**计算机软件基础实习报告**

**题目** 教学计划编制问题

**班级**  4231090504

**姓名** 甘硕奇

**学号**  423146150210

**指导老师** 张敬敏

**完成日期**  2025年1月10日

## 一、需求分析

1. 本程序中，需要完成对多门课程输入并按照学期最大分数限制和课程关系安排每学期的课程。首先得获取学期数和每学期最大学分，再按照顺序输入课程（自动跳过重名课程）。依据课程的链接关系，生成拓扑图。选择使课程尽可能地分散在各个学期中，对其进行遍历，累计达到足够分数后跳转至下一学期安排。

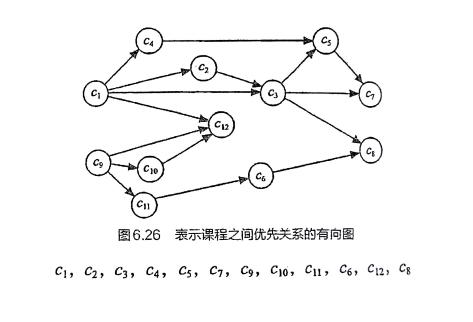
2. 演示程序以用户与计算机交互方式执行，即在计算机终端上显示“提示信息”之后，由用户在键盘上输入演示程序中规定的运算命令；

3. 程序执行的命令包括：(1)输入学期数、学期学分上限与课程数量。(2)输入课程名、学分及其前驱与后继课程 (3)删除课程(4)查找课程(5)输出所有课程详情 （6）输出计划编制 （7）退出

4. 测试数据：

学期总数：4学分上限：20；该专业共开设 12 门课，课程号从 C01 到 C12，

学分顺序为 2，3，4，3，2，3，4，4，7，5，2，3。先修关系见教科书图 6.26。



输出结果：第1学期课程为：

c1 c9 总分为：9

第2学期课程为：

c2 c4 c10 c11 总分为：13

第3学期课程为：

c3 c5 c6 c12 总分为：12

第4学期课程为：

c7 c8 总分为：8

## 二、概要设计

### 1. 程序使用的抽象数据类型

为实现上述程序功能，需要两个抽象数据类型：图和线性表。

1）线性表的抽象数据类型定义为：

ADT LinearList

{

**数据对象：**D = {ai|ai∈ElemSet,i=1,2,…,n,n≥0}

**数据关系：**R = {<ai-1,ai>| ai-1,ai∈D,i=2,…,n}

**基本操作：**

Listinit(&L)

操作结果：构造一个空的线性表L。

ListSeach( &L, C)

初始条件：线性表L已存在。

操作结果：在L中查找到课程名为C的节点，输出其所有信息。

ListAdd (&L)

初始条件：线性表L已存在。

操作结果：在有线性表L的未尾插入新节点并输入元素。

ListDel(&L,Class C)

初始条件：线性表已存在，表内存在Class类型元素C。

操作结果：删除表内元素C；

ListVisit(&L)

初始条件：线性表L已存在。

操作结果：遍历线性表并输出各个节点的课程及后继课程。

}ADT LinearList

2）图的抽象数据类型定义为：

ADT Graph

{

**数据对象**：V是具有相同特征的数据元素的集合，称为顶点集；

**数据关系：**

R={VR}

VR={<v,w>|v,w属于V且P(v,w)<v,w>表示从v到w的弧，

谓词P{(c,w)定义了弧<v,w>的意义或信息}

基本操作：

Graphinit(Graph& G)

操作结果：初始化图。

GraphCreate(&G,&L,classnub)

初始条件：图G存在，线性表L存在，classnub为课程数量。

操作结果: 根据每个线性表中每个课程的前驱、后继等信息构造拓扑图G。

GraphNodeDel( &G,int i)

初始条件：图G存在，i是G中某个顶点的序号。

操作结果：删除该节点，并将其所有后继节点的入度减1。

GraphOut(Graph &G)

初始条件：图G存在。

操作结果：对图进行递归遍历，每次遍历输出并删除每个入度为一的顶点。

}ADT Graph;

