**《数据库原理》实验报告**

**实验名称 数据库接口实验－１**

**班 级 2014211302**

**学 号 2014211182**

**姓 名 曹桢**

## 实验四 数据库接口实验－１

### 一、实验目的

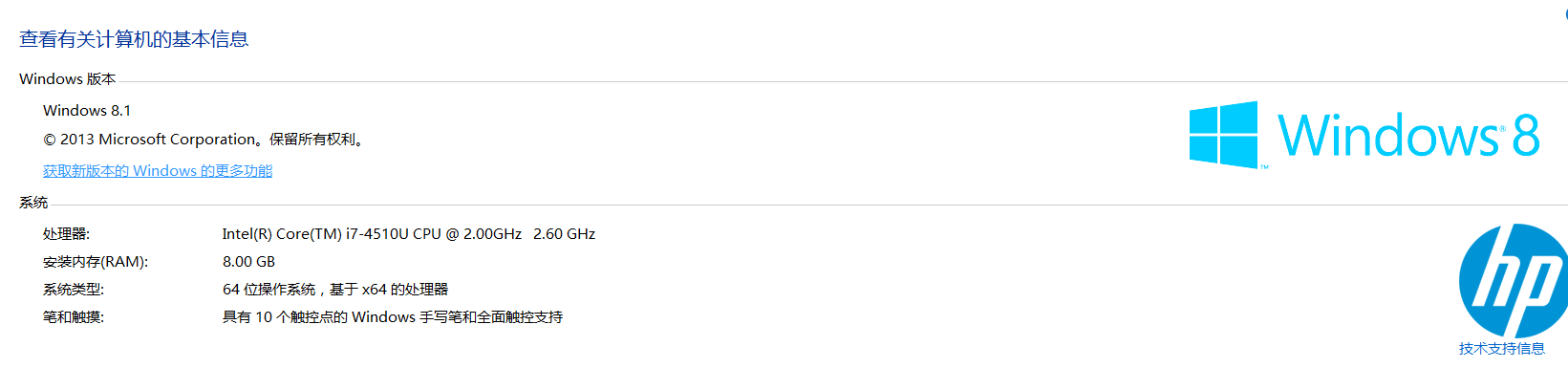
1．通过实验了解通用数据库应用编程接口ODBC的基本原理和实现机制，熟悉主要的ODBC接口的语法和使用方法；

2．利用C语言(或其它支持ODBC接口的高级程序设计语言)编程实现简单的数据库应用程序，掌握基于ODBC的数据库访问的基本原理和方法

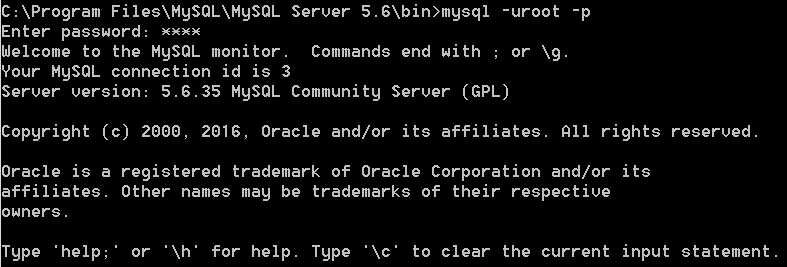
3．学习java语言，并采用jdbc接口方式对数据库进行访问

### 二、实验平台及环境

* 操作系统环境：windows8.1 64位



* 数据库版本：mysql5.6



* 编译器Dev-C++ 5.7.1



* ODBC：mysql-connector-odbc-5.3.7-winx64

### 三、实验内容

1. 以教科书第四章关于SQL语言相关内容为基础，课后查阅、自学ODBC接口有关内容，包括ODBC的体系结构、工作原理、数据访问过程、主要API接口的语法和使用方法等。

