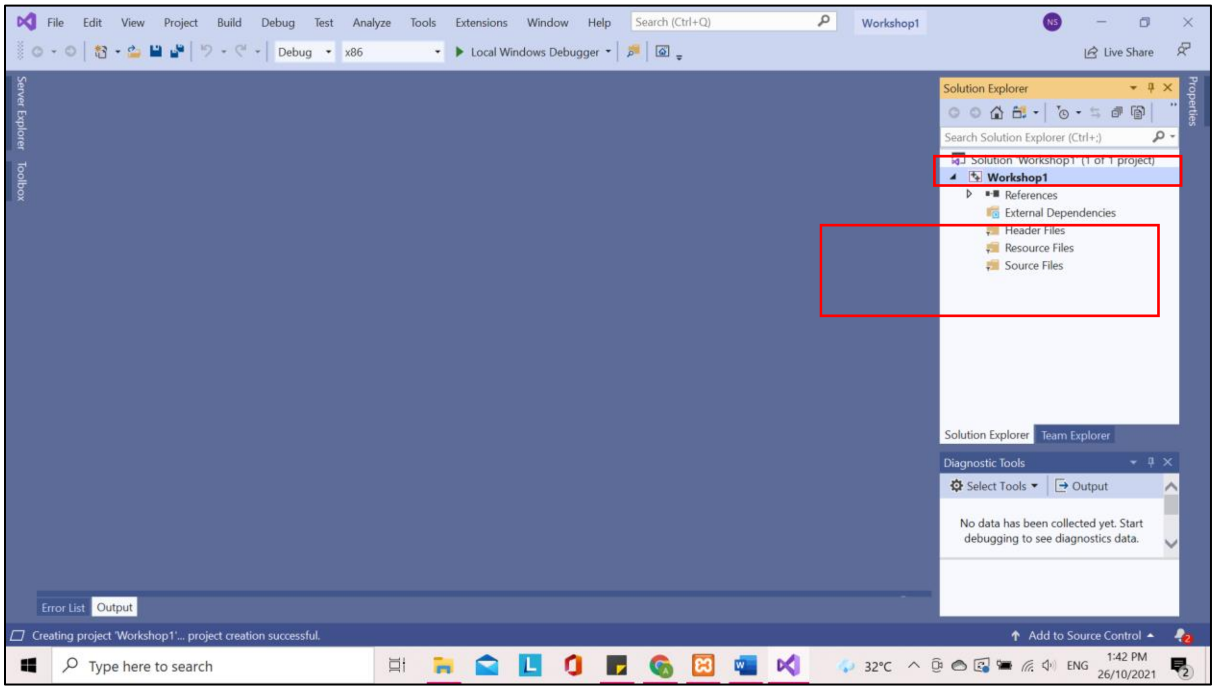
23. Now if you can compile and run without error then it means you have successfully imported a third-party library and the mysql/jdbc.h is now ready to be used which will be explained in the next sections.

Note: moving/deleting or any changes on your connector folder might affect your C++ application when you recompile it which will result to your IDE unable to locate the necessary dependencies. Thus, put your connector folder somewhere you might not change frequently.

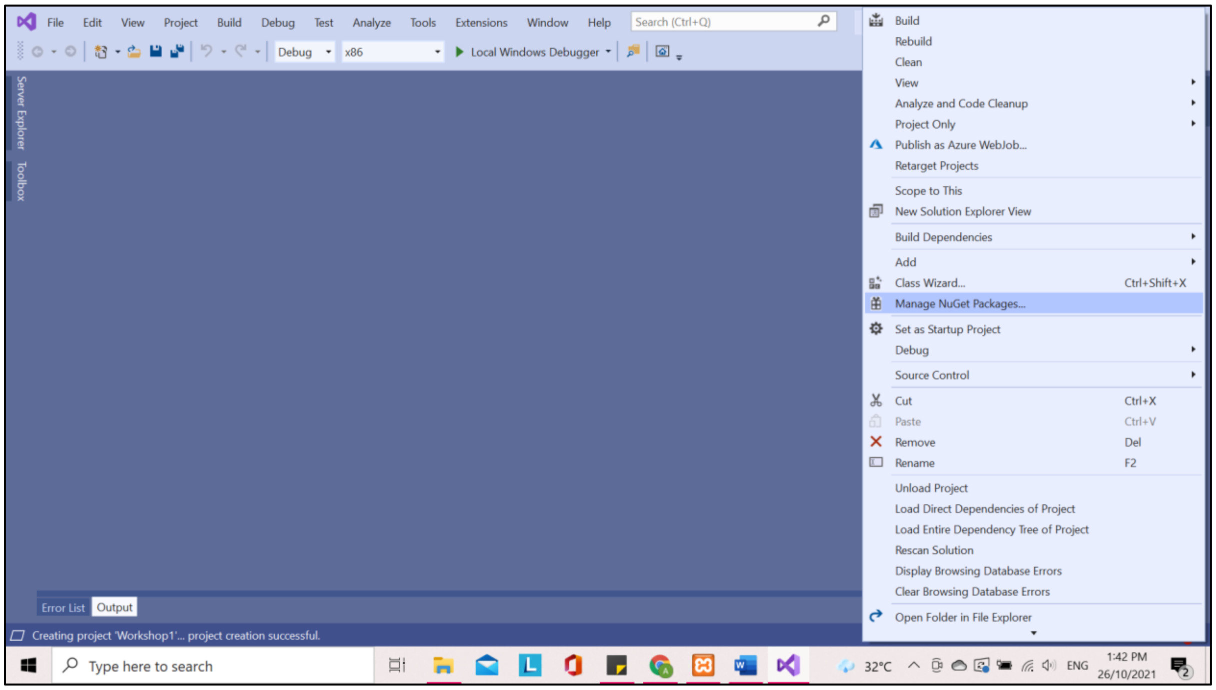
Also, importing other third-party library might follows similar steps which is adding entry in Additional Include Directories, Additional Library Directories, and Additional Dependencies. Thus, understanding this step could be helpful when you want to use other third-party libraries such as GNU plot integration to generate graph or other libraries.

1.4. Install MySQL library in Visual Studio

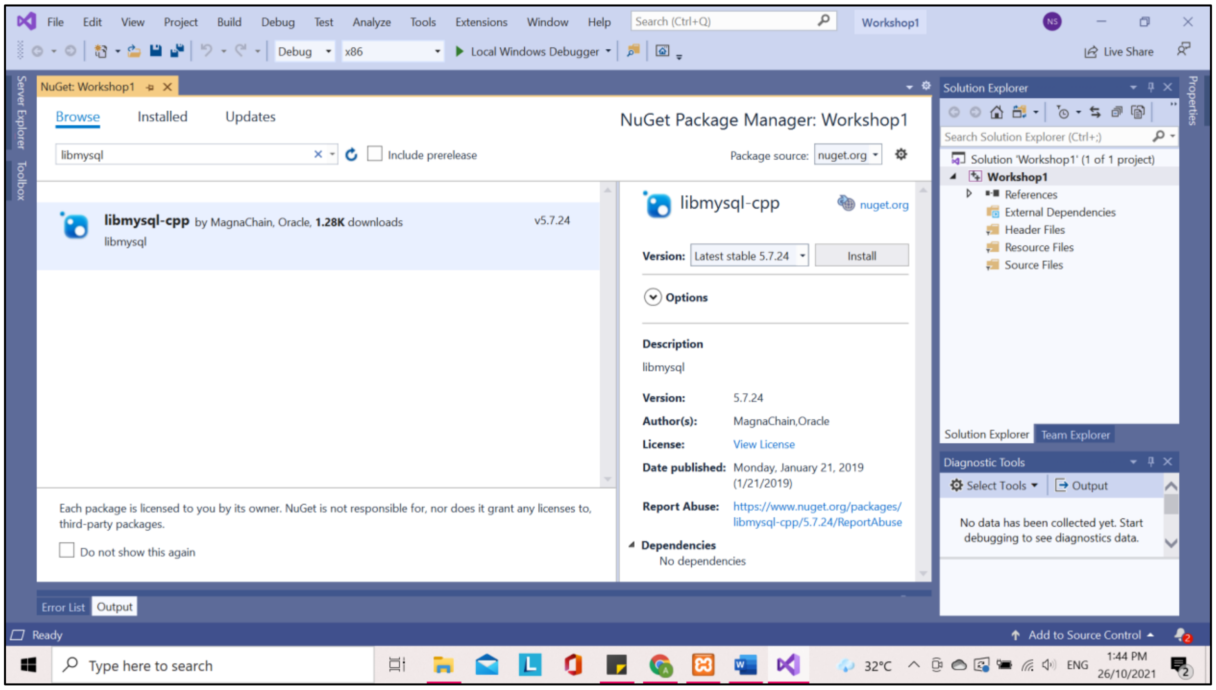
1. Click to your project name.



2. Click to Manage NuGet Package.


Build
Rebuild
Clean
View
Analyze and Code Cleanup
Project Only
Publish as Azure WebJob...
Retarget Projects
Scope to This
New Solution Explorer View
Build Dependencies
Add
Class Wizard... Ctrl+Shift+X
Manage NuGet Packages...
Set as Startup Project
Debug
Source Control
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename F2
Unload Project
Load Direct Dependencies of Project
Load Entire Dependency Tree of Project
Rescan Solution
Display Browsing Database Errors
Clear Browsing Database Errors
Open Folder in File Explorer

3. In Browse search libmysql and install libmysql-cpp



libmysql-cpp by MagnaChain, Oracle, 1.28K downloads v5.7.24
libmysql

libmysql-cpp nuget.org

Version: Latest stable 5.7.24

Description
libmysql

Version:	5.7.24
Author(s):	MagnaChain,Oracle
License:	View License
Date published:	Monday, January 21, 2019 (1/21/2019)
Report Abuse:	https://www.nuget.org/packages/libmysql-cpp/5.7.24/ReportAbuse

Dependencies
No dependencies

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

No data has been collected yet. Start debugging to see diagnostics data.

2. Development "Implementation (Coding)"

In this section, the implementation in form of coding will be explained in detail including the logic behind the codes to help you understand how to perform the fundamental database operations (CRUD) and implement it into your own specific project needs.

