西安电子科技大学
XIDIAN UNIVERSITY

面向对象程序设计

Object Oriented Programming

实验报告

Experimental Report

学号	19030500024	姓名	马子豪
班级	1903058	任课老师	褚华
实验名称	校园一卡通乘车模拟系统		
实验学期	2020-2021 学年第二学期		
实验日期	2021.5.29	实验地点	G 楼 314
报告成绩		评分教师	

西安电子科技大学计算机科学与技术学院

一、实验目的

采用面向对象程序设计范型、C++语言实现一个完整的简化系统——校园一卡通乘车模拟系统。通过编写该系统熟悉掌握面向对象的编程思想。

二、实验环境

- **硬件环境：**

  Lenovo Legion Y7000P

  处理器：Intel(R) Core(TM) i5-9300H CPU @2.40GHz

  Windows 10 64 位操作系统

- **集成开发环境：**

  Visual Studio 2019

- **数据存储：**

  MySQL Ver 8.0.24

- **代码版本管理：**

  Github

三、实验内容

1. **系统增加对三种一卡通类型的管理**，每种一卡通所有的属性不尽相同。在一卡通持有者刷卡乘车时，根据所持有的一卡通种类的不同，对每张一卡通的属性值（余额、乘车次数等）进行相对应的更新。

2. **系统增加对班车信息的管理**，每辆班车都有各自的基本信息。根据班车的基本信息编制运行时刻表。乘客刷卡上车时，根据班车的载乘人数进行相应的操作。

3. **系统增加对一卡通持卡人信息的管理。**增加对每个人乘车的记录，和对一卡通的业务