2．．以实验二建立的学生数据库为基础，编写 C语言(或其它支持ODBC接口的高级程序设计语言) 数据库应用程序，按照如下步骤访问数据库

* 1. Step1. ODBC初始化，为ODBC分配环境句柄
  2. Step2. 建立应用程序与ODBC数据源的连接
  3. Step3. 利用SQLExecDirect语句，实现数据库应用程序对数据库的建立、查询、修改、删除等操作
  4. Step4. 检索查询结果集
  5. Step5. 结束数据库应用程序

### 四、实验要求

1．要求所编写的数据库访问应用程序中使用到以下主要的ODBC API函数：

1. SQLALLocEnv：初始化ODBC环境，返回环境句柄
2. SQLALLocConnect：为连接句柄分配内存并返回连接句柄
3. SQLConnect：连接一个SQL数据资源

(4) SQLDriverConnect

连接一个SQL数据资源，允许驱动器向用户询问信息   
(5) SQLALLocStmt

为语句句柄分配内存, 并返回语句句柄

(6) **SQLExecDirect**

把SQL语句送到数据库服务器，请求执行由SQL语句定义的数据库访问

(7) SQLFetchAdvances

将游标移动到到查询结果集的下一行(或第一行)

(8) SQLGetData

按照游标指向的位置，从查询结果集的特定的一列取回数据

(9) SQLFreeStmt

释放与语句句柄相关的资源

(10) SQLDisconnect

切断连接

(11) SQLFreeConnect

释放与连接句柄相关的资源

(12) SQLFreeEnv

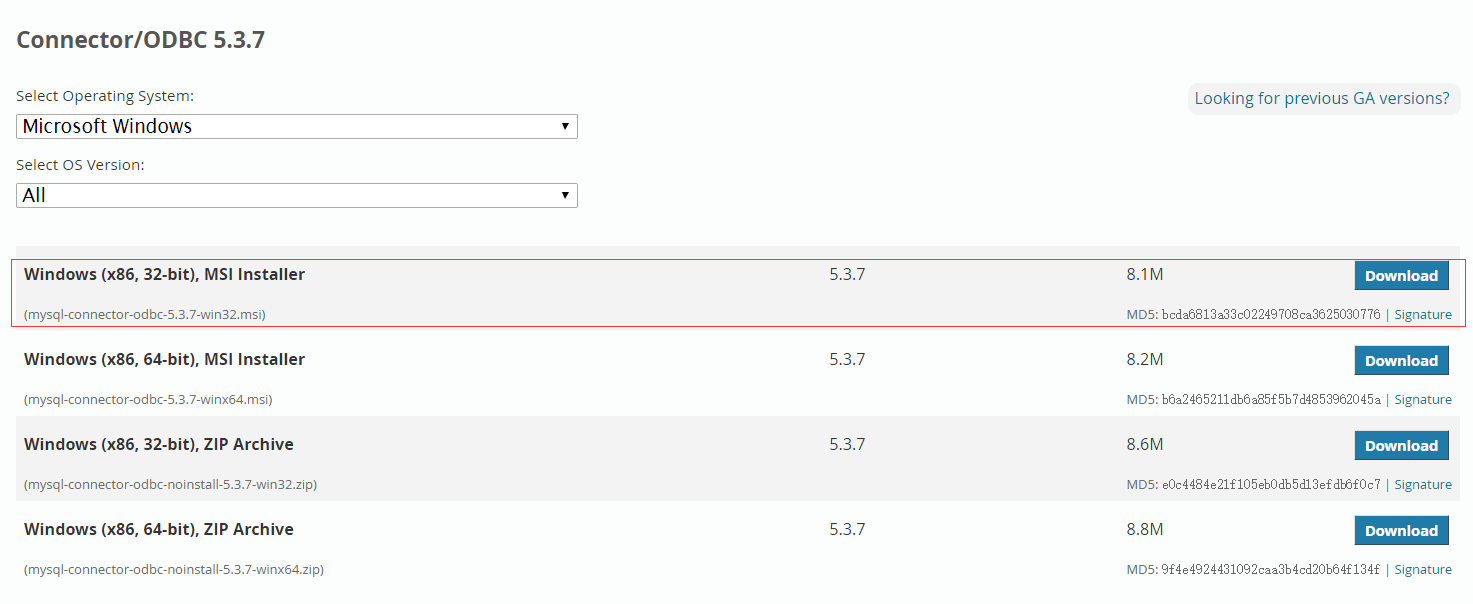
1. 释放与环境句柄相关的资源

### 五、实验步骤

1、下载安装配置ODBC

* 下载ODBC

在官网 <https://dev.mysql.com/downloads/connector/odbc/> 下载对应版本。



* 安装ODBC

参照教程 <https://jingyan.baidu.com/article/4b07be3cb1974e48b380f304.html>



