

Actividad | #2 | Conexión y Tablas Lenguajes de Programación II

Ingeniería en Desarrollo de Software

TUTOR: Félix Acosta Hernández

ALUMNO: Alejandra Ibarra Carmona

FECHA:30/11/2024

Indicé

Introducción…………………………………………….. 3

Descripción……………………………………………. .3

Justificación…………………………………………….3

Desarrollo………………………………………………4

○ Conexión……………………………………………..4

○ Tablas…………………………………………………4

○ Código…………………………………………………5

Conclusión………………………………………………9

Referencias…………………………………………….9

Introducción

En esta actividad se estará conectando nuestra base de datos de SQL con nuestro visual studio para poder ver las tablas creadas de diferente manera. Cabe mencionar que esta también está orientada a objetos y que podemos denominar polimorfismo que es la capacidad que tienen los métodos para responder de una manera distinta a partir de su invocación, claro que se clasifica en dos clases las cuales son: dinámico y estático

* Polimorfismo dinámico: también es conocido como paramétrico, en este el código no incluye ningún tipo de especificación acerca del tipo que se trabaja.
* Polimorfismo estático: este es conocido como ad hoc, es aquel en el que los tipos de polimorfismo deben de ser declarados uno por uno antes de ser utilizados.

A su vez cada uno de los polimorfismos puede dividirse ya que cuentan con una carga de trabajo muy grande sus bloques son sobrecarga y sobreescritura.

Descripción

Un método virtual es considerado una función miembro que es publica y protegida de una clase base que se redefine en cada una de las clases derivadas de esta. En la programación orientada a objetos cuando una de las clases es derivada se hereda de la clase base un objeto de aquella, esto referido tanto al tipo de clase base como la derivada, a lo largo de estas dos actividades hemos mencionado un objeto derivado como el tipo de clase base, este cuenta con un comportamiento de la llamada a la función deseada.

Cuando tenemos una clase que incluya una función virtual se puede llamar una clase polimórfica, este término también se puede aplicar a una clase que hereda una clase base que a su vez contiene una virtual.

