**门禁系统——基于面向对象的设计方法**

实验人: 邱抱朴 161100036

实验时间：11.15

一、实验要求

进入系楼门禁开启由刷卡和人脸识别两种方式，刷卡器（人脸识别器）预存了有效卡（人脸）的数据信息，当某人试图进入系楼，需刷卡或人脸识别进入。出系楼无需身份验证。进出系楼人员包含学生、老师两类。学生只可通过刷卡进入，老师可选择任意一种方式（优先人脸识别，否则刷卡）。每位人员会指定一个进出系楼的时间，进出人员需按照时间先后出入，同样时间先出后入，相同时间出（或入）则根据排入队列顺序出（或入）。

二、整体设计思路以及类的概述

显然，这道题要用到类继承的方法，这促使我去思考类继承的本质。

首先，它可能是一种“具体化”，比如把“形状”作为基类，把具体的形状如“三角形”“圆形”作为它的继承类。在本题中，比较显见的是我们要为进出楼的人员建立一些类，题目中提到“进出西楼的人员包含学生、老师两类”，那么我们就可以设计Student, Teacher 这两个类，它们都继承自基类person。

其次，类继承还可能是一种功能的增加。基类是一个有一些功能的类，同时为了满足具体情境下的一些需求，我们用类继承的方法设计一些新的类，增加额外的功能。本题中比较基础的功能是实现卡号、人脸ID的录入、增、删、清空、打印，即所谓的“系统管理功能”。我们用一个基类IDSystem实现它。同时在本题的情境中，我们要实现进出人员的模拟，那么就建立一个类Manage继承上述类的功能并增加其它的处理函数。

三、类的具体实现与算法描述

1.person student teacher三个类在person.h中实现

代码：

class person

{

public:

string name;

int cardid;

char entertime[10];

string action;

};

class student :public person

{

public:

string status;

int priority = 0;

student() {}

student(string name, int cardid, char s[], string action)

{

int i, k;

k = 0;

this->name = name;

this->cardid = cardid;

strcpy(entertime, s);

this->action = action;

status = "学生";

for (i = 0; entertime[i] != ':'; i++)

priority = priority \* 10 + (entertime[i] - '0');

priority \*= 60;

for (i = i + 1; entertime[i] != '\0'; i++)

k = k \* 10 + (entertime[i] - '0');

priority += k;

priority \*= 10;

if (action == "出")priority -= 5;

cout << priority << endl;

}

};

class teacher :public person

{

public:

int faceid;

string status;

int priority = 0;

teacher() {}

teacher(string name, int faceid, int cardid, char s[], string action)

{

int i, k;

k = 0;

this->name = name;

this->cardid = cardid;

strcpy(entertime, s);

this->action = action;

this->faceid = faceid;

status = "老师";

for (i = 0; entertime[i] != ':'; i++)

priority = priority \* 10 + (entertime[i] - '0');

priority \*= 60;

for (i = i + 1; entertime[i] != '\0'; i++)

k = k \* 10 + (entertime[i] - '0');

priority += k;

priority \*= 10;

if (action == "出")priority -= 5;

priority -= 3;

cout << priority << endl;

}

};

说明：基类person 包含了几个基本的信息: name, cardid, entertime, action 分别表示姓名、卡号、进入时间（用字符串表示），动作

在继承出来的student类的构造函数中，我们通过用字符串表示的时间来计算一个优先值，这个优先值跟进入时间、动作（先出后进原则），身份（学生会让老师先出去）有关。

teacher类中除了计算优先值还添加了一个faceID成员。

2.IDSystem实现在IDSystem.h中

代码：

class IDSystem {

protected:

int cardID[50];

int n1 = 0;

int faceID[50];

int n2 = 0;

public:

IDSystem()

{

n1 = 0;

n2 = 0;

}

~IDSystem() {}

void inputid()

{

int x, y;

cout << "录入有效刷卡ID：" << endl;

while (cin >> x)

cardID[n1++] = x;

cout << "完成" << endl;

cin.clear();

cout << "录入有效人脸ID：" << endl;

while (cin >> y)

faceID[n2++] = y;

cin.clear();

}

void addid()

{

int x, flag, i;

cout << "选择要增加的是人脸还是卡号，卡号输入0，人脸输入1" << endl;

cin >> flag;

cout << "输入要增加的ID:" << endl;

while (cin >> x) {

if (flag == 0) {

for (i = 0; i < n1; i++)

if (cardID[i] == x) {

cout << "输入的ID " << x << " 已经在有效ID中！" << endl; break;

}

if (n1 == i) cardID[n1++] = x;