* 配置ODBC
* 进入控制面板->管理工具->ODBC数据源（64位）



* 添加ODBC到DNS

DSN：数据源名称

用户DSN：对当前用户可见，只能用于当前计算机上的数据源。

系统DSN：对当前计算机上所有的用户可见，包括NT服务器。

文件DSN：可有安装了相同驱动程序的用户共享的数据源。

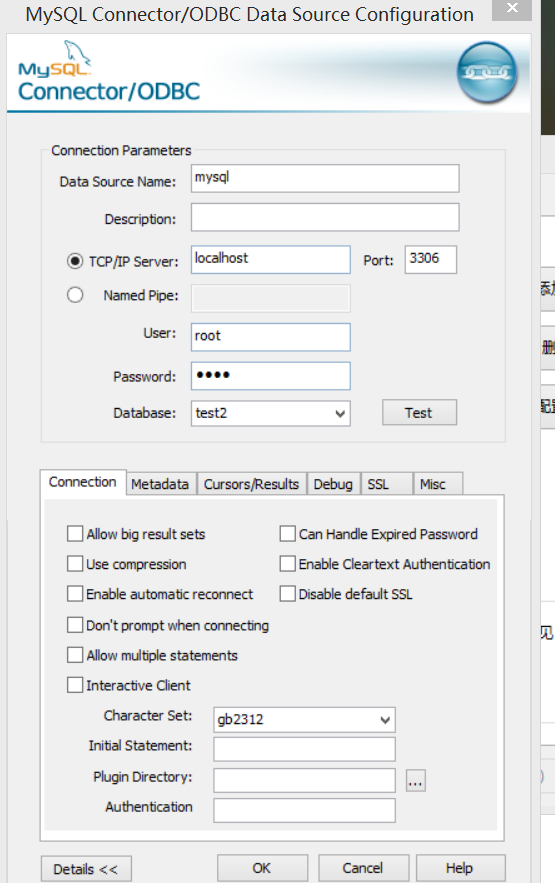
设置用户DSN，连接到本地mysql的数据库。



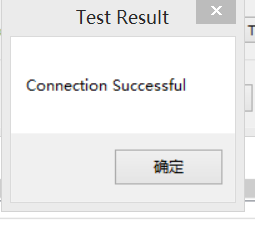


* 配置ODBC

将设置中的编码格式“character set”设置为gb2312



Test，成功



2、配置Devcpp环境

* 新建工程
* 配置环境

工具->编译选项



* 编写代码

#include<stdio.h>

#include<stdlib.h>

#include"windows.h"

#include"sql.h"

#include"sqlext.h"

#include"sqltypes.h"

#include"odbcinst.h"

#pragma comment(lib, "odbc32.lib")

#pragma comment(lib, "libmysql.lib")

struct course\_cell

{

char cno[5];

char cname[20];

int lhour;

int credit;

char semester[5];

};

unsigned char sql\_select[] = "select \* from test2.course"; //SQL查询语句

unsigned char sql\_dolist[3][100] = {

"insert into test2.course values ('C06','数据库原理',48, 3,'')",

"update test2.course set semester='#' where cno='C06'",

"delete from test2.course where cno='C06'"