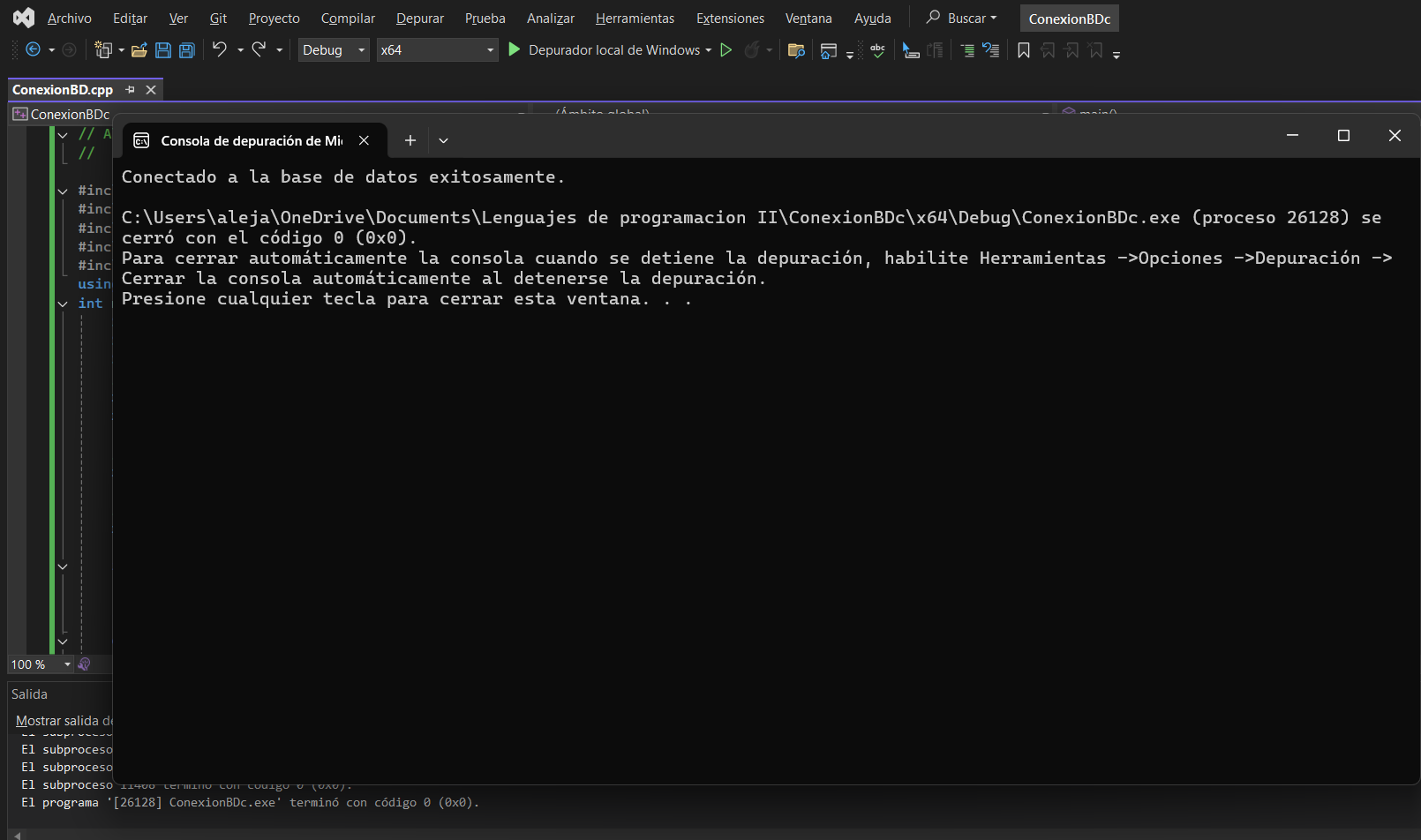
Justificación

La declaración de un método virtual ofrece solo la cabecera, claro que normalmente ofrece una implementación de la función virtual pura en una clase abstracta, toda clase hija no abstracta continúa necesitando redefinir el método. Cuando se escribe en el compilador en automático sabe a qué implementación del método llamar en el tiempo de la ejecución creando una tabla de punteros a todos los métodos virtuales de una clase llamada vtable o tabla virtual.

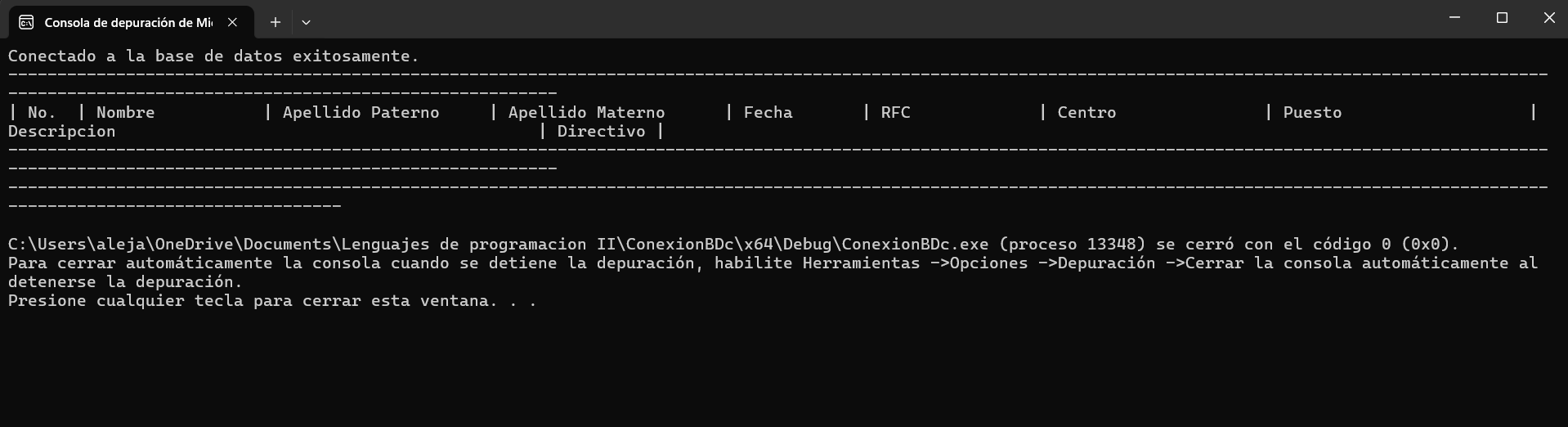
Las clases definen atributos y métodos, los métodos de una clase sirven para manipular atributos de la clase, si bien sabemos que in método puede recibir valores para que con esto se efectúen operaciones con estos y así poder retornarlos después, los métodos siempre presentan un paréntesis al final de su nombre, con esto es necesario para la ejecución de los métodos y también se le llaman argumentos

