**《数据结构》**

**课程设计报告**

|  |  |
| --- | --- |
| 课程名称： | 《数据结构》课程设计 |
| 课程设计题目： | 学生教学计划编制 |
| 姓 名： | 应宇杰 |
| 院 系： | 计算机学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | 19052321 |
| 学 号： | 19151633 |
| 指导教师： | 葛瑞泉 |

2020年12月23日

1. **需求分析**

#### 功能需求：

大学的每个专业都要制订教学计划。假设任何专业都有固定的学习年限，每学年含两学期，每学期的时间长度和学分上限均相等。每个专业开设的课程都是确定的，而且课程在开设时间的安排必须满足先修关系。每门课程有哪些先修课程是确定的，可以有任意多门，也可以没有。每门课恰好占一个学期。试在这样的前提下设计一个教学计划编制程序。

1. **概要设计**

测试用例：

学期总数：6；

学分上限：10；

该专业共开设课数：12

课程号：c01  c02  c03  c04  c05  c06  c07  c08  c09  c10  c11  c12

学分顺序：2  3  4  3  2  3  4  4  7  5  2  3

先修关系如下图：

c01 c04

c01 c02

c01 c03

c02 c03

c04 c05

c03 c05

c03 c07

c01 c12

c03 c08

c09 c12

c09 c10

c09 c11

c10 c12

c11 c06

c06 c08

c05 c07

#### 接口设计

int LocateVex(ALGraph G,VertexType u)

//操作结果: 若G中存在顶点u,则返回该顶点在图中位置;否则返回-1 ；

Status CreateGraph(ALGraph &G)

//采用邻接表存储结构,构造没有相关信息的图G(用一个函数构造种图)；

void Display(ALGraph G)

//输出图的邻接矩阵G ；

void FindInDegree(ALGraph G,int indegree[])

//求顶点的入度；

typedef struct SqStack{

}SqStack;

//顺序栈；

Status InitStack(SqStack \*S)

//构造一个空栈S；

void ClearStack(SqStack \*S)

//清空栈的操作；

Status StackEmpty(SqStack S)

//若栈S为空栈，则返回TRUE，否则返回FALSE ；

Status Pop(SqStack \*S,SElemType \*e)

//若栈不空，则删除S的栈顶元素，用e返回其值，并返回OK；否则返回ERROR ；

Status Push(SqStack \*S,SElemType e)

//插入元素e为新的栈顶元素；

Status zxf(ALGraph G)

//求大学所有课程总学分；

Status TopologicalSort(ALGraph G)

//程序的核心函数：

有向图G采用邻接表存储结构，若G无回路,则按用户选择的方案输出G的顶点的一个拓扑序列并返回OK, 否则返回ERROR；

int main()

主函数；

**界面设计**

void Display(ALGraph G){ //显示函数

#### 数据结构设计

typedef struct ArcNode{ //弧结构；

int adjvex; //该弧所指向的顶点的位置;

struct ArcNode \* nextarc; //指向下一条弧的指针;

InfoType \* info; //网的权值指针;

}ArcNode; //表结点;

typedef struct{

VertexType data; //顶点信息;

ArcNode \*firstarc; //第一个表结点的地址,指向第一条依附该顶点的弧的指针;

}VNode, AdjList[MAX\_VERTEX\_NUM];

//创建一个图

typedef struct{

AdjList vertices,vertices2; //分别存课程名和学分；

int vexnum,arcnum; //课程数和先修关系数

int kind;

}ALGraph;

//strcmp如果相等返回0，查找这个先修是否存在和这个 先修关系的下标

int LocateVex(ALGraph G,VertexType u){

int i;

for(i=0;i<G.vexnum;++i)

if(strcmp(u, G.vertices[i].data)==0)

return i;

return -1;

}

1. **详细设计**

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

#include<iomanip>

#include<math.h>

#include <iostream>

using namespace std;

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define MAX\_NAME 10 //顶点字符串的最大长度;

#define MAX\_CLASS\_NUM 100

#define MAX\_VERTEX\_NUM 100

typedef int Status; // Status是函数的返回类型；

typedef int Boolean;

int Z=0;

int X=0;

int term\_num,credit\_lim,q=1;

typedef int InfoType;

typedef char VertexType[MAX\_NAME]; //字符串类型;

typedef enum{DG}GraphKind; //{有向图,有向网,无向图,无向网};

typedef struct ArcNode{ //弧结构；

int adjvex; //该弧所指向的顶点的位置;

struct ArcNode \* nextarc; //指向下一条弧的指针;

InfoType \* info; //网的权值指针;

}ArcNode; //表结点;

typedef struct{

VertexType data; //顶点信息;

ArcNode \*firstarc; //第一个表结点的地址,指向第一条依附该顶点的弧的指针;