};

//插入、修改、删除语句

unsigned char do\_name[3][10] = { "insert", "update", "delete" };

void show\_course(HDBC hdbc)

//显示course的内容

{

HSTMT hstmt;

RETCODE retcode;

long long lenOut1, lenOut2, lenOut3, lenOut4, lenOut5;

struct course\_cell\* ccell;

ccell = (struct course\_cell\*)malloc(sizeof(struct course\_cell));

retcode = SQLAllocStmt(hdbc, &hstmt);//SQLALLocStmt；为语句句柄分配内存, 并返回语句句柄

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

//SQLBindCol将应用程序的数据缓冲绑定到结果集的各列

SQLBindCol(hstmt, 1, SQL\_C\_CHAR, ccell->cno, 5, &lenOut1);//将数据缓冲绑定到结果集的列

SQLBindCol(hstmt, 2, SQL\_C\_CHAR, ccell->cname, 20, &lenOut2);

SQLBindCol(hstmt, 3, SQL\_C\_SLONG, &ccell->lhour, sizeof(int), &lenOut3);

SQLBindCol(hstmt, 4, SQL\_C\_SLONG, &ccell->credit, sizeof(int), &lenOut4);

SQLBindCol(hstmt, 5, SQL\_C\_CHAR, ccell->semester, 5, &lenOut5);

retcode = SQLExecDirect(hstmt, sql\_select, SQL\_NTS);

//把SQL语句送到数据库服务器，请求执行由SQL语句定义的数据库访问

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

printf("test2.course is as following:\n");

printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("cno cname lhour credit semester\n");

retcode = SQLFetch(hstmt);

/\*SQLFetch fetches the next rowset of data from the result set and returns data for all bound columns.

相当与SQLFetchAdvances和SQLGetData两个函数\*/

while (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

printf("%-10s%-20s%-15d%-16d%-5s\n", ccell->cno, ccell->cname, ccell->lhour, ccell->credit, ccell->semester);

retcode = SQLFetch(hstmt);

}

printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n \n");

}

}

SQLFreeStmt(hstmt, SQL\_DROP);//释放语句句柄

}

int main()

{

HENV env; //定义环境句柄

HDBC hdbc; //定义链接句柄

HSTMT hstmt; //定义语句句柄

RETCODE retcode;

retcode = SQLAllocEnv(&env); //初始化ODBC环境，返回环境句柄

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

retcode = SQLAllocConnect(env, &hdbc); //为连接句柄分配内存并返回链接句柄

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

retcode = SQLConnect(hdbc, (SQLCHAR\*)"mysql", SQL\_NTS, (SQLCHAR\*)"root", SQL\_NTS, (SQLCHAR\*)"emma", SQL\_NTS); //连接一个SQL数据资料

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

show\_course(hdbc); //显示初始的course内容

for (int i = 0; i<3; i++) //依次执行插入、修改、删除操作

{

retcode = SQLAllocStmt(hdbc, &hstmt);

//为语句句柄分配内存，并返回语句句柄

retcode = SQLExecDirect(hstmt, (SQLCHAR\*)sql\_dolist[i], SQL\_NTS);

//把SQL语句送到数据库服务器，请求执行由SQL语句定义的数据库访问

printf("%s ", do\_name[i]);

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

printf("success!\n");

else printf("fail!\n");

SQLFreeStmt(hstmt, SQL\_DROP); //释放与 语句句柄相关的资源

show\_course(hdbc);//显示操作后的新course 内容

}

SQLDisconnect(hdbc); //切断连接

}

SQLFreeConnect(hdbc); //释放与连接句柄相关的资

}

SQLFreeEnv(env); //释放与环境句柄相关的资源

}

system("pause");

return 0;

