COMP 244: Database Management Systems

Fall 2014

MySQL Connection Handout

1. **Python Connector:**

**Part 1: Installation**

Download and install the following library:

<http://dev.mysql.com/downloads/connector/python/>

Choose the following option:

Windows (Architecture Independent), MSI Installer Python 3.3

Once the installation is complete, you are done.

**Part 2: Testing the connector**

Open idle and type the following code:

import mysql.connector

cnx = mysql.connector.connect(user='uXXXXXX', password='pXXXXXX',

host='COMPDBS300',

database='schemaXXXXXX')

cnx.close()

Replace in the above code every XXXXXX with your six-digit ID number.

**Part 3: Sample code reading data from the database:**

#import the datetime package to use dates

import datetime

#import the mysql connector to access the database

import mysql.connector

#establish a connection to the database

cnx = mysql.connector.connect(user='uXXXXXX', password='pXXXXXX',

host='COMPDBS300', database='schemaXXXXXX')

#retrieve a cursor to execute queries

cursor = cnx.cursor()

#setup a variable for the query text

query = ("SELECT account\_no, startdate, balance from account")

#execute the query

cursor.execute(query)

# read the results line-by-line and print the results

for (account\_no, startdate, balance) in cursor:

print(str(account\_no) + "\t" + str(startdate) + "\t"+str(balance))

#close the cursor

cursor.close()

#close the connection to the DBMS

cnx.close()

**Part 4: Sample code updating the database**

#import the datetime package to use dates

import datetime

#import the mysql connector to access the database

import mysql.connector

#establish a connection to the database

cnx = mysql.connector.connect(user='uXXXXXX', password='pXXXXXX',

host='COMPDBS300', database='schemaXXXXXX')

#retrieve a cursor to execute queries

cursor = cnx.cursor()

#setup a variable for the query text

query = ("update account set balance = %s where account\_no = %s")

#a tuple of values to fill in the %s values which are missing

qdata = (200.00, '121234345610')

try:

#execute the query

cursor.execute(query,qdata)

# get the number of rows updated

numrows = int(cursor.rowcount)

print(str(numrows) + " were updated")

# Make sure data is committed to the database

cnx.commit()

except mysql.connector.Error as err:

print("Update was not successful " + str(err))

#close the cursor

cursor.close()

#close the connection to the DBMS

cnx.close()

1. **C++ Connector:**

**Part 1: Libraries needed**

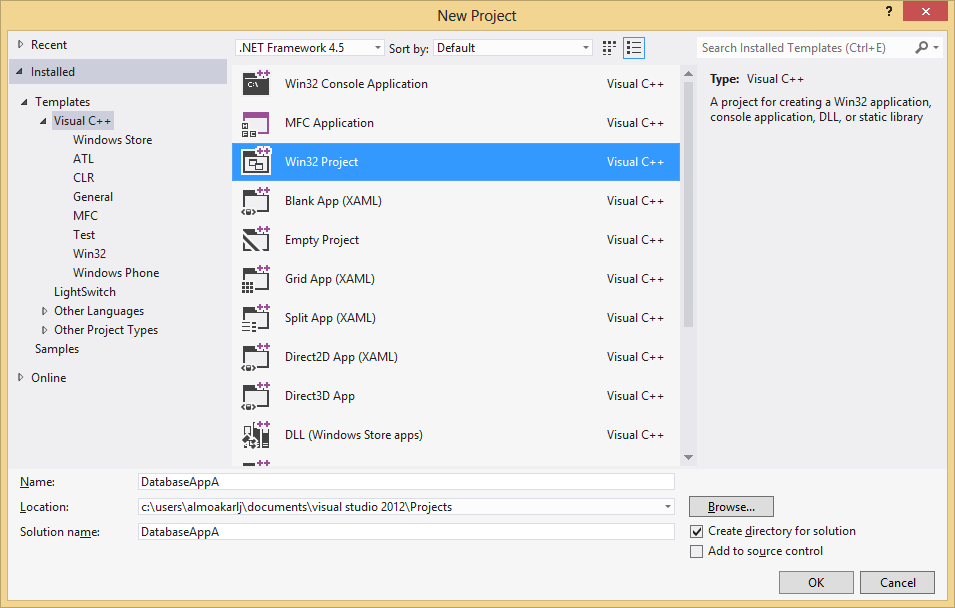
Download and Install the following libraries:

* MySQL C connector (32 bit):
  + <http://dev.mysql.com/downloads/connector/c/>
  + **Windows (x86, 32-bit), MSI Installer**
* MySQL C++ connector (32 bit):
  + <http://dev.mysql.com/downloads/connector/cpp/>
  + **Windows (x86, 32-bit), MSI Installer**
* Boost library
  + <http://www.boost.org/users/history/version_1_56_0.html>
  + **boost\_1\_56\_0.zip**

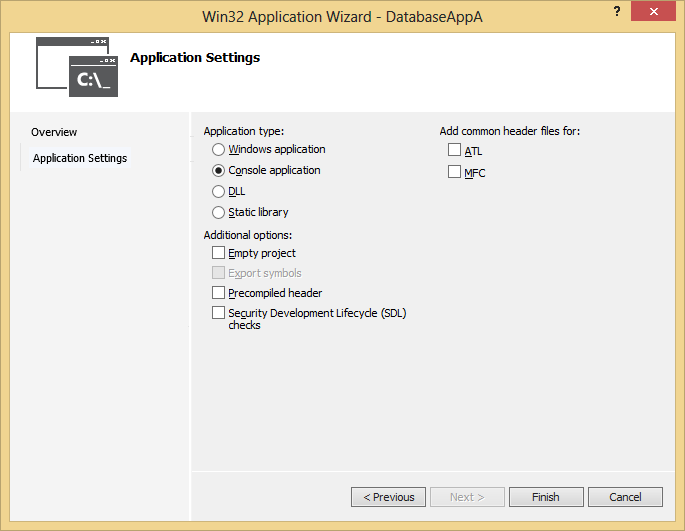
**Part 2: Visual Studio Configuration**

Follow the following steps:

