**暨南大学本科实验报告专用纸**

课程名称 数据库系统原理实验 成绩评定

实验项目名称 自动随机考勤系统 指导教师 武建华

实验项目编号 3 实验项目类型 操作型 实验地点

学生姓名 李炜鹏 学号 2017052544

学院 智能科学与工程学院 系 专业 信息安全

实验时间2019年12月1日 午～12月10日 午 温度 ℃湿度

1. **实验目的**
2. 理解并熟悉数据库编程
3. 利用嵌入式SQL设计一个数据库系统，深入理解嵌入式SQL编程
4. **实验内容和要求**
5. 实验介绍：随机读取数据库中的记录来生成一份可以任意指定规模到名单，来实现随机考勤。数据库中存放的是职工信息，包括职工号、职工姓名、职工所在部门、部门领导、每一次考勤记录、考勤成绩等。采用嵌入式的SQL编程实现。
6. 基本规则：1）每一次考勤，不在名单中的职工默认考勤通过 2）考勤通过者，相应的考勤成绩+1，且在相应的考勤记录中记录“通过” 3）连续三次考勤不通过，考勤成绩置0 4）累计5次考勤不通过，考勤成绩永久置0。
7. 附加部分：1）若不同的职工信息放在不同的DBMS中。例如，部门A的职工信息放在Kingbase中，部门B的职工信息放在Oracle中，采用ODBC/JDBC等编程实现上述规则，例如突然想考察某位职工，修改某位职工的信息（这些在事先是不确定的，所以要用动态SQL） 3）对比两者之间存在的效率差异
8. 实验说明：自定义测试案例，一次给出程序对上述规则的实现情况，实现报告附加源代码。（抽查代码的相关性检测，独自完成）
9. **主要仪器设备**

**仪器：**计算机

**实验环境：**Dev-C++、MySQL、CMD命令提示符

1. **源程序**

**attendence\_system.h**

#pragma once

#include <iostream>

#include <iomanip>

#include <vector>

#include <string>

#include "mysql.h"

using namespace std;

void show\_database(MYSQL mysql);//数据库操作系统

void attendence(MYSQL mysql);//考勤功能

void change\_data(MYSQL mysql);//修改数据功能

bool ConnectDatabase(MYSQL &mysql);//初始化

bool InsertData(MYSQL& mysql, const int id, const char \*name, const char \*department, const char \*leader, int record, int grade, const char \*attdence);//插入数据函数

bool SelectData(MYSQL &mysql, const char \*columns\_name, const char \*table\_name);//选择函数,没有where的情况

bool SelectData(MYSQL &mysql, const char \*columns\_name, const char \*table\_name, const char \*limited\_info);//重载有where的情况

bool UpdateData(MYSQL &mysql, const char \*table\_name, const char \*modified\_info, const char \*limitde\_info);//修改操作

bool DeleteData(MYSQL &mysql, const char \*table\_name, const char \*limited\_info);//删除数据函数

**attendence\_system.cpp**

#include "attendence\_system.h"

#include <string>

using namespace std;

void show\_database(MYSQL mysql)

{

SelectData(mysql, "\*", "clerk");

cout << endl;

cout << "服务类型表如下:" << endl

<< "1. 考勤" << endl

<< "2. 修改数据" << endl

<< "3. 退出系统" << endl << endl;

cout << "请输入您所需要的服务类型: " << endl;

int input\_server;

cin >> input\_server;

while (input\_server != 3)

{

switch (input\_server)

{

case 1:

{

attendence(mysql);

break;

}

case 2:

{

change\_data(mysql);

break;

}

case 3:

{

break;

}

default:

{

cout << "Unvalid select number!" << endl;

break;

}

}

cout << "服务类型表如下:" << endl

<< "1. 考勤" << endl

<< "2. 修改数据" << endl

<< "3. 退出系统" << endl << endl;

cout << "请输入您所需要的服务类型: " << endl;

cin >> input\_server;

}

cout << endl << "请您确认操作后的数据库，表格如下:" << endl << endl;

SelectData(mysql, "\*", "clerk");

cout << endl << endl << "本次数据库服务结束，感谢您的使用！" << endl << endl;

}

void attendence(MYSQL mysql)

{

MYSQL\_RES \* result; //保存结果集

MYSQL\_ROW row; //代表的是结果集中的一行(返回string类的数据)

cout << endl << "可参与考勤的教职工的职工号如下：" << endl;

const char \* i\_query = "select \* from clerk"; //查询语句

mysql\_query(&mysql, i\_query);

result = mysql\_store\_result(&mysql);

while ((row = mysql\_fetch\_row(result)) != NULL)

cout << row[0] << endl;

cout << endl << "考勤名单如下：\n请输入本次考勤的职工号(未参与本次考勤的教职工默认通过考勤/使用Crtl+Z结束输入)：" << endl;

int temp;

vector<int> input\_id;

while (cin >> temp)

{

input\_id.push\_back(temp);

}

cout << "输入完成！" << endl << endl;

cout << "请输入本次考勤情况(y代表到场,n代表缺席):" << endl;

char attendence;

for (vector<int>::iterator iter = input\_id.begin(); iter < input\_id.end(); iter++) //本次考勤名单的考勤

{

mysql\_query(&mysql, i\_query);

result = mysql\_store\_result(&mysql);

while ((row = mysql\_fetch\_row(result)) != NULL)

{

//WHERE后面的条件

const string limitde\_info = "id=";

const string limit = limitde\_info + row[0];

//C++提供了两个函数将string转化为char \*：c\_str()和data()

const char \* lim = limit.c\_str();

if (\*iter == stoi(row[0], nullptr, 10))

{

cout << \*iter << ":";

cin.clear(); //错误位重置

cin >> attendence;

if (attendence == 'y')

{

UpdateData(mysql, "clerk", "attendence='pass'", lim);

UpdateData(mysql, "clerk", "grade = grade + 1", lim);

UpdateData(mysql, "clerk", "not\_pass\_times = 0", lim);

const string record = row[4];

if (stoi(record, nullptr, 10) >= 5) //超过五次考勤不通过永久置零

{

UpdateData(mysql, "clerk", "grade = 0", lim);

}

cout << \*iter << " " << row[1] << ": pass!" << endl << endl;

break;

}

else if (attendence == 'n')

{

UpdateData(mysql, "clerk", "attendence='not pass'", lim);

// UpdateData(mysql, "clerk", "record += 1", lim); //+=在数据库里面不可用

UpdateData(mysql, "clerk", "record = record + 1", lim);

UpdateData(mysql, "clerk", "not\_pass\_times = not\_pass\_times + 1", lim);

const string record = row[4];

//超过五次考勤不通过永久置零

if (stoi(record, nullptr, 10) >= 5)

{

UpdateData(mysql, "clerk", "grade = 0", lim);

}

//连续三次考勤不通过，grade置零

if (stoi(row[7], nullptr, 10) >= 3)

{

UpdateData(mysql, "clerk", "grade = 0", lim);

}

cout << \*iter << " " << row[1] << ": not pass!" << endl << endl;

break;

}

}

}

}

mysql\_query(&mysql, i\_query);

result = mysql\_store\_result(&mysql);

vector<int> not\_input\_id;

int exist = 0;

cout << "未参与本次考勤的名单如下(默认考勤通过)：" << endl;

//不在考勤名单上的默认考勤通过

while ((row = mysql\_fetch\_row(result)) != NULL)

{

const string limitde\_info = "id=";

const string limit = limitde\_info + row[0];

const char \* lim = limit.c\_str();

for (vector<int>::iterator iter = input\_id.begin(); iter < input\_id.end(); iter++) //这里不能input\_id.end()，所以不能取等号

{

if (\*iter == stoi(row[0], nullptr, 10))

{

exist = 1;

break;

}

else

exist = 0;

}

if (exist == 0)

{

UpdateData(mysql, "clerk", "attendence='pass'", lim);

UpdateData(mysql, "clerk", "grade = grade + 1", lim);

UpdateData(mysql, "clerk", "not\_pass\_times = 0", lim);

const string record = row[4];

//超过五次考勤不通过永久置零

if (stoi(record, nullptr, 10) >= 5)

{

UpdateData(mysql, "clerk", "grade = 0", lim);

}

cout << row[0] << " " << row[1] << ": pass!" << endl << endl;

not\_input\_id.push\_back(stoi(row[0], nullptr, 10));

}

}

}

void change\_data(MYSQL mysql)

{

cout << endl;

cout << "请输入要修改的数据对应的职工号:" << endl;

int input\_number;

cin >> input\_number;

MYSQL\_RES \* result;