}

else {

for (i = 0; i < n2; i++)

if (faceID[i] == x) {

cout << "输入的ID " << x << " 已经在有效ID中！" << endl; break;

}

if (n2 == i) faceID[n2++] = x;

}

}

cin.clear();

}

void eraseid()

{

int x, flag, i, j, find;

cout << "选择要删除的是人脸还是卡号，卡号输入0，人脸输入1" << endl;

cin >> flag;

cout << "输入要删除的ID:" << endl;

while (cin >> x) {

find = 0;

if (flag == 0) {

for (i = 0; i < n1; i++)

if (cardID[i] == x) {

for (j = i; j < n1; j++)

cardID[j] = cardID[j + 1];

n1--;

find = 1;

break;

}

if (find == 0) cout << "抱歉，你要删除的ID " << x << " 不在系统中！" << endl;

}

else {

for (i = 0; i < n2; i++)

if (faceID[i] == x) {

for (j = i; j < n2; j++)

faceID[j] = faceID[j + 1];

n2--;

break;

}

if (i == n2) cout << "抱歉，你要删除的ID" << x << "不在系统中！" << endl;

}

}

cin.clear();

}

void clearid()

{

n1 = 0; n2 = 0;

}

void showid()

{

int i;

cout << "有效卡ID：" << endl;

for (i = 0; i < n1; i++)

cout << cardID[i] << '\t';

cout << endl;

cout << "有效人脸ID：" << endl;

for (i = 0; i < n2; i++)

cout << faceID[i] << '\t';

cout << endl;

}

};

说明：类中的两个数组cardID faceID分别存储着系统中有效的卡号和人脸识别ID，同时实现普通的增删算法，如果要增加/删除的ID已经在系统中，那么输出提示。

3.Manage类代码：

#include "person.h"

#include "IDSystem.h"

class Manage :public IDSystem

{

protected:

student students[20];

teacher teachers[20];

int m1;

int m2;

public:

Manage() { n1 = 0; n2 = 0; m1 = 0; m2 = 0; }

void input()

{

string name, identity, method1, method2, act;

char time[20];

int id1, id2;

cout << "输入进出人员信息:" << endl;

while (cin >> name) {

cin >> identity;

if (identity == "学生") {

cin >> method1; cin >> id1;

}

else if (identity == "老师") {

cin >> method1; cin >> id1;

cin >> method2; cin >> id2;

}

cin >> time >> act;

if (identity == "学生") {

student stu(name, id1, time, act);

students[m1++] = stu;

}

else if (identity == "老师") {

teacher tea(name, id1, id2, time, act);

teachers[m2++] = tea;

}

}

cin.clear();

int i, j, k;

student t; teacher r;

for (i = 0; i < m1; i++) {

for (k = i, j = i + 1; j < m1; j++) if (students[j].priority < students[k].priority) k = j;

if (k != i) { t = students[k]; students[k] = students[i]; students[i] = t; }

}

for (i = 0; i < m2; i++) {

for (k = i, j = i + 1; j < m2; j++) if (teachers[j].priority < teachers[k].priority) k = j;

if (k != i) { r = teachers[k]; teachers[k] = teachers[i]; teachers[i] = r; }

}

}

void infooutput()

{

int i;

for (i = 0; i < m1; i++)

cout << students[i].name << ' ' << "学生 卡号 " << students[i].cardid << ' ' << students[i].entertime << ' ' << students[i].action << endl;

for (i = 0; i < m2; i++)

cout << teachers[i].name << ' ' << "老师 人脸 " << teachers[i].faceid << " 卡号 " << teachers[i].cardid << ' ' << teachers[i].entertime << ' ' << teachers[i].action << endl;

}

void resultoutput()

{

cout << "显示模拟结果:" << endl;

int i, j;

for (i = 0, j = 0; (i < m1) && (j < m2);) {

if (students[i].priority < teachers[j].priority) { studentout(i); i++; }

else { teacherout(j); j++; }

}

if (i < m1)

for (; i < m1; i++)

studentout(i);

else if (j < m2)

for (; j < m2; j++)

teacherout(j);

}

void clear()

{

m1 = 0; m2 = 0;

}

private:

void studentout(int i)

{

int k;

cout << students[i].status << students[i].name;

if (students[i].action == "出") cout << ' ' << students[i].entertime << ' ' << "出系楼" << endl;

else {

for (k = 0; k < n1; k++)

if (cardID[k] == students[i].cardid) { cout << ' ' << "有效刷卡" << ' ' << students[i].entertime << ' ' << "进系楼" << endl; break; }

if (k == n1) cout << ' ' << "无效刷卡" << ' ' << students[i].entertime << ' ' << "未进系楼" << endl;

