DBS211
Week 8 - Database Application Development

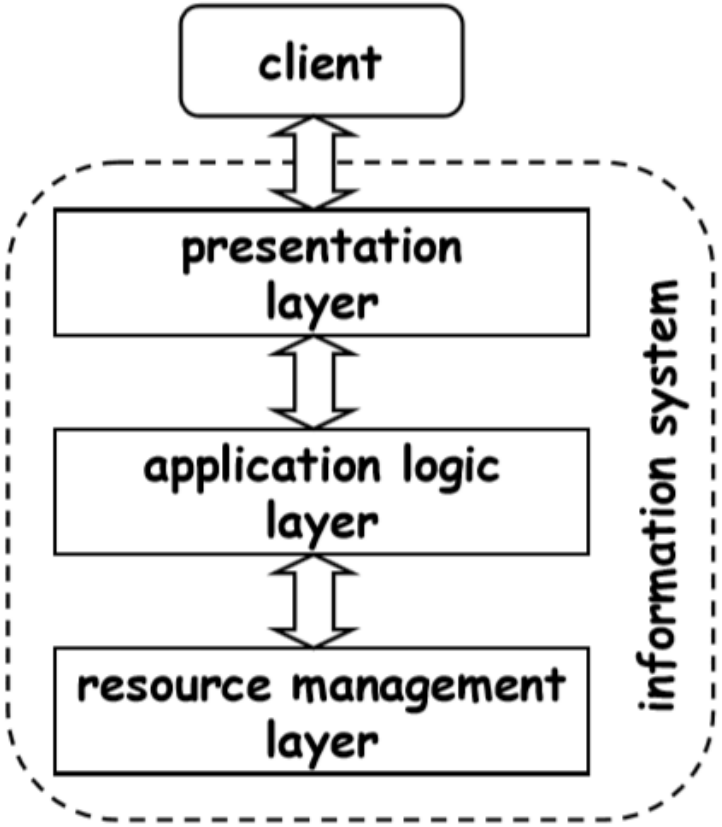
Agenda

- Application Development and Design
- Roll of Databases in Software
- N-Tier Development
- Design Methodologies
- Database Access and Connections (AND INSTALLATIONS)
- Summary

Application Layers

- Applications are typically broken into three layers

1. Presentation / User Interface / View Layer
2. Business Logic / Application Logic / Controller Layer
3. Resource Management / Data Access / Model Layer



Presentation Layer

- The presentation or the user interface provides the user interaction.
- Different form of the presentation layers could be
  - Web browsers
  - Mobile phone user interface

Business Layer

- The business or application logic layer provide the high level view and access of data.
- takes input from the user interface
- forward it to the database layer for storage or retrieval.
- In this layer, business rules are enforced, changes can be made to the business rules with little or no impact on the other layers

Data Access Layer

- The data access later is the interface between business layer and data resources. It deals with data storage and retrieval to support the business layer.

- The structure of the database and the DML statements and routines needed to manipulate the data, both reading and writing.

Modern Software Development

- Every modern software application has a database involved in the background. The purpose of the database could be:

- data storage
- data retrieval
- logging and tracking
- inventory management

- In smaller development companies, where employees are often involved in many areas of the Software Development Life Cycle (SDLC). including database development, software development and the entire design process.

Application Architecture

- Two-tier architecture
- Three-tier architecture

2 Tiered Architecture

- The two-tier application is based on client server approach.
- There is a direct communication between client and server.
- Client request data from server. Further data processing can be done on the client side.
- The two-tier architecture has two main parts:
  - Database
  - Client application
- Advantages:
  - Since there is not any intermediate layer between client and server, two-tier application runs faster. Business and data managements are in one layer.
- Disadvantages:
  - It supports limited number of clients.
  - The server cannot respond multiple request same time

3 Tiered Architecture

- In this architecture, there is an intermediate layer between presentation (user interface) and data layers.
- Three separate layers:
  - Presentation
    - In this layer, the users send requests and receive data.
  - Business/application logic
    - In this layer, business rules are applied on
      - data calculation
      - data validation
      - data input
  - Data
      - This layer includes methods to
      - connect to the data resource (database)
      - manipulate data such as insertion, deletion, and modification

N-Tier Development

- The n-tier architecture is such that various aspects of the application can be kept independent for **object oriented** purposes.
- In some cases, software is divided into addition tiers to further separate concerns. One example of this: is to **create a security layer** that controls and centralizes user access, administration, role management, and permissions to perform specific actions within the business and data tiers.