Desarrollo

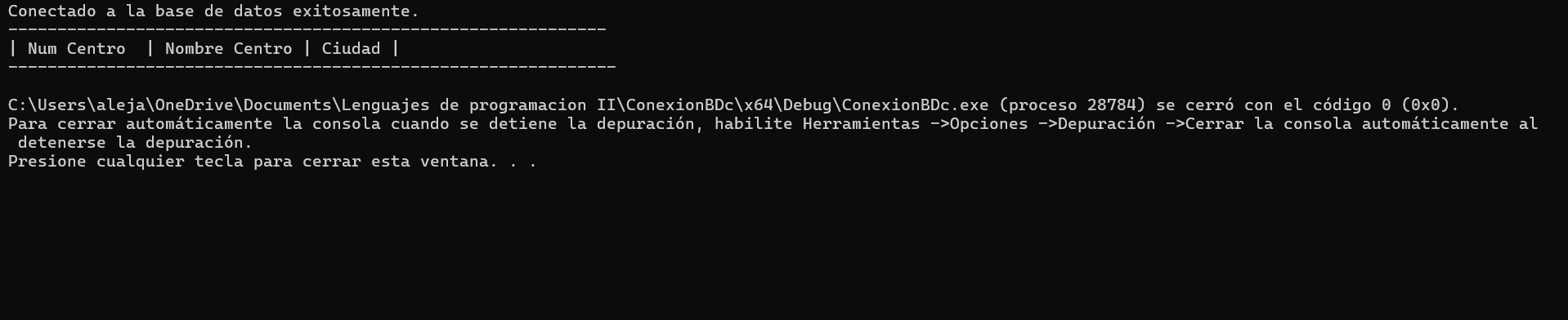
* Conexión exitosa con la base de datos



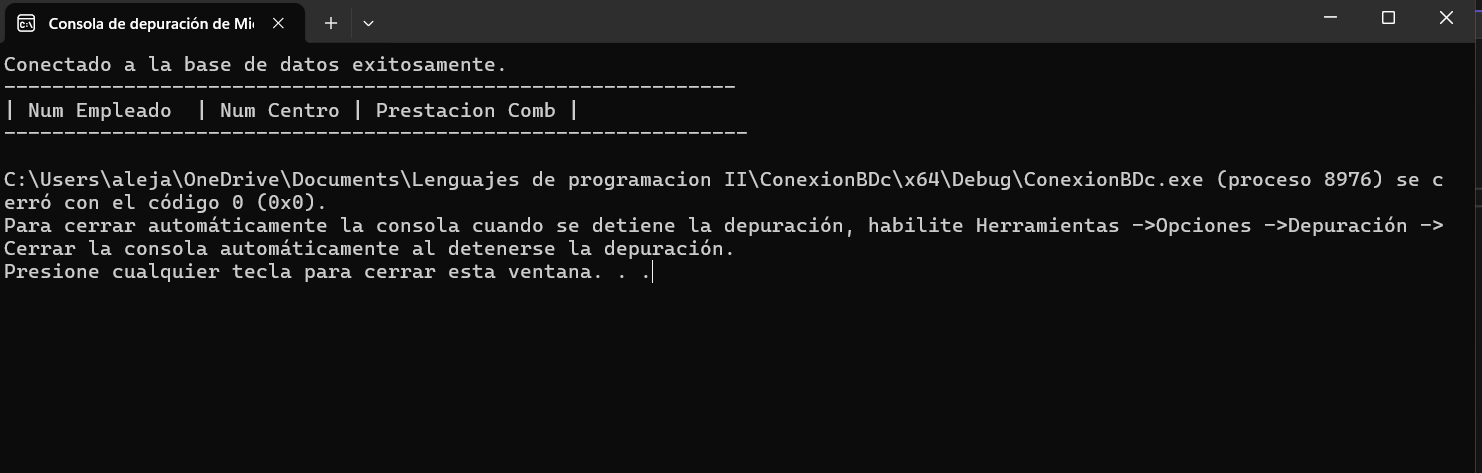
* Tabla Empleados



* Tabla Puestos



* Tabla Directivos



* Codigo para la conexión de la base de datos

// AlejandraIbarra\_A2.cpp : Este archivo contiene la función “main”. La ejecución del programa comienza y termina ahí.