2.1. Connect to Database

To perform database operation in your C++ application we will have to make sure to run Apache and MySQL in xampp.

```
#include <iostream>
#include <string>
#include <conio.h>
#include <mysql.h> //libmysql
using namespace std;

void Register();
void Login();
void AdminMenu();
void ViewCar();
void InsertCar();
void EditCar();
void DeleteCar();
void SearchCar();

int qstate;
MYSQL* conn;
MYSQL_ROW row;
MYSQL_RES* res;

string UserID, CarID;

class db_response
{
public:
    static void ConnectionFunction()
    {
        conn = mysql_init(0);
        if (conn)
            cout << "Database Connected" << endl;
        else
            cout << "Failed To Connect!" << endl;

        conn = mysql_real_connect(conn, "localhost", "root", "", "car_rental", 3306, NULL, 0);
        if (conn)
            cout << "Database Connected To MySql" << endl;
        else
            cout << "Failed To Connect!" << endl;
    }
};
```

```
int main() {
    system("cls");
    system("title My Project");
    db_response::ConnectionFunction();

return 0;
}
```

2.2. Main Function

```
int main() {
    system("cls");
    system("title My Project");
    db_response::ConnectionFunction();

    int menu;

    cout << "1. Login" << endl;
    cout << "2. Register" << endl;
    cout << "3. Exit" << endl;
    cout << "Choose: ";
    cin >> menu;

    switch (menu)
    {
    case 1:
        Login();
        break;
    case 2:
        Register();
        break;
    case 3:
        cout << "Program Exiting..." << endl;
        exit(0);
    default:
        cout << "Please choose between 1 - 3. Press Enter To Continue...";
        _getch(); //get char - pause console
        system("cls");
        main();
        break;
    }

return 0;
}
```

2.3. Registration

This is implementation for registration. All these data are depending on your system requirement. Commonly, we would store username and password for login session later. As we know, the username of account should be unique. So, we should check first if the entered username already exists and display the error. Otherwise, customer will be registered successfully.

```
void Register()
{
    system("cls");
    string name, phone, age, username, password;
    cout << "--- Admin Register ---" << endl;
    cout << "Enter Name: ";
    cin.ignore(1, '\n');
    getline(cin, name);
    cout << "Enter Phone: ";
    getline(cin, phone);
    cout << "Enter age: ";
    getline(cin, age);
    cout << "Enter Username: ";
    getline(cin, username);
    cout << "Enter Password: ";
    getline(cin, password);

    string checkUser_query = "select * from Admin where Username = '" + username + "'";
    const char* cu = checkUser_query.c_str();
    qstate = mysql_query(conn, cu);

    if (!qstate)
    {
        res = mysql_store_result(conn);
        if (res->row_count == 1)
        {
            cout << "Username is already exist. Press Enter to Try Again...";
            _getch();
            Register();
        }
        else
        {
            string insertCustomer_query = "insert into Admin (name, contact_no, age, username, 
password) values ('" + name + "', '" + phone + "', '" + age + "', '" + username + "', '" + 
password + "')";
            const char* q = insertCustomer_query.c_str();
            qstate = mysql_query(conn, q);

            if (!qstate)
            {
                cout << endl << "You have been registered. Press Enter to Continue...";
                _getch();
                main();
            }
            else
            {
                cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
            }
        }
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    }
}
```

2.4. Login

This is implementation of login page. Depends on your system, how many account roles do you have. For example, we have 2 role which is Admin & Customer. As we know, login page will require the username, password and role. System will check the user information by username and password for authentication. If fail, error will be display. Otherwise, user will be redirect to menu page based on their role.

```
void Login()
{
     system("cls");
    string password;
    string username;

    cout << "--- Login Page ---" << endl;
    cout << "Username : ";
    cin >> username;
    cout << "Password : ";
    char ch;
    while ((ch = _getch()) != 13)
    {
        password += ch;
        cout << '*';
    }

    string checkUser_query = "select admin_id from Admin where username = '" + username + "'
and password = '" + password + "'";
    const char* cu = checkUser_query.c_str();
    qstate = mysql_query(conn, cu);

    if (!qstate)
    {
        res = mysql_store_result(conn);
        if (res->row_count == 1)
        {
            while (row = mysql_fetch_row(res))
                UserID = row[0];
            AdminMenu();
        }
        else
        {
            char c;
            cout << "Invalid username or password. Want to try again? (y/n): ";
            cin >> c;
            if (c == 'y' || c == 'Y')
                Login();
            else
```

```
                main();
            }
        }
        else
            cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
}
```

2.5. Admin Menu Page

This is a page after login

```
void AdminMenu()
{
    int menu;

    system("cls");
    cout << "-----Admin Page------" << endl;

    cout << "1. View Car" << endl;
    cout << "2. Insert New Car" << endl;
    cout << "3. Edit Car" << endl;
    cout << "4. Delete Car" << endl;
    cout << "5. Search Car" << endl;
    cout << "6. Logout" << endl;
    cout << "Choose menu: ";

    cin >> menu;

    switch (menu)
    {
    case 1:
        ViewCar();
        break;
    case 2:
        InsertCar();
        break;
    case 3:
        EditCar();
        break;
    case 4:
        DeleteCar();
        break;
    case 5:
        SearchCar();
        break;
    case 6:
        main();
        break;
    default:
        cout << "Please choose between 1 - 5. Press Enter To Continue...";
        _getch();
        system("cls");
        AdminMenu();
        break;
    }
}
```

2.6. Insert Data

This is implementation for insert data. Data will be saved in database.

```cpp
void InsertCar()
{
    system("cls");

    string registration_no, type, brand, year;
    char choose;
    cout << "-----Insert New Car-----" << endl;

    cin.ignore(1, '\n');
    cout << "Car Registration No.: ";
    getline(cin, registration_no);
    cout << "Type of Car: ";
    getline(cin, type);
    cout << "Car Brand: ";
    getline(cin, brand);
    cout << "Manafacturer Year: ";
    getline(cin, year);

    string insert_query = "insert into Car (admin_id, registration_no, type, brand, year) values
('" + UserID + "', '" + registration_no + "', '" + type + "', '" + brand + "', '" + year +
"')";
    const char* q = insert_query.c_str();
    qstate = mysql_query(conn, q);

    if (!qstate)
    {
        cout << endl << "Car is successful added in database." << endl;
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    }

    do
    {
        cout << "Do you want add another equipment? (y/n): ";
        cin >> choose;
        if (choose == 'y' || choose == 'Y')
        {
            InsertCar();
        }
        else if (choose == 'n' || choose == 'N')
        {
            ViewCar();
        }
    } while (choose == 'y' || choose == 'Y' || choose == 'n' || choose == 'N');
}
```

2.7. View Data

```
void ViewCar()
{
    system("cls");
    char choose;
    cout << "\t\t\t\t--- List of All Car ---" << endl;

    //date_format() used to convert the format of date.
    qstate = mysql_query(conn, "select car_id, registration_no, type, brand, year from Car");

    if (!qstate)
    {
        cout << setw(11) << "ID" << setw(17) << "Registration No" << setw(20) << "Car Type" <<
setw(20) << "Car Brand" << setw(25) << "Manafacturer Year" << endl;

        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            cout << setw(11) << row[0] << setw(17) << row[1] << setw(20) << row[2] << setw(20)
<< row[3] << setw(20) << row[4] << endl;
        }
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    }

    cout << "Press Enter To Back...";
    _getch();
    AdminMenu();
}
```

2.8. Edit Data

Choose id then choose the column that you want to edit.

```
void EditCar()
{
    system("cls");

    string data;
    int choose;
    char option;
    cout << "-----Edit Car-----" << endl;

    qstate = mysql_query(conn, "select car_id, registration_no, type, brand, year from Car");

    if (!qstate)
    {
        cout << setw(11) << "ID" << setw(17) << "Registration No" << setw(20) << "Car Type" << 
setw(20) << "Car Brand" << setw(25) << "Manafacturer Year" << endl;

        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            cout << setw(11) << row[0] << setw(17) << row[1] << setw(20) << row[2] << setw(20) 
<< row[3] << setw(20) << row[4] << endl;
        }

        cout << "Choose ID: ";
        cin >> CarID;

        cout << endl << "-----Category-----" << endl;
        cout << "1. Registration no." << endl;
        cout << "2. Car Type" << endl;
        cout << "3. Car Brand" << endl;
        cout << "4. Manafacturer Year" << endl;
        cout << "Choose 1-4: ";
        cin >> choose;

        cin.ignore(1, '\n');
        if (choose == 1)
        {
            cout << "New Registration No.: ";
            getline(cin, data);
            string update_query = "update car set registration_no = '" + data + "' where car_id 
= '" + CarID + "'";
            const char* q = update_query.c_str();
            qstate = mysql_query(conn, q);
        }
        else if (choose == 2)
        {
            cout << "New Car Type: ";
            getline(cin, data);
            string update_query = "update car set type = '" + data + "' where car_id = '" + 
CarID + "'";
            const char* q = update_query.c_str();
            qstate = mysql_query(conn, q);
        }
        else if (choose == 3)
        {
            cout << "New Car Brand: ";
            getline(cin, data);
            string update_query = "update car set brand = '" + data + "' where car_id = '" +
CarID + "'";
            const char* q = update_query.c_str();
            qstate = mysql_query(conn, q);
        }
        else if (choose == 4)
        {
            cout << "New Manafacturer Year: ";
            getline(cin, data);
            string update_query = "update car set year = '" + data + "' where car_id = '" +
CarID + "'";
            const char* q = update_query.c_str();
            qstate = mysql_query(conn, q);
        }

        cout << "Do you want to edit other detail? (y/n): ";
        cin >> option;

        if (option == 'y' || option == 'Y')
            EditCar();
        else
            ViewCar();
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    }

}
```

2.9. Delete Data

Delete by car id.

```
void DeleteCar()
{
    system("cls");
    char choose;
    cout << "--- Delete Car ---" << endl;

    qstate = mysql_query(conn, "select car_id, registration_no, type, brand, year from Car");

    if (!qstate)
    {
        cout << setw(11) << "ID" << setw(17) << "Registration No" << setw(20) << "Car Type" <<
setw(20) << "Car Brand" << setw(25) << "Manafacturer Year" << endl;

        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            cout << setw(11) << row[0] << setw(17) << row[1] << setw(20) << row[2] << setw(20)
<< row[3] << setw(20) << row[4] << endl;
        }

        cout << "Choose ID: ";
        cin >> CarID;

        string delete_query = "delete from car where car_id = '" + CarID + "'";
        const char* q = delete_query.c_str();
        qstate = mysql_query(conn, q);

        cout << "Do you want to delete other car? (y/n): ";
        cin >> choose;

        if (choose == 'y' || choose == 'Y')
            DeleteCar();
        else
            ViewCar();
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    }
}
```