MYSQL\_ROW row;

string limitde\_info;

string limit;

const char \* i\_query = "select \* from clerk";

mysql\_query(&mysql, i\_query);

result = mysql\_store\_result(&mysql);

while ((row = mysql\_fetch\_row(result)) != NULL)

{

if (input\_number == stoi(row[0], nullptr, 10))

{

limitde\_info = "id=";

limit = limitde\_info + row[0];

break;

}

}

const char \* lim = limit.c\_str();

cout << endl;

cout << "请输入要修改的属性对应的序号:" << endl

<< "1. id" << endl

<< "2. name" << endl

<< "3. department" << endl

<< "4. leader" << endl

<< "5. record" << endl

<< "6. grade" << endl

<< "7. attendence" << endl

<< "8. not\_pass\_times" << endl << endl;

int input\_select;

cin >> input\_select;

switch (input\_select)

{

case 1:

{//这个大括号不可少，不然会报错

cout << endl << "Please enter a new id: " << endl;

int new\_id;

cin >> new\_id;

cout << endl;

const string id1 = "id=";

const string id2 = id1 + to\_string(new\_id);

const char \* id = id2.c\_str();

UpdateData(mysql, "clerk", id, lim);

cout << "Change id success!" << endl;

break;

}

case 2:

{

cout << endl << "Please enter a new name: " << endl;

string new\_name;

cin >> new\_name;

cout << endl;

const string name1 = "name=";

const string name2 = name1 + new\_name;

const char \* name = name2.c\_str();

UpdateData(mysql, "clerk", name, lim);

cout << "Change name success!" << endl;

break;

}

case 3:

{

cout << endl << "Please enter a new department: " << endl;

string new\_department;

cin >> new\_department;

cout << endl;

const string department1 = "department=";

const string department2 = department1 + new\_department;

const char \* department = department2.c\_str();

UpdateData(mysql, "clerk", department, lim);

cout << "Change department success!" << endl;

break;

}

case 4:

{

cout << endl << "Please enter a new leader: " << endl;

string new\_leader;

cin >> new\_leader;

cout << endl;

const string leader1 = "leader=";

const string leader2 = leader1 + new\_leader;

const char \* leader = leader2.c\_str();

UpdateData(mysql, "clerk", leader, lim);

cout << "Change leader success!" << endl;

break;

}

case 5:

{

cout << endl << "Please enter a new record: " << endl;

int new\_record;

cin >> new\_record;

cout << endl;

const string record1 = "record=";

const string record2 = record1 + to\_string(new\_record);

const char \* record = record2.c\_str();

UpdateData(mysql, "clerk", record, lim);

cout << "Change record success!" << endl;

break;

}

case 6:

{

cout << endl << "Please enter a new grade: " << endl;

int new\_grade;

cin >> new\_grade;

cout << endl;

const string grade1 = "grade=";

const string grade2 = grade1 + to\_string(new\_grade);

const char \* grade = grade2.c\_str();

UpdateData(mysql, "clerk", grade, lim);

cout << "Change grade success!" << endl;

break;

}

case 7:

{

cout << endl << "Please enter a new attendence: " << endl;

string new\_attendence;

cin >> new\_attendence;

cout << endl;

const string attendence1 = "attendence=";

const string attendence2 = attendence1 + new\_attendence;

const char \* attendence = attendence2.c\_str();

UpdateData(mysql, "clerk", attendence, lim);

cout << "Change attendence success!" << endl;

break;

}

case 8:

{

cout << endl << "Please enter a new not\_pass\_times: " << endl;

int new\_not\_pass\_times;

cin >> new\_not\_pass\_times;

cout << endl;

const string not\_pass\_times1 = "not\_pass\_times=";

const string not\_pass\_times2 = not\_pass\_times1 + to\_string(new\_not\_pass\_times);

const char \* not\_pass\_times = not\_pass\_times2.c\_str();

UpdateData(mysql, "clerk", not\_pass\_times, lim);

cout << "Change not\_pass\_times success!" << endl;

break;

}

default:

{

cout << endl << "Unvalid select number!" << endl;

break;

}

}

}

//初始化

bool ConnectDatabase(MYSQL &mysql)

{

mysql\_init(&mysql);

if (!mysql\_real\_connect(&mysql, "192.168.209.1(备注一下：这里填写服务器的ip)", "root", "password", "attendence\_check", 3306, NULL, 0)) {

printf("Error connecting to database:%s\n", mysql\_error(&mysql));

return false;

}

else {

mysql\_query(&mysql, "set names gbk");

printf("Connected..\n");

return true;

}

}

//插入数据函数

bool InsertData(MYSQL& mysql, const int id, const char \*name, const char \*department, const char \*leader, int record, int grade, const char \*attdence)

{

char insertsql[1024];

int len = sprintf\_s(insertsql, 256, "INSERT INTO table\_name values (%d,'%s','%s','%s',%d,%d,'%s');", id, name, department, leader, record, grade, attdence);

printf("%s\n", insertsql);

if (len < 0) {

printf("Insert data failed!\n");

return false;

}

if (mysql\_query(&mysql, insertsql) == 0) {

printf("Insert data success!\n");

return true;

}

else {

printf("Insert failed (%s)\n", mysql\_error(&mysql));

return false;

}

}

//选择函数（没有where的情况）

bool SelectData(MYSQL &mysql, const char \*columns\_name, const char \*table\_name)

{

char querysql[1024];

int len = sprintf\_s(querysql, 256, "SELECT %s FROM %s;", columns\_name, table\_name);

printf("%s\n", querysql);

if (len < 0) {

printf("Select data failed!\n");

return false;

}

if (mysql\_query(&mysql, querysql) == 0) {

printf("Select data success!\n");

}

else {

printf("Select failed (%s)\n", mysql\_error(&mysql));

return false;

}

MYSQL\_RES \*res; //返回行的查询结果集

MYSQL\_FIELD \*fd; //字段列数组

MYSQL\_ROW column;

char field[32][32];

//查询返回的数据

res = mysql\_store\_result(&mysql);

//打印数据行数

printf("number of dataline returned:%d\n", mysql\_affected\_rows(&mysql));

//获取列数

int column\_num = mysql\_num\_fields(res);

//输出选择完的列名称

for (int i = 0; fd = mysql\_fetch\_field(res); i++)

strcpy\_s(field[i], strlen(fd->name) + 1, fd->name);

for (int i = 0; i < column\_num; i++)

printf("%10s\t", field[i]);

printf("\n");

//输出每一行的数据

while (column = mysql\_fetch\_row(res))

{

for (int i = 0; i < column\_num; i++)

printf("%10s\t", column[i]);

printf("\n");

}

return true;

}

//重载查询（有where的情况）

bool SelectData(MYSQL &mysql, const char \*columns\_name, const char \*table\_name, const char \*limited\_info)

{

char querysql[1024];

int len = sprintf\_s(querysql, 256, "SELECT %s FROM %s WHERE %s;", columns\_name, table\_name, limited\_info);

printf("%s\n", querysql);

if (len < 0) {

printf("Select data failed!\n");

return false;

}

if (mysql\_query(&mysql, querysql) == 0) {

printf("Select data success!\n");

}

else {

printf("Select failed (%s)\n", mysql\_error(&mysql));

return false;

}

MYSQL\_RES \*res; //返回行的查询结果集

MYSQL\_FIELD \*fd; //字段列数组

MYSQL\_ROW column;

char field[32][32];

//查询返回的数据

res = mysql\_store\_result(&mysql);

//打印数据行数

printf("number of dataline returned:%d\n", mysql\_affected\_rows(&mysql));

//获取列数

int column\_num = mysql\_num\_fields(res);

//输出选择完的列名称

for (int i = 0; fd = mysql\_fetch\_field(res); i++)

strcpy\_s(field[i], strlen(fd->name) + 1, fd->name);

for (int i = 0; i < column\_num; i++)

printf("%10s\t", field[i]);

printf("\n");

//输出每一行的数据

while (column = mysql\_fetch\_row(res)) //返回结果集中的下一行

{

for (int i = 0; i < column\_num; i++)

printf("%10s\t", column[i]);

printf("\n");

}

return true;

}

//修改操作

bool UpdateData(MYSQL &mysql, const char \*table\_name, const char \*modified\_info, const char \*limitde\_info)

{

char updatesql[1024];

int len = sprintf\_s(updatesql, 256, "UPDATE %s SET %s WHERE %s;", table\_name, modified\_info, limitde\_info);

printf("%s\n", updatesql);

if (len < 0) {

printf("Update data failed!\n");

return false;

}

if (mysql\_query(&mysql, updatesql) == 0) {

printf("Update data success!\n");

return true;

}

else {

printf("Update failed (%s)\n", mysql\_error(&mysql));

return false;