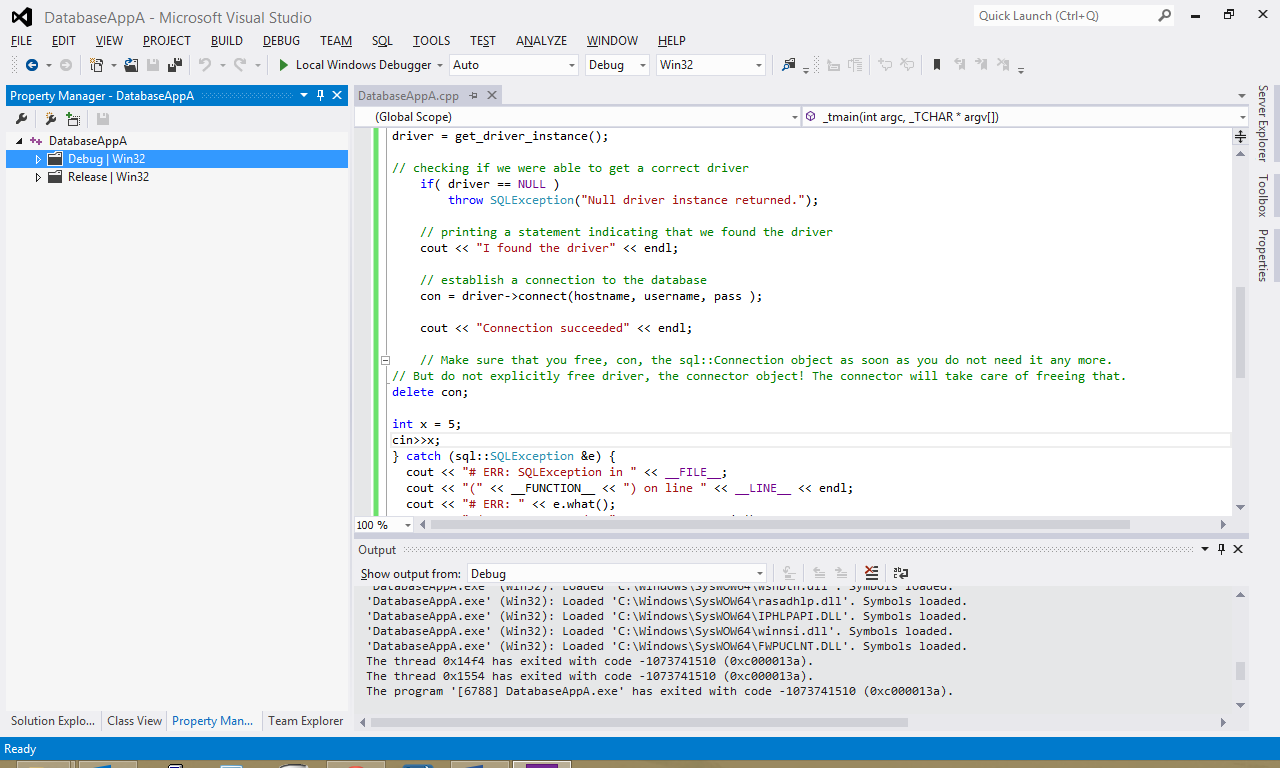
1. Create a new project, select **Visual C++, Win32 Project**



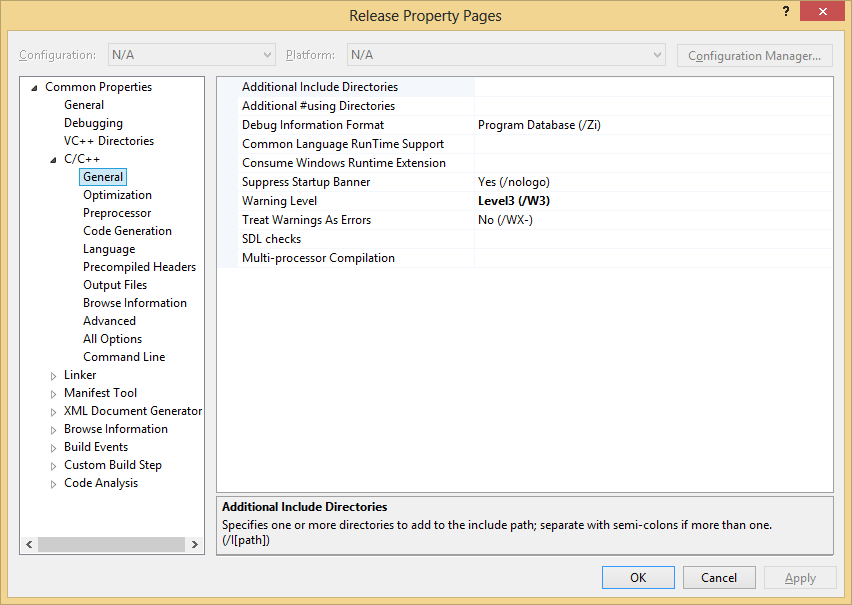
1. In the wizard - application settings, select the application type **Console application**.



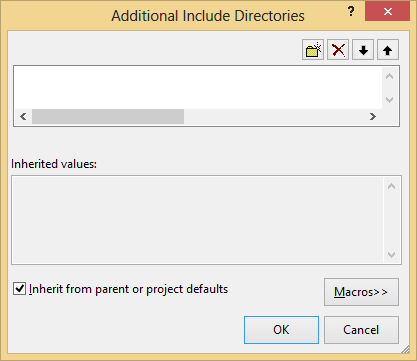
1. Choose the **property manager** tab on the left as shown below and double click on **debug**



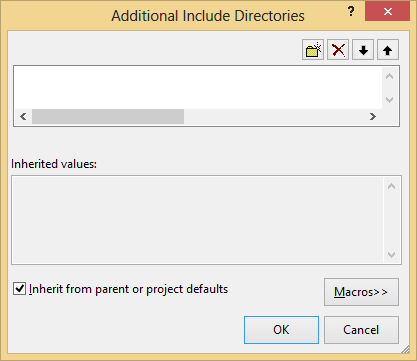
1. Select **C++**, **General** in the tree view.