}

* 编译运行

### 六、结果分析

1、对数据库的操作为：

unsigned char sql\_dolist[3][100] = {

"insert into test2.course values ('C06','数据库原理',48, 3,'春')",

"update test2.course set semester='#' where cno='C06'",

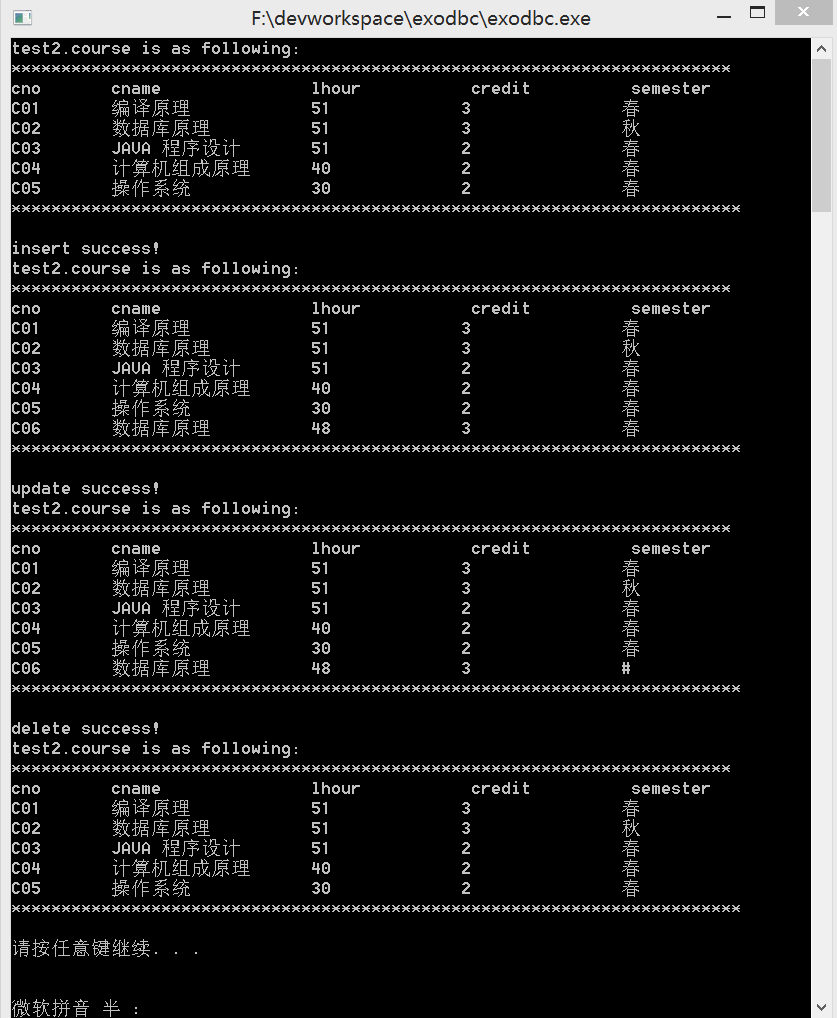
"delete from test2.course where cno='C06'"

};

（1）"insert into test2.course values ('C06','数据库原理',48, 3,'春')",

（2）"update test2.course set semester='#' where cno='C06'",

（3）"delete from test2.course where cno='C06'"



添加一个实体('C06','数据库原理',48, 3,'春')后，course表里多了一行相应信息，添加成功。

修改cno='C06'的semester后，这一项变为'#'，修改成功。

删除后，不再有cno='C06'的行，删除成功。

2、源程序中，有

SQLALLocEnv：初始化ODBC环境，返回环境句柄



SQLALLocConnect：为连接句柄分配内存并返回连接句柄



SQLConnect：连接一个SQL数据资源

SQLDriverConnect连接一个SQL数据资源，允许驱动器向用户询问信息 

SQLALLocStmt为语句句柄分配内存, 并返回语句句柄



**SQLExecDirect**把SQL语句送到数据库服务器，请求执行由SQL语句定义的数据库访问



SQLFetchAdvances将游标移动到查询结果集的下一行(或第一行)