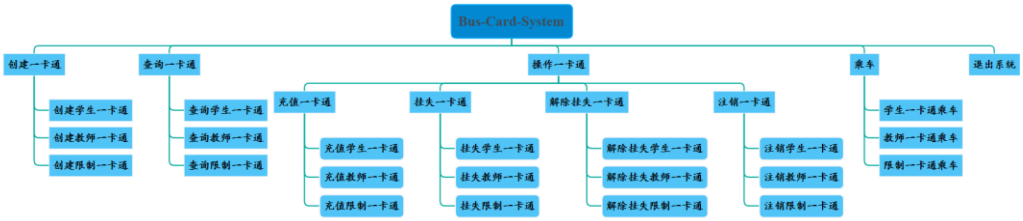
操作：申请一卡通、注销一卡通、挂失一卡通等。

4. **从相应文件中获取信息并完成对班车的仿真。**并设置中途停车点。

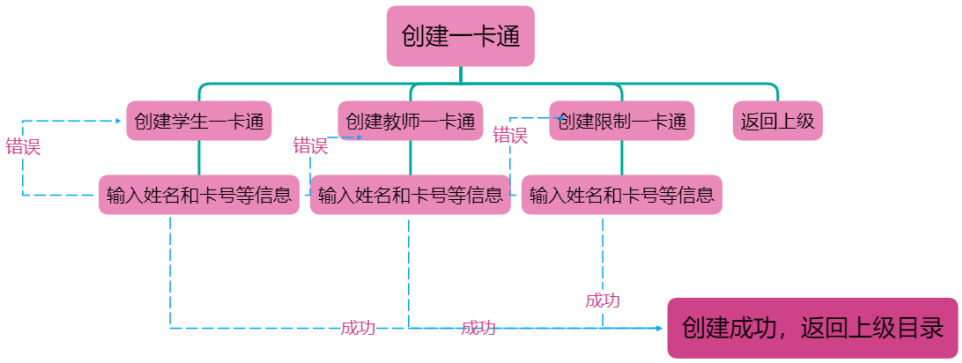
四、实验步骤

4.1 实验思路

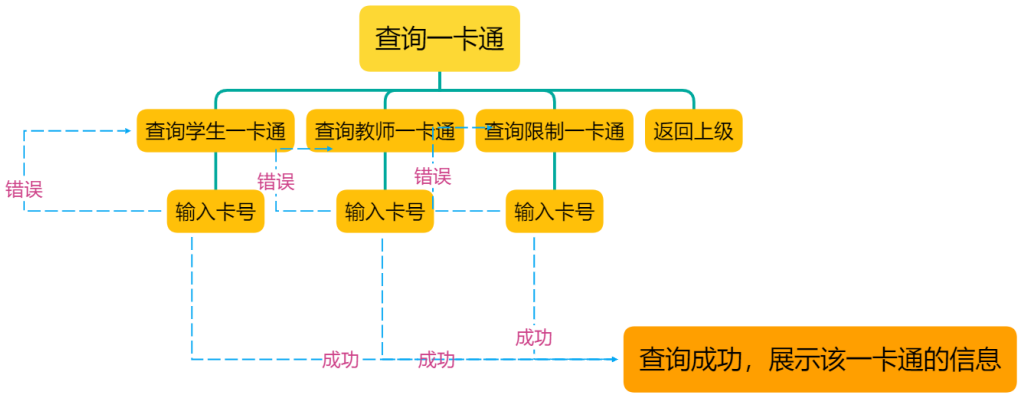
4.1.1 系统整体结构设计



4.1.2 创建一卡通功能逻辑设计

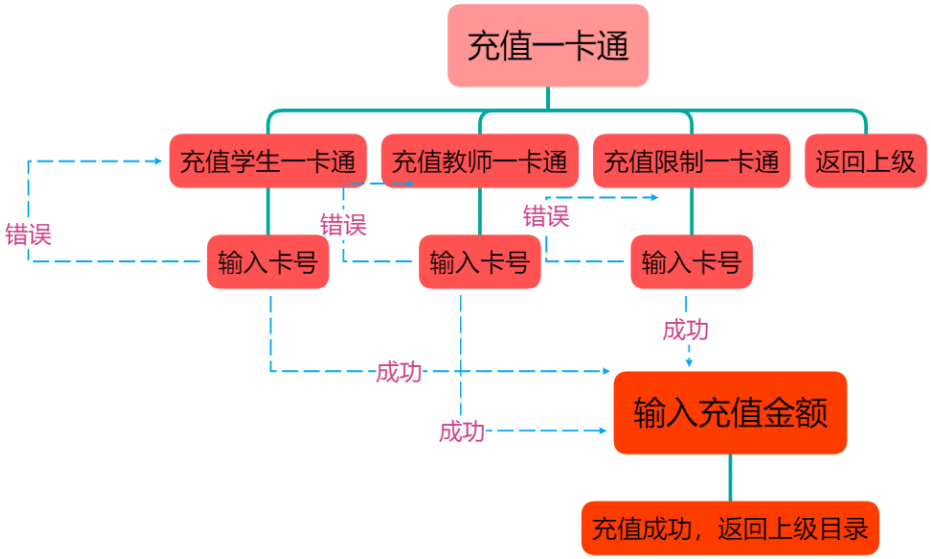


4.1.3 查询一卡通功能逻辑设计

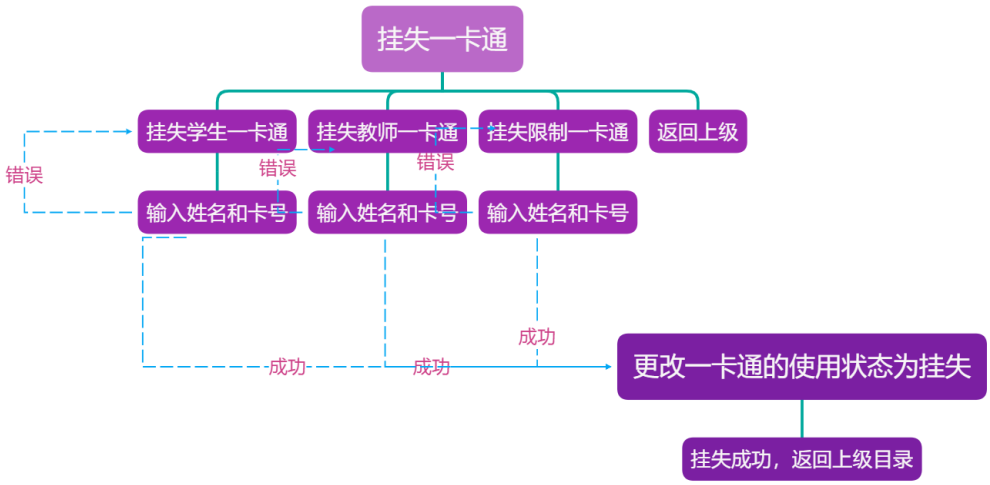


4.1.4 操作一卡通功能逻辑设计

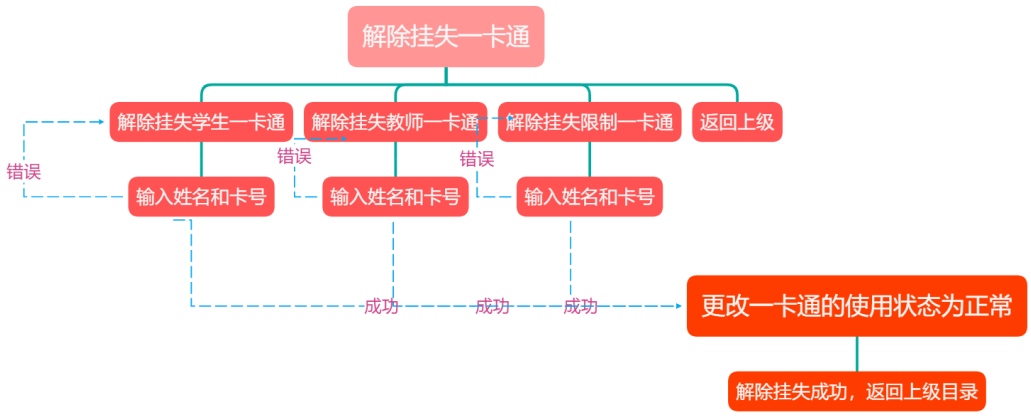
4.1.4.1 充值一卡通功能逻辑设计



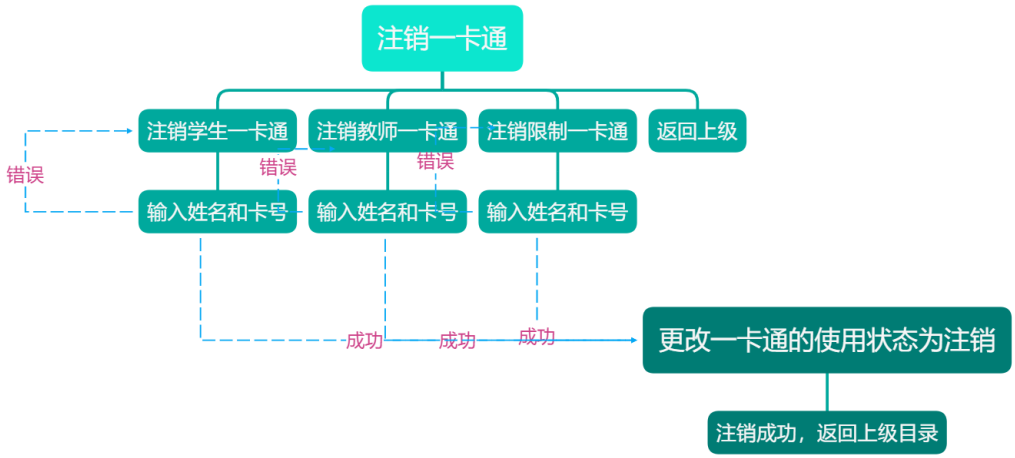
4.1.4.2 挂失一卡通功能逻辑设计



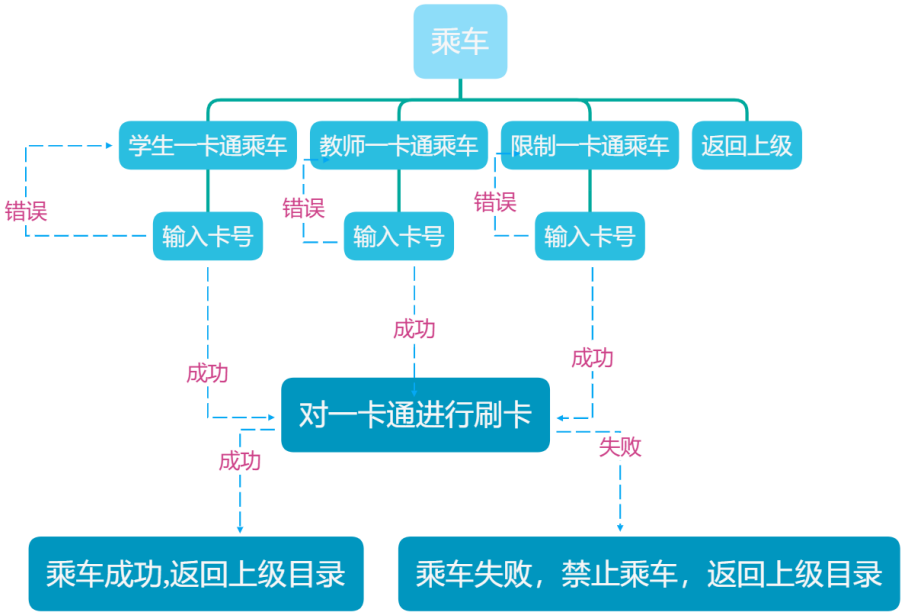
4.1.4.3 解除挂失一卡通功能逻辑设计



4.1.4.4 注销一卡通功能逻辑设计



4.1.5 乘车功能逻辑设计



4.2 关键数据结构的定义

该校园一卡通乘车模拟系统编写了五个类来实现。

4.2.1 class Student 学生一卡通

Student 类有 8 个数据成员和 8 个成员函数。

分类	名称	类型	含义
private **数据成员**	status	int	一卡通状态
	name	string	持卡人姓名
	number	string	卡号
	gender	string	持卡人性别
	times	int	乘车次数
	remain	int	一卡通余额
	subordinate	string	持卡人所属单位
	date	string	一卡通有效期

分类	名称	参数列表	返回值	功能
public **成员函数**	Student	(int,string,string,string, int,int,string,string)	—	构造函数
	~Student	(void)	—	析构函数
	show_info	(void)	void	展示一卡通信息
	top_up	(void)	void	充值一卡通
	get_on	(char[ ])	void	使用一卡通乘车
	get_off	(char[ ])	void	下车
	log_out	(void)	void	一卡通注销
	report_loss	(void)	void	一卡通挂失
	unlock_report_loss	(void)	void	一卡通解除挂失

4.2.2 class Teacher 教师一卡通

Teacher 类有 7 个数据成员和 9 个成员函数。

分类	名称	类型	含义
private **数据成员**	status	int	一卡通状态
	name	string	持卡人姓名
	number	string	卡号
	gender	string	持卡人性别
	times	int	乘车次数
	subordinate	string	持卡人所属单位
	date	string	一卡通有效期

