# Database Application Development

## Objective:

In this lab students learn:

- How to connect to a Oracle server from a C++ program.
- How to write and execute SQL queries in a C++ program.

## Getting Started Instructions:

## Create Oracle Server Connection:

To connect to the Oracle server from your C++ program in Visual Studio, you need to follow these steps:

Download the Oracle Instant Client SDK and libraries from for version 12.2.0.1.0 from http://dbs211.ca/files/instantclient_12_2.zip. Extract the zip archive to a local folder on your hard drive.

Open Visual Studio and create a new C++ console project. If you do not have visual studio installed, you can obtain it from MyApps at Seneca College or you can download the free Visual Studio Community Edition at: https://visualstudio.microsoft.com/vs/community/

Once you have created the project:

The first step in the project is to establish that it is a .cpp project, so create a cpp file by right clicking on the "Source Files" folder and choosing "Add – new Item". Choose a C++ File (.cpp) and enter a name of choice, I suggest something like "*DBS211_DB_Connection_Test.cpp*".

In the top of the file write the 2 includes below:

```
#include <iostream>
#include <occi.h>
```

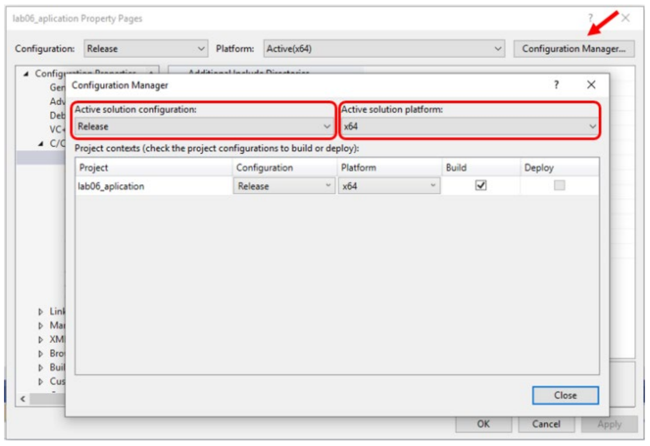
and save the file.

To connect and work with databases in Oracle server, we use the <*occi.h*> header file or library. To set up the visual configuration, right click on your project name in the navigation bar and go to "Properties":

### Project/Properties

In the "Configuration" dropdown on the top-left, change to "release" and change "Platform" to "x64" and select "Apply".

This previous step ensures that the Visual Studio project is set to release so other properties that are set apply to this version. In real life programming, you would work in Debug version until all testing is done, then change to release and test again.



lab06_aplication Property Pages

Configuration: Release
Platform: Active(x64)

Configuration Manager

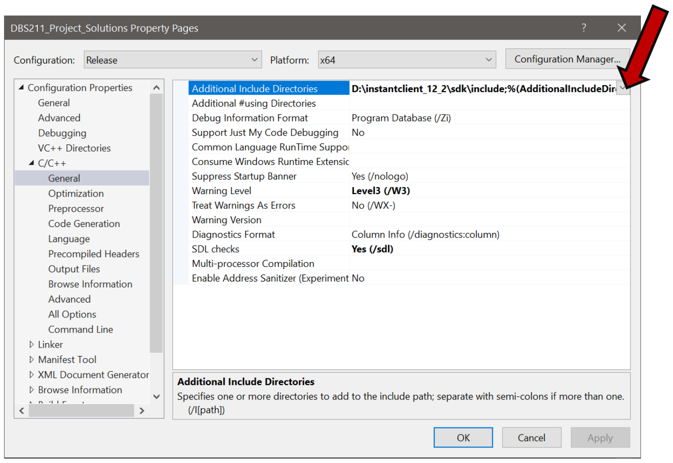
Active solution configuration: Release
Active solution platform: x64

Project contexts (check the project configurations to build or deploy):

| Project | Configuration | Platform | Build | Deploy |
|---|---|---|---|---|
| lab06_aplication | Release | x64 | ✓ | ☐ |

## Configuration Properties

In property Page, under Configuration Properties click on “C/C++” and then “General”. On the right side click the drop-down arrow beside “Include Directories” and choose “edit”.