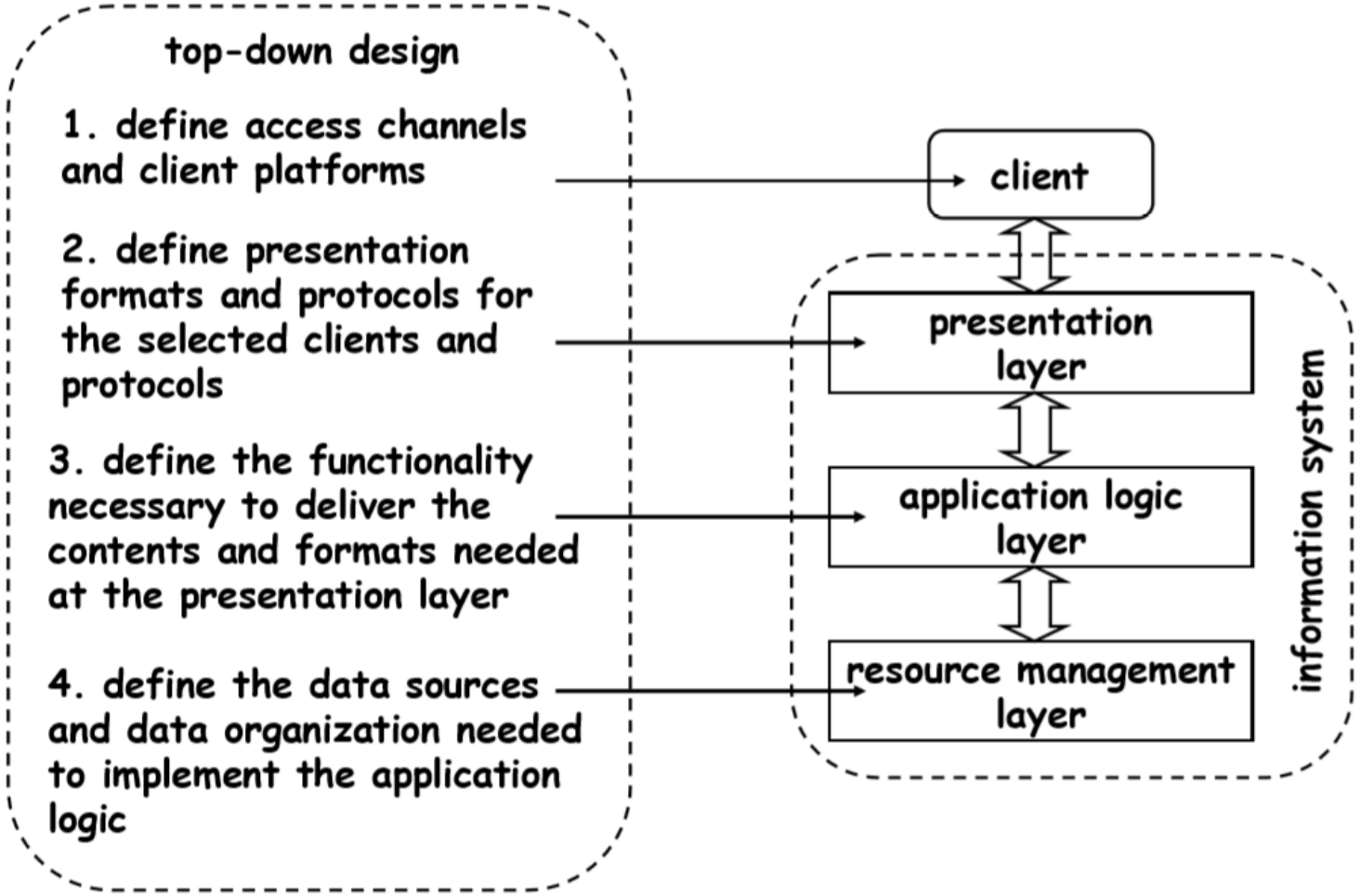
- Some of the approaches to database driven applications include:

- Bottom-up or database-first approach
- Top-Down or UI-first approach
- Inside-Out or Code-First approach
- Outside-In or Client Centric approach

Top-Down

- The functionality of the system is defined from the client's point of view.
  - It determines how client interact with the system.
  - It focuses on high-level requirements.
- Advantages:
    - It focuses on final goals.
    - It addresses
      - Functional issues – the operations that are supported
      - Non-functional issues – the performance and availability
- Disadvantages
  - The system has to be entirely developed from scratch.