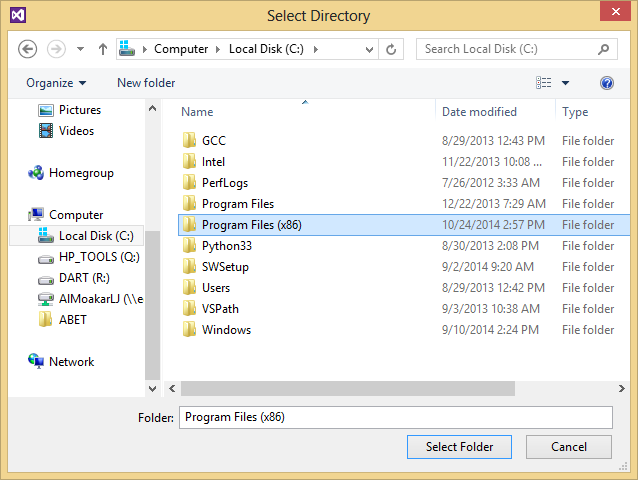
1. Edit **Additional Include Directories** text field, to get the following window

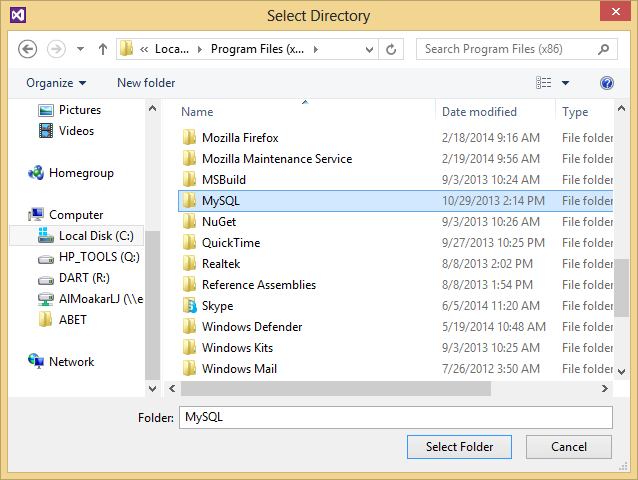


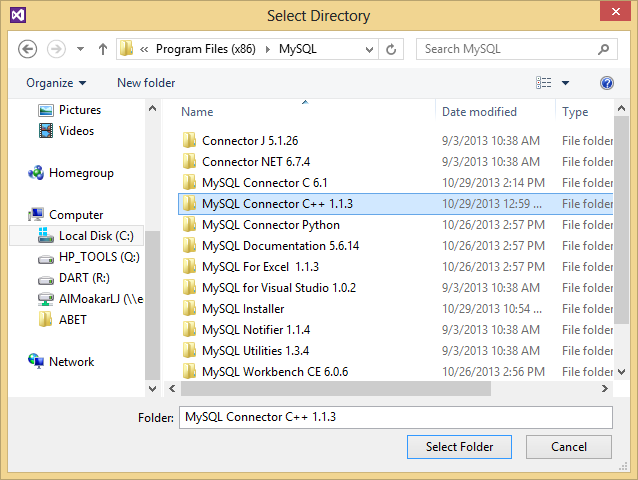
1. add the MySQL C++ connector include\ directory, follow the following steps:

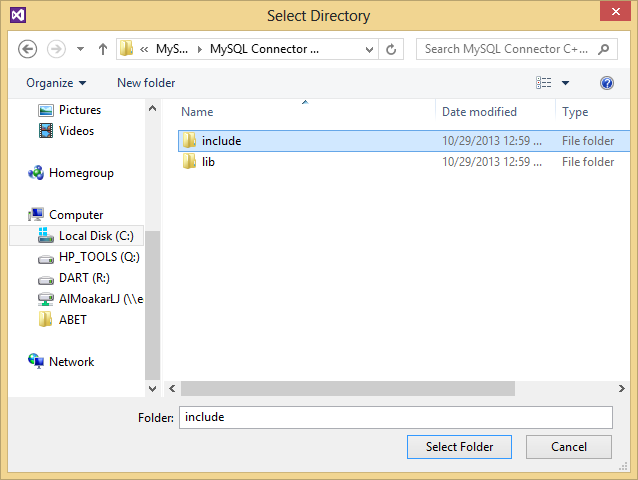




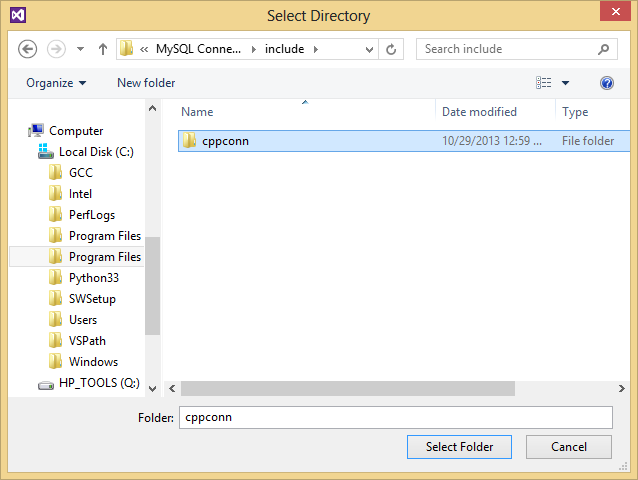


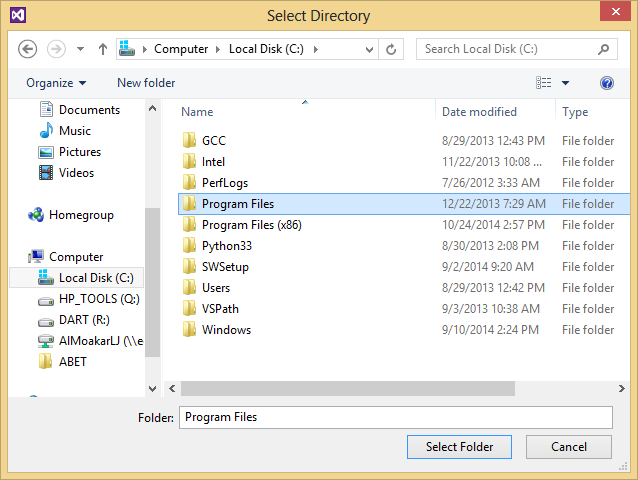


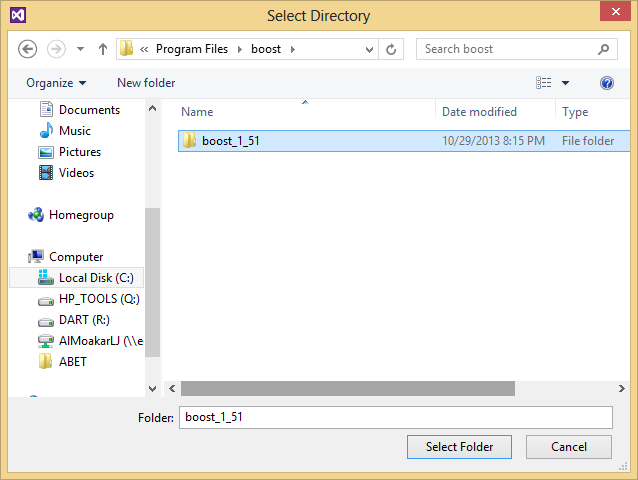


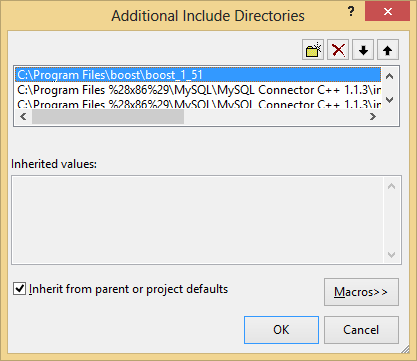


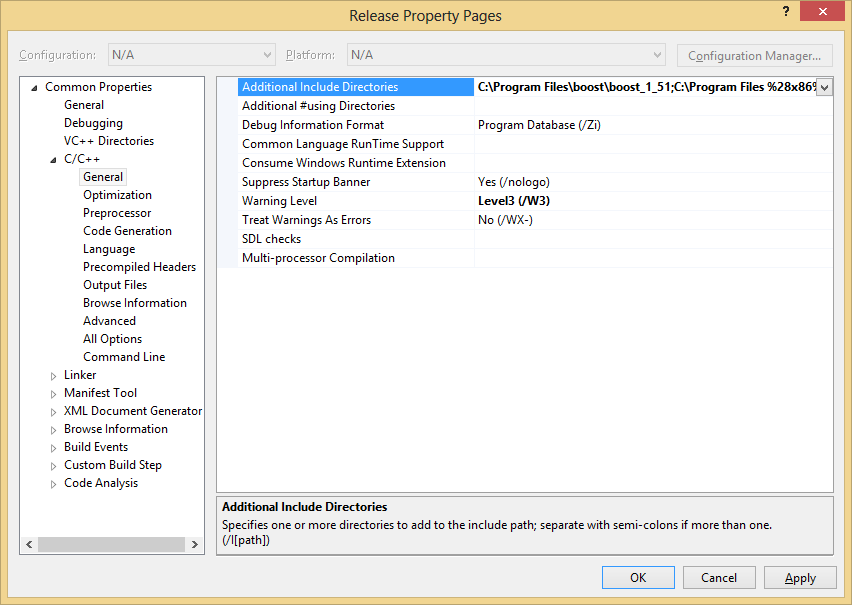
1. add the MySQL C++ connector include\cppconn directory. Follow same steps as number 6 but double click on include to get the following:



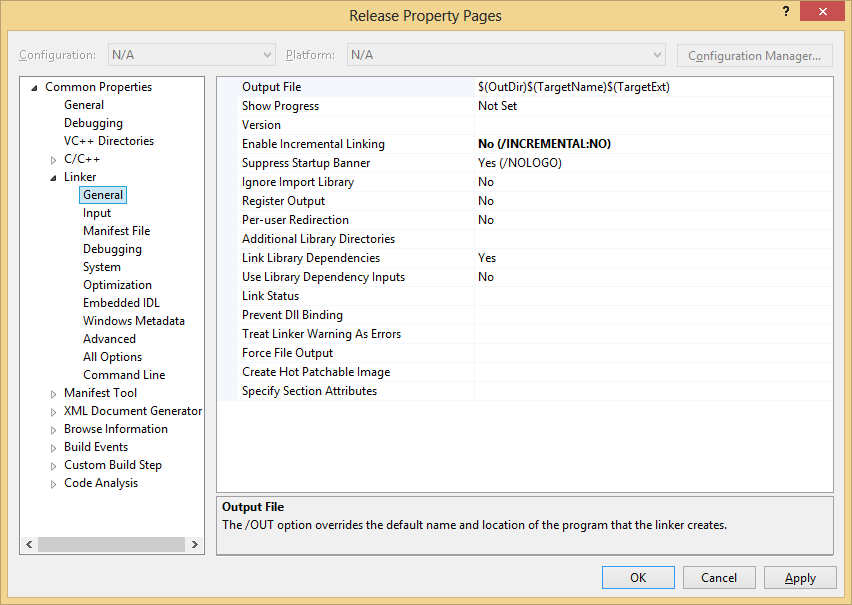
1. add the boost main directory path by following the steps below