分类	名称	参数列表	返回值	功能
public **成员函数**	Teacher	(int,string,string,string, int,string,string)	—	构造函数
	~Teacher	(void)	—	析构函数
	show_info	(void)	void	展示一卡通信息
	top_up	(void)	void	一卡通充值
	get_on	(char[ ])	void	使用一卡通乘车
	get_off	(char[ ])	void	下车
	log_out	(void)	void	一卡通注销
	report_loss	(void)	void	一卡通挂失
	unlock_report_loss	(void)	void	一卡通解除挂失

4.2.3 class Limit 限制一卡通

Limit 类有 9 个数据成员和 9 个成员函数。

分类	名称	类型	含义
private **数据成员**	status	int	一卡通状态
	name	string	持卡人姓名
	number	string	卡号
	gender	string	持卡人性别
	times	int	乘车次数
	remain	int	余额
	free_times	int	剩余免费次数
	subordinate	string	持卡人所属单位
	date	string	一卡通有效期

分类	名称	参数列表	返回值	功能
public **成员函数**	Limit	(int,string,string,string, int,int,int,string,string)	—	构造函数
	~Limit	(void)	—	析构函数
	show_info	(void)	void	展示一卡通信息
	top_up	(void)	void	充值一卡通
	get_on	(char[ ])	void	使用一卡通乘车
	get_off	(char[ ])	void	下车
	log_out	(void)	void	一卡通注销

	report_loss	(void)	void	一卡通挂失
	unlock_report_loss	(void)	void	一卡通解除挂失

4.2.4 class Bus 班车

Bus 类有 6 个成员函数和 9 个成员函数。

分类	名称	类型	含义
private 数据成员	number	string	车牌号
	type	string	车类型
	driver	string	司机名字
	max_num	int	最大乘客数
	cur_num	int	现有乘客数
	start_time	int	发车时间

分类	名称	参数列表	返回值	功能
public 成员函数	Bus	(string,string,string,int,int,int)	—	构造函数
	~Bus	(void)	—	析构函数
	show_bus	(void)	void	展示该车的信息
	get_cur_num	(void)	int	获得当前该车上乘客数
	check_bus	(void)	void	乘客乘车
	add_people	(string,string)	void	增加乘客

	delete_people	(string,string)	void	删除乘客
	bus_detail	(void)	void	展示该车上当前乘客具体信息
	clear_bus	(void)	void	清空该车

4.2.5 class System 系统

System 类较为特殊，无数据成员，只有成员函数。由于该系统中并未对 System 类进行实例化，因此仅有静态成员函数。

分类	名称	参数列表	返回值	功能
	start	(void)	static void	进入系统主页
	moni	(void)	static void	模拟班车运行
	get_student	(char[])	static Student	获得一个 Student 对象
	get_teacher	(char[])	static Teacher	获得一个 Teacher 对象
	get_limit	(char[])	static Limit	获得一个 Limit 对象
	create_card	(void)	static void	进入创建一卡通页面
	create_student	(void)	static void	创建学生一卡通
	create_teacher	(void)	static void	创建教师一卡通
	create_limit	(void)	static void	创建限制一卡通
	show_card	(void)	static void	进入展示一卡通页面
	show_student_card	(void)	static void	展示学生一卡通

public **成员函数**	show_teacher_card	(void)	static void	展示教师一卡通
	show_limit_card	(void)	static void	展示限制一卡通
	top_up	(void)	static void	充值一卡通
	top_up_student	(void)	static void	充值学生一卡通
	top_up_teacher	(void)	static void	充值教师一卡通
	top_up_limit	(void)	static void	充值限制一卡通
	get_on	(void)	static void	进入乘车页面
	get_on_student	(char[],char[])	static void	学生一卡通乘车
	get_on_teacher	(char[],char[])	static void	教师一卡通乘车
	get_on_limit	(char[],char[])	static void	限制一卡通乘车
	operate_card	(void)	static void	进入操作一卡通页面
	log_out	(void)	static void	进入注销一卡通页面
	log_out_student	(void)	static void	学生一卡通注销
	log_out_teacher	(void)	static void	教师一卡通注销
	log_out_limit	(void)	static void	限制一卡通注销
	report_loss	(void)	static void	进入挂失一卡通页面
	report_loss_student	(void)	static void	学生一卡通挂失
	report_loss_teacher	(void)	static void	教师一卡通挂失
	report_loss_limit	(void)	static void	限制一卡通挂失
	unlock_report_loss	(void)	static void	进入解除挂失一卡通页
	unlock_student	(void)	static void	学生一卡通解除挂失
	unlock_teacher	(void)	static void	教师一卡通解除挂失

	unlock_limit	(void)	static void	限制一卡通解除挂失
	get_bus	(char[])	static Bus	获得一个 Bus 对象
	add_bus	(void)	static void	添加一个班车