}

}

void teacherout(int j)

{

int k;

cout << teachers[j].status << teachers[j].name;

if (teachers[j].action == "出") cout << ' ' << teachers[j].entertime << ' ' << "出系楼" << endl;

else {

for (k = 0; k < n2; k++)

if (faceID[k] == teachers[j].faceid) { cout << ' ' << "有效人脸识别" << ' ' << teachers[j].entertime << ' ' << "进系楼" << endl; break; }

if (k == n2) {

cout << ' ' << "无效人脸识别";

for (k = 0; k<n1; k++)

if (cardID[k] == teachers[j].cardid) { cout << ' ' << "有效刷卡" << ' ' << teachers[j].entertime << ' ' << "进系楼" << endl; break; }

if (k == n1) cout << ' ' << "无效刷卡" << ' ' << teachers[j].entertime << ' ' << "未进系楼" << endl;

}

}

}

};

说明：继承自IDSystem同时又基于student、teacher类，主要实现了三个成员函数

input：逐个读入信息，把他们都存到类成员数组中，然后按照优先值进行从小到大的排列，优先值越小的优先输出

infooutput:把排好之后的信息输出

resultoutput:显示模拟结果，按照优先值的大小把student 和teacher混在一起输出，优先值小的先输出，输出的时候在系统中查找是否符合有效ID或人脸，以此来区分不同的输出方式。

main函数中实现交互：

int main()

{

int select;

int s;

Manage manager;

cout << "欢迎使用门禁系统！ " << endl;

while (true) {

cout << "下面输入数字选择你要进行的操作：（0：输入进出人员信息 1：进入管理员权限 2：退出)" << endl;

cin >> s;

cin.clear();

if (s == 0) { manager.input(); manager.infooutput(); manager.resultoutput(); manager.clear(); }

else if (s == 1) {

while (true) {

cout << "管理员权限：下面输入数字选择你要进行的操作：（0：录入 1：增加 2：删除 3：清空 4：打印现有有效ID 5：退出）" << endl;

cin >> select;

if (select == 0) manager.inputid();

else if (select == 1) manager.addid();

else if (select == 2) manager.eraseid();

else if (select == 3) manager.clearid();

else if (select == 4) manager.showid();

else if (select == 5) break;

else cout << "输入的数字错误，请重新输入！" << endl;

}

}

else break;

}

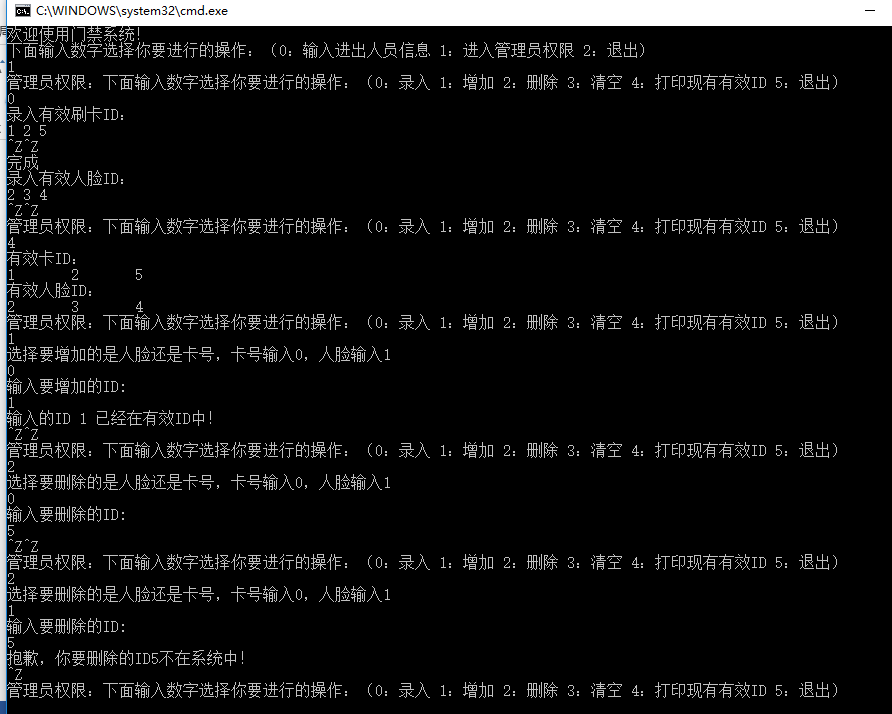
return 0;