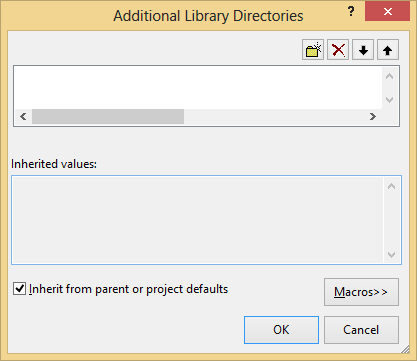


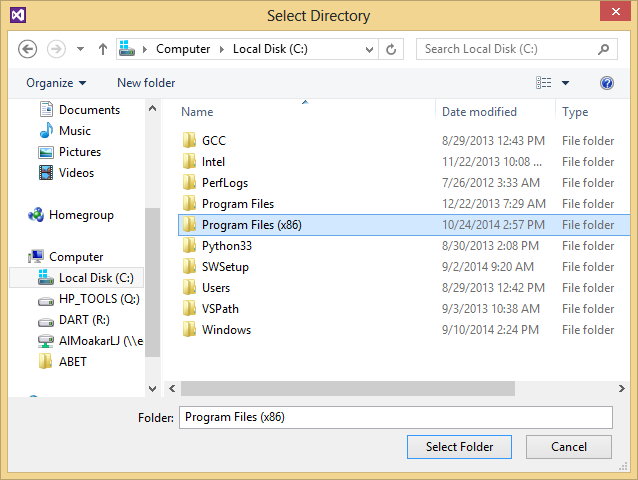


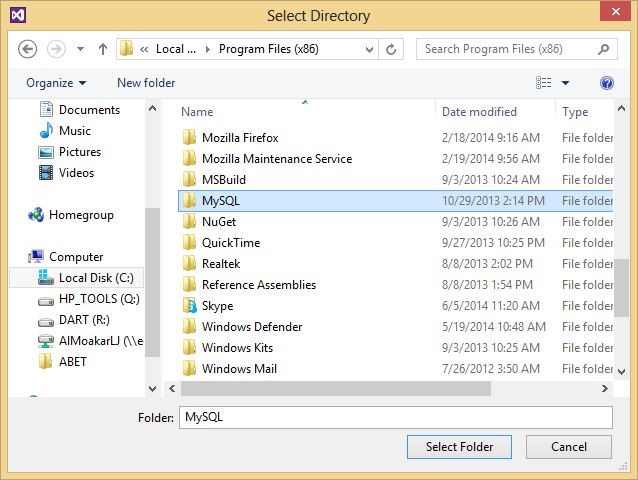
1. In the tree view, open **Linker**, **General**, **Additional Library Directories**.

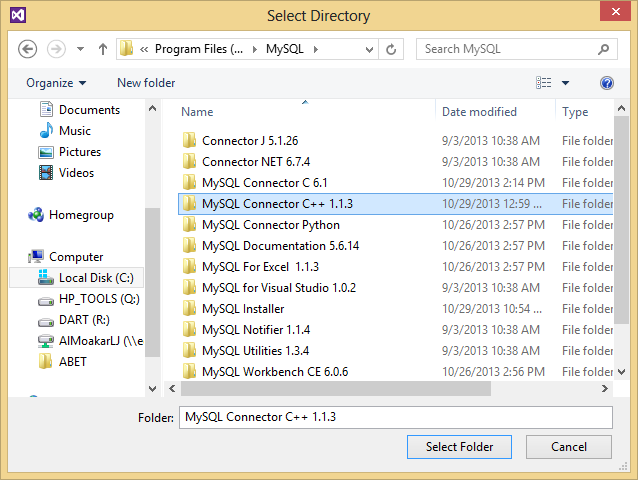


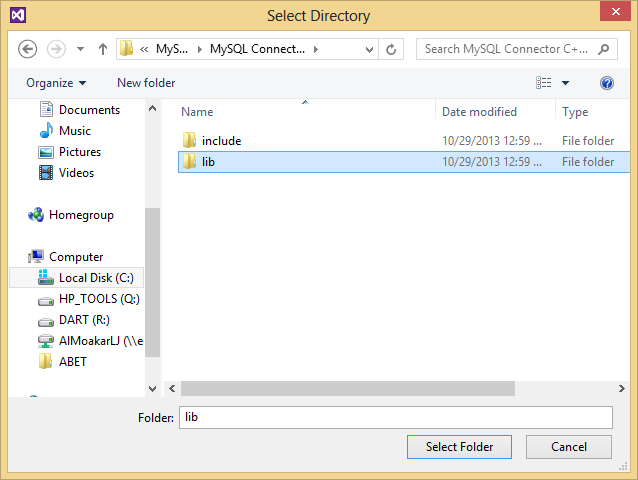
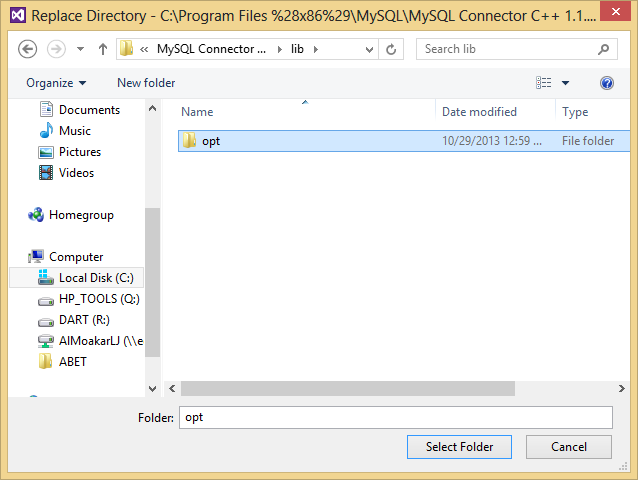
1. Add the MySQL C++ connector lib\opt directory by following the steps below