}VNode, AdjList[MAX\_VERTEX\_NUM];

//创建一个图

typedef struct{

AdjList vertices,vertices2; //分别存课程名和学分；

int vexnum,arcnum; //课程数和先修关系数

int kind;

}ALGraph;

//strcmp如果相等返回0，查找这个先修是否存在和这个 先修关系的下标

int LocateVex(ALGraph G,VertexType u){

int i;

for(i=0;i<G.vexnum;++i)

if(strcmp(u, G.vertices[i].data)==0)

return i;

return -1;

}

//创造一个图

Status CreateGraph(ALGraph &G){

int i,j,k;

VertexType v1,v2; //顶点信息；

ArcNode \* p; //指向第一条依附某顶点的弧的指针；

printf("请输入教学计划的课程数: ");

scanf("%d",& G.vexnum);

printf("请输入课程先修关系数: ");

scanf("%d",& G.arcnum);

printf("请输入%d个课程的代表值(如:c01):\n",G.vexnum);

for(i=0;i<G.vexnum;++i) {

scanf("%s",G.vertices[i].data);

G.vertices[i].firstarc=NULL;

}

printf("请输入%d个课程的学分值:\n",(G).vexnum);

for(i=0;i<G.vexnum;++i) {

scanf("%s",G.vertices2[i].data);

}

printf("请顺序输入全部先修关系(以空格作为间隔):\n");

for(k=0;k<G.arcnum;++k) {

scanf("%s%s",v1,v2);

i=LocateVex(G,v1);

j=LocateVex(G,v2);

p = (ArcNode\*)malloc(sizeof(ArcNode)); //新建一个节点；

p->adjvex = j;

p->info = NULL;

p->nextarc = G.vertices[i].firstarc;

G.vertices[i].firstarc = p;

}

return OK;

}

void Display(ALGraph G){

int i;

ArcNode \* p;

switch(G.kind){

case DG: printf("有向图\n");

}

printf("%d个顶点：\n",G.vexnum);

for(i=0;i<G.vexnum;++i)

printf("%s ",G.vertices[i].data);

printf("\n%d条弧:\n",G.arcnum);

for(i=0;i<G.vexnum;i++){

p=G.vertices[i].firstarc;

while(p){

printf("%s→%s ",G.vertices[i].data,G.vertices[p->adjvex].data);

p=p->nextarc;

}

printf("\n");

}

}

//求各个值的入度和出度

void FindInDegree(ALGraph G,int indegree[]){

int i;

ArcNode \*p;

//都置0

for(i=0;i<G.vexnum;i++)

indegree[i]=0;

for(i=0;i<G.vexnum;i++){

p=G.vertices[i].firstarc;

while(p){

indegree[p->adjvex]++;

p=p->nextarc;

}

}

}

typedef int SElemType;

#define STACK\_INIT\_SIZE 10

#define STACKINCREMENT 2

typedef struct SqStack{

SElemType \*base;

SElemType \*top;

int stacksize;

}SqStack;

Status InitStack(SqStack \*S){

(\*S).base=(SElemType \*)malloc(STACK\_INIT\_SIZE\*sizeof(SElemType));

if(!(\*S).base)

exit(OVERFLOW);

(\*S).top=(\*S).base;

(\*S).stacksize=STACK\_INIT\_SIZE;

return OK;

}

void ClearStack(SqStack \*S) { //清空栈的操作

S->top=S->base;

}

Status StackEmpty(SqStack S){

if(S.top==S.base)

return true;

else

return false;

}

Status Pop(SqStack \*S,SElemType \*e){

if((\*S).top==(\*S).base)

return ERROR;

\*e=\*--(\*S).top;

return OK;

}

Status Push(SqStack \*S,SElemType e){

if((\*S).top-(\*S).base>=(\*S).stacksize){

(\*S).base=(SElemType \*)realloc((\*S).base,((\*S).stacksize+STACKINCREMENT)\*sizeof(SElemType));

if(!(\*S).base)

exit(OVERFLOW);

(\*S).top=(\*S).base+(\*S).stacksize;

(\*S).stacksize+=STACKINCREMENT;

}

\*((\*S).top)++=e;

return OK;

}

Status zxf(ALGraph G){ //求大学所有课程总学分；

int z=0;

for(int i=0; i < G.vexnum; i++){

z += atoi(G.vertices2[i].data);

}

return z;

}

typedef int pathone[MAX\_CLASS\_NUM];

typedef int pathtwo[MAX\_CLASS\_NUM];

Status TopologicalSort(ALGraph G){

int i,k,count,indegree[MAX\_VERTEX\_NUM];

bool has = false;

SqStack S;

pathone a;

pathtwo b;

ArcNode \* p;

FindInDegree(G,indegree);