4.3 关键算法流程

4.3.1 class Student

类 Student 中，涉及的关键算法是对实例的析构函数。

- **~Student 算法：**

```
Student::~Student() {
    MYSQL mysql;
    mysql_init(&mysql);              //初始化数据库
    if 初始化失败:
        输出错误信息;
    else {
        mysql_real_connect(&mysql); //连接本地数据库
        if 连接本地数据库失败:
            输出错误信息;
        else{
            char query_delete[255]; //初始化MYSQL删除语句以删除当前对象原来的数据库中的信息
            mysql_query(&mysql,query_delete);
            char query_insert[255]; //初始化MYSQL插入语句以插入数据库当前Student对象的更新信息
        }
    }
    mysql_close(&mysql);             //关闭数据库
}
```

- **get_on 算法：**

```
void Student::get_on(char number_bus[]) {
    bus = get_bus(number_bus);  //根据输入的班车牌号，获得相应车牌号的Bus对象
    if (bus->最大人数 > 现有人数) {
        if (this->持卡状态 == 正常) {
            if (this->余额 >= 2) {
                this->余额 -= 2;
                if (this->余额 <= 7) {
                    print: 【注意】余额很低，请及时充值
                }
                this->乘车次数 += 1;
                b.add_people("stu_card", this->卡号); //将该乘客的具体信息加入到该班车中
            }
            else if (this->余额 < 2) {
                print: 【注意】余额不足，不能乘车
            }
        }
        else if (this->持卡状态 == 挂失) {
            print: 【注意】该一卡通挂失中，禁止使用
        }
        else if (this->持卡状态 == 注销) {
            print: 该一卡通已经注销
        }
    }

}
```

- **top_up 算法**

```
void Student::top_up(void) {
    scan: 充值的金额;
    this->余额 += 充值的余额;
    this->展示该一卡通的信息;
}
```

4.3.2 class Teacher

- **~Teacher 算法**

  和 Student 类的析构函数算法类似。

- **get_on 算法**

  和 Student 类的 get_on 函数算法相似

4.3.3 class Limit

- **~Limit:**

  和 Student 类的析构函数相似

- **get_on**

  和 Student 类的 get_on 函数相似

4.3.4 class Bus

- **get_cur_num**

```
int Bus::get_cur_num(void) {
    MYSQL mysql;
    mysql_init(&mysql);     //初始化数据库
    if 数据库初始化失败:
        print:错误信息;
    else{
        mysql_real_connect(&mysql); //连接本地数据库
        if 连接本地数据库失败:
            print:错误信息;
        else{
            char query[255]; //初始化MYSLQL查询语句
            mysql_query(&mysql,query);//查询该班车数据库表中所有的乘客
            if 查询失败:
                print:错误信息
            else{
                    MYSQL_RES* result=查询结果;      //定义数据库查询结果变量
                    return 查询结果的记录数;         //返回乘客数量
                    mysql_free_result(result);       //释放结果集
            }
            return false;
        }
        mysql_close(&mysql);                         //关闭数据库
        return false;
    }

}
```

- **add_people**

```
void Bus::add_people(string cardtype, string cardnumber) {
    MYSQL mysql;
    mysql_init(&mysql);      //初始化数据库
    if 数据库初始化失败:
        print:错误信息;
    else {
        mysql_real_connect(&mysql); //连接本地数据库
        if 连接本地数据库失败:
            print:错误信息;
        else{
            char query[255] = 将cardtype和cardnumber插入以this->车牌号为名字的数据库表中;
            mysql_query(&mysql,query);  //执行MYSQL语句
            if 插入失败:
                print:错误信息
            else{
                print:成功乘车
            }
        }

    }
    mysql_close(&mysql);     //关闭数据库

}
```

- **bus_detail**

```
void Bus::bus_detail(void) {
    MYSQL mysql;
    mysql_init(&mysql);     //初始化数据库
    if 数据库初始化失败:
        print:错误信息;
    else{
        mysql_real_connect(&mysql); //连接本地数据库
        if 连接本地数据库失败:
            print:错误信息;
        else {
            char query[255] = 查询以this->车牌号为名字的数据库表中所有的成员记录;
            mysql_query(&mysql,query);  //对本地数据库执行MYSQL语句
            if 查询失败:
                print:错误信息;
            else{
                MYSQL_RES* result = 查询结果集;
                while(res_row = mysql_fetch_row(result)){
                    print:每位乘客的信息;
                }
                mysql_free_result(result);
            }
        }
    }
    mysql_close(&mysql);
}
```

4.3.5 class System

- **get_setudent**

```
Student System::get_student(char number_student[]) {
    //1. 在相应数据库表中查找该一卡通号，若没有则返回
    //2. 若找到了该一卡通，先用数据库中的数据初始化出一个Student实例，并对该实例进行：钱-2；次数+1等操作
    MYSQL mysql;
    mysql_init(&mysql);
    if 数据库初始化失败:
        print:错误信息;
    else{
        mysql_real_connect(&mysql);
        if 连接本地数据库失败:
            print:错误信息;
        else{
            char query_select[255] = 查询学生一卡通数据库表中段名number为number_student的记录;
            mysql_query(&mysql,query_select);
            if 查询失败:
                print: 错误信息;
            else {
                MYSQL_RES* result;
                result = 查询结果集;
                if 结果集行数 == 0:
                    print:查无此卡，请重新输入或者注册;
                else{
                    Student s(查询得到的记录);      //根据数据库中的信息初始化出一个Student实例
                    mysql_free_result(result);
                    mysql_close(&mysql);
                    return s;
                }
            }

        }

    }
    return NULL;
}
```