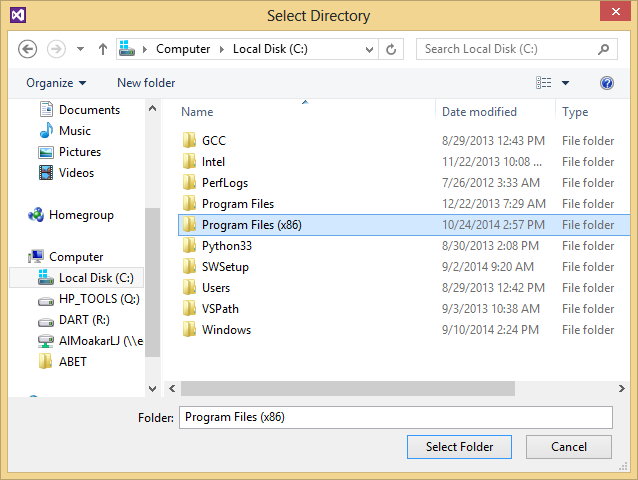
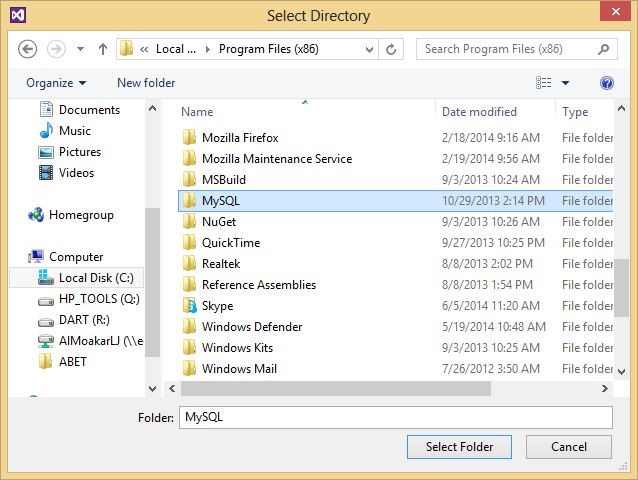