//

#include <windows.h>

#include <sql.h>

#include <sqlext.h>

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

SQLHENV hEnv;

SQLHDBC hDbc;

SQLRETURN ret;

// Asignar un gestor de entorno

ret = SQLAllocHandle(SQL\_HANDLE\_ENV, SQL\_NULL\_HANDLE, &hEnv);

ret = SQLSetEnvAttr(hEnv, SQL\_ATTR\_ODBC\_VERSION, (SQLPOINTER)SQL\_OV\_ODBC3, 0);

// Asignar un gestor de conexión

ret = SQLAllocHandle(SQL\_HANDLE\_DBC, hEnv, &hDbc);

// Conectarse a la base de datos

ret = SQLConnect(hDbc, (SQLWCHAR\*)L”localhost”, SQL\_NTS, (SQLWCHAR\*)L”Username”, SQL\_NTS, (SQLWCHAR\*)L”Password”, SQL\_NTS);

if (ret == SQL\_SUCCESS || ret == SQL\_SUCCESS\_WITH\_INFO) {

cout << “Conectado a la base de datos exitosamente.” << endl;

}

else {

cout << “Fallo la conexión a la base de datos” << endl;

}

// Desconectar y liberar gestores de entorno

SQLDisconnect(hDbc);

SQLFreeHandle(SQL\_HANDLE\_DBC, hDbc);

SQLFreeHandle(SQL\_HANDLE\_ENV, hEnv);

return 0;

}

* Código para la tabla empleados

if (ret == SQL\_SUCCESS || ret == SQL\_SUCCESS\_WITH\_INFO) {

// Imprimir encabezados de las columnas

std::cout << “---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n”;

std::cout << “| No. | Nombre | Apellido Paterno | Apellido Materno | Fecha | RFC | Centro | Puesto | Descripción | Directivo |\n”;

std::cout << “---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n”;

//Variables para almacenar datos de cada columna

int numEmpleado, directivo;

SQLCHAR nombre[50], apellidoP[50], apellidoM[50], fechaNac[50], rfc[20], centroT[50], puestoT[50], descripcionP[200];

// Obtener los resultados fila por fila

while (SQLFetch(hStmt) == SQL\_SUCCESS) {

SQLGetData(hStmt, 1, SQL\_C\_LONG, &numEmpleado, 0, NULL);

SQLGetData(hStmt, 2, SQL\_C\_CHAR, nombre, sizeof(nombre), NULL);

SQLGetData(hStmt, 3, SQL\_C\_CHAR, apellidoP, sizeof(apellidoP), NULL);

SQLGetData(hStmt, 4, SQL\_C\_CHAR, apellidoM, sizeof(apellidoM), NULL);

SQLGetData(hStmt, 5, SQL\_C\_CHAR, fechaNac, sizeof(fechaNac), NULL);

SQLGetData(hStmt, 6, SQL\_C\_CHAR, rfc, sizeof(rfc), NULL);

SQLGetData(hStmt, 7, SQL\_C\_CHAR, centroT, sizeof(centroT), NULL);

SQLGetData(hStmt, 8, SQL\_C\_CHAR, puestoT, sizeof(puestoT), NULL);

SQLGetData(hStmt, 9, SQL\_C\_CHAR, descripcionP, sizeof(descripcionP), NULL);

SQLGetData(hStmt, 10, SQL\_C\_LONG,&directivo, 0, NULL);

std::cout << std::setw(5) << std::left << numEmpleado << “ | “

<< std::setw(17) << std::left << nombre << “ | “

<< std::setw(15) << std::left << apellidoP << “ | “

<< std::setw(15) << std::left << apellidoM << “ | “

<< std::setw(12) << std::left << fechaNac << “ | “

<< std::setw(15) << std::left << rfc << “ | “

<< std::setw(20) << std::left << centroT << “ | “

<< std::setw(25) << std::left << puestoT << “ | “

<< std::setw(50) << std::left << descripcionP << “ | “ // Ancho ajustado para Descripcion

<< std::setw(10) << std::left << directivo << “ |\n”; // Ancho fijo para Directivo

}

// Línea final

std::cout << “-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\n”;

}

* Codigo para la tabla Directivos

#include <windows.h>

#include <sql.h>

#include <sqlext.h>

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

SQLHENV hEnv;

SQLHDBC hDbc;

SQLRETURN ret;

// Asignar un gestor de entorno

ret = SQLAllocHandle(SQL\_HANDLE\_ENV, SQL\_NULL\_HANDLE, &hEnv);

ret = SQLSetEnvAttr(hEnv, SQL\_ATTR\_ODBC\_VERSION, (SQLPOINTER)SQL\_OV\_ODBC3, 0);

// Asignar un gestor de conexión

ret = SQLAllocHandle(SQL\_HANDLE\_DBC, hEnv, &hDbc);

// Conectarse a la base de datos

ret = SQLConnect(hDbc, (SQLWCHAR\*)L”localhost”, SQL\_NTS, (SQLWCHAR\*)L”Username”, SQL\_NTS, (SQLWCHAR\*)L”Password”, SQL\_NTS);

if (ret == SQL\_SUCCESS || ret == SQL\_SUCCESS\_WITH\_INFO) {

cout << “Conectado a la base de datos exitosamente.” << endl;