}

}

//删除数据函数

bool DeleteData(MYSQL &mysql, const char \*table\_name, const char \*limited\_info)

{

char deletesql[1024];

int len = sprintf\_s(deletesql, 256, "DELETE FROM %s WHERE %s;", table\_name, limited\_info);

printf("%s\n", deletesql);

if (len < 0) {

printf("Delete data failed!\n");

return false;

}

if (mysql\_query(&mysql, deletesql) == 0) {

printf("Delete data success!\n");

return true;

}

else {

printf("Delete failed (%s)\n", mysql\_error(&mysql));

return false;

}

}

**main.cpp**

#include "mysql.h"

#include "attendence\_system.h"

#include <iostream>

#include<iomanip>

using namespace std;

MYSQL mysql, \*sock;//声明MySQL的句柄

int main(void)

{

ConnectDatabase(mysql); //连接数据库

cout << endl;

show\_database(mysql); //显示数据库操作系统

mysql\_close(sock); //关闭数据库

system("pause");

exit(EXIT\_SUCCESS);

}

**MySQL建表代码：**

CREATE TABLE clerk (

‘id’ char(12) NOT NULL,

‘name’ char(20) NOT NULL,

‘department’ char(20) NOT NULL,

‘leader’ char(20) NOT NULL,

‘record’ int(11) NOT NULL,

‘grade’ int(11) NOT NULL,

‘attendence’ char(10) check (attendence in ('pass','not pass')),

‘not\_pass\_times’ int(2),

PRIMARY KEY (‘id’),

UNIQUE KEY ‘id\_UNIQUE’ (‘id’),

UNIQUE KEY ‘name\_UNIQUE’ (‘name’)

)

**clerk表初始化代码：**

INSERT INTO attendence\_check.clerk

(‘id’,’name’,’department’,’leader’,’record’,’grade’,’attendence’,’not\_pass\_times’)