DBS211_Project_Solutions Property Pages

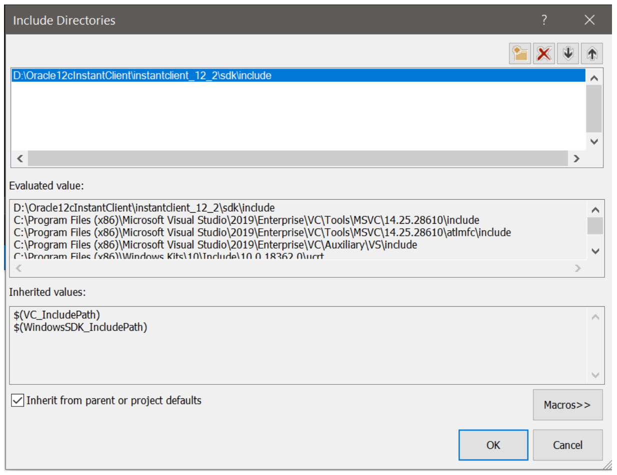
Configuration: Release Platform: x64

| | |
|---|---|
| Additional Include Directories | D:\instantclient_12_2\sdk\include;%(AdditionalIncludeDir |
| Additional #using Directories | |
| Debug Information Format | Program Database (/Zi) |
| Support Just My Code Debugging | No |
| Common Language RunTime Suppo | |
| Consume Windows Runtime Extensic | |
| Suppress Startup Banner | Yes (/nologo) |
| Warning Level | **Level3 (/W3)** |
| Treat Warnings As Errors | No (/WX-) |
| Warning Version | |
| Diagnostics Format | Column Info (/diagnostics:column) |
| SDL checks | **Yes (/sdl)** |
| Multi-processor Compilation | |
| Enable Address Sanitizer (Experiment | No |

**Additional Include Directories**
Specifies one or more directories to add to the include path; separate with semi-colons if more than one. (/I[path])

### *Linking to the Includes Folder to reference the “occi.h” header file required for the project.*

On the new screen (shown right), click on the currently blank top line of the first box and you will see a “…” button. Click on this button and navigate to the folder where you extracted the Instant Client SDK and then into the “include” folder, click “Select Folder”. Note: your path may be different then the one shown below as it will depend on where you extracted the instant client SDK zip archive.



Include Directories

D:\Oracle12cInstantClient\instantclient_12_2\sdk\include

Evaluated value:

```
D:\Oracle12cInstantClient\instantclient_12_2\sdk\include
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Tools\MSVC\14.25.28610\include
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Tools\MSVC\14.25.28610\atlmfc\include
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Auxiliary\VS\include
C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\ucrt
```

Inherited values:

```
$(VC_IncludePath)
$(WindowsSDK_IncludePath)
```

☑ Inherit from parent or project defaults

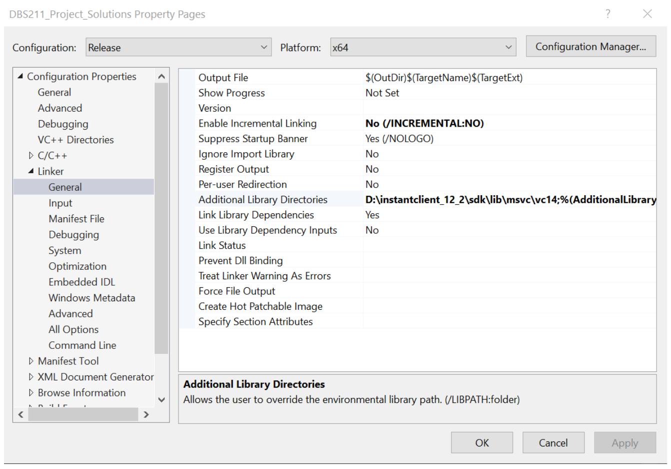
Click OK and then Apply.

## Linking to the Required Libraries

Now we need to tell the application where the library files we will be referencing are found. Navigation the Configuration Properties to Linker and then General

Under Additional Library Directories you will include a reference to the libraries folder:

`\instantclient_12_2\sdk\lib\msvc\vc14` - relative to your zip archive extraction location.