2.10. Search Data

```
void SearchCar()
{
    system("cls");
    char choose;
    string brand;
    cin.ignore(1, '\n');
    cout << "Search car by brand: ";
    getline(cin, brand);

    cout << "--- List of Searched Car ---" << endl;

    string search_query = "select car_id, registration_no, type, brand, year from Car where
brand like '%" + brand + "%'";
    const char* q = search_query.c_str();
    qstate = mysql_query(conn, q);

    if (!qstate)
    {
        cout << setw(11) << "ID" << setw(17) << "Registration No" << setw(20) << "Car Type" <<
setw(20) << "Car Brand" << setw(25) << "Manafacturer Year" << endl;

        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            cout << setw(11) << row[0] << setw(17) << row[1] << setw(20) << row[2] << setw(20)
<< row[3] << setw(20) << row[4] << endl;
        }

        cout << endl << "Do you want to search other brand?(y/n): ";
        cin >> choose;

        if (choose == 'y' || choose == 'Y')
            SearchCar();
        else if (choose == 'n' || choose == 'N')
            AdminMenu();
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    }
}
```

2.11. Input Validation

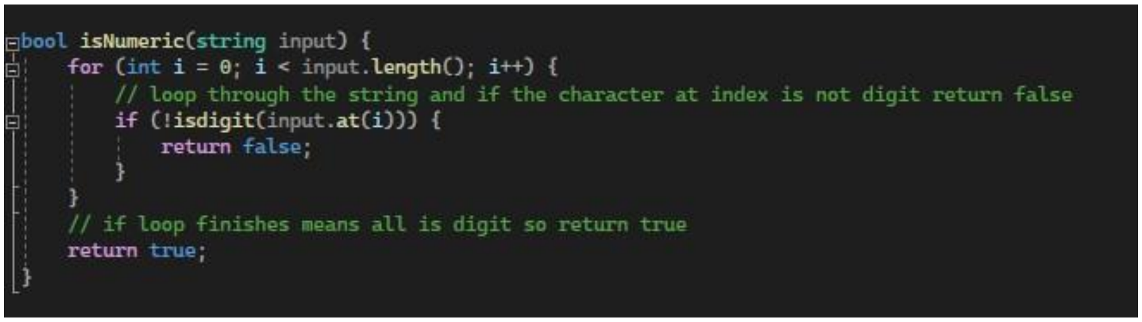
Input validation is a common thing to do. As a developer we have to increase the usability of our system to not only make it easier to use for a larger group of users by providing error messages as feedback to notify users of their mistake. Besides, input validation will also ensure the integrity of our data where no invalid data should be saved into the database. In this section, the two most basic types of validation will be explained.

2.11.1. Type validation

Input type validation is essential, and it concerns whether user inserted the correct input type or not. For example, let's look at the current application without validation. Users can insert invalid data into their year of birth with wrong type such as text when it supposed to be number. This causes unexpected behavior that might even crash the application. For that, we must ensure correct input before proceeding.

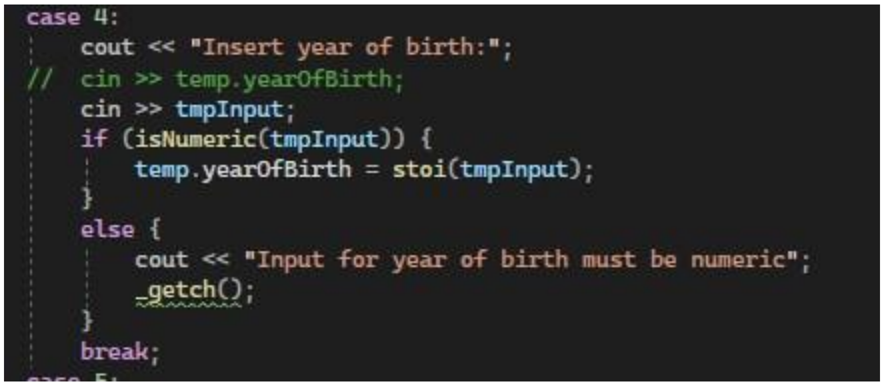
The most versatile data type that can hold almost anything is string, which is why it would be much safer to temporarily store our input data into a string first so that whatever user inserts it would not cause any error. Then only we check and validate the string whether it meets our requirements.

The following function will loop through the string and return false if any of the characters in the string is not digit. Although this method does work, it is not the best. There are other alternatives way which are mentioned in the additional note section.



```
bool isNumeric(string input) {
    for (int i = 0; i < input.length(); i++) {
        // loop through the string and if the character at index is not digit return false
        if (!isdigit(input.at(i))) {
            return false;
        }
    }
    // if loop finishes means all is digit so return true
    return true;
}
```

Then we can use it on our input:



```
case 4:
    cout << "Insert year of birth:";
//  cin >> temp.yearOfBirth;
    cin >> tmpInput;
    if (isNumeric(tmpInput)) {
        temp.yearOfBirth = stoi(tmpInput);
    }
    else {
        cout << "Input for year of birth must be numeric";
        _getch();
    }
    break;
```

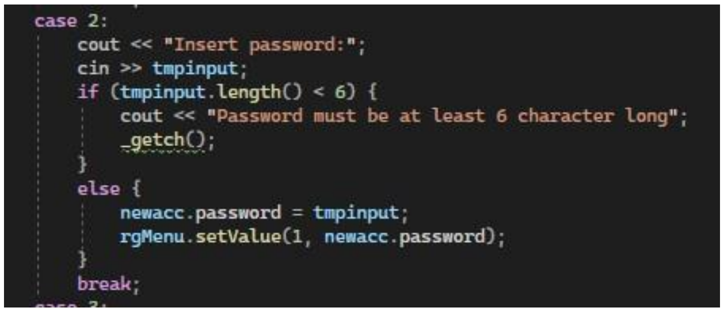
Since the function returns Boolean, we can directly call it inside if ().

- When the string is numeric, we proceed to convert the string into integer using stoi and stores it into yearOfBirth attribute of our newacc object.
- Also, do not forget to inform users regarding the error to guide them on how to correct it.
- _getch() is used here to wait for user to press any key so they have time to read the error message.

2.11.2. Format validation

The format validation is more to logical requirement rather than technical. For example, a system might have rules on its password minimum 6 characters, or username minimum 10 characters or email must be in valid format etc. In this section we only will cover the length format validation as the structural format will require using regular expression (Regex) that will be covered in the additional notes.

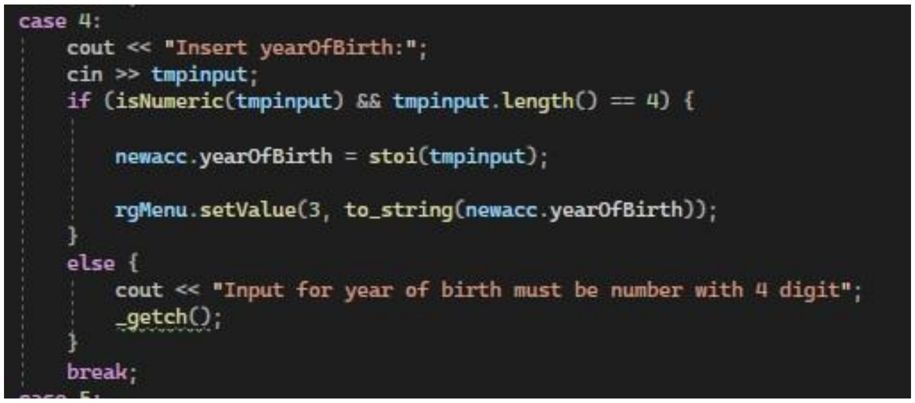
In this example we will be checking to ensure that the user input for password must be at least 6 characters long.



```
case 2:
    cout << "Insert password:";
    cin >> tmpinput;
    if (tmpinput.length() < 6) {
        cout << "Password must be at least 6 character long";
        _getch();
    }
    else {
        newacc.password = tmpinput;
        rgMenu.setValue(1, newacc.password);
    }
    break;
```

The implementation is rather simple since we can just use the. length () method of string to validate in. Besides text input like password, we can also apply the same logic for numerical input by using temporary input string.

For example, year of birth must be exactly 4 digit long. Then we can combine the checking with our previous checking as shown in the following diagram.



```
case 4:
    cout << "Insert yearOfBirth:";
    cin >> tmpinput;
    if (isNumeric(tmpinput) && tmpinput.length() == 4) {

        newacc.yearOfBirth = stoi(tmpinput);

        rgMenu.setValue(3, to_string(newacc.yearOfBirth));
    }
    else {
        cout << "Input for year of birth must be number with 4 digit";
        _getch();
    }
    break;
```

So now the year of birth must be number with exactly four digit.

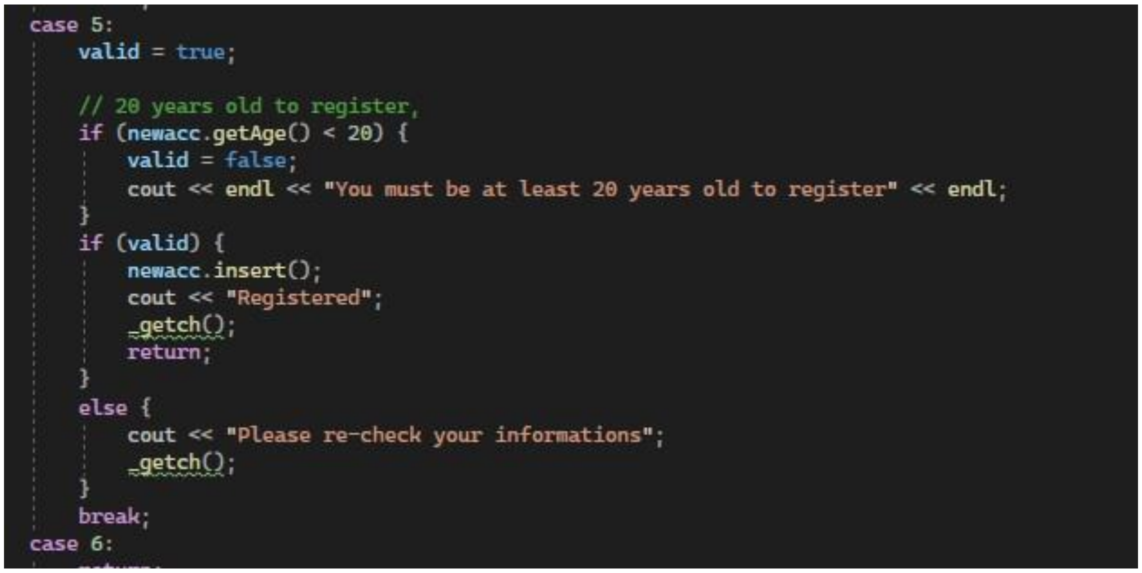
2.11.3. Range Validation

Similar to format validation, range validation is not up to technical requirements but rather your logical requirements on your data. For example, for price user must insert price greater than 0. In this example we will be doing range validation using the user age.