### 2. 本程序包含三个模块

1) 主程序模块

void main(){

do

{

输入学期总数、一学期的学分上限；

初始化线性表；

输入每门课的详细信息存储至线性表；

由线性表内容创建图；

对图进行拓扑排序，并根据学号及学习先后的限制，输出编制结果；

退出

}while(!退出)

｝

1. 图单元模块 实现图的抽象数据类型；
2. 线性表单元模块 实现线性表的抽线数据类型；

各模块之间的调用关系如下：

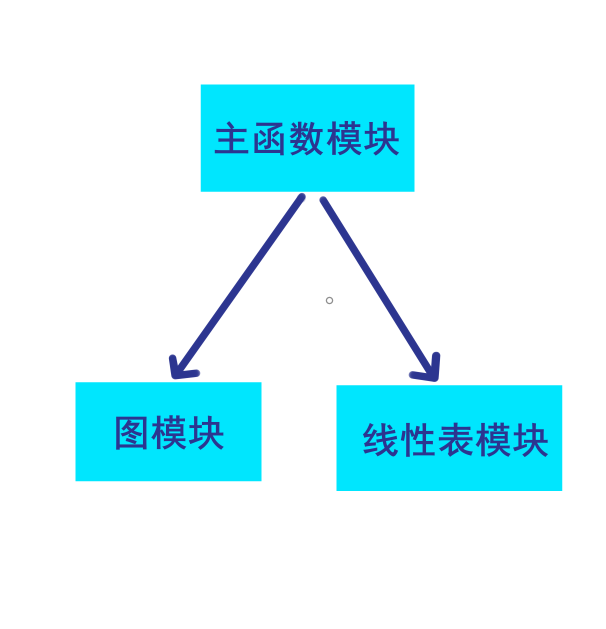


图2-1 模块调用关系图

## 三、详细设计

### 1. 链表ADT的 C语言形式

1）链表的C语言定义

集合中元素存放在以单链表存储的有序表中，集合元素类型、单链表结点类型的C语言定义如下：

#define OK 1

#define ERROR -1

#define MAXSIZE 50

typedef string Class; /\*课程的元素类型\*/

typedef struct ListNode {

Class C;/\*本课程名称\*/

Class subclass[MAXSIZE];/\*后继课程\*/

int subclassnub=0;/\*后继课程数量\*/

//Class preclass[MAXSIZE];/\*前驱课程\*/

int preclassnub = 0;/\*前驱课程数量\*/

int score = 0;

struct ListNode\* next;/\*链表节点后续\*/

}ListNode, \* LNode; /\*结点类型, \*/

2）. 链表的基本操作的C语言描述

int Listinit((LNode &L)

//构造一个带头结点的空的有序链表L,并返回OK;

//若分配空间失败,则令L为NULL,并返回ERROR;

ListSeach(LNode &L,Class C)

//查找课程名为C的节点，输出其全部信息

ListDel(LNode& L, Class C)

//查找课程名为C的文件，将其删除，返回OK。若未找到则返回ERROR;

void ListAdd ((LNode &L)

//在链表末尾添加新节点，并在节点中输入元素.

ListVisit((LNode &L)

//遍历链表所有节点；输出各个节点的课程、学分及后继课程，操作完成后返回OK

//若链表仅有头节点返回ERROR；

1. 链表表的基本操作C语言实现的伪代码

int Listinit(LNode &L){

1. >next=NULL；

}//Listinit

void ListAdd(LNode &L)

{

LNode New = new ListNode;

New->next = NULL;

LNode P = L;

while (P->next != NULL)//找到队尾

{

P = P->next;

}P->next = New;

cin >> New->C;//输入课程

cin >> New->score;//输入学分

cin >> New->preclassnub;//输入前驱数量

cin >> New->subclassnub;//输入后继数量

for (int i = 0; i < New->subclassnub; i++) {

cin >> New->subclass[i];//输入后继

}

}// ListAdd

int ListDel(LNode& L, Class C) {

LNode P = L;

LNode Q = P->next;

while (Q!= NULL) {

if (Q->C == C) {

P->next = Q->next;

Q->next = NULL;

free(Q);//释放空间

return OK;

}

P = P->next;

Q = Q->next;