VALUES

('2019000000','张三','教育部门','教育部部长', 0, 5,'pass',0),

('2019000001','李四','行政部门','行政部部长', 1, 4,'pass',0),

('2019000002','王五','科研部门','科研部部长', 2, 3,'pass',0),

('2019000003','胡六','艺术部门','艺术部部长', 3, 2,'pass',0),

('2019000004','曾七','教育部门','教育部部长', 0, 10,'pass',0),

('2019000005','陈八','艺术部门','艺术部部长', 1, 9,'pass',0),

('2019000006','黄九','体育部门','体育部部长', 2, 8,'pass',0),

('2019000007','牛十','宣传部门','宣传部部长', 3, 7,'pass',0),

('2019000008','张十一','艺术部门','艺术部部长', 0, 15,'pass',0),

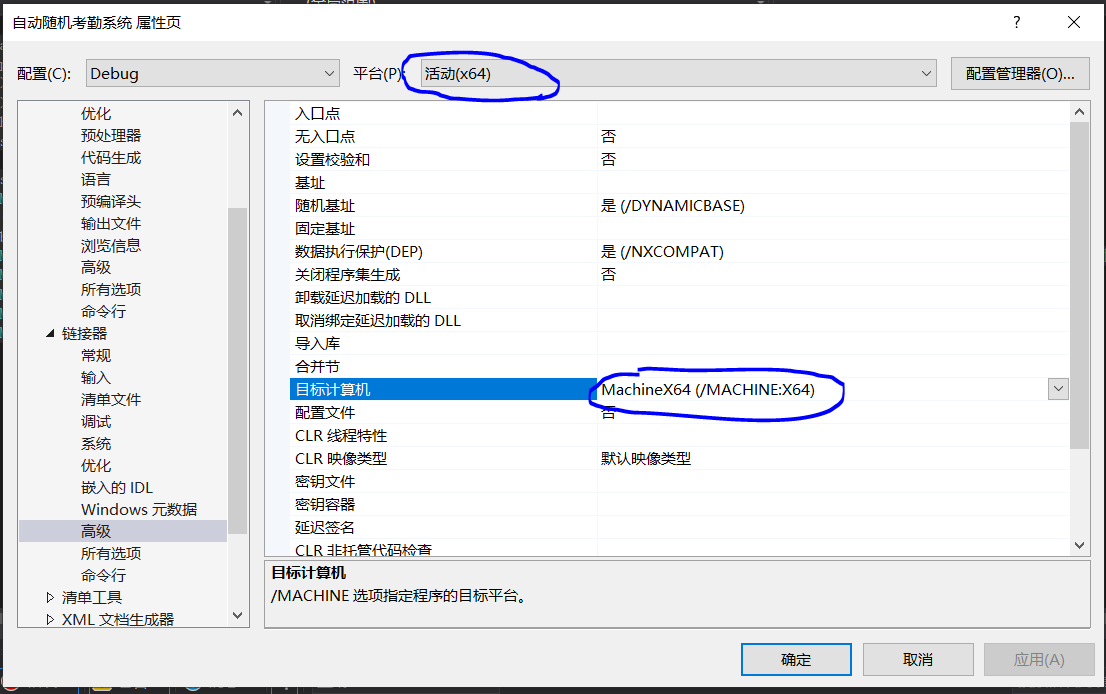
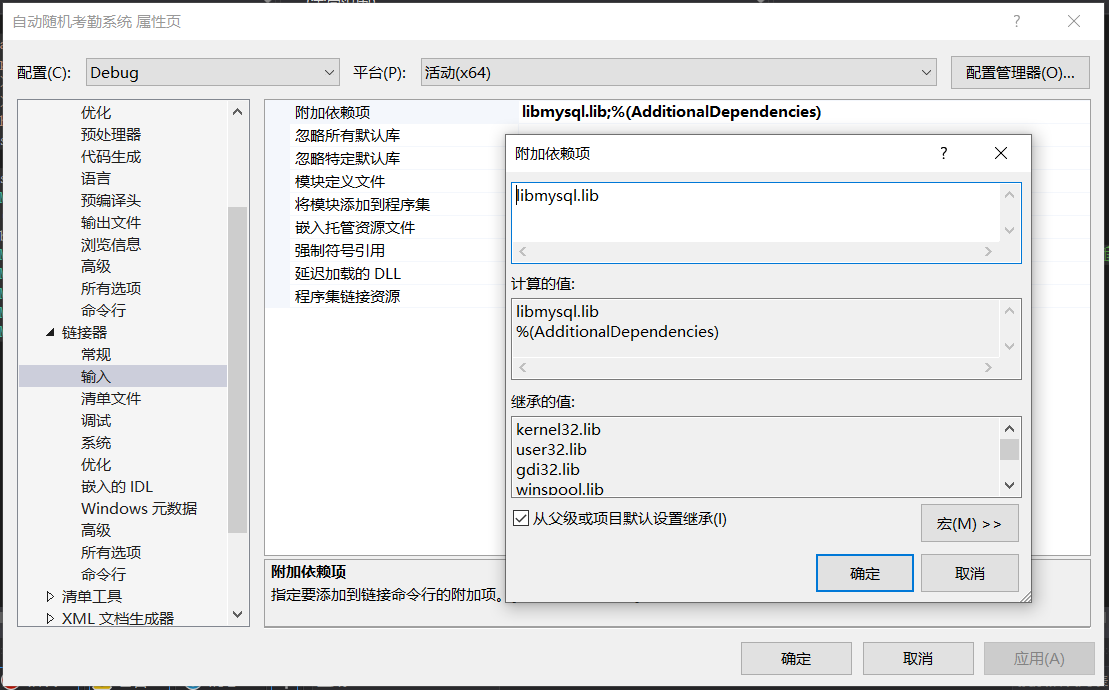
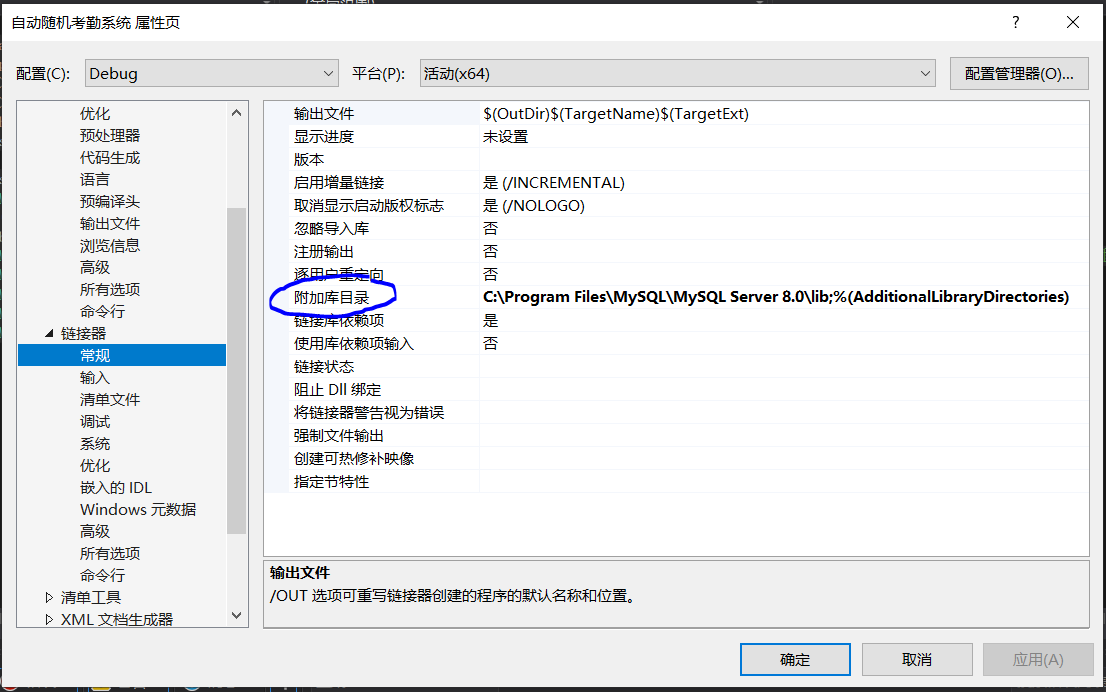
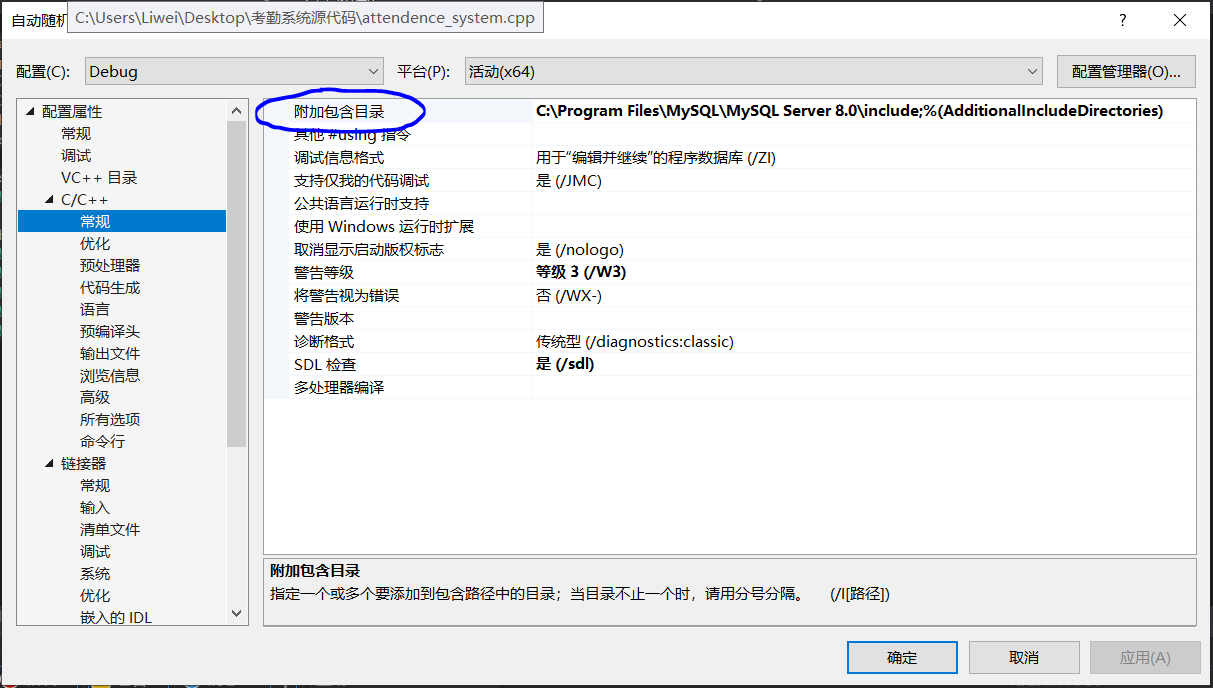
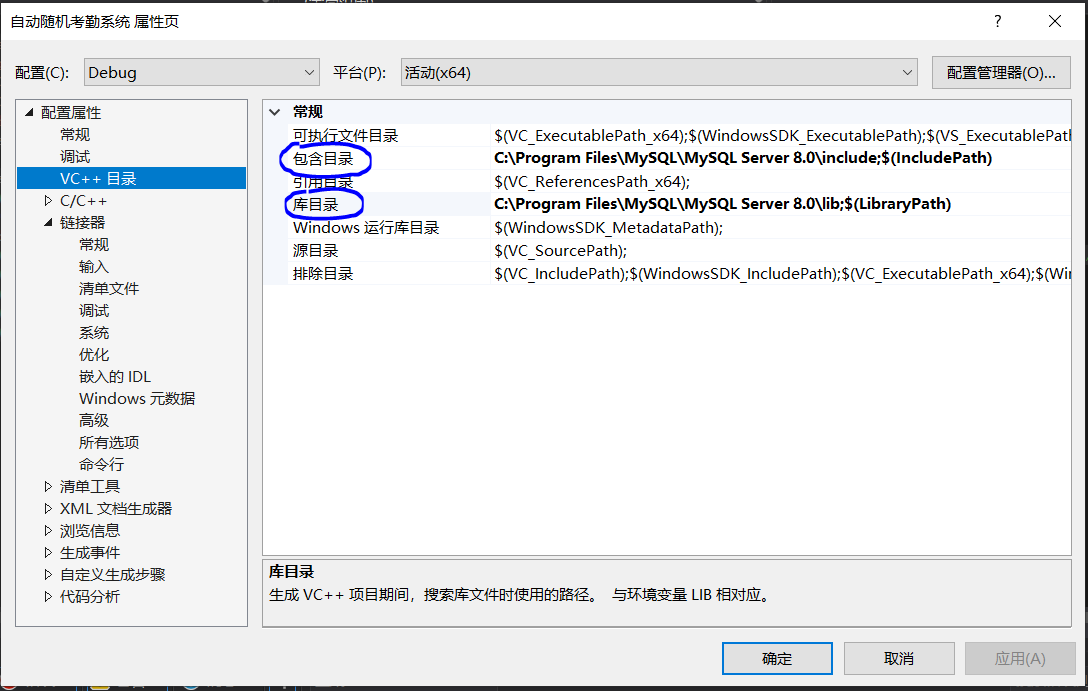
('2019000009','武十二','行政部门','行政部部长', 1, 14,'pass',0),

('2019000010','刘十三','科研部门','科研部部长', 2, 13,'pass',0),

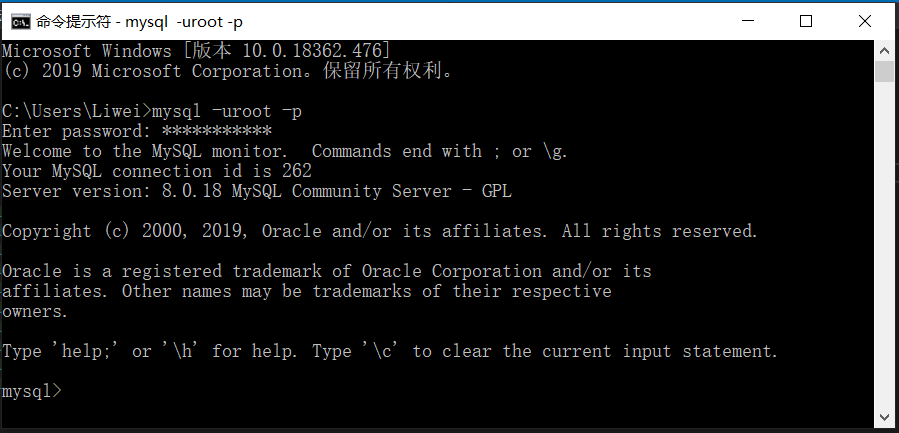
('2019000011','卢十四','体育部门','体育部部长', 3, 12,'pass',0);

select \* from clerk;

1. **实验步骤与调试**
2. 首先配置好开发环境，本实验使用的是嵌入式SQL，开发环境是Visual Studio2017和MySQL，要实现两者连接起来，则需要在Visual Studio2017的属性，如下图



1. 配置好开发环境之后，首先在命令提示符CMD里面查看数据库attendence\_check中表clerk的数据，以便和后面数据库系统中比对数据。
   1. 输入mysql –uroot –p，表示使用密码登录MySQL