Top-down Design

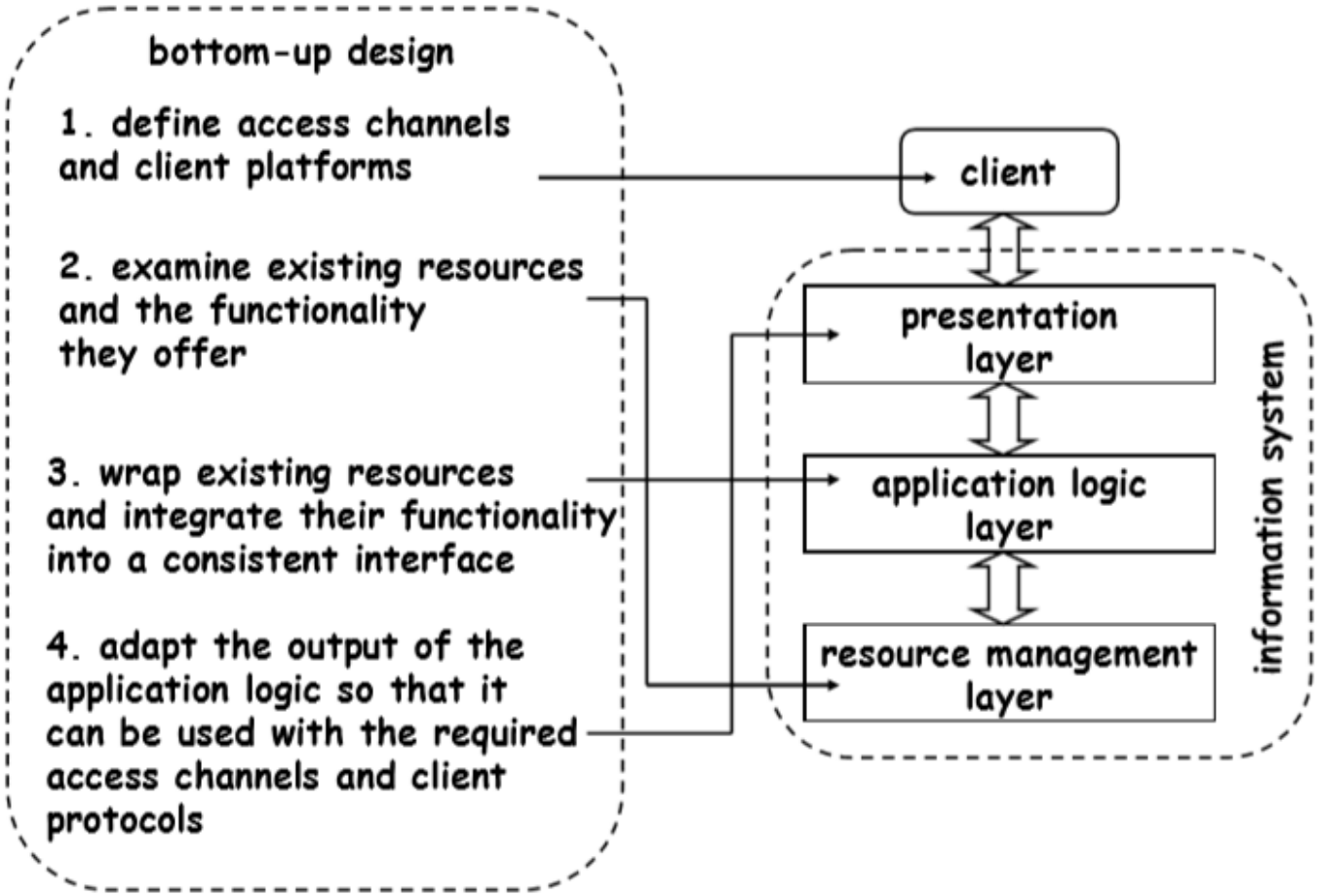


Bottom-Up

- This approach involves developing the database first in the application development process.
- Developing the database first can often result in the ability to automate the code writing process through code generation.
- The coding of the data access or model layers can almost be completely automated

- This often results in the client wondering if the software company is performing its tasks, and a relationship breakdown between client and vendor.
- This approach also can result in limitations in the design of the user interface due to code already being written. The iteration approach to the user-interface will be greatly limited.