}

}

int ListSeach(LNode &L,Class C) {//查找，找到链表中课程名和C相同的节点

LNode P = L->next;

int n = 0;

while (P->next != NULL) {

if (P->C == C) {

cout << P->C << endl;

cout <<P->preclassnub << endl;

cout <<P->subclassnub << endl;

for (int i = 0; i < P->subclassnub; i++) {//输出后继课程

cout << i + 1 << "、" << P->subclass[i] << endl;

}

cout << "该课程的学分：" <<P->score<< endl;

return OK;

}

P = P->next;

}

if (P->next == NULL) {

cout << "Not find!" << endl;

return ERROR;

}

}// ListSeach

int ListVisit(LNode& L) {

LNode P = L->next;

if (L->next == NULL) {

return ERROR;

}

else {

while (P!=NULL)

{

cout << "课程名：" << P->C << endl;

cout << "学分：" << P->score << endl;

cout << "后继课程：" << endl;

for (int i = 0; i < P->subclassnub; i++) {

cout << i + 1 << "、" << P->subclass[i] << endl;

}

cout << "前驱数量为：" << P->preclassnub << endl;

P = P->next;

}

}

}//ListVisit

### 2.图ADT的C语言描述

1）图的C语言定义

typedef struct topuGraph {

Vtype V[MAXSIZE];//顶点

int score[MAXSIZE];//学分

Edgetype E[MAXSIZE][MAXSIZE];//边

int Vertexnum;//顶点数

int outnub[MAXSIZE];//出度数

int innub[MAXSIZE];//入度数

}\*Graph;

int stack[MAXSIZE];//暂存节点序号数组，用于在拓扑排序中暂时存储

存储删除的节点序号

int top = 0;//数组顶

2）. 图基本操作的C语言描述

void Graphinit(Graph& G)

//图G的初始化

void GraphCreat(Graph& G, LNode& L,int classnub)

//根据链表L内容和课程数量classnub创建拓扑图

void GraphNodeDel(Graph &G,int i)

//删除节点及其边，并使其后继节点入度减1

void GraphOut(Graph &G)

//拓扑排序递归输出每个学期的课程

1. 图的基本操作C语言实现的伪代码

void Graphinit(Graph& G) {//图的初始化

G = new topuGraph;

}

void GraphCreat(Graph& G, LNode& L,int classnub) {/图的创建

G->Vertexnum = classnub;//将课程数赋值给节点数

LNode P = L->next;

for (int i = 0; i <G->Vertexnum; i++) {

G->V[i] = P->C;//给每个顶点赋值学科

G->outnub[i] = P->subclassnub;

G->innub[i] = P->preclassnub;

G->score[i] = P->score;

P = P->next;

}

for (int i = 0; i < G->Vertexnum; i++) {//将每条边赋值为零

for (int j = 0; j < G->Vertexnum; j++) {

G->E[i][j] = 0;

}

}

P = L->next;//P指针重新归位

for (int i = 0; i < G->Vertexnum; i++) {

for (int j = 0; j < G->Vertexnum; j++) {

if (P->subclassnub == 0) {

break;//若P节点中无后继，则直接跳转下一步

}

else {

for (int k = 0; k < P->subclassnub; k++) {

if (G->V[j] == P->subclass[k]) {//判断是否存在有向边

G->E[i][j] = 1;

}

}

}

}

P = P->next;

}

}

void GraphNodeDel(Graph &G,int i) {//删除节点

G->innub[i] = -1;//因为正常情况下节点入度不可能为-1，所以将访问过的节点标记为-1以区分

.

for (int j = 0; j<= G->Vertexnum; j++) {

if (G->E[i][j] != 0) {

G->E[i][j] = 0;//删除访问过的节点的边

G->innub[j]--;//后继节点的入度-1

}

}

}

void GraphOut(Graph &G) {//拓扑排序，输出编制结果

int totalscore = 0;

cout << "第" << times << "学期课程为：" << endl;

for (int i = 0; i < G->Vertexnum; i++) {

if (G->innub[i] == 0&&totalscore<=maxscore) {//

totalscore += G->score[i];

cout << G->V[i]<<" ";

stack[top] = i;//i保存到数组中

top++;

}

}

for (int i=0;i<=top;i++) {

GraphNodeDel(G, stack[i]);//删除入度为零的节点

}

times++;

cout << "总分为：" << totalscore << endl;

if (times == termnub + 1) {

times = 1;

return; }GraphOut(G);