In the registration menu, we have validated to year of birth to ensure that it should be 4 digits of numbers only. But is this enough? Is 1000 a valid year? No, it’s not since it is not logical to have user who was born at year 1000. What about when user inserts “0010”? using our simple validation which only checks type and length “0010” will pass as valid string, even “0000”.

Thus, we will implement another checking on the submission/registration to ensure that user are at least 20 years old to proceed.

Since we already have getAge() method in account class, to check user age is fairly simple as shown in the following diagram.



```
case 5:
    valid = true;

    // 20 years old to register,
    if (newacc.getAge() < 20) {
        valid = false;
        cout << endl << "You must be at least 20 years old to register" << endl;
    }
    if (valid) {
        newacc.insert();
        cout << "Registered";
        _getch();
        return;
    }
    else {
        cout << "Please re-check your informations";
        _getch();
    }
    break;
case 6:
```

- Note that our menu is in a loop, so we declared this valid Boolean variable outside of the loop.
- On case 5 which is the “register” option, we first re-initialize this valid value to true.
- Then we perform checking using the getAge() method, and if its less than 20 which doesn’t meet our requirements, we set the valid Boolean to false and display an error message.
- Finally, we check if the valid Boolean is true or not before proceeding to save user data using insert() or display error footer.

- Between re-initialization of valid = true and the final process, you can have any number of validation process which should change valid to false and display error message when the data does not meet the logical criteria.

  Thus, regardless of how many times you check if at least one error, the process will not proceed. Otherwise, when all validation is not error, the code will proceed to save the data into database.

This concludes the section for validation explanation. In this example, the validation is very minimal since the purpose is only to demonstrate and explain. Do note that in an actual application you will have to validate all input from user to ensure accuracy of your data and protect your system from unexpected behaviors.

3. Additional Notes

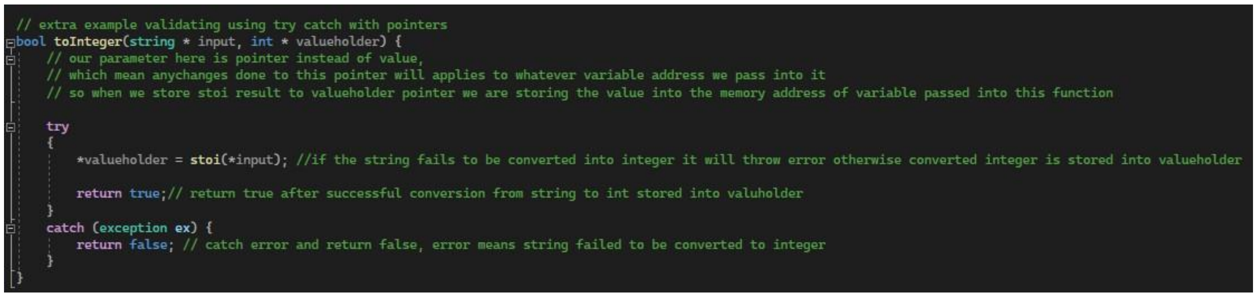
This section covers explanation of some features which might not be necessary or compulsory to have but good to have.

3.1. Validation using Exception handling (Try Catch)

Previously we have done basic validation using simple if. For the input type validation, we loop through each of the characters in the string to check if any of it is not digit. This approach definitely works but its implementation is limited. Not all data types have similar functions like isDigit to be used in checking each of the characters.

For example, what if the input is price with decimal place? Our validation using loop will reject the input the moment it encounters the dot "**.**". Should we add is digit or character == "." To allow dot? Then what if the user wrongly inserted string with multiple dots. There are so many things to consider when doing input validation since users' stupidity and carelessness is unmeasurable.

Thus, this section will explain an alternative method to perform input validation using exceptions as shown in the following code.



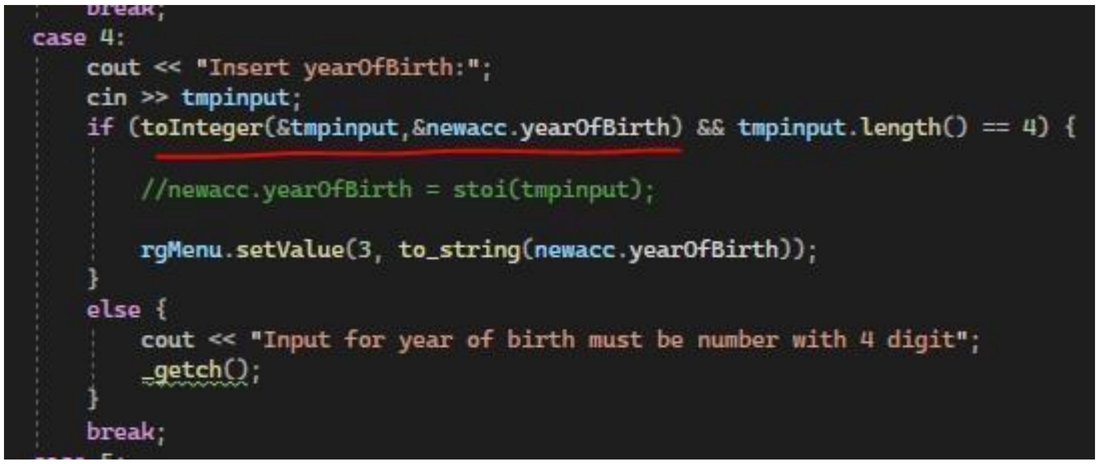
```
// extra example validating using try catch with pointers
bool toInteger(string * input, int * valueholder) {
    // our parameter here is pointer instead of value,
    // which mean anychanges done to this pointer will applies to whatever variable address we pass into it
    // so when we store stoi result to valueholder pointer we are storing the value into the memory address of variable passed into this function

    try
    {
        *valueholder = stoi(*input); //if the string fails to be converted into integer it will throw error otherwise converted integer is stored into valueholder

        return true;// return true after successful conversion from string to int stored into valuholder
    }
    catch (exception ex) {
        return false; // catch error and return false, error means string failed to be converted to integer
    }
}
```

- The purpose of this function is to convert a string into integer while also checking if the string is actually numeric or not.
- Note that the parameters are pointers (*), not value. Thus, when we refer to a pointer, we are referring to the original variable passed to this function. Basically, any changes you make to the pointer here will apply to the variable being passed, for example if you change value of valueholder, the variable passed into this parameter will have that value even though it's in different functions.
- Try block contain codes that may produce/throw an exception which is the stoi() in this case. If the string passed in the parameter is not numeric and cannot be converted into integer stoi will throw an exception which will crash and stop your application if not handled. Thus, if stoi causes error, the return true statement will not be executed.

- The catch block will handle the exception thrown from try block to prevent crashing. Additionally, we can add codes of what we want to do in case of error. In this case we only return false when error happens.

  The expected behavior of this function is, it will return true or false whether the string is numeric or not and it will also store the integer value into the variable passed as second parameter on successful conversion in case of valid string.

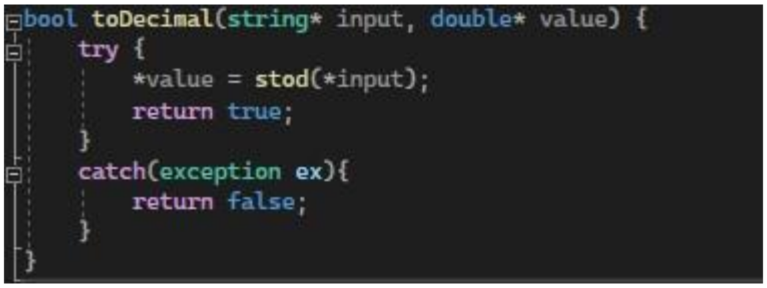
Example usage:



```
    break;
case 4:
    cout << "Insert yearOfBirth:";
    cin >> tmpinput;
    if (toInteger(&tmpinput,&newacc.yearOfBirth) && tmpinput.length() == 4) {

        //newacc.yearOfBirth = stoi(tmpinput);

        rgMenu.setValue(3, to_string(newacc.yearOfBirth));
    }
    else {
        cout << "Input for year of birth must be number with 4 digit";
        _getch();
    }
    break;
```

- Notice that we added (&) before the variable we pass into this function because the required parameters are pointed to variable address, not value. & symbol means address of. For example, &tmpinput means address of tmpinput. So, when we pass the address instead of value, the function will know where the data is actually stored and changes the value inside that same memory where the variable stores the data. Generally, this is called passing by reference which basically passing the exact same variable into the parameter.

Now applying the same logic, change the stoi to stod(string to double) to create a validation function for number with decimal places.



```
bool toDecimal(string* input, double* value) {
    try {
        *value = stod(*input);
        return true;
    }
    catch(exception ex){
        return false;
    }
}
```

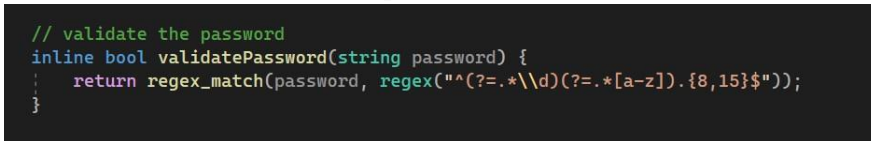
You can even use stof with float with the same logic for float value.

3.2. Regular Expression (Validation)

Regex is a widely used validation technology not only in C++ but also on most other programming languages. Regex is basically defining a pattern in which you want to string to match using a specific symbol and syntax to form an expression. This expression is then used to check against user input to see if it matches.