- **get_teacher**

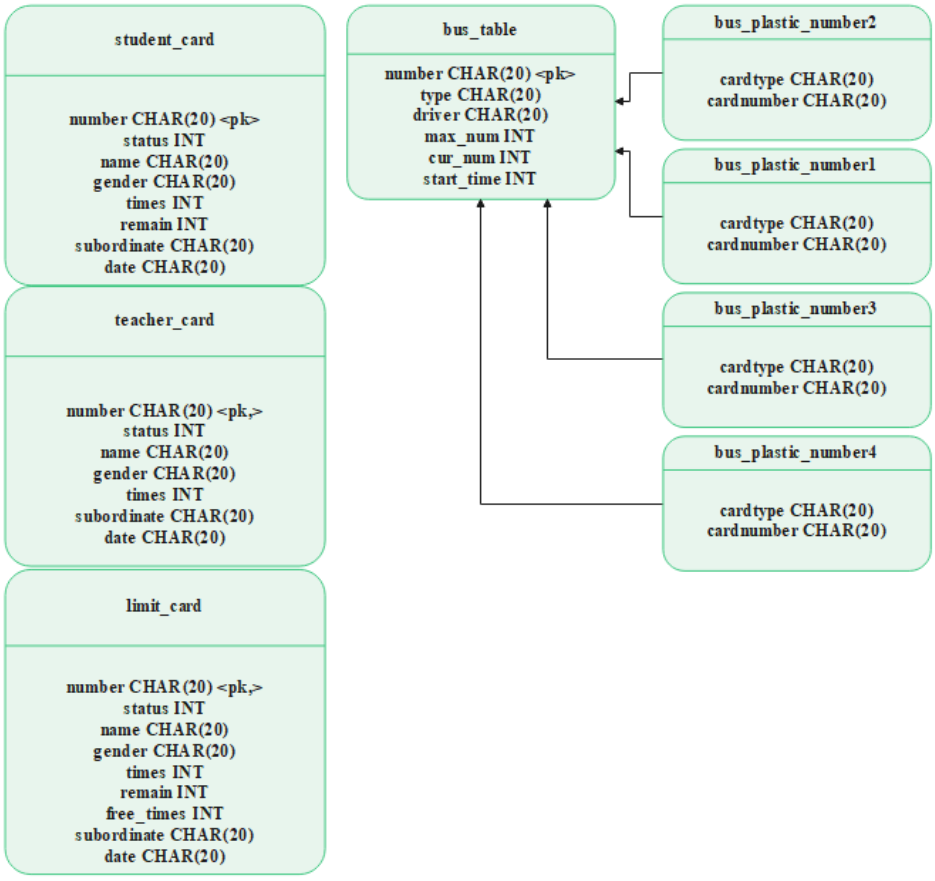
  和 get_student 类似

- **get_limit**

  和 get_student 类似

4.4 数据存储

使用 Visual Studio 2019 连接本地 MySQL 数据库，来存储学生一卡通、教师一卡通、限制一卡通、班车信息以及每辆班车当前所载的乘客信息。



4.5 遇到的问题

- 系统数据的存储问题

第一次实验时，使用 txt 文本文件进行存储一卡通信息以及班车信息，使用 C++的头文件 fstream 可以对文件进行读写。

但是 C++对于文本文件的操作十分繁琐。不能直接在 txt 文件中查找某个字段中的指定字符串，只能先将 txt 文件中每一行读入 C++的缓冲区中，然后在进行对比某个字段的值。除此之外，对于 txt 文件某条记录的更新，需要重写该 txt 文件，即先将原先内容保存，再修改内容，最后写入 txt 文件。

- 解决方法：

因此，我最终选择了使用 Visual Studio2019 集成开发环境连接本地 MySQL 数据库，

来进行数据的存储。只需要更改 Visual Studio 的项目设置中属性的 VC++ **包含目录**和**库目录**为 MySQL 的包含目录和库目录所在路径即可。