}

4. 主函数和其他函数的伪码算法

int main() {

int loop = 1;

int classnub;

LNode L;

Listinit(L);

Graph G;

Class temp;

Graphinit(G);

while (loop) {

cin >> loop;

switch (loop) {

case 1:

cout << "请输入学期数：" << endl;

cin >> termnub;

cout << "请输入每学期最大学分：" << endl;

cin >> maxscore;

cout << "请输入课程总数量：" << endl;

cin >> classnub;

break;

case 2:

ListAdd(L);

break;

case 3:

cout << "请输入要删除的课程名：" << endl;

cin >> temp;

ListDel(L, temp);

break;

case 4:

cout << "请输入要查询的课程名：" << endl;

cin >> temp;

ListSeach(L, temp);

break;

case 5:

ListVisit(L);

break;

case 6:

GraphCreat(G, L, classnub);

GraphOut(G);

//Gout(G);

break;

case 7:

loop = 0;

break;

default:loop = 0;

break;

}

}

return 0;

}

5. 函数的调用关系图反映了演示程序的层次结构 :

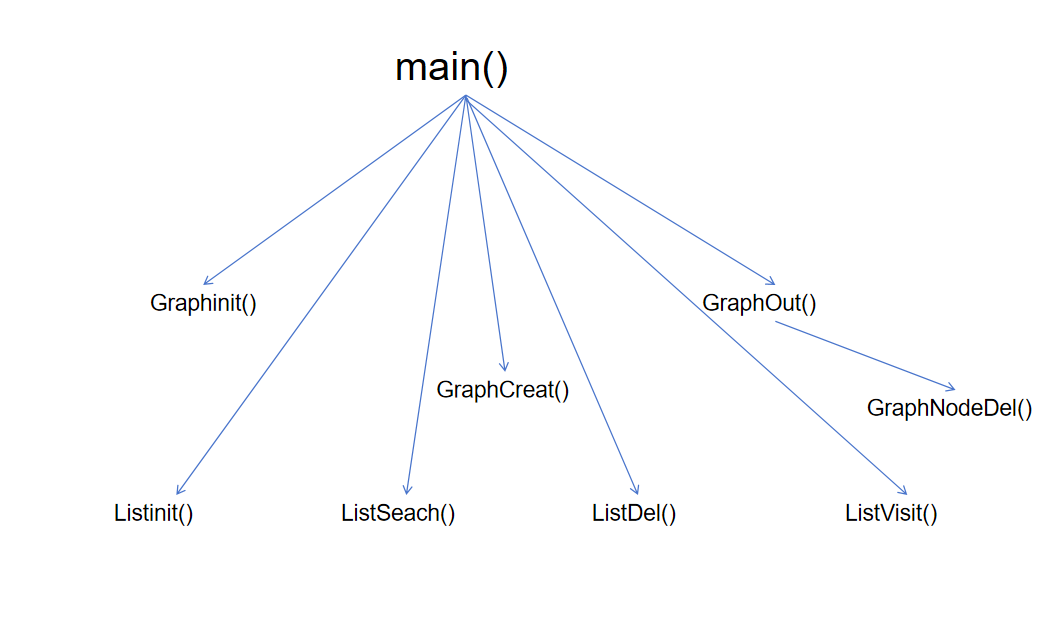


图3-1 函数调用图

## 调试分析

1. 问题回顾与设计分析：

1）每次在写图中的循环的函数GraphCreat(Graph& G, LNode& L,int classnub)时，忘记把节点指针归零，导致出现空指针。

2) 在写有关图的操作代码GraphOut(Graph &G)时，由于涉及多重嵌套循环，常常忘记起始值应从零开始还是从一开始，导致编译时经常出现错误，由多次进行反复测试，最终解决。

3)在实现拓扑排序GraphOut(Graph &G)函数时，未保存节点而直接删除，导致排序出错，后新建了一个全局的数组临时保存节点序号，保存后再删除节点及其邻边，解决该问题。

4）本程序的模块划分,首先用为方便数据的输入和更改，采用链表存储输入的节点及各节点信息，之后可以对链表进行遍历，查找，删除等操作，确保链表内的文件准确无误后课将其输入到有向图中，之后根据该有向图进行拓扑排序，并根据最大学分进行分配，输出每学期的课程。