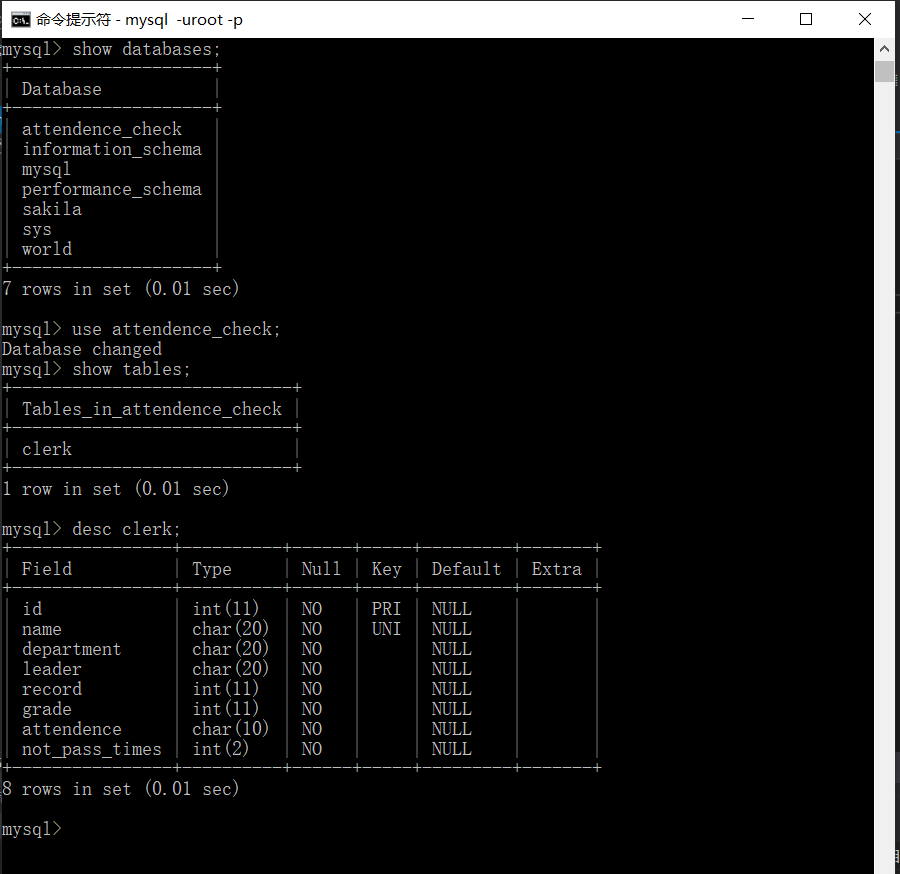


* 1. 输入show databases; //显示当前所有数据库

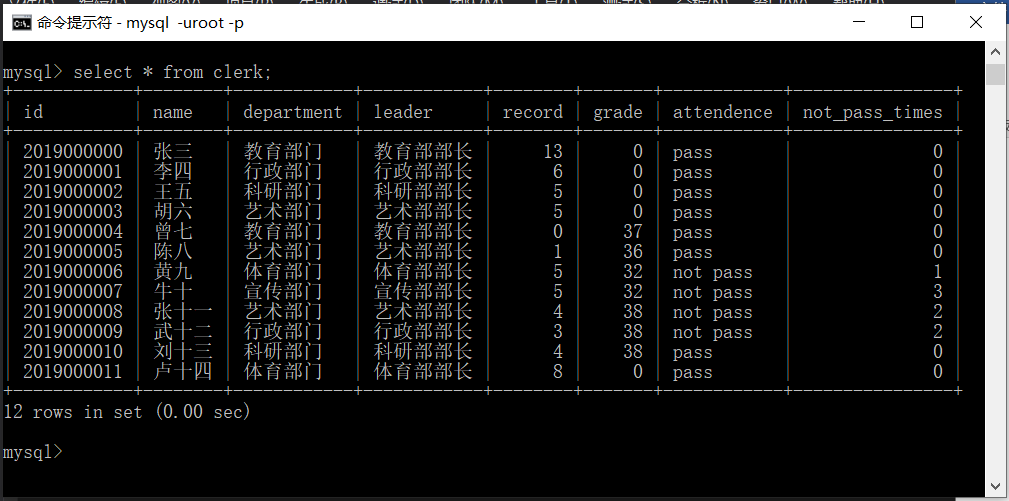
输入use attendance\_check; //进入数据库attendance\_check