Bottom-up Design



Inside-Out or Code-First Approach

- This approach is similar to the bottom-up approach but focuses on the database access or model layer first built based upon the data requirements gathering process. This object orientated approach allows developer to create the required classes, properties and methods first and then generate the database from the code.

- PROS: This approach works well for a team with strong developers, and can give the developers a deep understanding of all the code that will be accessed through the entire development process.

- CONS: This approach often results in an incomplete database design and many criteria and database features missing, such is indexes, referential integrity and the use of stored procedures and user-defined functions.

Outside-In or Client Centric Approach

- This approach uses the requirements gathering results and simultaneously designs both the database and user interface layers. Then as both ends are developed, they will work inwards towards the business logic layer.

- PROS: This balances the client interactive parts of the development with those parts where very little client interaction is involved. This means the client sees continuous and consistent progress through the project and keeps lines of communication open.

- CONS: This approach can often result in reiterating the design of both the user interface and data access layers because of considerations determined through the middle or business logic tiers.

Database Application Development

A First Course in Database Systems_3rd Edition

https://dev.mysql.com/doc/connector-cpp/8.0/en/Oracle document

MySQL Document

Database Application

- How SQL fits into a programming environment?
- How a typical application can access a database to fetch or manipulate data in a database?
- How to embed SQL to programs with programming languages such as C++?

Database Access from Application

- How SQL commands can be executed in a host language?
  - **Embedded SQL**
  - **Cursors**
  - **Dynamic SQL**

Embedded SQL

- SQL statements and commands can be used in the host language. In this course, we use C++ as the host language.
- In an Embedded SQL approach, the host language can consist of either static or dynamic SQL or a combination of both.
- One approaches to embed SQL statements in a host language:
  - Call Level Interface (CLI):
    - In this approach, a program uses some libraries to access the database. All SQL statements are set of strings that are passed to the database using a CLI function call.

SQL Application

- Connect to DBMS
- Retrieve data from the database
- Manipulate data
- Modify data in the database

Using C++ to connect to Oracle

C++ TOPICS TO GO THROUGH for the Database Application

- INPUT AND OUTPUT IN C++
- CREATING STRUCTURES
- CREATING FUNCTIONS
- USING OCCI TO CONNECT TO ORACLE

Using standard library header files

#include <iostream>, instructs the preprocessor to include a section of standard C++ code, known as header iostream, that allows to perform standard input and output operations, such as writing the output of this program (Hello DBS211 students!) to the screen.

The function named **main** is a special function in all C++ programs; it is the function called when the program is run. The execution of all C++ programs begins with the main function, regardless of where the function is actually located within the code.