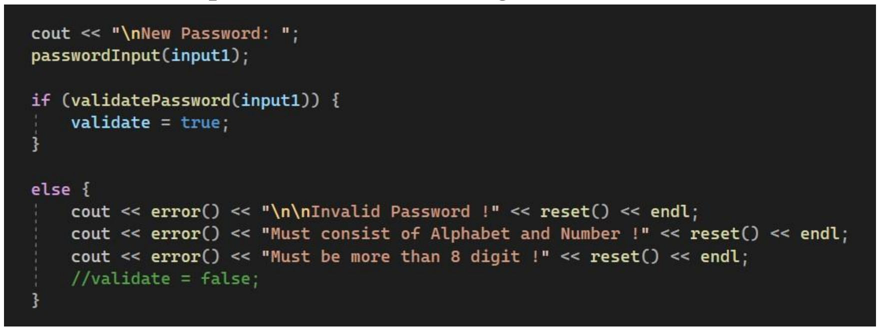
A Regular Expression (or Regex) is a pattern (or filter) that describes a set of strings that matches the pattern. In other words, a regex accepts a certain set of strings and rejects the rest. By using regex, we wouldn't have to code lot of chained ifs statement to validate an input since we can craft an expression and simply attempt to match the input with it. This is very useful to validate format sensitive inputs such as password, date and email.

- To utilize regex in C++, you need to include this library.
  #include <regex>
- Create a function/method for specific validation.



```
// validate the password
inline bool validatePassword(string password) {
    return regex_match(password, regex("^(?=.*\\d)(?=.*[a-z]).{8,15}$"));
}
```

- This example tests a password string to ensure it must contain alphabets, characters, and must have 8-15 characters only.
- Here is the example of the function usage:

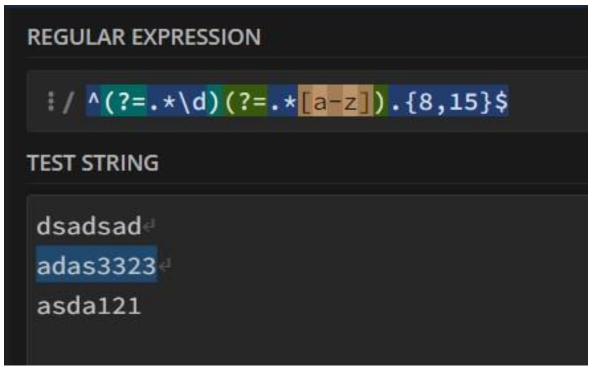


```
cout << "\nNew Password: ";
passwordInput(input1);

if (validatePassword(input1)) {
    validate = true;
}

else {
    cout << error() << "\n\nInvalid Password !" << reset() << endl;
    cout << error() << "Must consist of Alphabet and Number !" << reset() << endl;
    cout << error() << "Must be more than 8 digit !" << reset() << endl;
    //validate = false;
}
```

Instead of using multiple if to check for type and format, 1 line of code is enough for Regex. Since we make it as a function it can be re used for example on register and profile etc.

Regex is a very robust and convenient solution to validate user input and is widely used. Thus, we don't actually need to "re-invent the wheel" since there are lot of expressions out there ready to be used and some are tested and endorsed by many. Some of the reliable sources of expressions are https://regexlib.com where you can explore some Regex expression already made and used by others.

Although lot of Regex are readily available, it is obviously important to understand the inner working of Regex expressions itself since not all the Regex available online are 100% perfect and some might not fulfil your system needs. https://regex101.com is one of the most convenient websites where you can craft and test your expressions equipped with sufficient explanation and guide.



```
/ ^(?=.*\d)(?=.*[a-z]).{8,15}$
```

```
dsadsad
adas3323
asda121
```

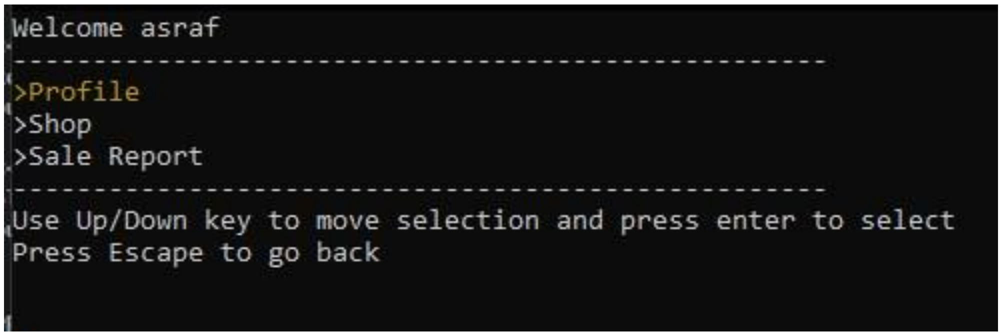
3.3. Arrow Menu

Code for this part is only available in "Additional" branch in the GitHub Repository

One of the challenges in creating a menu in CLI (command line interface) is the limitation of the CLI itself. Most implementations use simple menu selected using numbered option like what we did in the other part of the guide. However, this way of creating a menu has a clear limitation which is the number of options that you can add. In our example menu class, we use 1 digit for the option which means the menu can work for maximum of 10 options from 0 to 9.

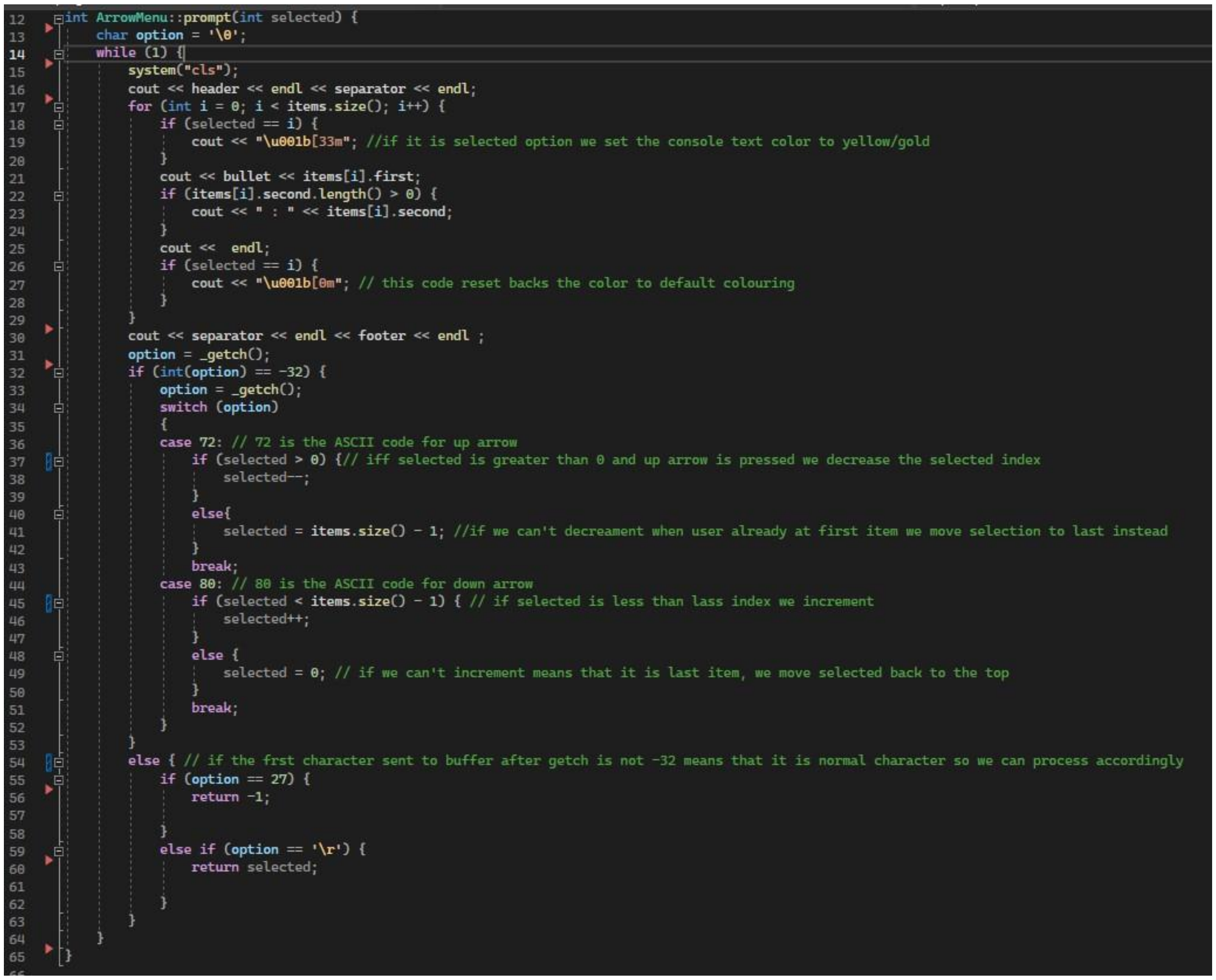
In this section, as an additional improvement, we will implement a new type of menu which will use arrow keys to navigate through the options instead of selecting using user input value. The primary purpose of this menu is to limit the display appropriately and allow any number of options to be shown.

The following diagram is an example output of what we are trying to develop in this section. The general idea is to display the menu while highlighting the option with different color then wait for user to press arrow keys, enter or escape.



```
Welcome asraf
--------------------------------------------------
>Profile
>Shop
>Sale Report
--------------------------------------------------
Use Up/Down key to move selection and press enter to select
Press Escape to go back
```

Most of the other behavior of this class is very similar to the Menu class. The only significant difference is in the prompt method. Refer to the source code in the GitHub repository in "Additional" branch to get clearer view.