2. 算法的时空分析

l)由于线性表中进行较多的为插入和删除，所以线性表采用链表，插入操作使用的函数ListAdd(LNode &L)及删除操作使用的函数ListDel(LNode& L, Class C)时间复杂度为O(n)，假设每个节点所占空间的大小为1,空间复杂度为O(n)。

查找函数ListSeach(LNode &L,Class C)时间复杂度为O(n),假设每个节点所占空间的大小为1,空间复杂度为O(n)

遍历函数 ListVisit(LNode& L)时间复杂度为O(n),

1. 图的创建函数GraphCreat(Graph& G, LNode& L,int classnub)，由于涉及多步操作嵌套循环，复杂度较高，时间复杂度为O(n2)空间复杂度为O(n2)。

图的顶点删除函数GraphNodeDel(Graph &G,int i)，需要完成删除边的操作，时间复杂度为O(n)，空间复杂度为O(n)。

图的输出函数GraphOut(Graph &G)时间复杂度为O(n)，空间复杂度为O(n2)。

1. 经验和体会

本实习作业采用数据抽象的程序设计方法，将程序划分为三个模块；链表、有向图和主控模块。通过链表来储存数据，通过图来运算并得出结果，完成输出。通过这次学习我学习并掌握了有向图的构建和拓扑排序，学会用递归算法去处理数据并得出结果，掌握了数据由链表到图的传入，深感受益匪浅。此外，我还更加巩固了对线性表，尤其是链表的使用操作，包括数据的输入、输出、查找、遍历。对其时空复杂度的分析也有了更加深刻的认识。深感得到了一次良好的程序设计训练。