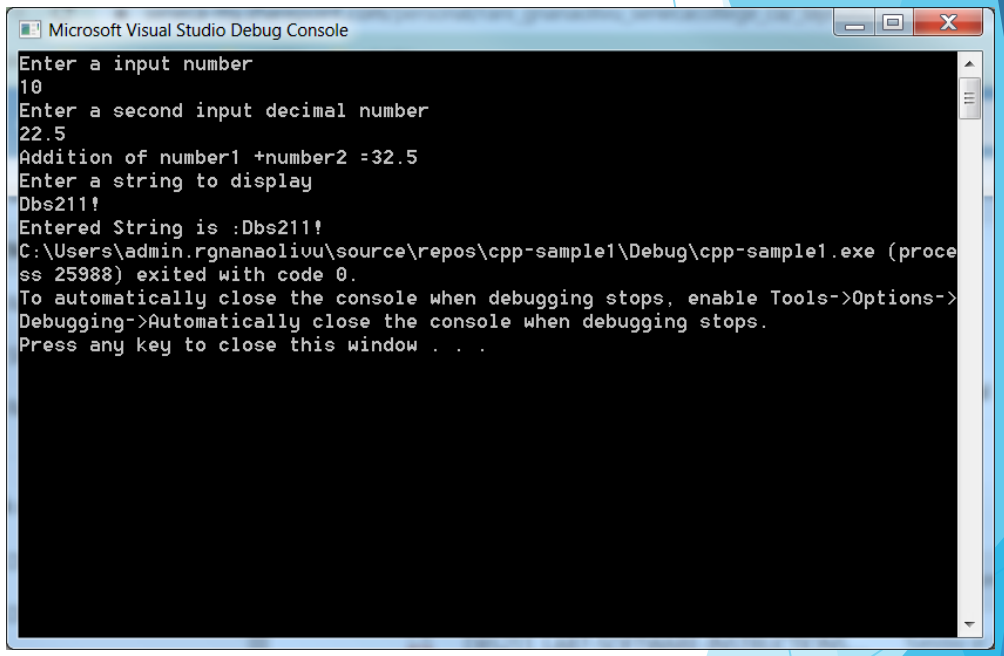
cout is part of the standard library, and all the elements in the standard C++ library are declared within what is called a **namespace:** the namespace std.

Using standard library header files

```
#include <iostream>
using namespace std;

int main() {
  cout << "Hello DBS211 students!";
  return 0;
}
```

Example of input with different datatypes

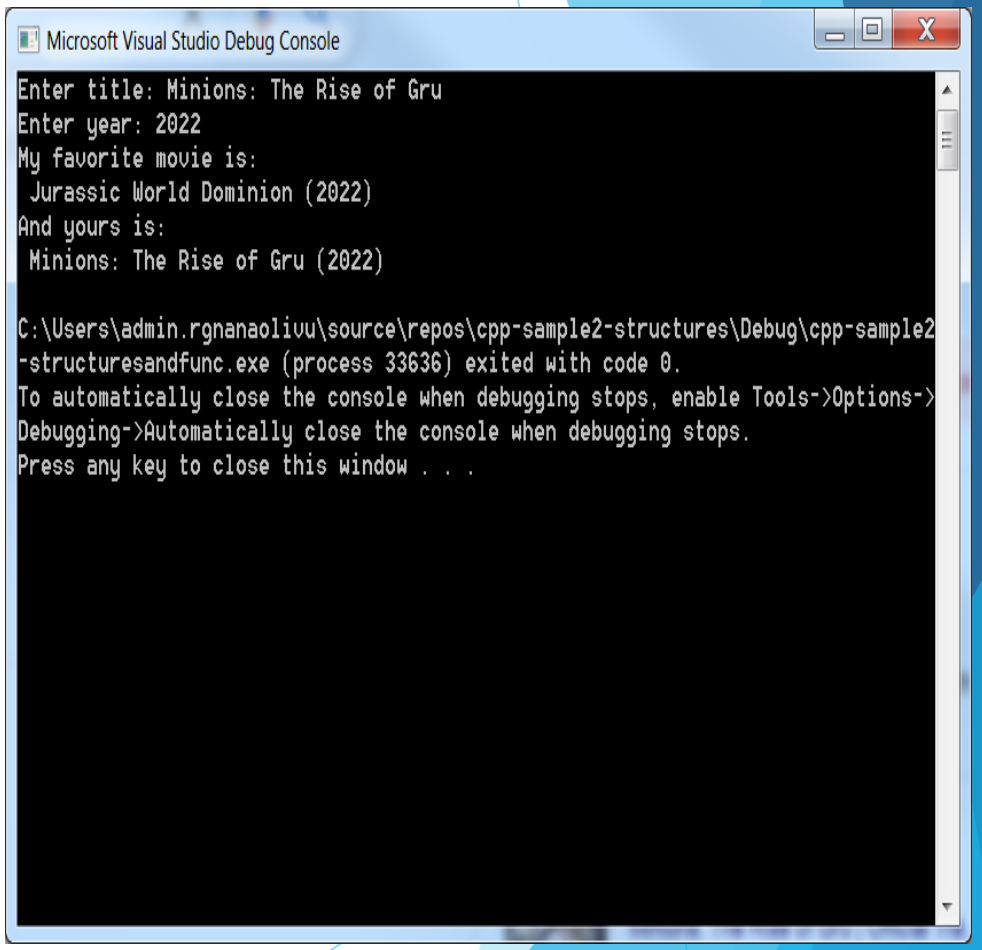
```
#include <iostream>

using namespace std;

int main() {
    //Declaration of variables
    int number1;
    double number2;
    string strofchars;
    time_t timetoday = time(0);
    cout << "Enter a input number" << "\n";
    cin >> number1;
    cout << "Enter a second input decimal number" << "\n";
    cin >> number2;
    cout << "Addition of number1 +number2 ="<<number1 + number2 << "\n";

    cout << "Enter a string to display" << "\n";
    cin >> strofchars;
    cout<<"Entered String is :"<< strofchars;
    return 0;
}
```