SQLGetData 按照游标指向的位置，从查询结果集的特定的一列取回数据



/\*SQLFetch fetches the next rowset of data from the result set and returns data for all bound columns.相当与SQLFetchAdvances和SQLGetData两个函数\*/

SQLFreeStmt释放与语句句柄相关的资源



SQLDisconnect 切断连接



SQLFreeConnect 释放与连接句柄相关的资源



SQLFreeEnv

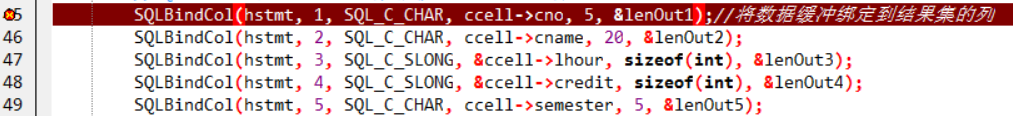


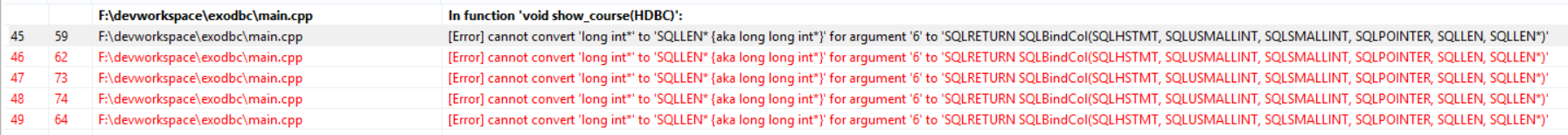
实验要求操作均在代码中体现。

### 七、实验小结

(一)问题和收获

1、声明long lenOut1, lenOut2, lenOut3, lenOut4, lenOut5;报错





解决：将long改成long long，则可编译

2、一开始在vs2010中编译运行，修改所有编译的问题后，代码如下

#include<stdio.h>

#include<stdlib.h>

#include"windows.h"

#include"sql.h"

#include"sqlext.h"

#include"sqltypes.h"

#include"odbcinst.h"

#pragma comment(lib, "odbc32.lib")

#pragma comment(lib, "libmysql.lib")

struct course\_cell

{

char cno[5];

char cname[20];

int lhour;

int credit;

char semester[5];

};

unsigned char sql\_select[] = "select \* from test2.course"; //SQL查询语句

unsigned char sql\_dolist[3][100] = {

"insert into test2.course values ('C06','数据库原理',48, 3,'')",

"update test2.course set semester='#' where cno='C06'",

"delete from test2.course where cno='C06'"

};

//插入、修改、删除语句

unsigned char do\_name[3][10] = { "insert", "update", "delete" };

void show\_course(HDBC hdbc)

//显示course的内容

{

HSTMT hstmt;

RETCODE retcode;

long lenOut1, lenOut2, lenOut3, lenOut4, lenOut5;

struct course\_cell\* ccell;

ccell = (struct course\_cell\*)malloc(sizeof(struct course\_cell));

retcode = SQLAllocStmt(hdbc, &hstmt);//SQLALLocStmt；为语句句柄分配内存, 并返回语句句柄

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

//SQLBindCol将应用程序的数据缓冲绑定到结果集的各列

SQLBindCol(hstmt, 1, SQL\_C\_CHAR, ccell->cno, 5, &lenOut1);//将数据缓冲绑定到结果集的列

SQLBindCol(hstmt, 2, SQL\_C\_CHAR, ccell->cname, 20, &lenOut2);

SQLBindCol(hstmt, 3, SQL\_C\_SLONG, &ccell->lhour, sizeof(int), &lenOut3);

SQLBindCol(hstmt, 4, SQL\_C\_SLONG, &ccell->credit, sizeof(int), &lenOut4);

SQLBindCol(hstmt, 5, SQL\_C\_CHAR, ccell->semester, 5, &lenOut5);

retcode = SQLExecDirect(hstmt,(SQLWCHAR\*)sql\_select, SQL\_NTS);

//把SQL语句送到数据库服务器，请求执行由SQL语句定义的数据库访问

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

printf("test2.course is as following:\n");

printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("cno cname lhour credit semester\n");

retcode = SQLFetch(hstmt);

/\*SQLFetch fetches the next rowset of data from the result set and returns data for all bound columns.