InitStack(&S);

for(i=0;i<G.vexnum;++i){

if(!indegree[i])

Push(&S,i);

}

count = 0;

while(!StackEmpty(S)){

Pop(&S,&i);

a[i]=\*G.vertices[i].data; //课程名；

b[i]=\*G.vertices2[i].data; //学分；

printf("课程%s→学分%s ",G.vertices[i].data,G.vertices2[i].data);

++count;

for(p=G.vertices[i].firstarc; p; p=p->nextarc){

k=p->adjvex;

//如果入度不等于0就push

if(!(--indegree[k]))

Push(&S,k);

}

}

if(count<G.vexnum){

printf("此有向图有回路\n");

return ERROR;

}

else{

printf("为一个拓扑序列。\n\n");

has=true;

}

printf("请问您想使学生在各学期中的学习负担尽量均匀（输入1）\n");

printf("还是想使课程尽可能地集中在前几个学期中（输入2）？\n");

int pattern;

printf("请选择(1 or 2):");

scanf("%d",&pattern);

FindInDegree(G,indegree); //对各顶点求入度indegree[0..vernum-1] ;

ClearStack(&S);

printf("=====================================================\n");

printf("教学计划如下：\n");

int xq = 1; //学期数;

int xfh; //学分和；

int now=0;

int top = G.vexnum / term\_num ; //平均每学期课程数；

int pjxf = zxf(G) / term\_num ; //每学期平均学分；

while(xq <= term\_num + 1){

int result[20]; //某个学期的课程；

int m = 0;

xfh = 0;

now=0; //当前学期课程数 ；

for(i=0;i<G.vexnum;++i){

//输入入度为0的点

if(0 == indegree[i])

Push(&S,i);

}

if(xq == term\_num+1 && !StackEmpty(S)){

printf("还有课程未安排！\n");

}

while(!StackEmpty(S) && xq <= term\_num){

int xf;

Pop(&S,&i); //弹出栈顶元素；

xf = atoi(G.vertices2[i].data); //atoi:字符串转换成整型数, xf:学分;

xfh = xfh+xf;

now++;

//如果这学期学分大于最大学分，就退出

if(xfh > credit\_lim){

ClearStack(&S);

break;

}

if(pattern == 1){

//如果学期不等于二分之一总学期而且现在课程大于平均每学期课程就跳出

if(xq!=term\_num/2&&now>top){

ClearStack(&S); //该操作使程序跳出此内层的while循环;

now = 0;

break;

}

}

if(pattern == 2){

if(xq!=1&&xq!=term\_num/2&&xq!=term\_num/2-1&&now>top){

ClearStack(&S);

now = 0;

break;

}

}

indegree[i]--; //减为-1,避免被再次计入；

for(p=G.vertices[i].firstarc; p; p=p->nextarc){

k=p->adjvex;

//k头节点的入度减1

indegree[k]--;

if(indegree[k]==0)

Push(&S,k);

}

result[m]=i;

m++;

}

if(xq <= term\_num){

printf("第%d个学期的课程为：",xq);

for(int j=0; j<m; j++){

printf("课程%s ",G.vertices[result[j]].data);

}

}

printf("\n");

xq++;

ClearStack(&S);

}

printf("=====================================================\n");

return OK;

}

int main(){

printf(" 教学计划编制问题(拓扑排序AOV-网)\n");

//AOV-网:顶点表示活动，弧表示活动间优先关系的有向图；

int CONTINUE = 1;

printf("------------------------------------------------\n");

ALGraph f; //图的邻接表存储；

printf("请输入学期总数: ");

scanf("%d",&term\_num);

printf("请输入每学期的学分上限: ");

scanf("%d",&credit\_lim);

CreateGraph(f);

Display(f);

while(CONTINUE != 0){

TopologicalSort(f);

printf("\n按1继续，按0结束:");

scanf("%d",&CONTINUE);

}

return 0;