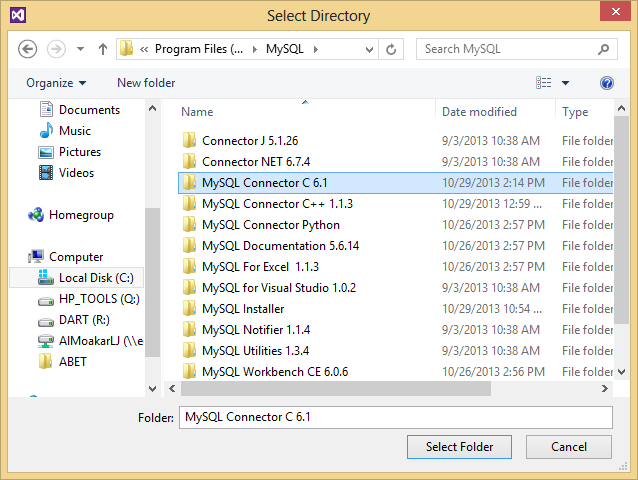


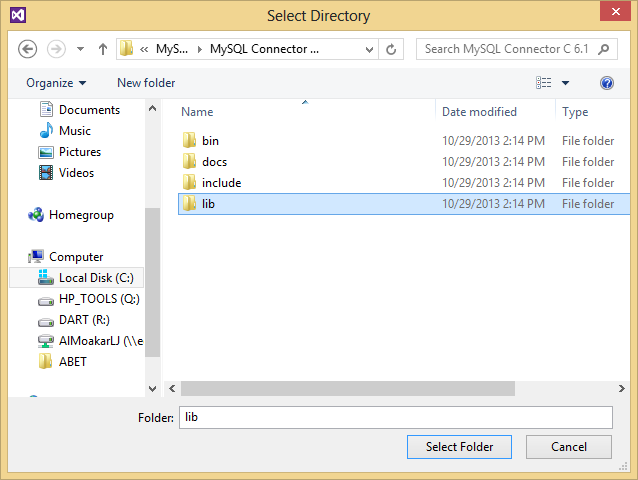


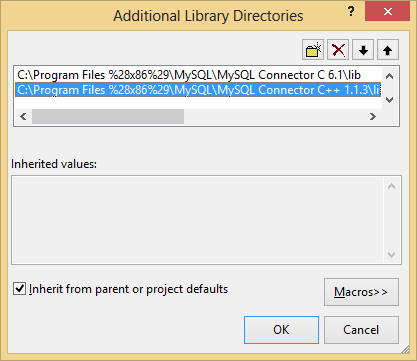


1. Add the MySQL C connector lib directory path

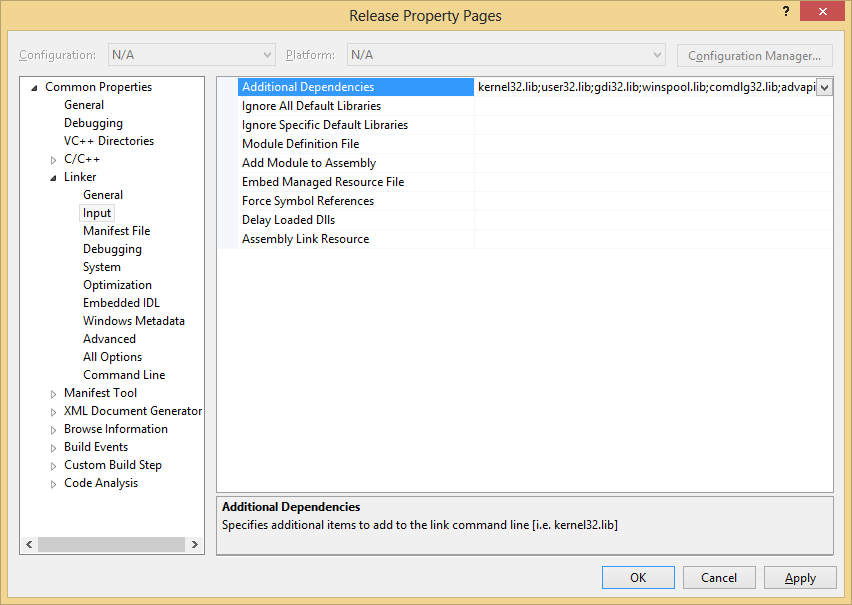


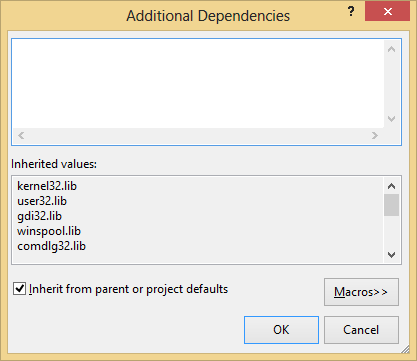






1. Under **Linker**, **Input**, add mysqlcppconn.lib into the **Additional Dependencies** text field.





1. Move mysqlcppconn.dll and libmysql.dll into the folder where the executables is located

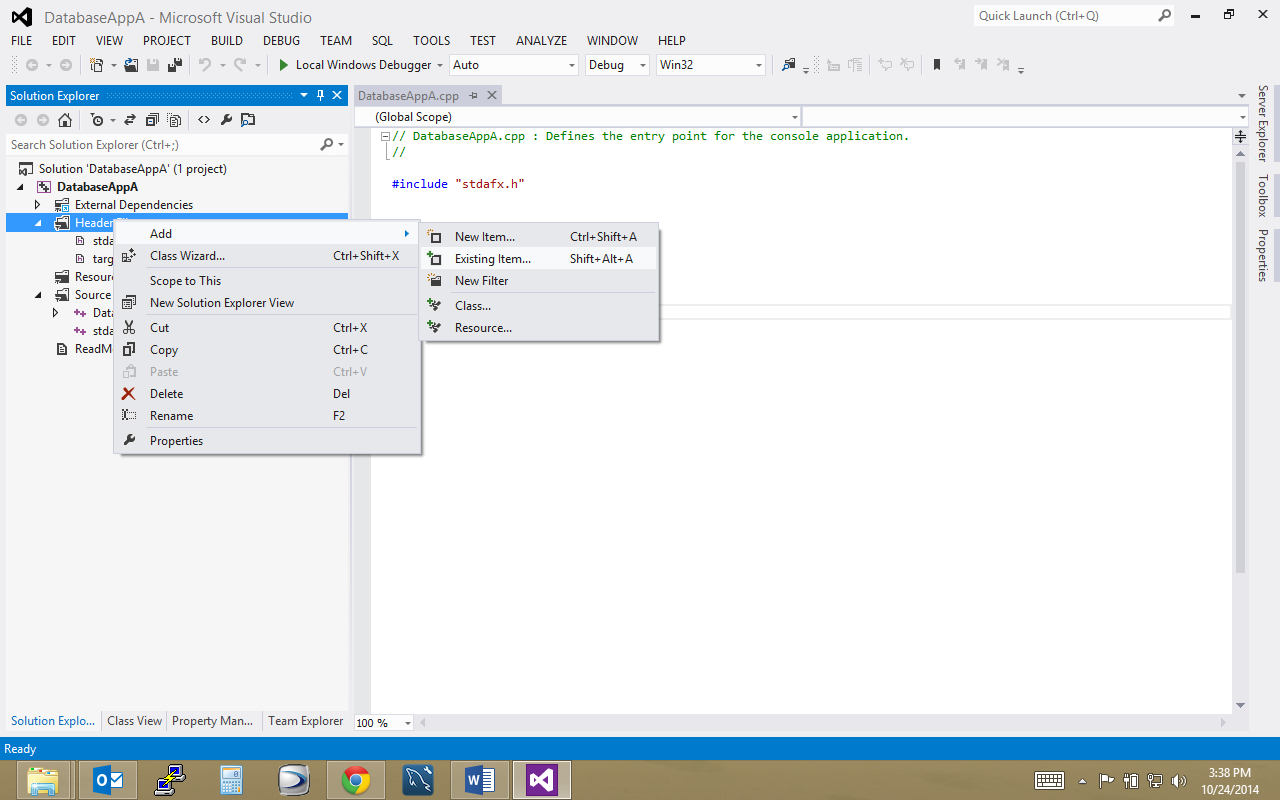
The path to mysqlcppconn.dll should be similar to:

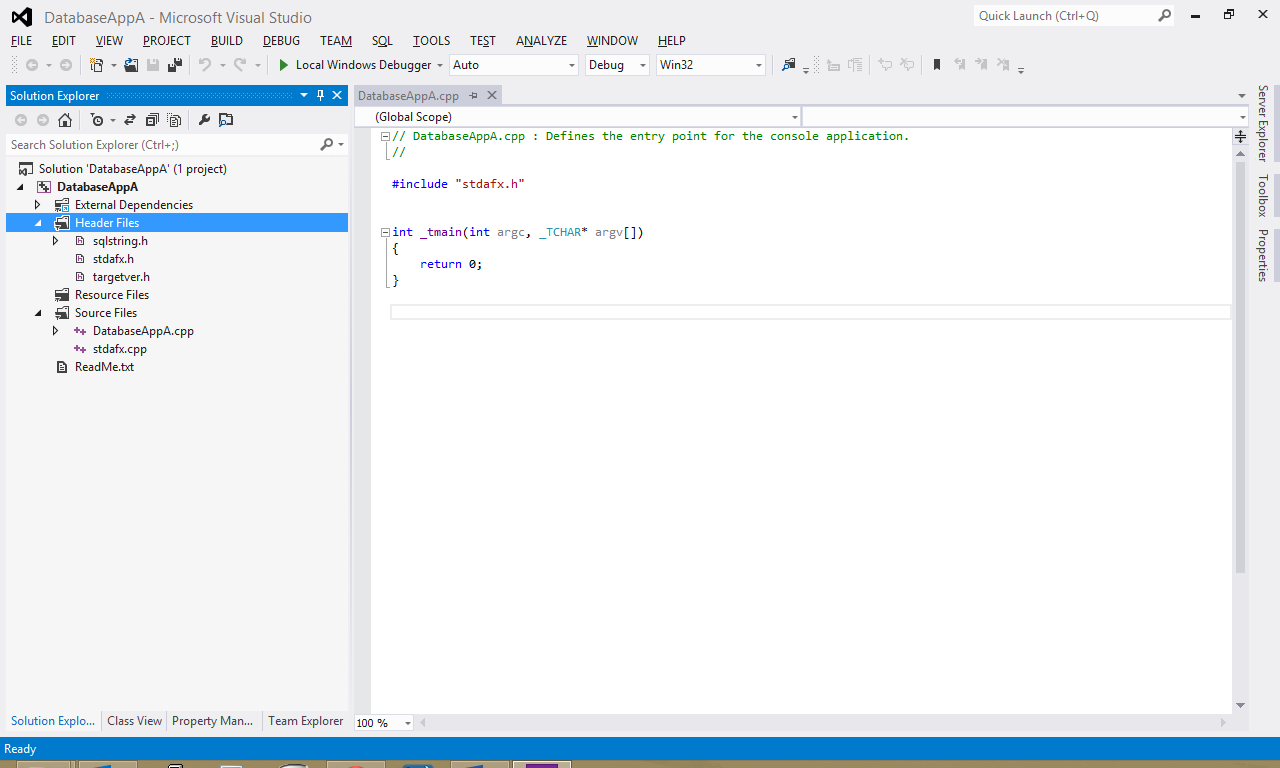
C:\Program Files (x86)\MySQL\MySQL Connector C++ 1.1.3\lib\opt