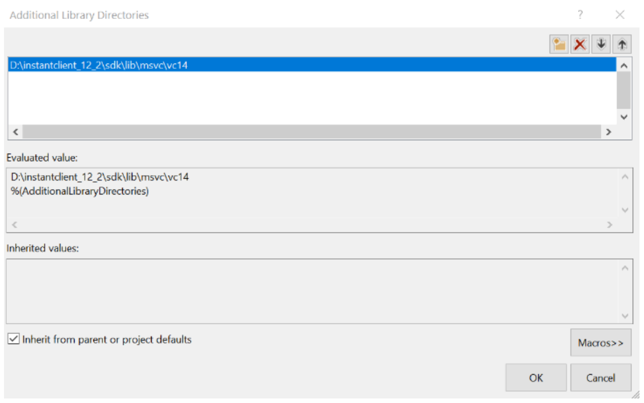


DBS211_Project_Solutions Property Pages

Configuration: Release Platform: x64

| | |
|---|---|
| Output File | $(OutDir)$(TargetName)$(TargetExt) |
| Show Progress | Not Set |
| Version | |
| Enable Incremental Linking | **No (/INCREMENTAL:NO)** |
| Suppress Startup Banner | Yes (/NOLOGO) |
| Ignore Import Library | No |
| Register Output | No |
| Per-user Redirection | No |
| Additional Library Directories | **D:\instantclient_12_2\sdk\lib\msvc\vc14;%(AdditionalLibrary** |
| Link Library Dependencies | Yes |
| Use Library Dependency Inputs | No |
| Link Status | |
| Prevent Dll Binding | |
| Treat Linker Warning As Errors | |
| Force File Output | |
| Create Hot Patchable Image | |
| Specify Section Attributes | |

**Additional Library Directories**
Allows the user to override the environmental library path. (/LIBPATH:folder)


Additional Library Directories

D:\instantclient_12_2\sdk\lib\msvc\vc14

Evaluated value:

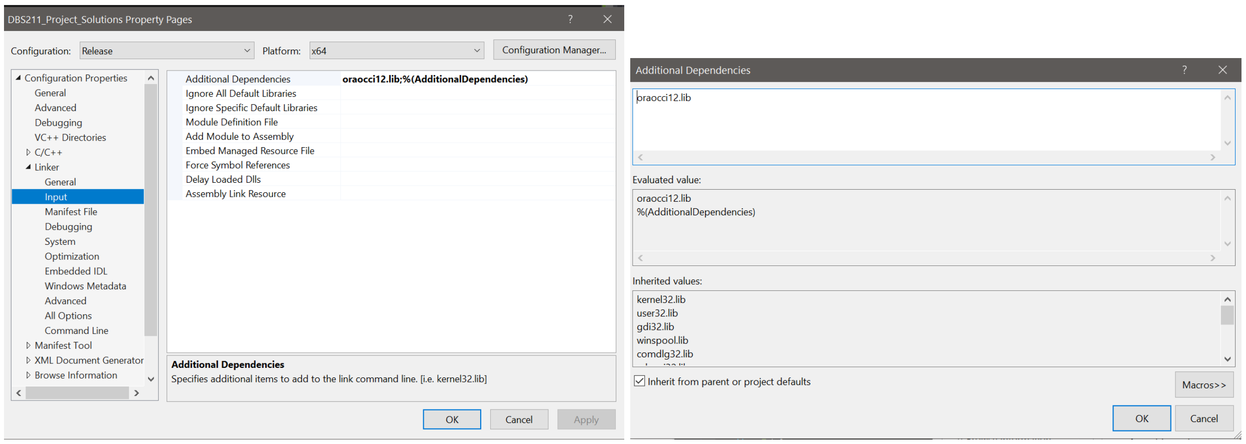
D:\instantclient_12_2\sdk\lib\msvc\vc14
%(AdditionalLibraryDirectories)

Inherited values:

☑ Inherit from parent or project defaults

You will then be required to tell the application which library we need to use.

Under Linker / Input / Additional Dependencies, you will need to add `oraocci12.lib`.


DBS211_Project_Solutions Property Pages

Configuration: Release  Platform: x64

| | |
|---|---|
| Additional Dependencies | oraocci12.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 | |

**Additional Dependencies**
Specifies additional items to add to the link command line. [i.e. kernel32.lib]

Additional Dependencies

oraocci12.lib

Evaluated value:

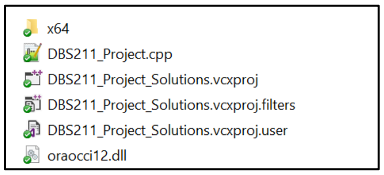
oraocci12.lib
%(AdditionalDependencies)