SQLHSTMT hStmt;

ret = SQLAllocHandle(SQL\_HANDLE\_STMT, hDbc, &hStmt);

if (ret == SQL\_SUCCESS || ret == SQL\_SUCCESS\_WITH\_INFO) {

// Imprimir encabezados de las columnas

std::cout << “-------------------------------------------------------------\n”;

std::cout << “| Num Empleado | Num Centro | Prestacion Comb |\n”;

std::cout << “--------------------------------------------------------------\n”;

//Variables para almacenar datos de cada columna

int numEmpleado; int numCentro;

SQLCHAR PrestacionComb[50] ;

// Obtener los resultados fila por fila

while (SQLFetch(hStmt) == SQL\_SUCCESS) {

SQLGetData(hStmt, 1, SQL\_C\_LONG, &numEmpleado, 0, NULL);

SQLGetData(hStmt, 2, SQL\_C\_CHAR, &numCentro, 0, NULL);

SQLGetData(hStmt, 3, SQL\_C\_CHAR, PrestacionComb, sizeof(PrestacionComb), NULL);

std::cout << std::setw(5) << std::left << numEmpleado << “ | “

<< std::setw(17) << std::left << numCentro << “ | “

<< std::setw(15) << std::left << PrestacionComb << “ | “;

// Línea final

std::cout << “-------------------------------------------------------------------------------------------------------------------------\n”;

}

}

* Codigo para la tabla Puestos

#include <windows.h>

#include <sql.h>

#include <sqlext.h>

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

SQLHENV hEnv;

SQLHDBC hDbc;

SQLRETURN ret;

// Asignar un gestor de entorno

ret = SQLAllocHandle(SQL\_HANDLE\_ENV, SQL\_NULL\_HANDLE, &hEnv);

ret = SQLSetEnvAttr(hEnv, SQL\_ATTR\_ODBC\_VERSION, (SQLPOINTER)SQL\_OV\_ODBC3, 0);

// Asignar un gestor de conexión

ret = SQLAllocHandle(SQL\_HANDLE\_DBC, hEnv, &hDbc);

// Conectarse a la base de datos

ret = SQLConnect(hDbc, (SQLWCHAR\*)L”localhost”, SQL\_NTS, (SQLWCHAR\*)L”Username”, SQL\_NTS, (SQLWCHAR\*)L”Password”, SQL\_NTS);

if (ret == SQL\_SUCCESS || ret == SQL\_SUCCESS\_WITH\_INFO) {

cout << “Conectado a la base de datos exitosamente.” << endl;

SQLHSTMT hStmt;

ret = SQLAllocHandle(SQL\_HANDLE\_STMT, hDbc, &hStmt);

if (ret == SQL\_SUCCESS || ret == SQL\_SUCCESS\_WITH\_INFO) {

// Imprimir encabezados de las columnas

std::cout << “-------------------------------------------------------------\n”;

std::cout << “| Num Centro | Nombre Centro | Ciudad |\n”;

std::cout << “--------------------------------------------------------------\n”;

//Variables para almacenar datos de cada columna

int numCentro;

SQLCHAR NombreCentro[50];

SQLCHAR Ciudad[50];

// Obtener los resultados fila por fila

while (SQLFetch(hStmt) == SQL\_SUCCESS) {

SQLGetData(hStmt, 1, SQL\_C\_LONG, &numCentro, 0, NULL);

SQLGetData(hStmt, 2, SQL\_C\_CHAR, NombreCentro, sizeof(NombreCentro), NULL);

SQLGetData(hStmt, 3, SQL\_C\_CHAR, Ciudad, sizeof(Ciudad), NULL);

std::cout << std::setw(5) << std::left << numCentro << “ | “

<< std::setw(17) << std::left << NombreCentro << “ | “

<< std::setw(15) << std::left << Ciudad << “ | “;

// Línea final

std::cout << “-------------------------------------------------------------------------------------------------------------------------\n”;

}

}

Conclusiones

Podemos llegar a la conclusión que las declaraciones para un método son se suma importancia, de igual manera dentro de esta actividad pudimos aprender a como conectar nuestra BD con visual en lenguaje C++, esta conexión nos permitió ya crear las tablas de forma rápida con el código escrito y claro al estar conectada comprueba de manera exitosa el hecho de la información que incluye. Los métodos se declaran y se definen de la misma manera que una función cualquiera en una aplicación convencional de C++ en otras palabras en su firma expresa el valor de retorno.

Usualmente se hace la declaración de los métodos al interior de la clase y por su parte la definición se hace por fuera

Referencias