**实验报告**

题目：模拟选课系统

姓名：马瑞辰 学号；2019202387 完成日期：2020.10.21

1. 需求分析
2. 抽取出2016-2017 秋季学期所有班号作为模拟开设的教学班；对应的有效评估数作为该班最大能够容纳的学生人数；在后面的派课算法中，不允许给一个课程派超过容量限制的学生人数；例如一个教学班的容量是50，最多只能派给改班50个学生；
3. 撰写代码模拟约2000-4000名学生，按照学院分布，每个学院学生至少要大于50名；
4. 输入学院名字，可以打印该学院所有学生，按照学号排序；输入学号，可以检索该学生的信息，包括学号、所在学院等。
5. 撰写代码生成模拟bidding数据（选课志愿记录），按照学号、选课时间、志愿班号的形式生成模拟测试数据。控制该测试数据中每个学生最多可以选择50个班级作为志愿，最少需要选择5个班；请尽量设计一些逻辑（随机，参考课程容量等）模拟真实情况，而不是为每个学生分配一些顺序的班号；
6. 设计派课算法，满足：（1）每个班不允许超过最大能够容纳的学生人数，（2）让尽量多的学生都能够派上课，每个学生被派课的门数差异不太大（尽量少出现一个学生上了几十门课，而有的学生没课）（3）学科基础、专业必修、专业选修只允许自己学院的学生选修；（4）考虑学生选课时间的先后顺序，出现冲突时，优先考虑先选课的时间（但还需要综合考虑班的综合分配）
7. 派课算法执行后，可以通过输入一个班号，可以输出该班所有bidding了的学生以及最后派了的学生
8. 派课算法执行后，可以输入一个学生学号（模拟），输出该学生所有bidding的班号以及每个班是否被分配到该班；
9. 概要设计
10. 自定义队列

ADT Queue{

数据对象：D={}

基本操作：

void InitQueue(LinkQueue &Q)

操作结果：构造一个空队列

void DestroyQueue(LinkQueue &Q)

初始条件：队列Q已存在

操作结果：队列Q被销毁，不再存在

void Push(LinkQueue &Q,int e)

初始条件：队列Q已存在

操作结果：插入元素e为Q的新的队列的元素

void Pop(LinkQueue &Q)

初始条件：队列Q为非空队列

操作结果：删除Q的队头元素

bool Empty(LinkQueue Q)

初始条件：队列Q已存在

操作结果：若Q为空队列，则返回TRUE，否则返回FALSE

int Front(LinkQueue Q)

初始条件：队列Q为非空队列

操作结果：返回队首元素的值

1. 定义的学生和班级结构体数组

struct A{

int num; 学号

int eva; 课程容纳量

int curnum=0; 班级现有人数

int stu[100]; 选上该班的学生数组下标数组

string type; 课程类别

string coll; 开课学院

}cla[3000];

struct B{

int num; 班号

int volnum; 志愿报的班数

int actnum=0; 实际选上的班数

int vol[50]; 志愿报的班数组下标数组

int whether[50]={0}; 志愿报的班是否选上

int act[50]; 实际报的班数组下标数组

int count=0; 推进器

string coll; 所属学院

}stu[4000];

1. meun交互式界面

void printcoll()

输入学院号，输出该学院所有学生，按照学号排序，并统计总人数

void printstu()

输入学号，输出该学生的学号、所属学院、志愿报了哪些班级，这些班级是否选上和实际选上了哪些班级

void printcla()

可以通过输入一个班号，可以输出该班所有bidding了的学生以及最后派了的学生

1. 主程序

int main(){

数据读入、预处理、结构体数组初始化赋值；

do{

根据bidding分配课程；

menu()

}

}

1. 详细设计

typedef struct QNode{

int data;

struct QNode \*next;

}QNode,\*QueuePtr;

typedef struct{

QueuePtr front;

QueuePtr rear;

}LinkQueue;

void InitQueue(LinkQueue &Q){

Q.front=Q.rear=(QueuePtr)malloc(sizeof (QNode));

if(!Q.front) cout<<"OVERFLOW";

Q.front->next=NULL;

}

void DestroyQueue(LinkQueue &Q){

while(Q.front){

Q.rear=Q.front->next;

free(Q.front);

Q.front=Q.rear;

}

cout<<"删除完毕";

}

void push(LinkQueue &Q,int e){

QueuePtr p=(QueuePtr)malloc(sizeof (QNode));

if(!p) cout<<"OVERFLOW";

p->data=e; p->next=NULL;

Q.rear->next=p;

Q.rear=p;

}

void pop(LinkQueue &Q){

if(Q.front==Q.rear) cout<<"ERROR";

QueuePtr p=Q.front->next;