Inherited values:

kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib

☑ Inherit from parent or project defaults

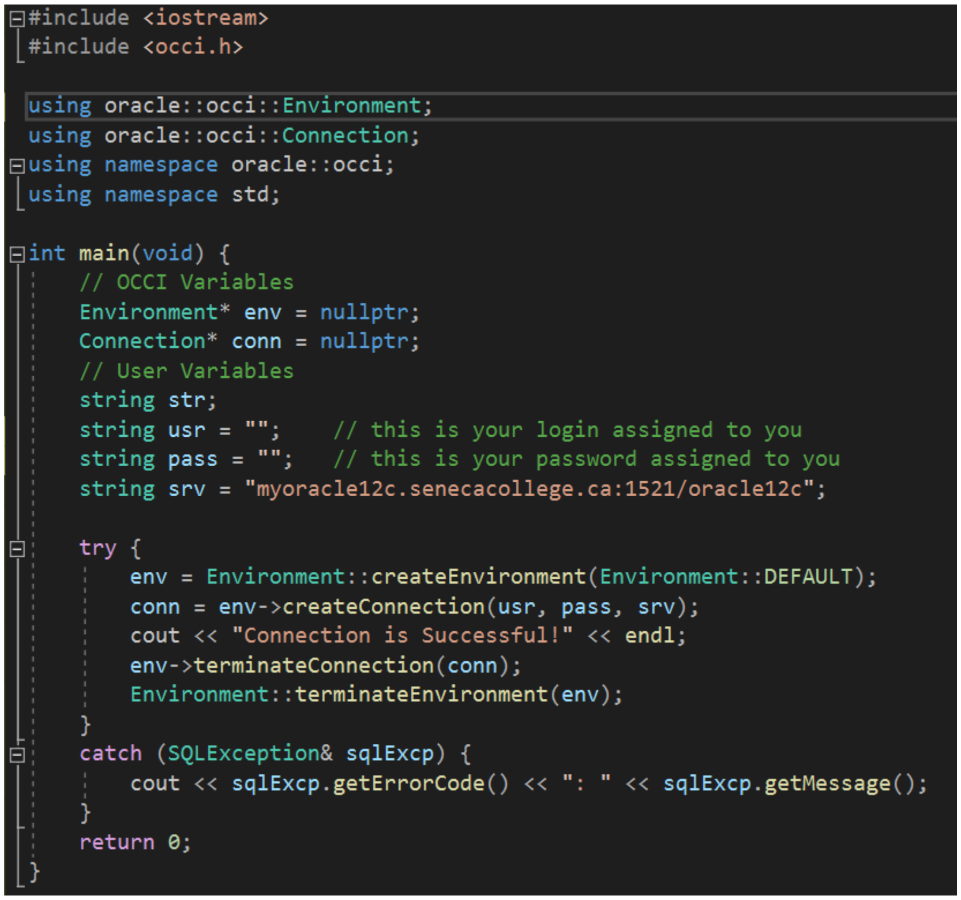
And lastly you will need to add the `oraocci12.dll` file to your project folder directory. Right click on the Project and choose: "Open Folder in File Explorer". This is the folder where the .dll file must be placed. The required file can be copied from the main `instantclient_12_2` folder.


x64
DBS211_Project.cpp
DBS211_Project_Solutions.vcxproj
DBS211_Project_Solutions.vcxproj.filters
DBS211_Project_Solutions.vcxproj.user
oraocci12.dll

Finally make sure that everything seems okay to this point by trying to build the project. An error free build is the first good sign.

## *Connecting to an Oracle database from a C++ Program*

So now we will add some code to make sure we can connect we can do a simple test. In the .cpp file we created earlier, type in the following code to test your connection. It is important that you type it as to learn what is happening to make completing the project tasks much easier. (Hence why I am providing a screen shot rather than text so you cannot copy and paste)