然后，再从 MySQL 官方网站学习 MySQL 给予 C++的 API 接口即可。

- C++ 中使用 MySQL API 接口时遇到的问题

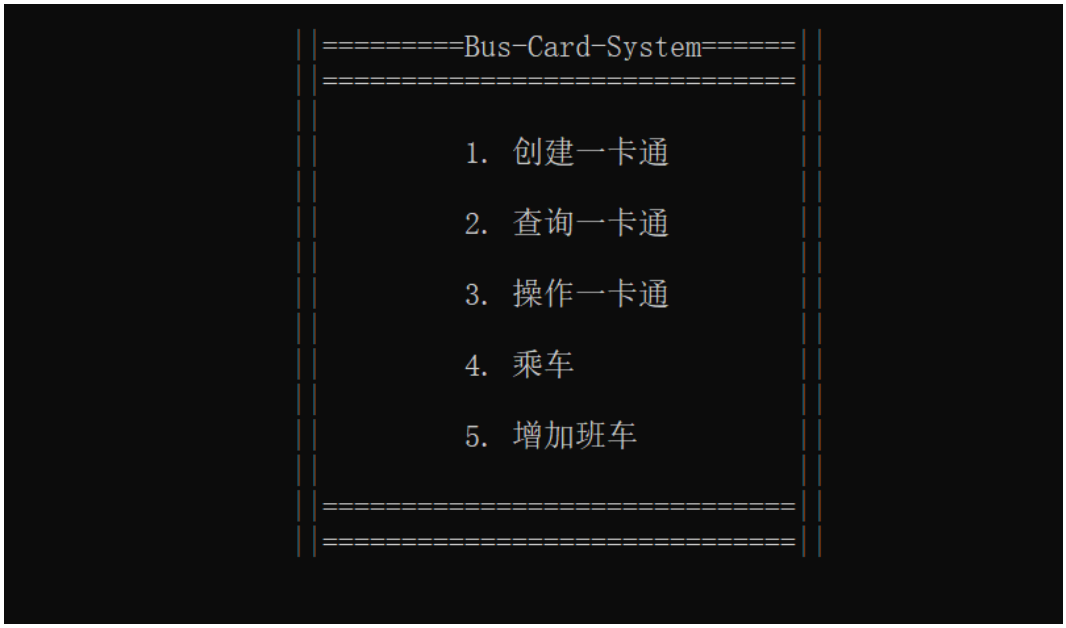
1. 在使用 mysql_query()查询数据库中的信息时，所显示出的中文乱码。
2. 使用 sprintf_s( ) 将 C++中的变量传入 MySQL 语句时报错
3. 访问数据库冲突

- 解决方法

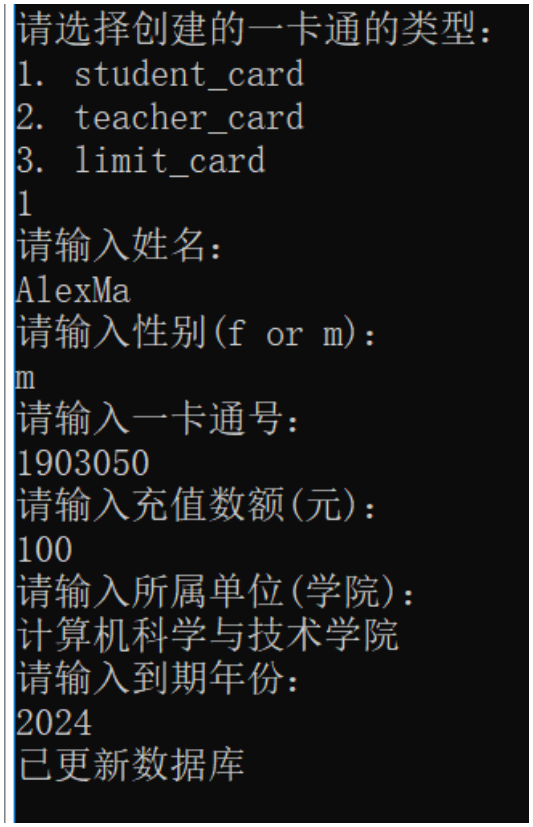
1. 使用 mysql_options() 语句将数据库编码方式设置为 gbk
2. 传入变量为字符串时，需要传入 char[]字符串，须将 string 转换成 char[]。此外，在传入数据库表名时，不需要加单引号
3. 在使用完 MySQL 本地数据库后，需要将数据库使用 API 接口函数 mysql_close() 关闭,和 mysql_real_connect() 前后呼应,不然会引发访问冲突。

五、实验结果

- **主页面**



- **创建一卡通**



- **创建后，本地数据库表 stu_card 中:**

```
mysql> select * from stu_card;
+--------+---------+-------------+--------+-------+--------+----------------------+------+
| status | name    | number      | gender | times | remain | subordinate          | date |
+--------+---------+-------------+--------+-------+--------+----------------------+------+
|      1 | 张宁    | 19030500025 | m      |    43 |      0 | 计科院               | 2022 |
|      1 | 郑亦韬  | 19030500012 | m      |    46 |      1 | 计算机学院           | 2023 |
|      1 | 杨翔    | 19030500027 | m      |     7 |     86 | 计科院               | 2022 |
|      1 | 唐子辰  | 19030500017 | m      |     9 |     82 | 计科院               | 2023 |
|      1 | 正好天  | 20030500024 | m      |     1 |     98 | 计科院               | 2023 |
|      1 | 小白    | 19030500026 | m      |    42 |      0 | 计科院               | 2022 |
|      1 | mazihao | 19030500024 | f      |    41 |     12 | 计算机科学与技术学院 | 2022 |
|      1 | AlexMa  | 1903050     | m      |     0 |    100 | 计算机科学与技术学院 | 2024 |
+--------+---------+-------------+--------+-------+--------+----------------------+------+
8 rows in set (0.00 sec)
```