输入show tables; //显示当前数据库所有的表

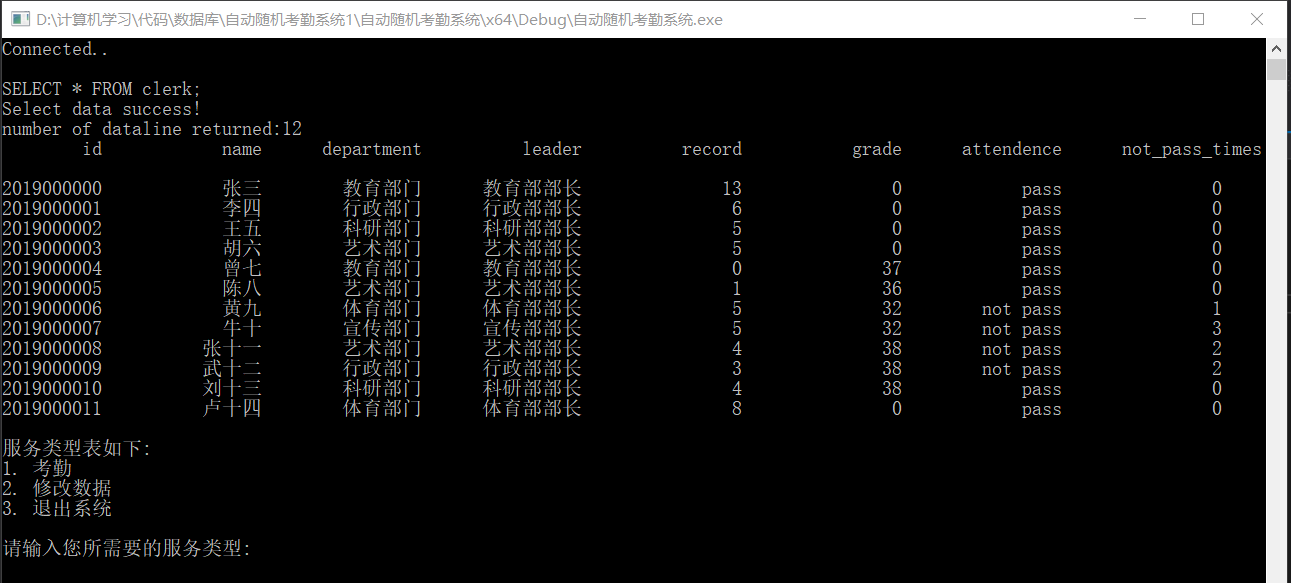
输入desc clerk; //进入表clerk



* 1. 输入select \* from clerk; //显示clerk表的所有数据



1. 回到Visual Studio2017里面，运行程序：



* 1. 输入1，回车；

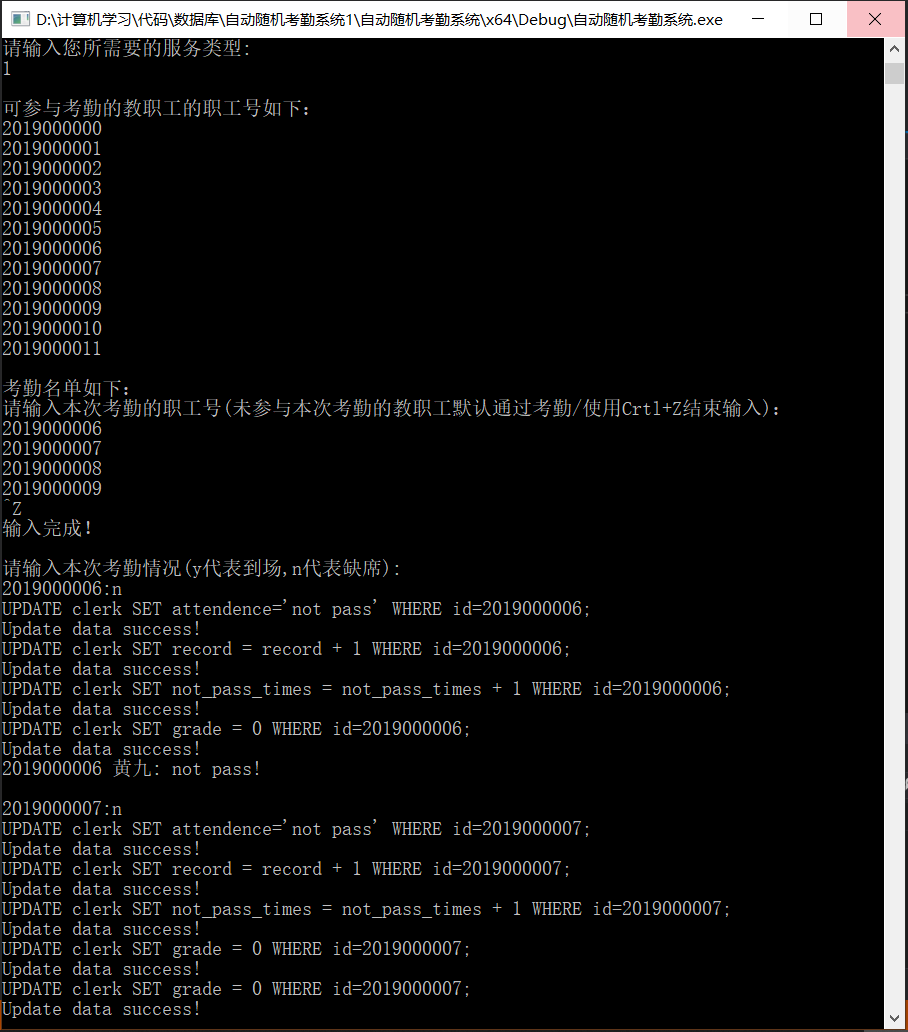
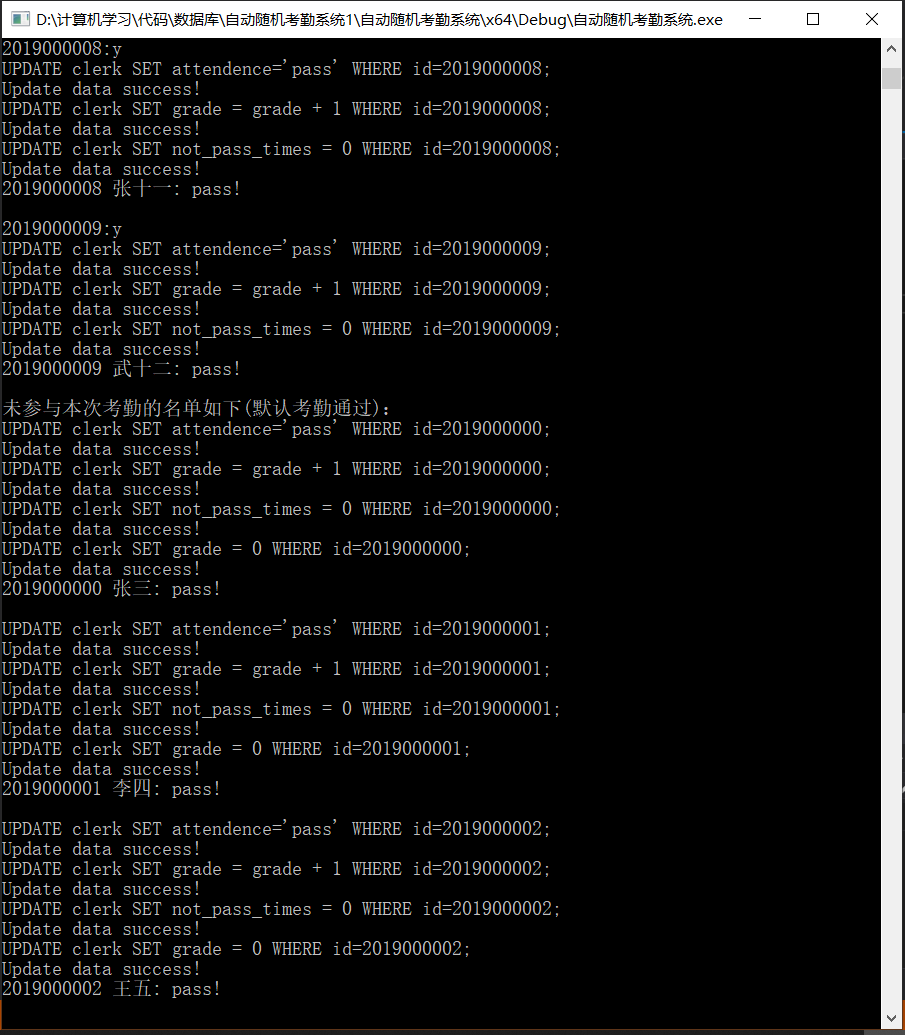
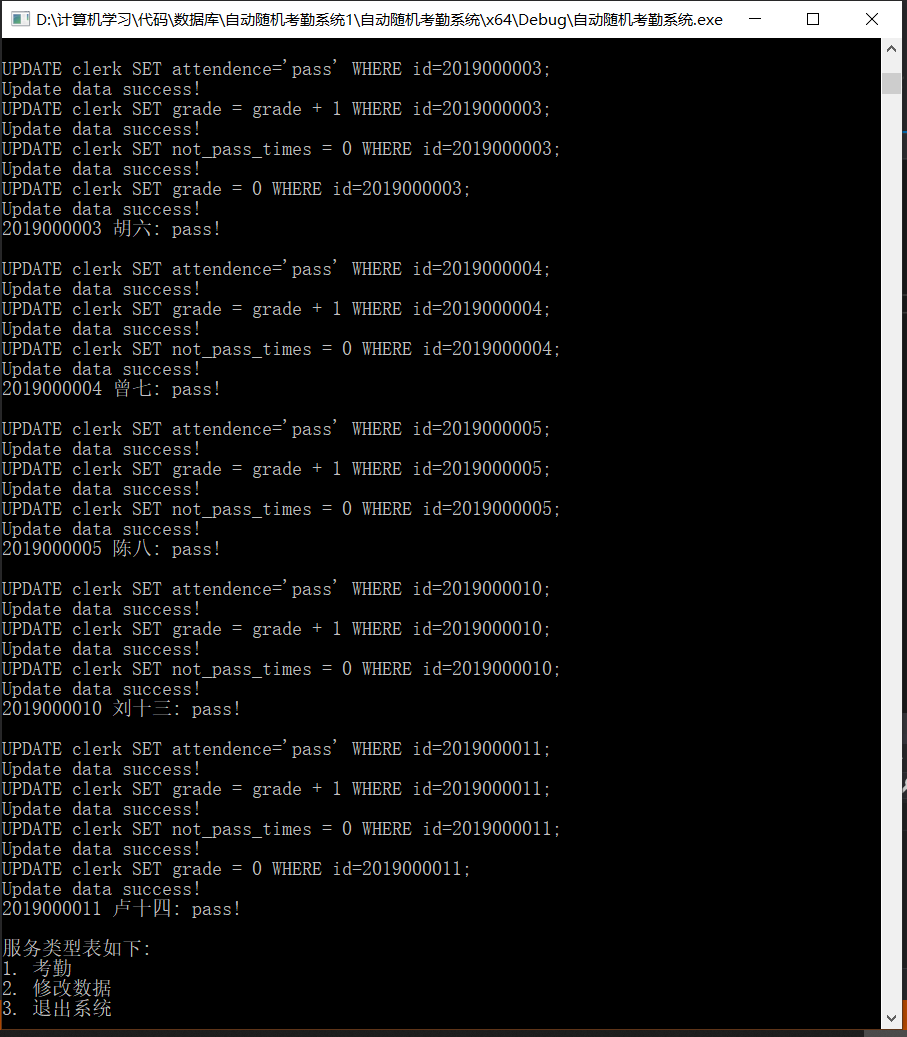
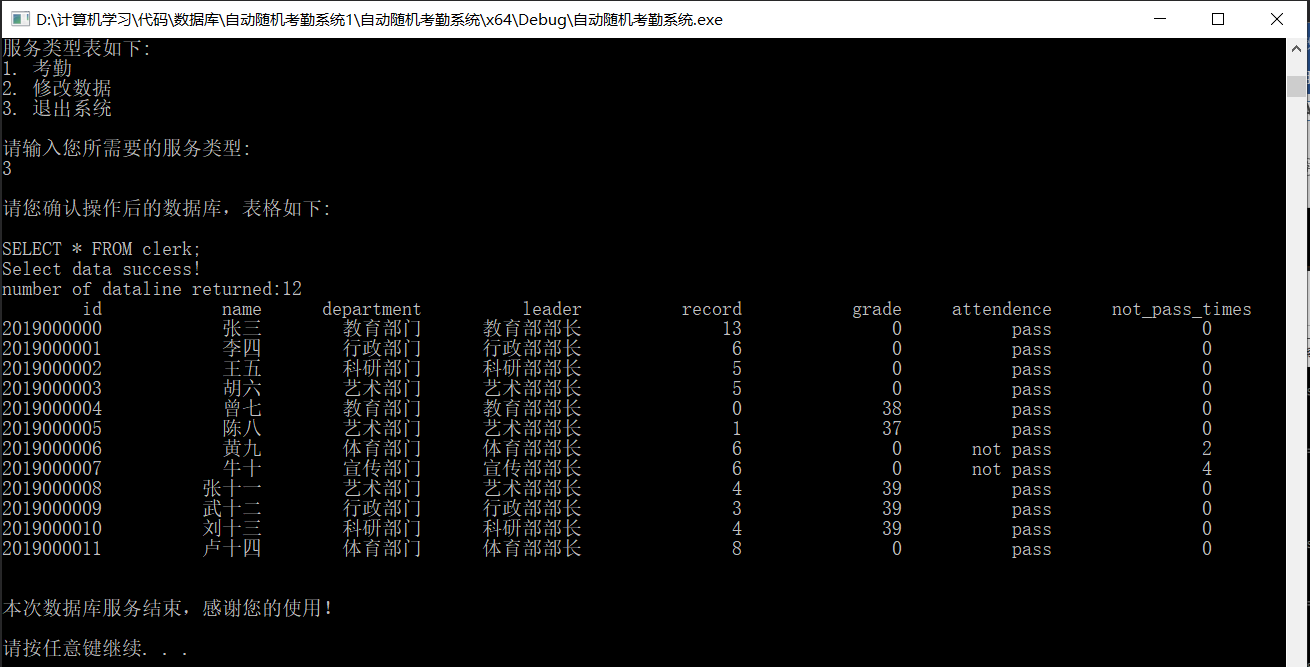
输入2019000006，回车；