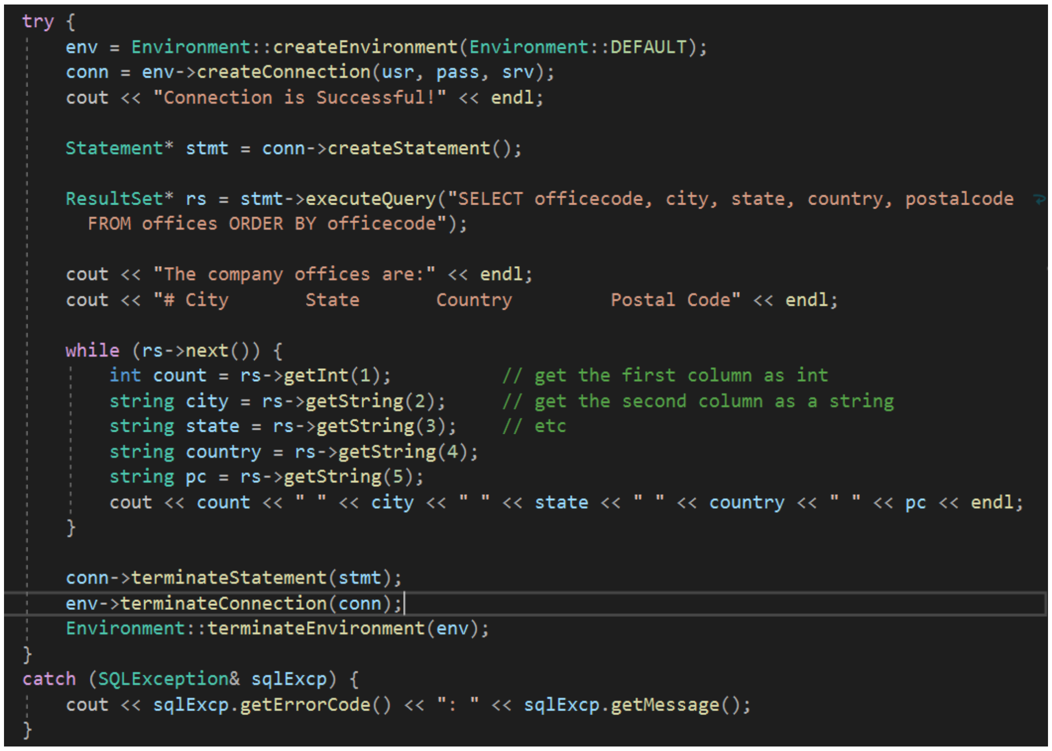
```
#include <iostream>
#include <occi.h>

using oracle::occi::Environment;
using oracle::occi::Connection;
using namespace oracle::occi;
using namespace std;

int main(void) {
    // OCCI Variables
    Environment* env = nullptr;
    Connection* conn = nullptr;
    // User Variables
    string str;
    string usr = "";     // this is your login assigned to you
    string pass = "";    // this is your password assigned to you
    string srv = "myoracle12c.senecacollege.ca:1521/oracle12c";

    try {
        env = Environment::createEnvironment(Environment::DEFAULT);
        conn = env->createConnection(usr, pass, srv);
        cout << "Connection is Successful!" << endl;
        env->terminateConnection(conn);
        Environment::terminateEnvironment(env);
    }
    catch (SQLException& sqlExcp) {
        cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();
    }
    return 0;
}
```

If you see the console screen come up with Connection is Successful, then you are ready to move to the next steps.

## Running a Simple SELECT query and outputting the results

So we will go through a series of steps to get some output. Type in the following code as shown within the code shown above…. (i.e. merge the two). Use the provided information in class (likely a PPT slideshow) to understand what each line is doing. Some key concepts you will want to understand include:

- Connection (conn)
- RecordSet (rs)
- next() method and how it works
- and all the terminate statements and why they are necessary.



```
try {
    env = Environment::createEnvironment(Environment::DEFAULT);
    conn = env->createConnection(usr, pass, srv);
    cout << "Connection is Successful!" << endl;

    Statement* stmt = conn->createStatement();

    ResultSet* rs = stmt->executeQuery("SELECT officecode, city, state, country, postalcode
      FROM offices ORDER BY officecode");

    cout << "The company offices are:" << endl;
    cout << "# City       State       Country         Postal Code" << endl;

    while (rs->next()) {
        int count = rs->getInt(1);          // get the first column as int
        string city = rs->getString(2);     // get the second column as a string
        string state = rs->getString(3);    // etc
        string country = rs->getString(4);
        string pc = rs->getString(5);
        cout << count << " " << city << " " << state << " " << country << " " << pc << endl;
    }

    conn->terminateStatement(stmt);
    env->terminateConnection(conn);
    Environment::terminateEnvironment(env);
}
catch (SQLException& sqlExcp) {
    cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();
}
```

Now to learn this.  Complete the same concept, but for the employees table.

Some tasks that should be completed as part of working towards the first project milestone.

- the output should be formatted in a way that columns line up nicely
- the appropriate data types are used for the respective database data type
- all open objects should be terminated
- try to modify the SQL statement to include a WHERE clause to return a specific employee number and only show the output for the employee number typed in from a console prompt line.
- Error check the code for a couple things, 1 – the entered employee number is the right data type before trying to query the database and 2 – the employee number exists in the database before trying to output the values to the console window.   Give appropriate error messages to the user in both cases.