## 五、测试

执行命令1

输入学期数、每学期最大学分、课程总数量后，将这些信息保存

执行命令2

输入课程的详细信息，将其存入链表中，多次重复输入各个课程

执行命令3

输入课程名，在线性表中查找并删除

执行命令4

输入课程名，在线性表中查找并输出其详细信息

执行命令5

查看链表中所有课程

执行命令6

输出计划安排

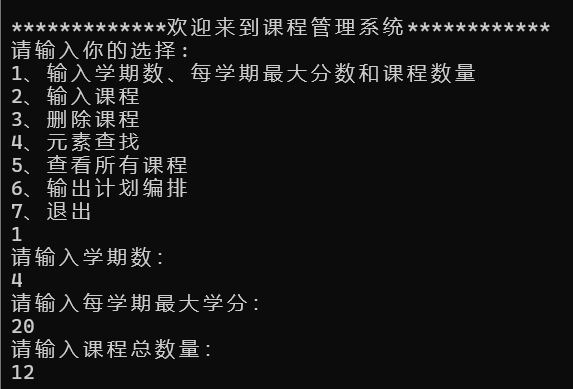


图5-1 基本数据输入

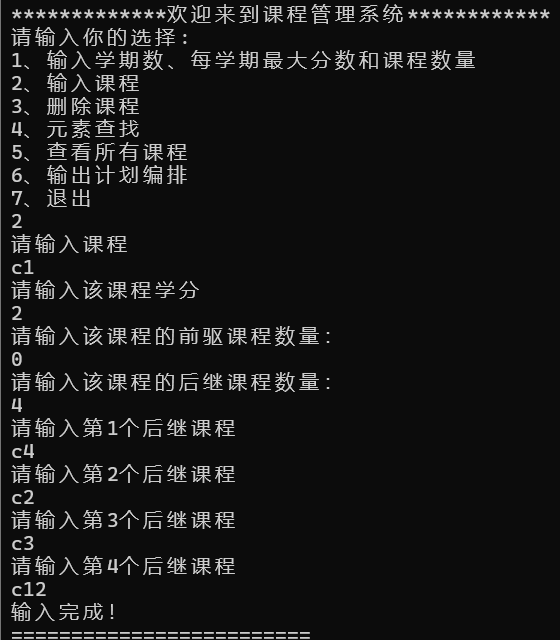


图5-2 课程数据输入

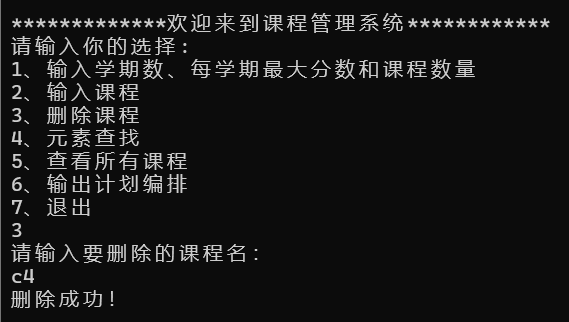


图5-3 课程删除

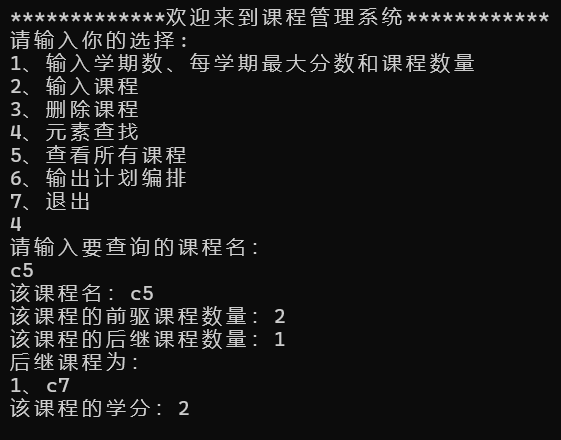


图5-4 课程查找

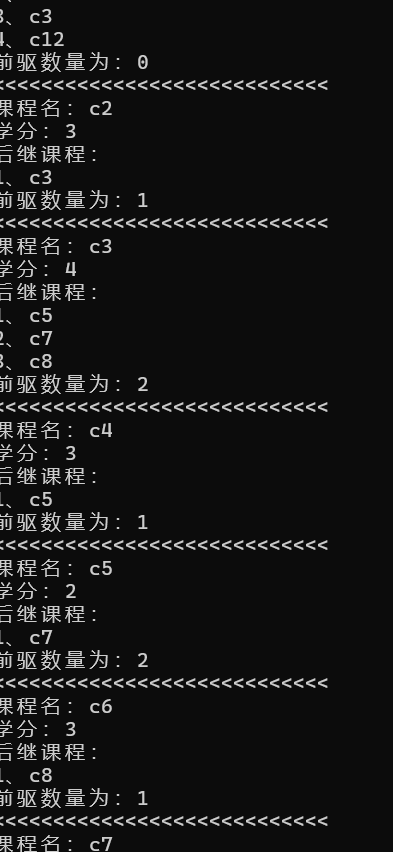


图5-5 输出所有课程数据

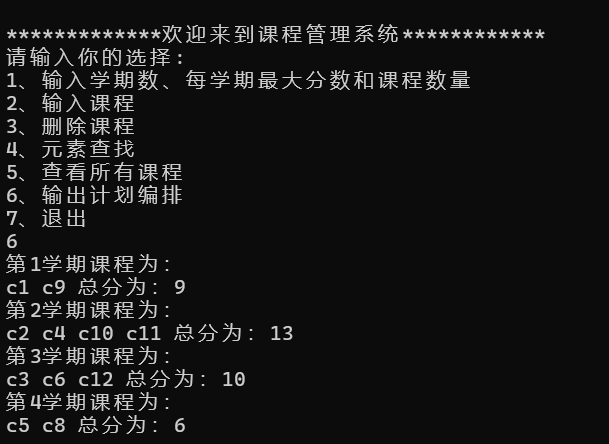


图5-6 输出计划编制

## 六、附录 源程序文件名清单

Graph.h //图单元。

List.h //有序链表单元

main.cpp //主程序单元

# 课程设计报告排版要求

一、页面设置

纸张大小：A4型（297mm×210mm）。

页边距：上为2.5cm，左2.5cm，下、右各为2cm。

装订线：左侧0.5cm。

正文行间距采固定值18磅行距，小四号宋体。

二、课程设计报告正文

1、一级标题：三号、宋体加粗；段前、段后间距各9磅

2、正文：首行缩进两个字符，小四号宋体，两端对齐，行间距为18磅。伪码算法缩进不受此限制。

3、图题：按大标题编号，字体宋体五号字居中。