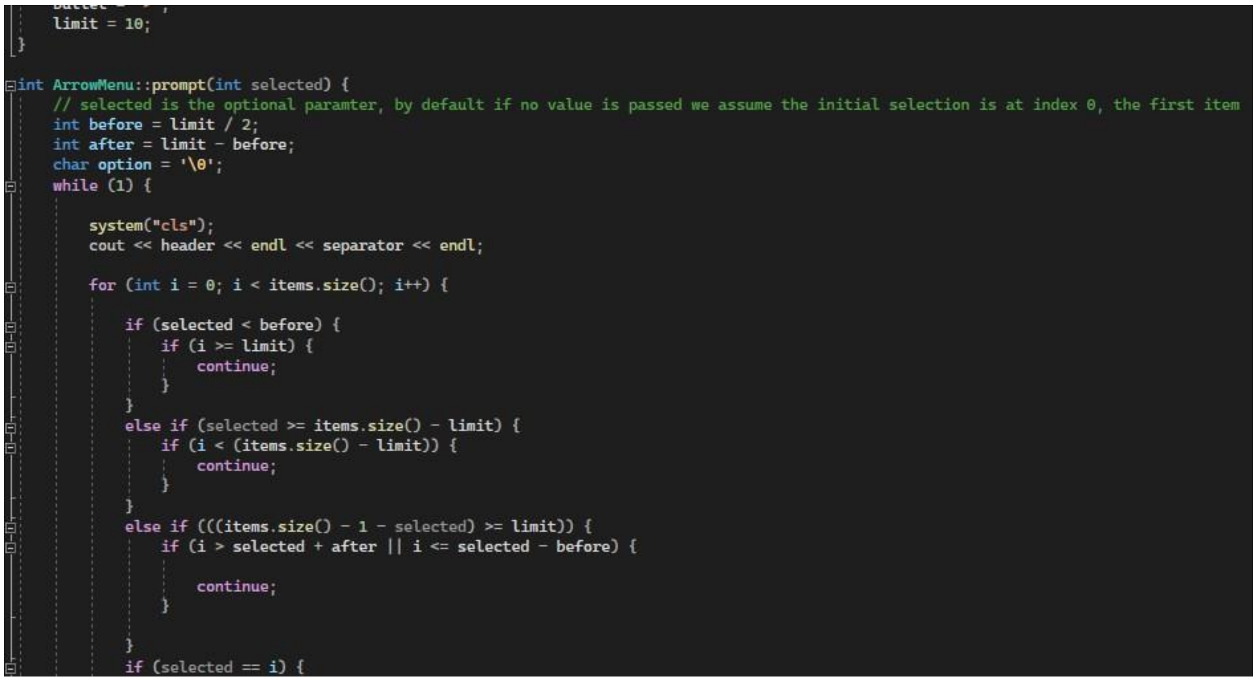
```cpp
int ArrowMenu::prompt(int selected) {
    char option = '\0';
    while (1) {
        system("cls");
        cout << header << endl << separator << endl;
        for (int i = 0; i < items.size(); i++) {
            if (selected == i) {
                cout << "\u001b[33m"; //if it is selected option we set the console text color to yellow/gold
            }
            cout << bullet << items[i].first;
            if (items[i].second.length() > 0) {
                cout << " : " << items[i].second;
            }
            cout <<  endl;
            if (selected == i) {
                cout << "\u001b[0m"; // this code reset backs the color to default colouring
            }
        }
        cout << separator << endl << footer << endl ;
        option = _getch();
        if (int(option) == -32) {
            option = _getch();
            switch (option)
            {
            case 72: // 72 is the ASCII code for up arrow
                if (selected > 0) {// iff selected is greater than 0 and up arrow is pressed we decrease the selected index
                    selected--;
                }
                else{
                    selected = items.size() - 1; //if we can't decreament when user already at first item we move selection to last instead
                }
                break;
            case 80: // 80 is the ASCII code for down arrow
                if (selected < items.size() - 1) { // if selected is less than lass index we increment
                    selected++;
                }
                else {
                    selected = 0; // if we can't increment means that it is last item, we move selected back to the top
                }
                break;
            }
        }
        else { // if the frst character sent to buffer after getch is not -32 means that it is normal character so we can process accordingly
            if (option == 27) {
                return -1;

            }
            else if (option == '\r') {
                return selected;

            }
        }
    }
}
```

- Firstly, we initialize a char variable to store the user input.
- Then we loop through the items to display each of them.
- If the index of the item equals the current index, we highlight it with yellow color.
- \u001b[33m tells the console to start printing text in yellow.
- \u001b[0m tells the console to start printing text in default color (reset).
- Getch is used to wait and read user keypress. The difference here is we utilized more ASCII codes. To understand more about ASCII you can refer to other sources but basically it is the code that represents each character.
- -32 is the escape character which we use to identify if user pressing normal or special characters.
- 72 and 80 are the ASCII code of up and down arrow key. The actual characters sent to input buffer when arrow keys are pressed are like -32,72 and -32,80 because in

ASCII table, the code for 72 and 80 overlaps with normal characters. Hence the needs of -32 scape character identifier.

- 27 is ASCII code for the Escape button on keyboard which we use to break the loop and return -1 to indicate that user cancel selection.
- Option == ‘\r’ will detect user key press on Enter key which indicates that user would like to proceed with current selection.

Then by improving the code with the following addition, we can limit the menu display so that not all the data will be displayed at once. Instead the data will be limited and user can scroll through the data using the selection.



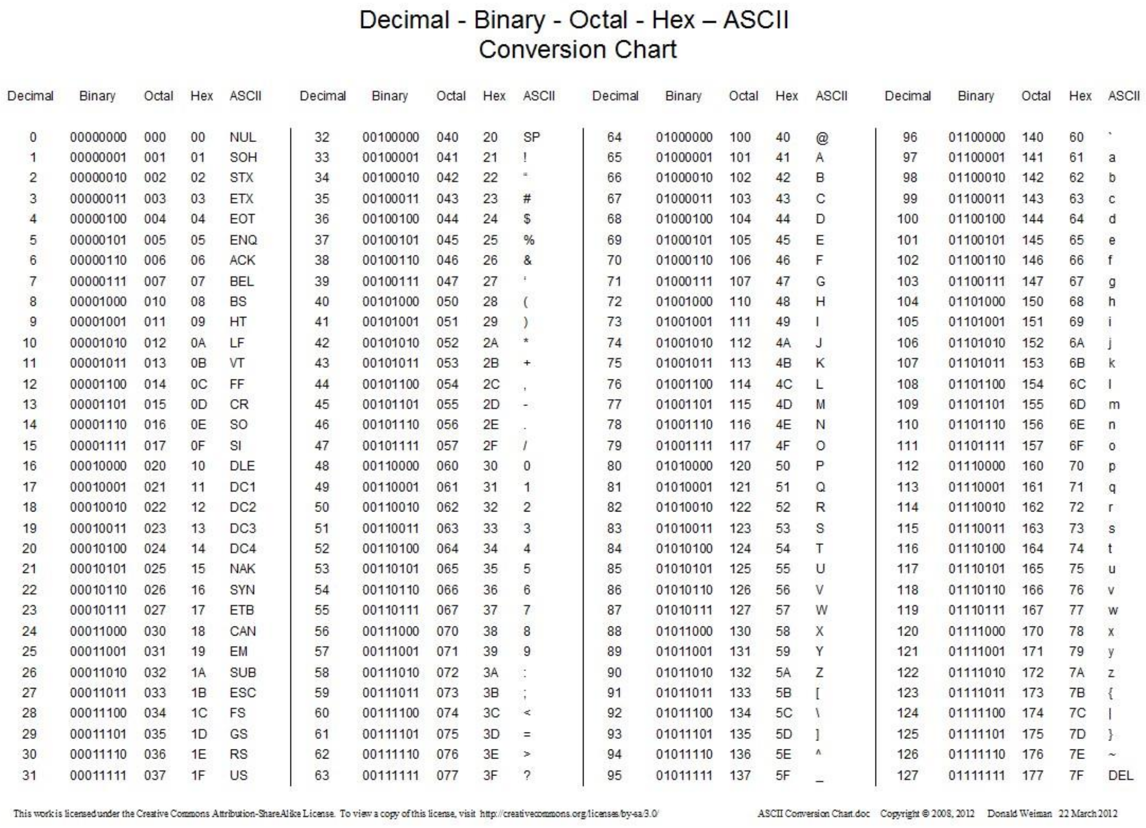
```
    limit = 10;
}

int ArrowMenu::prompt(int selected) {
    // selected is the optional paramter, by default if no value is passed we assume the initial selection is at index 0, the first item
    int before = limit / 2;
    int after = limit - before;
    char option = '\0';
    while (1) {

        system("cls");
        cout << header << endl << separator << endl;

        for (int i = 0; i < items.size(); i++) {

            if (selected < before) {
                if (i >= limit) {
                    continue;
                }
            }
            else if (selected >= items.size() - limit) {
                if (i < (items.size() - limit)) {
                    continue;
                }
            }
            else if (((items.size() - 1 - selected) >= limit)) {
                if (i > selected + after || i <= selected - before) {

                    continue;
                }

            }
            if (selected == i) {
```

- A new attribute limit is added.
- Before displaying we calculate the number of items to display before and after the selected index.
- Then 3 if conditions are added inside the loop.
- The first if handles when selection is at the beginning of the items list.
- Second if handle when selection is at the end of the item list.
- Last if handle when selection is somewhere else in the middle.
- The idea is to use continue, to skip items at indexes that are outside of the limit that we want to display.
- For example, if selection is at index 50 of total 100 item, we only want to display 45 until 65.

3.4. Hidden Input and ASCII codes

ASCII stands for American Standard Code for Information Interchange. It is globally standardized codes on how computers understand a character. On machine level, computer only understand binary bits consisting of 1 and 0. To represent the characters we know in our life to computer understandable format, we need to convert it into binary which is why ASCII is necessary.

ASCII already stated the number which represents a character for all characters for example 48 represents character ‘0’. This means that our computer will understand characters ‘0’ as 48 but not in decimal, computer will read it in binary bits format consisting of 8 bits which is 00110000 that equals to 48 in decimal and represent character ‘0’ according to ASCII standard. The following table shows the mapping of value for each character where the ASCII column is the character that we understand and hex, octa, binary and decimal is its representation at machine level.