Example of functions and structures

```
// example about structures and functions
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

//declaration of structures
struct movies_t {
string title;
int year;
};

//declaration of functions
void printmovie(movies_t movie);

//function definition part
void printmovie(movies_t movie)
{
cout << movie.title;
cout << " (" << movie.year << ")\n";
}
```

```
int main()
{
string mystr;
movies_t mine, yours;
mine.title = "Jurassic World
Dominion";
mine.year = 2022;

cout << "Enter title: ";
getline(cin, yours.title);
cout << "Enter year: ";
getline(cin, mystr);
stringstream(mystr) >> yours.year;

cout << "My favorite movie is:\n ";
printmovie(mine);
cout << "And yours is:\n ";
printmovie(yours);
return 0;
}
```



Connecting to ORACLE : Using OCCI header files

Oracle C++ Call Interface (OCCI) is an Application Programming Interface (API) that provides C++ applications access to data in an Oracle database. OCCI enables C++ programmers to use the full range of Oracle database operations, including SQL statement processing and object manipulation.

The Environment class provides an OCCI environment to manage memory and other resources for OCCI objects.
OCCI provides **a library of standard database access and retrieval functions** in the form of a dynamic runtime library (OCCI classes) that can be linked in a C++ application at runtime.

```
#include <occi.h>

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

Creating an Environment

```
Environment *env = Environment::createEnvironment();
```

All OCCI objects created with the create*xxx*() methods (connections, connection pools, statements) must be explicitly terminated. When appropriate, you must also explicitly terminate the environment.

Terminating an Environment

If the application requires access to objects in the global scope, such as static or global variables, these objects must be set to NULL before the environment is terminated.

```
Environment::terminateEnvironment(env);
```

createConnection()

This method establishes a connection to the database specified.

Syntax

Connection * createConnection(const string &username, const string &password, const string &connectString);

terminateConnection(conn);

```
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();
}
```

C++ User Input

```
int x;
cout << "Type a number: "; // Type a number and press enter
cin >> x; // Get user input from the keyboard
cout << "Your number is: " << x; // Display the input value
```

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

int count=rs->getInt(1);
string city=rs->getString(2);
string state=rs->getString(3);
string country=rs->getString(4);
string pc=rs->getString(5);
cout<<count<<" "<<city<<" "<<state<<" "<<country<<" "<<pc<<endl;
}
```

Example in C++ to connect to Oracle

```cpp
#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;
}
```