}

四、运行结果

**注：与实验指导上不同，本系统所有的输入都要用空格而非逗号隔开，像”人脸 2”, “卡号 3”这样的也需要用空格隔开，同时，输入的时间中间的冒号是英文的冒号。**

首先验证系统权限的录入、增、删、清、打印功能。



再输入如下进出人员信息（注意，输入之后下面显示的数字就是该人员的优先值，可以看出确实按照优先值排列了）

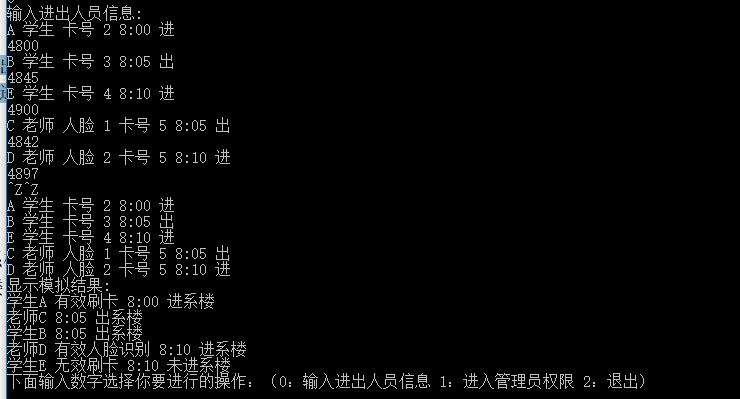
A 学生 卡号 2 8:00 进

B 学生 卡号 3 8:05 出

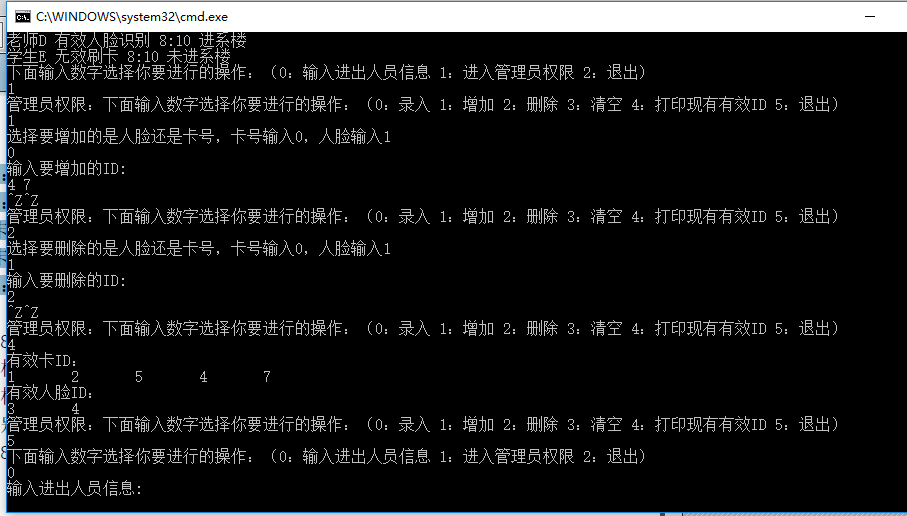
E 学生 卡号 4 8:10 进

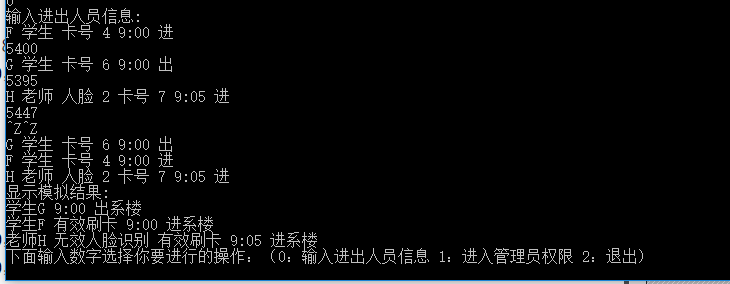
C 老师 人脸 1 卡号 5 8:05 出

D 老师 人脸 2 卡号 5 8:10 进



转过来对系统进行管理，增加一些ID，删除一些人脸，再次验证结果





五、实验中遇到的问题

在对Manage类进行设计的过程中，发现把学生和老师分开的操作非常繁琐，并且，学生与老师唯一的不同就是身份以及老师多出来一个faceID，那么还要不要把这两个类分开呢？

我经过思考认为还是要分开的，因为如果后续添加其他功能必然会对学生和老师产生区分，如果仅仅是设计成一个person类，那么以后再修改就麻烦了。