}

**四、调试分析**

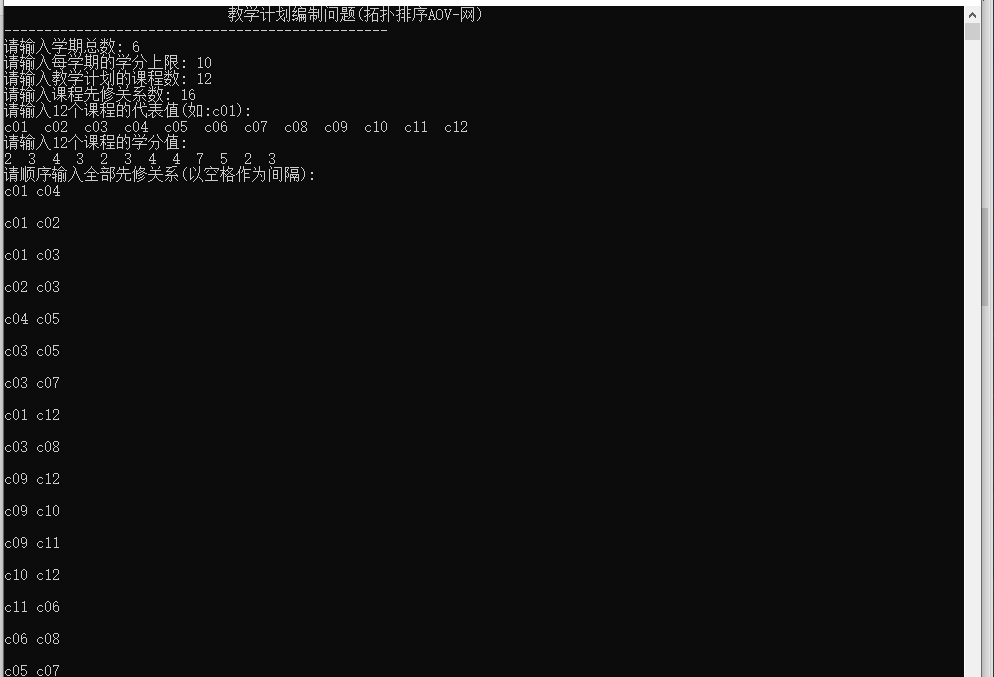
1、 过程中出现一门课程重复输出的现象，经过调试分析，发现是因为输出一轮后，零入度顶点栈未清空。

2、 按各学期均匀的方案进行时，有课程未安排的现象，最后让个别学期（中间偏前）不按方案进行。

**五、用户手册**

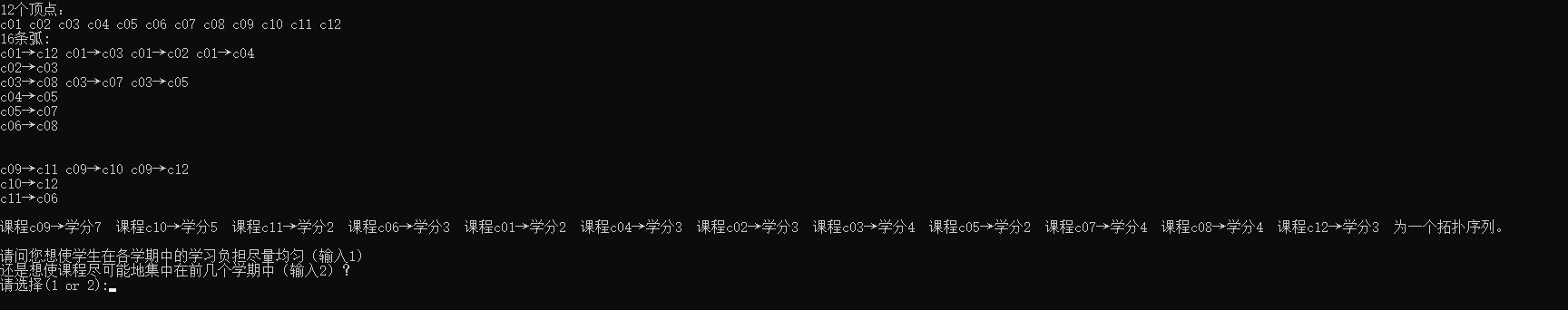
1. 本程序的执行文件为：学生教学计划编制）.exe
2. 进入演示程序后，将显示如下的界面



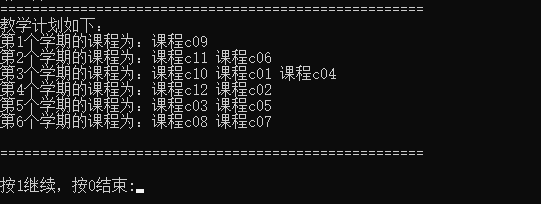
3、输入学期总数、学分上限、教学计划的课程数、输入课程先修关系数、输入课程代表值和课程学分、最后输入先修关系

**六、测试结果**

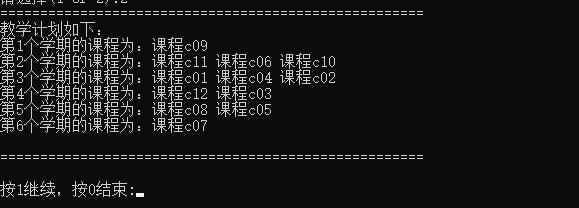
1、首先分别输入十二个点的弧，在输出各个弧的学分是多少，选择1为尽量均匀，选择2为尽量集中



2、选择1



3、选择2



4、按0结束

**七、附录**

源程序文件名清单：学生教学计划编制.cpp，学生教学计划编制.exe