The path to libmysql.dll should be similar to:

C:\Program Files (x86)\MySQL\MySQL Connector C 6.1\lib

1. Copy sqlstring.h into the project folder and add it to the project





**Part 3: Testing the MySQL C++ connector:**

Edit the cpp file. Below is a basic program that connects to the database:

#include "stdafx.h"

#include <stdlib.h>

#include <iostream>

#include "mysql\_connection.h"

#include "mysql\_driver.h"

#include <cppconn/exception.h>

#include <cppconn/resultset.h>

#include <cppconn/statement.h>

using namespace std;

using namespace sql;

int \_tmain(int argc, \_TCHAR\* argv[])

{

try {

sql::Driver \*driver;

sql::Connection \*con;

SQLString hostname("COMPDBS300.gcc.edu:3306");

SQLString username("uXXXXXX"); // put your username here

SQLString pass("pXXXXXX"); // put your password here

// To establish a connection to MySQL Server, retrieve an instance of sql::Connection from a

// sql::mysql::MySQL\_Driver object. A sql::mysql::MySQL\_Driver object is returned by

// sql::mysql::MySQL\_Driver::get\_mysql\_driver\_instance().

driver = get\_driver\_instance();

// checking if we were able to get a correct driver

if( driver == NULL )

throw SQLException("Null driver instance returned.");

// printing a statement indicating that we found the driver

cout << "I found the driver" << endl;

// establish a connection to the database

con = driver->connect(hostname, username, pass );

cout << "Connection succeeded" << endl;

// Make sure that you free, con, the sql::Connection object as soon as you do not need it any more.

// But do not explicitly free driver, the connector object! The connector will take care of freeing that.

delete con;

int x = 5;

cin>>x;

} catch (sql::SQLException &e) {

cout << "# ERR: SQLException in " << \_\_FILE\_\_;

cout << "(" << \_\_FUNCTION\_\_ << ") on line " << \_\_LINE\_\_ << endl;

cout << "# ERR: " << e.what();

cout << " (MySQL error code: " << e.getErrorCode();

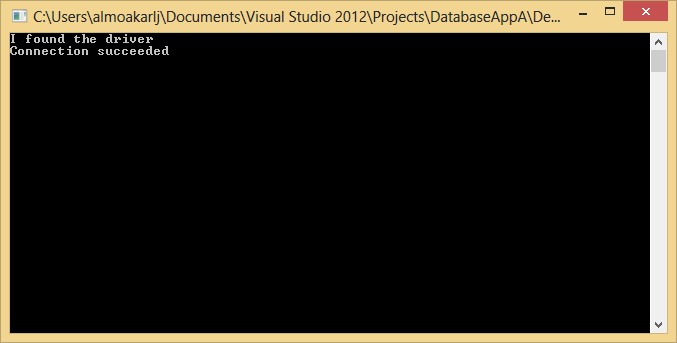
cout << ", SQLState: " << e.getSQLState() << " )" << endl;

}

cout << endl;

return EXIT\_SUCCESS;

}



**Part 4: Querying the Database**

Now that we connected to the Database. Let’s attempt to read a relation:

1. declare an sql statement so we can run SQL statements:

sql::Statement \*stmt;

1. Declare a result set in order to store the results of an SQL query

sql::ResultSet \*res;

1. After the connection succeeded statement, specify the schema to use

con->setSchema("put your schema name here");

1. Then, create a statement

stmt = con->createStatement();

1. Specify the SQL query to run and execute the query

res = stmt->executeQuery("SELECT \* from bank");

1. Iterate over the result set and print out the results

while (res->next()) {

cout << res->getString(1) << "\t";

cout << res->getString(2) << "\t";

cout << res->getString(3) << endl;

}

1. Before the end of main, destroy both the result set and the statement

delete res;

delete stmt;

**Part 5: Modifying the Database using PreparedStatements**

1. Include the prepared statement header file

#include <cppconn/prepared\_statement.h>

1. Declare a PreparedStatement

sql::PreparedStatement \*pstmt;

1. Set the transaction mode to be true for autocommits

con->setAutoCommit(true);

1. Specify the SQL statement to run. Values inputted by the program/user are replaced by question marks (?)

pstmt = con->prepareStatement("INSERT INTO bank VALUES (?,?,?)");

1. Populate the values

pstmt->setString(1, "XYZ");

pstmt->setString(2, "XYZ Bank");

pstmt->setString(3, "XYZ street, Grove City, PA");

1. Run the SQL statement:

int rows = pstmt->executeUpdate();

1. Determine if the insertion was successful or not

if (rows > 0 )

cout << "Insertion Successful!" << endl;

1. Do not forget to destroy the PreparedStatement

delete pstmt;

**Part 6: Useful functionality**

The following code allows the application to retrieve the names of the attributes stored in the resultset.

ResultSetMetaData \*res\_meta = res -> getMetaData();

for (unsigned int i=0; i < res\_meta->getColumnCount(); i++)

cout << res\_meta->getColumnLabel(i+1) << "\t";

cout << endl;