- **查询一卡通**

```
||=========Bus-Card-System======||
||==============================||
||                              ||
||          1. 创建一卡通       ||
||                              ||
||          2. 查询一卡通       ||
||                              ||
||          3. 操作一卡通       ||
||                              ||
||          4. 乘车             ||
||                              ||
||          5. 增加班车         ||
||                              ||
||==============================||
||==============================||
2
请输入要查询的一卡通种类
1. student card
2. teacher card
3. limit card
1
请输入一卡通卡号
1903050
已更新数据库
|=====================================|
|一卡通状态:    1                     |
| 姓名:         AlexMa                |
| 一卡通号:     1903050               |
| 余额:         100                   |
| 乘车次数:     0                     |
|=====================================|
已更新数据库
```

- **使用一卡通乘车**

```
请选择一卡通类型
1. student_card
2. teacher_card
3. limit_card
1
请输入一卡通的卡号
1903050
已更新数据库
...车辆未满,准许上车...
刷卡中...
扣费成功!
1903050成功乘上shanA
已更新数据库
```

- **乘车后 shanA 的数据库表**

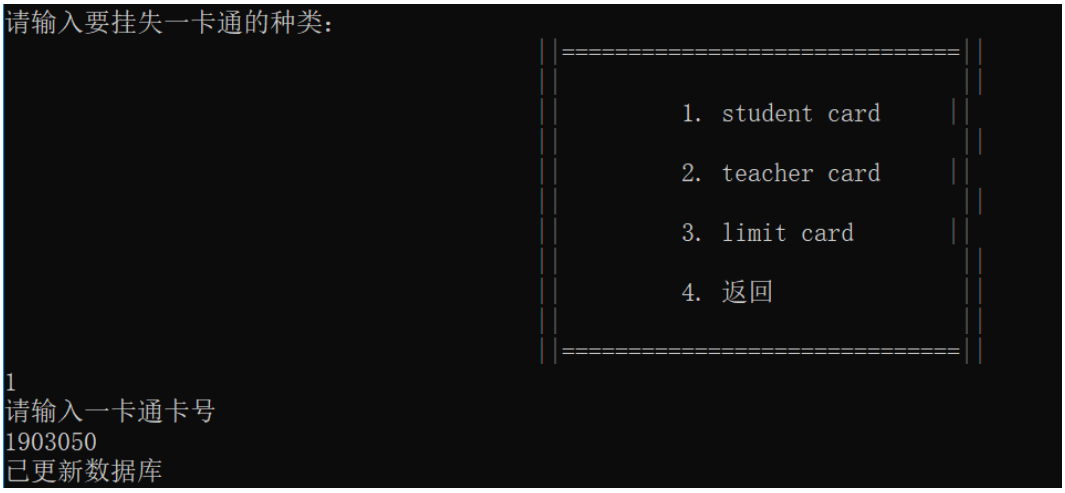
```
mysql> select * from shana;
+----------+------------+
| cardtype | cardnumber |
+----------+------------+
| stu_card | 1903050    |
+----------+------------+
1 row in set (0.00 sec)
```

- **操作一卡通页面**

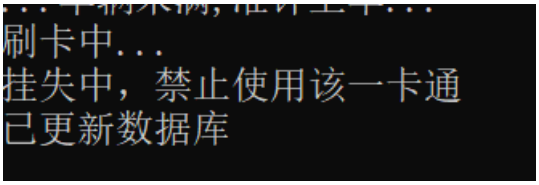
```
||==============================||
||                              ||
||         1. 充值一卡通        ||
||                              ||
||         2. 挂失一卡通        ||
||                              ||
||         3. 注销一卡通        ||
||                              ||
||         4. 解除挂失一卡通    ||
||                              ||
||         5. 返回              ||
||                              ||
||==============================||
```

- **充值一卡通**

- **挂失一卡通**



- **挂失一卡通后乘车**



六、实验总结

通过本次使用 C++对西电乘车一卡通系统的模拟，对 C++的面向对象程序设计范型有了更深入的理解。面向对象你的程序设计范型和封装机制将定义和实现分离，将复杂的内部细节隐藏，提高了程序的可复用性和可维护性。这次实验中，主函数 main() 之外，基本只定义了类、类的非静态成员函数、类的静态成员函数。

类中最特别的**成员函数**应该就是**构造函数、析构函数和虚拟函数**了，本实验中最令我印象深刻的是，对三个一卡通类的析构函数的定义。由于每次对一卡通对象的信息的编辑，**一定**需要及时更改到相应的数据库表中，因此我想到可以直接在析构函数中将该一卡通对象生命周期的最后的状态更新到相对应的数据库表中。这样不需要繁琐地在每次对一卡通的操作后更新数据库。

除此之外，之前只是将 Hexo 博客部署在 Github 上，在本次实验时我首次使用 Github 用于项目的版本管理。一个人的使用优点并不突出，但是当一个团队来维护一个项目时，我觉得会极大的提高开发项目的效率和正确性。