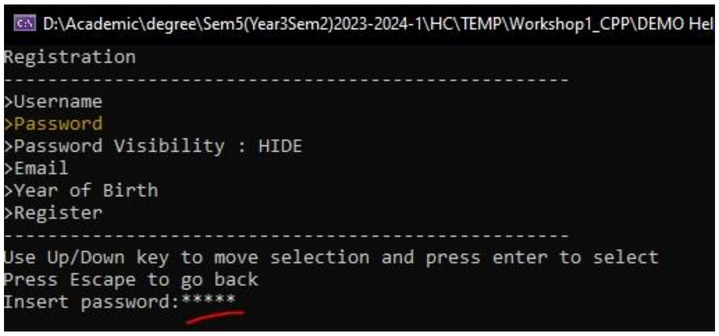


Decimal - Binary - Octal - Hex – ASCII
Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

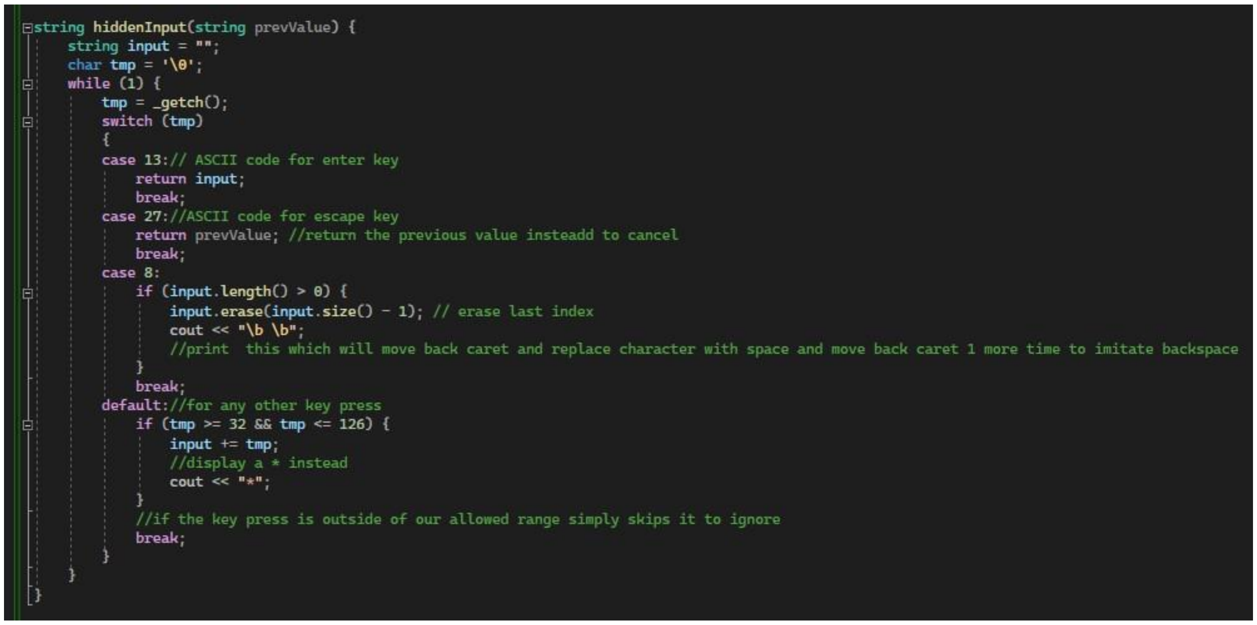
The table above only shows up to 127. However, nowadays we already have more ASCII codes for more special characters which makes ASCII code increased from 7 bits with maximum 128 characters to 8 bits to have maximum of 256 characters. You may google the “ASCII table”.

Knowing and understanding ASCII code and value comes in handy in programming since it provides us with more option to manipulate characters. This section will explain an alternative way to get input from user without displaying what the user is typing as commonly used for password input fields.



```
D:\Academic\degree\Sem5(Year3Sem2)2023-2024-1\HC\TEMP\Workshop1_CPP\DEMO Hel
Registration
--------------------------------------------------
>Username
>Password
>Password Visibility : HIDE
>Email
>Year of Birth
>Register
--------------------------------------------------
Use Up/Down key to move selection and press enter to select
Press Escape to go back
Insert password:*****
```

The general idea is to read user key press one by one as character and process it using its ASCII.



```cpp
string hiddenInput(string prevValue) {
    string input = "";
    char tmp = '\0';
    while (1) {
        tmp = _getch();
        switch (tmp)
        {
        case 13:// ASCII code for enter key
            return input;
            break;
        case 27://ASCII code for escape key
            return prevValue; //return the previous value insteadd to cancel
            break;
        case 8:
            if (input.length() > 0) {
                input.erase(input.size() - 1); // erase last index
                cout << "\b \b";
                //print  this which will move back caret and replace character with space and move back caret 1 more time to imitate backspace
            }
            break;
        default://for any other key press
            if (tmp >= 32 && tmp <= 126) {
                input += tmp;
                //display a * instead
                cout << "*";
            }
            //if the key press is outside of our allowed range simply skips it to ignore
            break;
        }
    }
}
```

- Firstly, we declare a string to hold our input value.
- Inside and infinite loop, we use _getch to wait for user key press and read it as characters which is passed into switch case.
- As you can see, eventhough we passed character into the switch, our case use integer number as case. This is possible because in C++ when characters are compared with integer, it will compare the ASCII decimal with that integer instead.
  • Case 13 is Enter, case 27 is Escape and case 8 is backspace (refer to ASCII table)
- When user press the stated key, the program goes into each specific case.
- When other key is pressed, it will go to default instead.

- In default we check the character, again comparing it with integer. 32 is space, and 126 is tilda (~). If you observer the ASCII table, 32 until 126 is basically the printable characters in the table. Thus, this condition means that we only append the character into our input string if user press any key that has ASCII decimal value from 32 and 126 which is all characters including digits and specials. Other key presses will be ignored.
- Changing the value in this if condition will allow you to customize your input function to accept what value. For example, 65 until 90 will only allow uppercase character, 48 until 57 allow only numbers.
- Unlike cin, getch does not display user key press by default. We utilize this feature to display our own placeholder to show asterisk (*) instead of what user actually press.

This section explained on how to handle input utilizing getch() and ACII code to have custom behavior in our input reader. The implementation and usage of ASCII code is very wide and not only limited to identifying user input.

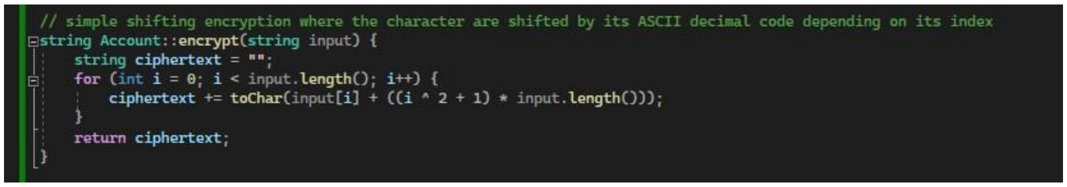
3.5. Password Encryption

Encrypting a password is not only best practice but something that legally developers has to do since regardless of who you are, even admin should never know users' password in most systems to promotes confidentiality and privacy of user information.

There are lot of cryptography technologies/algorithm out there which you can use to encrypt your passwords with such as Blowfish, AES, SHA, RSA etc. but for the sake of simplicity, in this example we will be using a very simple function to obfuscate a string by shifting its ASCII decimal value to substitute it to a different character.

For this example, we put the encryption method inside account class since it is the only class that needs it. Ideally, you would have a separate class for encryptor which is how most 3-rd party encryption libraries does.

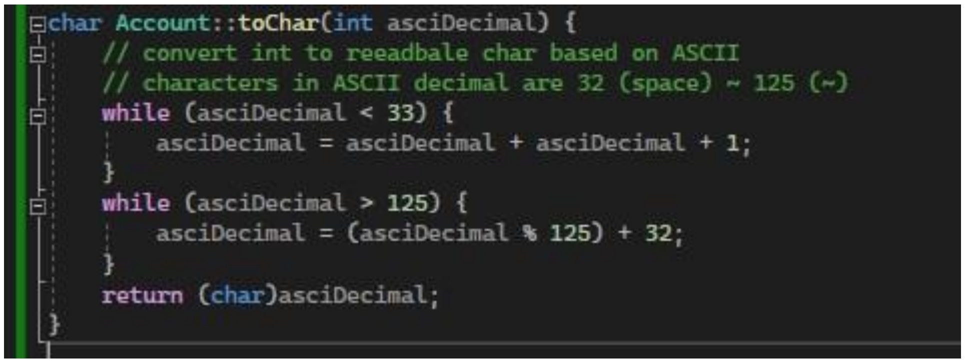
- Firstly, we need a method to modify a string changing each of its characters as follows:



```
// simple shifting encryption where the character are shifted by its ASCII decimal code depending on its index
string Account::encrypt(string input) {
    string ciphertext = "";
    for (int i = 0; i < input.length(); i++) {
        ciphertext += toChar(input[i] + ((i ^ 2 + 1) * input.length()));
    }
    return ciphertext;
}
```

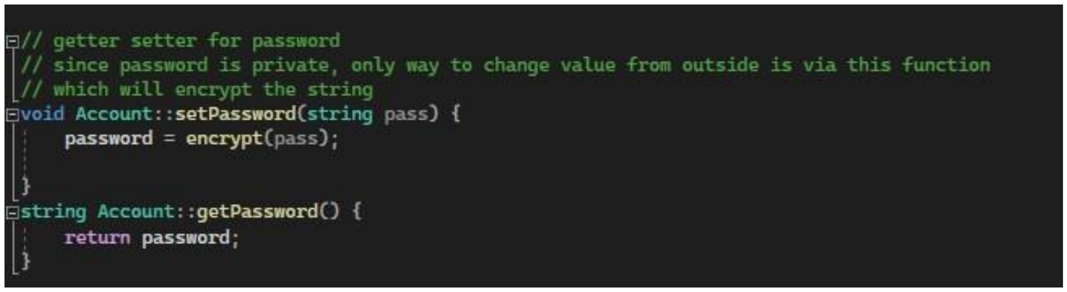
- Inside the loop, we use the index of the characters and the length of the string to create some irregular pattern of number which won't be exactly same for every string.

- We then add this number to the character at current index. When adding integer into character, what happens is, it will add to the ASCII decimal value instead. For example, '0' + 10. '0' decimal value in ASCII is 48. +10 will be 58 which is semi colon ;
- The toChar method is a safe method to convert integer into readable characters. We need this method since ASCII also include key such as enter, backspace etc. We don't want "backspace" to be part of password. Its not even valid or possible to begin with which is why we need to add 32 when the ASCII decimal to small to make sure it become readable character.
- Also, 8bits binary means ASCII has max value of 256. Since we only consider normal characters, we use 125 as the max. When the value exceed this, we divide by 125 and get the remainder and again + 32 to ensure it becomes readable character.