输入2019000007，回车；

输入2019000008，回车；

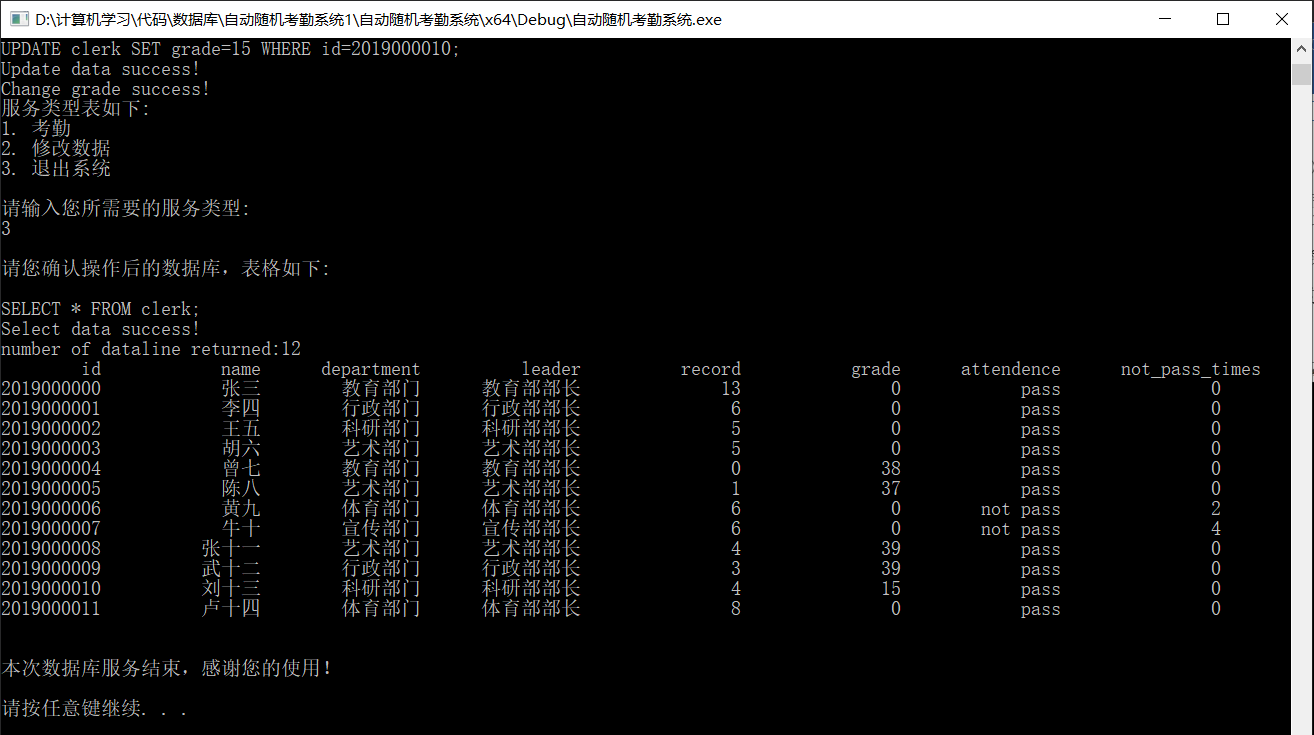
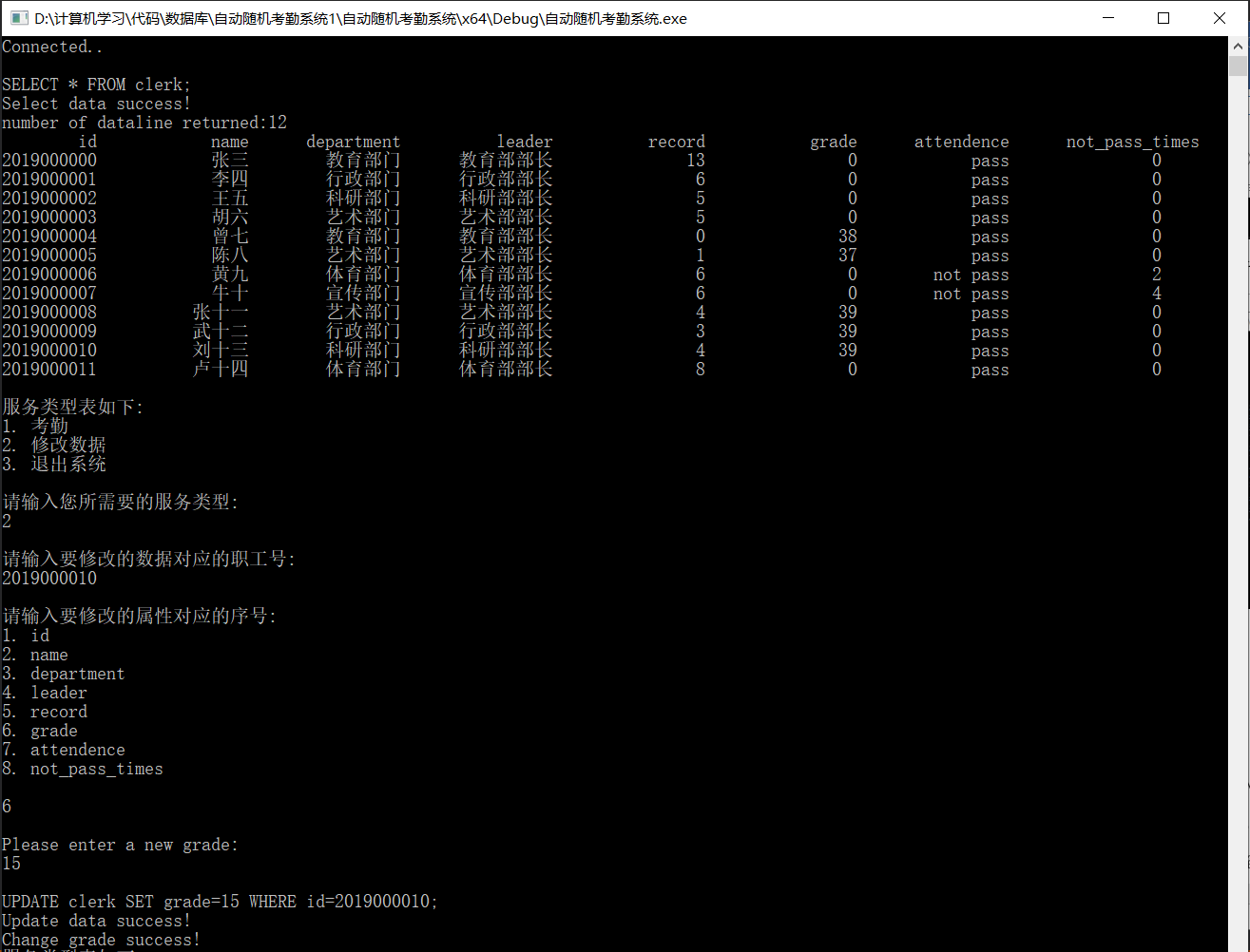
输入2019000009，回车；输入Ctrl+Z结束输入