Q.front->next=p->next;

if(Q.rear==p) Q.rear=Q.front;

free(p);

}

bool empty(LinkQueue Q){

if(Q.front==Q.rear) return true;

return false;

}

int front(LinkQueue Q){

return Q.rear->data;

}

void printcoll()

{

int num,count=0;

cout<<"请输入学院号：";

cin>>num;

cout<<total\_coll[num]<<":"<<endl;

int temp[2000],j=0;

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

{

if(stu[i].coll==total\_coll[num])

{

temp[j++]=stu[i].num;

count++;

}

}

sort(temp,temp+j);

for(int i=0;i<j;i++)

cout<<temp[i]<<" ";

cout<<endl;

cout<<"共："<<count<<"人"<<endl;

}

void printstu()

{

int num;

cout<<"请输入学生学号：";

cin>>num;

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

{

if(stu[i].num==num)

{

cout<<"学号为："<<num<<" 学院为："<<stu[i].coll<<endl;

cout<<"该生志愿填报的班级为："<<endl;

for(int j=0;j<stu[i].volnum;j++)

{

cout<<cla[stu[i].vol[j]].num<<" ";

if(stu[i].whether[j]) cout<<"选上了";

else cout<<"未选上";

cout<<endl;

}

cout<<"该生实际选上的班级为："<<endl;

for(int j=0;j<stu[i].actnum;j++)

cout<<cla[stu[i].act[j]].num<<" ";

cout<<endl;

break;

}

}

}

void printcla() //可以通过输入一个班号，可以输出该班所有bidding了的学生以及最后派了的学生

{

int num,whether\_find=0;

cout<<"请输入一个班号：";

cin>>num;

for(int i=0;i<total\_class;i++)

{

if(cla[i].num==num)

{

whether\_find=1;

cout<<"选报该班的学生："<<endl;

for(int j=0;j<ran\_stu\_num;j++)

{

for(int k=0;k<stu[j].volnum;k++)

{

if(stu[j].vol[k]==i)

{

cout<<stu[j].num<<" ";

break;

}

}

}

cout<<endl;

cout<<"该班实际选上的学生："<<endl;

for(int j=0;j<cla[i].curnum;j++)

cout<<stu[cla[i].stu[j]].num<<" ";

cout<<endl;

break;

}

}

if(whether\_find==0)

cout<<"输入的班号不存在"<<endl;

}

void menu()

{

int button;

cout<<"1 输入学院编号，可以打印该学院所有学生，按照学号排序"<<endl;

cout<<"2 输入学号，可以检索该学生的信息，包括学号、所在学院，并输出该学生所有bidding的班号以及每个班是否被分配给该生"<<endl;

cout<<"3 输入一个班号，可以输出该班所有bidding了的学生以及最后派了的学生"<<endl;

cout<<"其它数字 退出程序"<<endl;

cout<<"请输入一个数字来执行一个功能："<<endl;

cin>>button;

switch (button) {

case 1:

printcoll();

menu();

break;

case 2:

printstu();

menu();

break;

case 3:

printcla();

menu();

break;

default:

cout<<"退出成功"<<endl;

break;

}

}

派课算法实现：

while(1)

{

//先将队列清空

while(!empty(q))

{

while(1) //一定要给队首学生分配某个志愿的课

{

if(stu[front(q)].count==stu[front(q)].volnum)

break;

if(cla[stu[front(q)].count].curnum<cla[stu[front(q)].count].eva)

{

cla[stu[front(q)].vol[stu[front(q)].count]].stu[cla[stu[front(q)].vol[stu[front(q)].count]].curnum++]=front(q);

stu[front(q)].whether[stu[front(q)].count]=1;

stu[front(q)].act[stu[front(q)].actnum++]=stu[front(q)].vol[stu[front(q)].count++];

break;

}

else

{

stu[front(q)].count++;

}

}

pop(*q*);

}

for(i=0;i<ran\_stu\_num;i++)

{

if(stu[i].count==stu[i].volnum) //该学生志愿已满足

{

dele\_stu\_num++;

continue;

}

if(cla[stu[i].vol[stu[i].count]].curnum<cla[stu[i].vol[stu[i].count]].eva)

{

cla[stu[i].vol[stu[i].count]].stu[cla[stu[i].vol[stu[i].count]].curnum++]=i;

stu[i].whether[stu[i].count]=1;

stu[i].act[stu[i].actnum++]=stu[i].vol[stu[i].count++];

}

else

{

stu[i].count++;

push(*q*,i);

}

}

if(dele\_stu\_num==ran\_stu\_num)

break;

dele\_stu\_num=0;

}

1. 附录

course.txt //原始数据

main.cpp //源代码