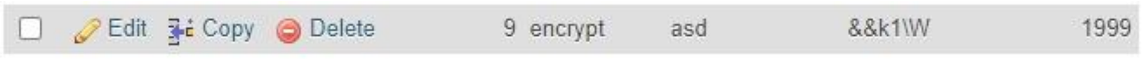
```
char Account::toChar(int asciDecimal) {
    // convert int to reeadbale char based on ASCII
    // characters in ASCII decimal are 32 (space) ~ 125 (~)
    while (asciDecimal < 33) {
        asciDecimal = asciDecimal + asciDecimal + 1;
    }
    while (asciDecimal > 125) {
        asciDecimal = (asciDecimal % 125) + 32;
    }
    return (char)asciDecimal;
}
```

- After the simple encryption, our text will no longer be the same as what user actually inserted. No one can tell the actual value at a glance. If you use some other better and robust algorithms it might even be unbreakable in term of security.
- Now, encapsulates the password attribute in Account class as private so that it can only be set via setter function in which we implement the encryption process.



```
// getter setter for password
// since password is private, only way to change value from outside is via this function
// which will encrypt the string
void Account::setPassword(string pass) {
    password = encrypt(pass);

}
string Account::getPassword() {
    return password;
}
```

Now that we have obfuscated the string, database admin will not be able to tell what user password is at a glance. But still this custom simple encryption is close to 0 security since it is easy to be cracked. If possible, you should opt to widely known cryptography algorithms.



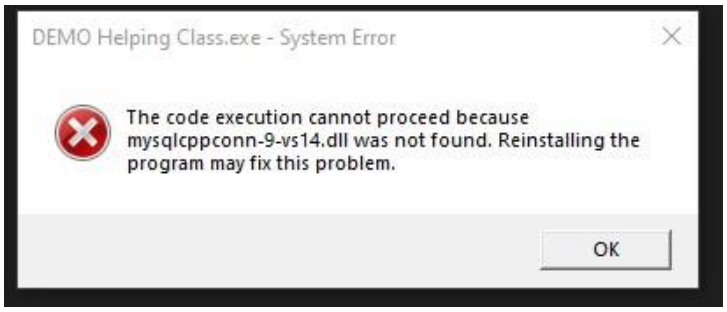
☐	Edit	Copy	Delete	9	encrypt	asd	&&k1\W	1999

Actual value of &&k1\w is asd123

Querying for login does not change in this example since we are using very straightforward encryption. But in case of using complex algorithms, you might need to query out the password value and do the matching comparison in coding instead of in database query.

3.6. mysqlcpp-9-vs14.dll not found Error.

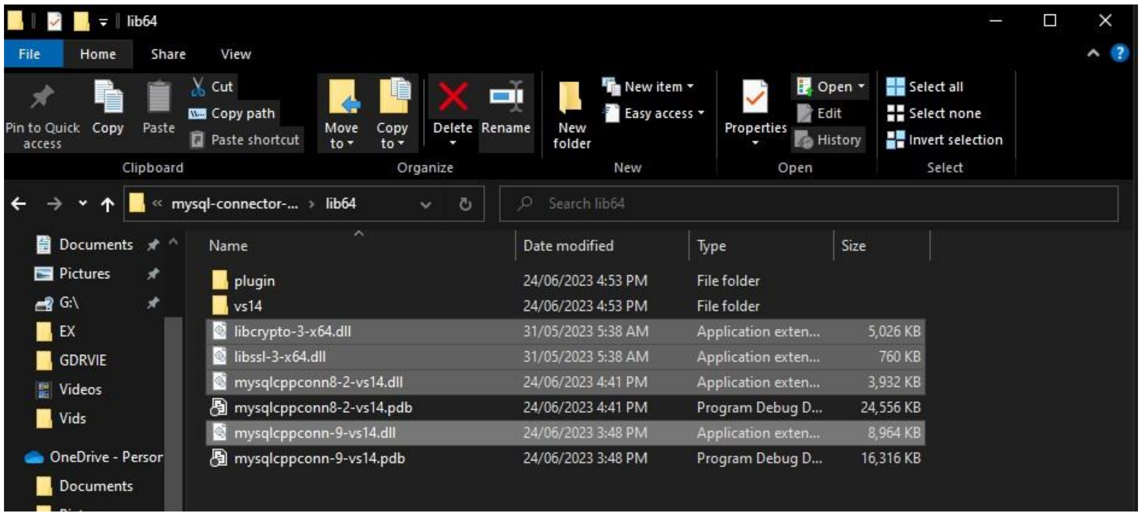
In case you encounter a similar error as shown in following diagram, you can continue to read this section which explains how to solve it.



DEMO Helping Class.exe - System Error

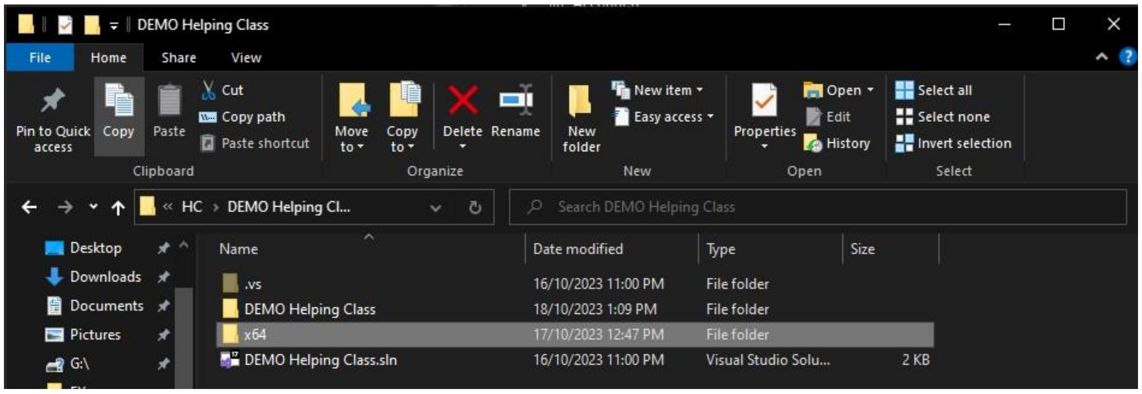
The code execution cannot proceed because mysqlcppconn-9-vs14.dll was not found. Reinstalling the program may fix this problem.

Locate the folder in which you extracted the downloaded MySQL connector and go into the lib64 folder to copy the .dll files.



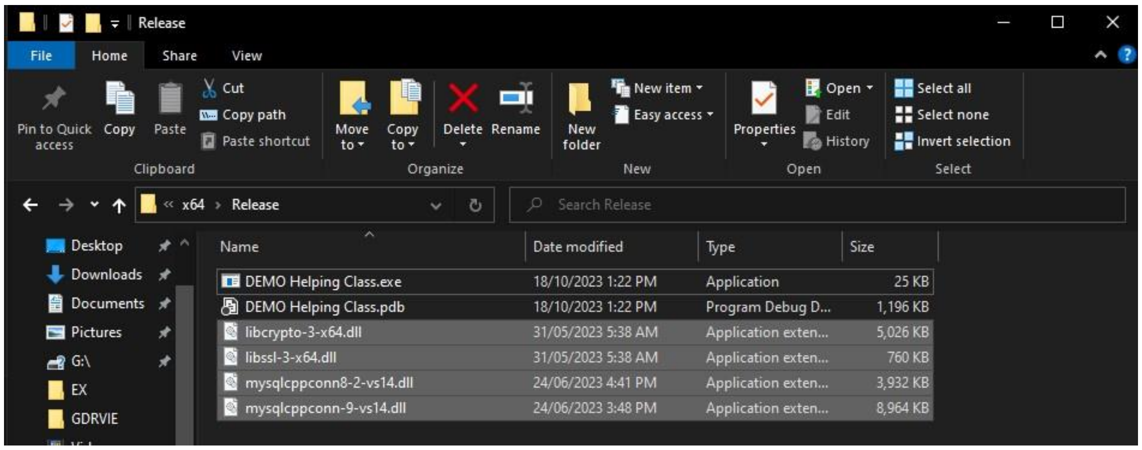
Name	Date modified	Type	Size
plugin	24/06/2023 4:53 PM	File folder	
vs14	24/06/2023 4:53 PM	File folder	
libcrypto-3-x64.dll	31/05/2023 5:38 AM	Application exten...	5,026 KB
libssl-3-x64.dll	31/05/2023 5:38 AM	Application exten...	760 KB
mysqlcppconn8-2-vs14.dll	24/06/2023 4:41 PM	Application exten...	3,932 KB
mysqlcppconn8-2-vs14.pdb	24/06/2023 4:41 PM	Program Debug D...	24,556 KB
mysqlcppconn-9-vs14.dll	24/06/2023 3:48 PM	Application exten...	8,964 KB
mysqlcppconn-9-vs14.pdb	24/06/2023 3:48 PM	Program Debug D...	16,316 KB

Go to your project folder root directory where you can see the project name.sln file and go into the x64 folder in that same directory.



Name	Date modified	Type	Size
.vs	16/10/2023 11:00 PM	File folder	
DEMO Helping Class	18/10/2023 1:09 PM	File folder	
x64	17/10/2023 12:47 PM	File folder	
DEMO Helping Class.sln	16/10/2023 11:00 PM	Visual Studio Solu...	2 KB

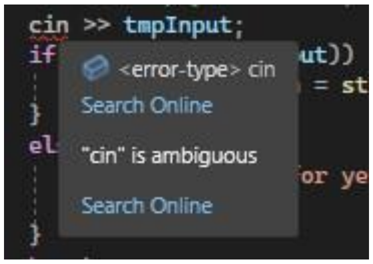
There should be a Release folder in the x64 directory, if not you can create one with the same name. Paste the .dll files you have copied here.



Name	Date modified	Type	Size
DEMO Helping Class.exe	18/10/2023 1:22 PM	Application	25 KB
DEMO Helping Class.pdb	18/10/2023 1:22 PM	Program Debug D...	1,196 KB
libcrypto-3-x64.dll	31/05/2023 5:38 AM	Application exten...	5,026 KB
libssl-3-x64.dll	31/05/2023 5:38 AM	Application exten...	760 KB
mysqlcppconn8-2-vs14.dll	24/06/2023 4:41 PM	Application exten...	3,932 KB
mysqlcppconn-9-vs14.dll	24/06/2023 3:48 PM	Application exten...	8,964 KB

Now when you compile your program it should be able to detect the necessary .dll files.

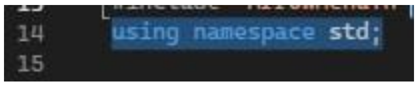
3.7. Cin / Cout ambiguous error



```
cin >> tmpInput;
```

This error is a very common error when developing C++ applications using visual studio. It usually happens at random. This section will show one of the possible fixes which is simple.

- In your file, find the line where you put using namespace std;



```
using namespace std;
```

- Cut this line.
- Save your file.
- Now paste it back.
- Should've fixed the ambiguous error.

If this method does not work, you might have to find an alternative on your own for your specific issue.