分别输入：n回车（2019000006缺席），n回车（2019000007缺席），y回车（2019000008到场），y回车（2019000009到场），此时，系统按照输入为考勤名单上的教职工进行考勤，并且自动为不在考勤名单上的教职工默认设置为考勤通过，查看最终考勤结果，可以看到本数据库系统实现了实验要求的全部四个基本规则（I每一次考勤，不在名单中的职工默认考勤通过；II考勤通过者，相应的考勤成绩+1，且在相应的考勤记录中记录“pass”；III连续三次考勤不通过，考勤成绩置0 ；IV累计5次考勤不通过，考勤成绩永久置0）

经过手动计算，验证得数据库考勤系统计算数据正确（2019000006 黄九：考勤不通过，record+1（不通过次数>5）,grade置0，attendence置” not pass”，not\_pass\_times+1；其他考勤名单上和非考勤名单上的教务人员运算规则同理）

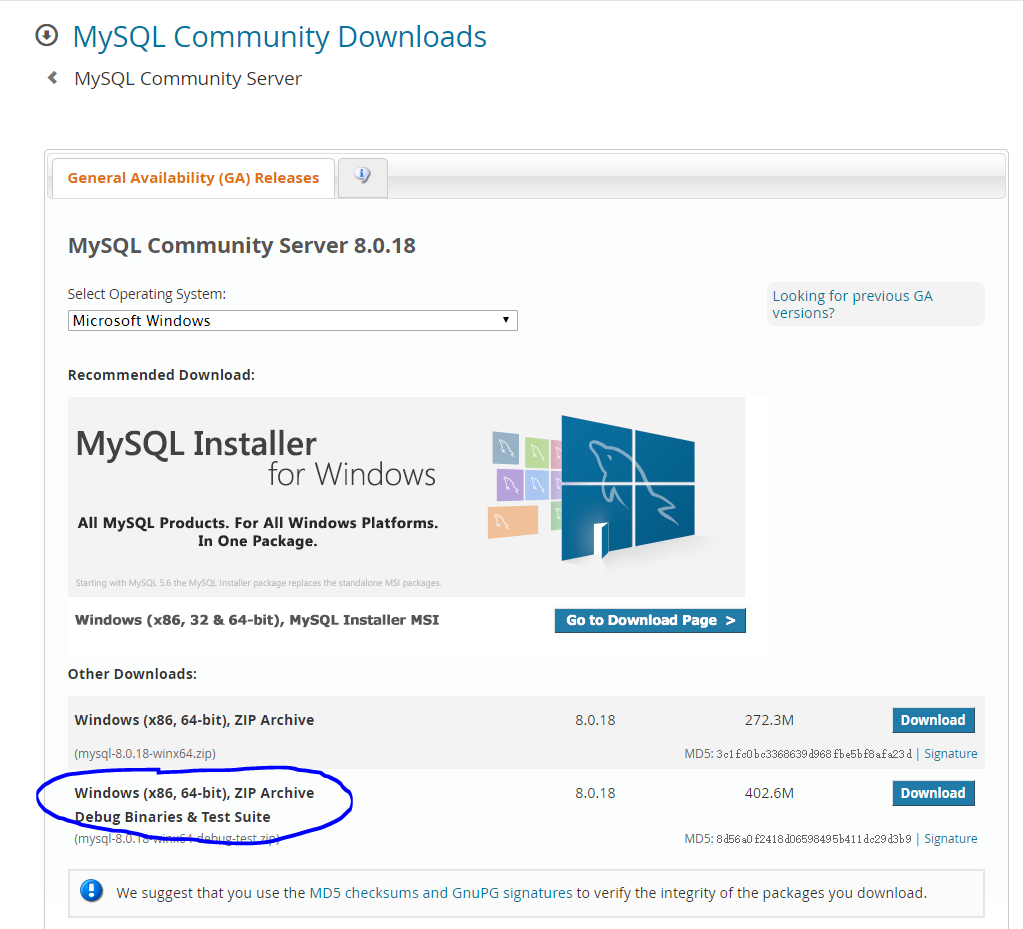
* 1. 输入2修改数据，输入修改的职工号2019000010，输入修改的属性号6（grade），输入新数据15，修改成功！（本次操作实现了本实验的附加功能，修改数据库的数据）



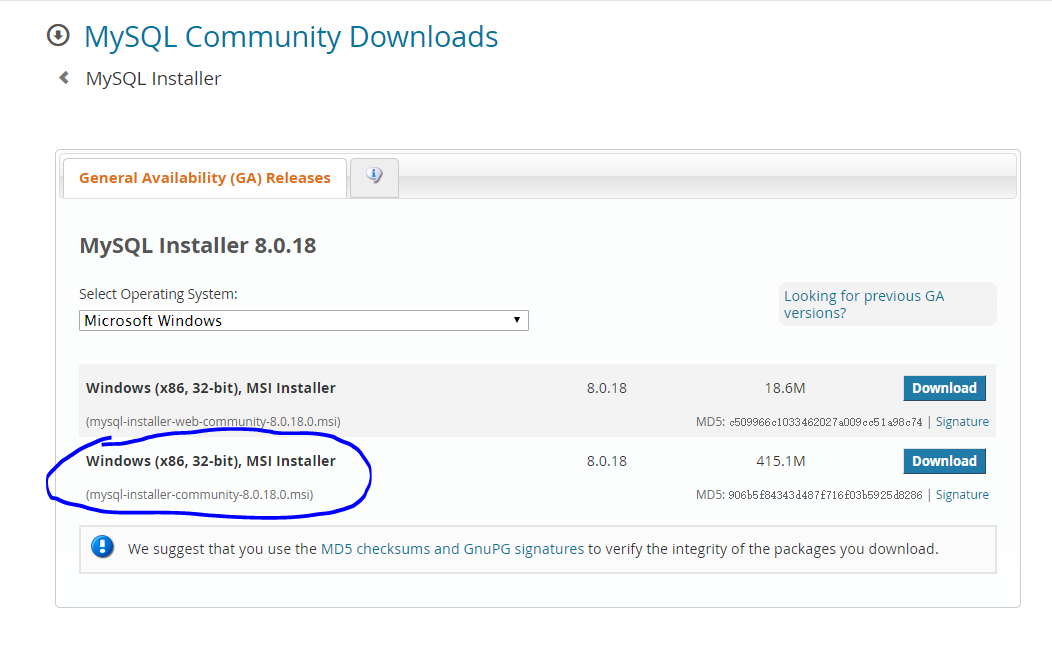
1. **实验总结**

本次实验作为这门课程的课程大项目，是验证本学期对本课程的综合考察，鉴于本实验的重要性，我十分认真地完成了本次任务，收获良多，总结如下几点：

1. 实验中遇到的困难：
   1. 鉴于前两次实验都是使用SQL server，本次实验本打算使用C++连接SQL server，但经多次查阅网站和书籍之后，发现连接方法过于复杂，也认识到自己的能力有限，于是打算使用MySQL数据库，并且另外安装了MySQL数据库（安装MySQL数据库的时候也遇到了令人烦恼的问题，第一次安装的时候下载的是zip文件，解压后要自己通过CMD配置相关参数，过程十分麻烦；于是后来直接下载MySQL的msi安装文件，通过图形化界面进行安装）



zip文件



msi安装文件（对应网站：https://dev.mysql.com/downloads/windows/installer/8.0.html）

* 1. 嵌入式SQL需要连接数据库MySQL，连接过程较为繁琐，花费了大量时间在这个过程，现在通过写下大致步骤理一下思路：
     1. 配置Visual Studio2017引入的包和参数（步骤如上面的（五）第1点）
     2. 在C++里面使用连接代码（上面有相关源代码，这里附上最关键了两行代码）

mysql\_init(&mysql);

mysql\_real\_connect(&mysql,"192.168.209.1(备注一下：这里填写服务器的ip)","root", "password", "attendence\_check",3306, NULL, 0)

* + 1. 使用C++编写数据库功能

1. 本综合实验的收获：
   1. 学会了MySQL的安装以及配置
   2. 熟悉了MySQL增删查改等基本操作的语句
   3. 学会了使用C++连接数据库MySQL
   4. 开发出一个小型数据库系统（自动随机考勤系统）
2. 总体来说，本次实验收获很大，既巩固了数据库这门课程的基本知识，也练习了C++的编程，学到了不少知识，也认识到自己的不足之处，借鉴了相关网站和书籍的知识，理解并学会欣赏技术大牛的思路，希望在以后的探索当中，继续保持这份求知热情，不断提升自身的能力。