Example in C++ to connect to Oracle

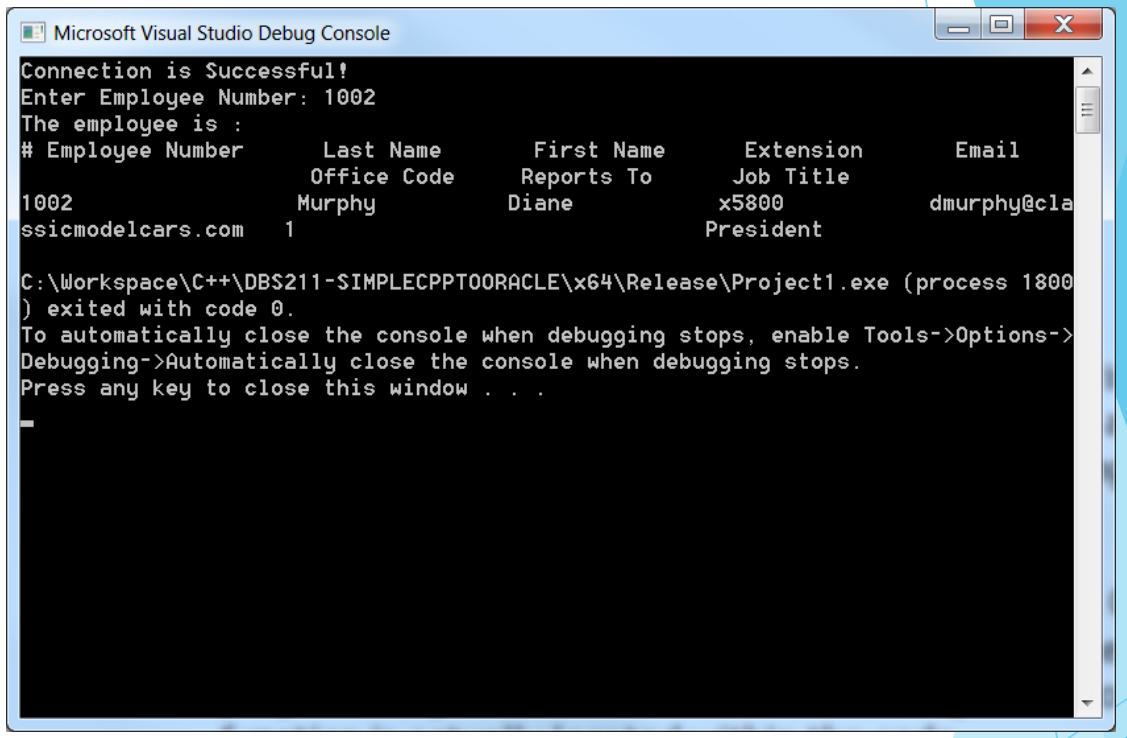
```
#include <iostream>
#include <occi.h>


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


struct Employee
{
    int employeeNumber;
    string lastName;
    string firstName;
    string extension;
    string email;
    string officeCode;
    int reportsTo;
    string jobTitle;
};


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

Example in C++ to connect to Oracle

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


        Statement* stmt = conn->createStatement();

        string employeenum;

                cout << "Enter Employee Number: ";
                cin >> employeenum;


        ResultSet* rsss = stmt->executeQuery("SELECT employeenumber, lastname, firstname,
                                                extension,email, officecode, reportsto, jobtitle FROM retailemployees
                                                WHERE employeenumber=" + employeenum);



        cout << "The employee is :" << endl;
        cout << "# Employee Number      Last Name      First Name      Extension      Email      Office Code      Reports To      Job Title" << endl;
```

Example in C++ to connect to Oracle

```
while (rsss->next()) {

int employeenumber = rsss->getInt(1);
string lastname = rsss->getString(2);
string firstname = rsss->getString(3);
string extension = rsss->getString(4);
string email = rsss->getString(5);
int officecode = rsss->getInt(6);
string reportsto = rsss->getString(7);
string jobtitle = rsss->getString(8);

cout.width(20); cout << left << employeenumber << " ";
cout.width(15); cout << left << lastname << " ";
cout.width(15); cout << firstname << " ";
cout.width(15); cout << extension << " ";
cout.width(30); cout << email << " ";
cout.width(15); cout << officecode << " ";
cout.width(15); cout << reportsto << " ";
cout.width(15); cout << jobtitle << endl;
}

conn->terminateStatement(stmt);
env->terminateConnection(conn);
Environment::terminateEnvironment(env);
}
catch (SQLException& sqlExcp) {
cout << sqlExcp.getErrorCode() << ": " << sqlExcp.getMessage();
}
return 0;
}
```



Some reading about databases and c++

C++ Language - C++ Tutorials - Cplusplus.com
https://www.cplusplus.com/doc/tutorial/

C++ Tutorial - W3Schools
https://www.w3schools.com/CPP/default.asp

Introduction to OCCI
https://docs.oracle.com/cd/B12037_01/appdev.101/b10778/introduction.htm

Accessing Oracle Database Using C++

https://docs.oracle.com/database/121/LNCPP/relational.htm#LNCPP003