相当与SQLFetchAdvances和SQLGetData两个函数\*/

while (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

printf("%-10s%-20s%-15d%-16d%-5s\n", ccell->cno, ccell->cname, ccell->lhour, ccell->credit, ccell->semester);

retcode = SQLFetch(hstmt);

}

printf ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n \n");

}

}

SQLFreeStmt(hstmt, SQL\_DROP);//释放语句句柄

}

int main()

{

HENV env; //定义环境句柄

HDBC hdbc; //定义链接句柄

HSTMT hstmt; //定义语句句柄

RETCODE retcode;

retcode = SQLAllocEnv(&env); //初始化ODBC环境，返回环境句柄

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

retcode = SQLAllocConnect(env, &hdbc); //为连接句柄分配内存并返回链接句柄

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

retcode = SQLConnect(hdbc, (SQLWCHAR\*)"mysql", SQL\_NTS, (SQLWCHAR\*)"root", SQL\_NTS, (SQLWCHAR\*)"emma", SQL\_NTS); //连接一个SQL数据资料

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

{

printf("in if");

show\_course(hdbc); //显示初始的course内容

for (int i = 0; i<3; i++) //依次执行插入、修改、删除操作

{

retcode = SQLAllocStmt(hdbc, &hstmt);

//为语句句柄分配内存，并返回语句句柄

retcode = SQLExecDirect(hstmt, (SQLWCHAR\*)sql\_dolist[i], SQL\_NTS);

//把SQL语句送到数据库服务器，请求执行由SQL语句定义的数据库访问

printf("%s ", do\_name[i]);

if (retcode == SQL\_SUCCESS || retcode == SQL\_SUCCESS\_WITH\_INFO)

printf("success!\n");

else printf("fail!\n");

SQLFreeStmt(hstmt, SQL\_DROP); //释放与 语句句柄相关的资源

show\_course(hdbc);//显示操作后的新course 内容

}

SQLDisconnect(hdbc); //切断连接

}

SQLFreeConnect(hdbc); //释放与连接句柄相关的资

}

SQLFreeEnv(env); //释放与环境句柄相关的资源

}

system("pause");

return 0;

}

* 可以用long lenOut1, lenOut2, lenOut3, lenOut4, lenOut5;
* 需要强制类型转化retcode = SQLConnect(hdbc, (SQLWCHAR\*)"mysql", SQL\_NTS, (SQLWCHAR\*)"root", SQL\_NTS, (SQLWCHAR\*)"emma", SQL\_NTS);
* 但是，在下面不能成功打开数据库，无法进入if循环

3、最开始设定字符集编解码方式为utf8，这样程序运行打印的是乱码，需要设置为gb2312

(二)心得

这次实验，我熟悉了通用数据库应用变成接口ODBC的使用方法，并对MySQL中的ODBC的使用和ODBC接口的语法和函数等有了进一步的了解。在devcpp中编写C语言，通过字符串编写SQL代码，并在数据库中进行相应操作。链接ODBC，我们通过编写的C/C++语言代码可以访问数据库，这样在编写大作业时也可以通过创建和增删查该数据库来操作我们的信息。事实上，在大二数据结构大作业用C++和qt编码旅行模拟系统时，已经使用了数据库来存取用户信息；在这学期java编写的软件工程分布式空调大作业中，我们使用了数据库来保存日志信息。但是，这次实验让我系统地理解了ODBC的原理和使用，对于各个接口和函数有了更深的理解，相信在今后的学习中，能够在编码中更加熟练地运用。