

## 实验八 有向图及其应用

### 一、实验目的

1. 掌握基于邻接表存储结构的图的定义与实现。
2. 掌握图的拓扑排序算法的实现以及应用

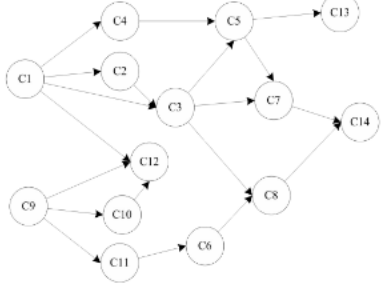
### 二、实验内容

**项目名称：**教学计划编制系统

**项目内容：**大学的每个专业都要制定教学计划。假设任何专业都有固定的学习年限，每学年包含两个学期，每个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上必须满足先修关系。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系统，该系统需要满足以下功能。

- (1) 完成课程进修目录信息的读取。

表 1 计算机专业进修课程

课程进修关系图		课程编号	课程名称	学分
		C1	程序设计基础	2
		C2	离散数学	3
		C3	数据结构	4
		C4	汇编语言	3
		C5	程序设计与分析	2
		C6	计算机原理	3
		C7	编译原理	4
		C8	操作系统	4
		C9	高等数学	7
		C10	线性代数	5
		C11	普通物理	2
		C12	数值分析	3
		C13	软件工程	3
		C14	数据库原理	3

- (2) 完成课程进修目录信息的编辑，包括课程的增加，删除，信息修改等。
- (3) 学生的教学计划学期为 6，每个学期的学分上限为 10 分，允许用户指定下列编排策略进行教学计划的输出
  - 1) 使学生在各个学期中的负担尽量均匀；
  - 2) 使课程尽可能地集中在前几个学期中若根据给定的条件问题无解，则报告适当信息；否则将教学计划输出到用户指定的文件中。计划的表格格式自行设计

测试数据不限于此。

### 三、实验要求

#### 1、线性表的应用

可以直接调用在前面课程中实现的线性表的类，该类中包含线性表的初始化操作，线性表中元素的增、删、改、查等功能。该线性表主要用来存放课程进修目录信息，完成课程进修目录信息的编辑功能，包括课程的增加，删除，信息修改。

线性表的定义参考如图 1 所示。

```

template <class T> <T>
class SqList
{
private:
    T *elem; //保持不变, NULL 不存在
    int length; //实际存放元素的个数
    int listsize; //可以容纳的最大元素的个数
public:
    SqList();
    ~SqList();
    void InputList();
    void OutputList();
    Status Insert(int i, T e); //在i位置插入一个元素
    Status Delete(int i, T &e); //删除i位置的元素
    Status Update(int i, T e); //在i位置更新一个元素
    int Locate(T e); //根据元素查找在线性表中的位置
};

```

图 1 线性表的定义参考图

## 2、堆栈的应用

可以直接调用在前面课程中实现的栈的类，该类中包含栈的初始化操作，判断栈是否为空操作，入栈和出栈操作。该栈主要用来存放入度为 0 的顶点，即当前没有先修关系，可以编排的课程。

栈的定义参考如图 2 所示。

```

template <class T> <T>
class SqStack
{
    T *base; //保持不变, NULL 不存在栈
    T *top; //栈顶, 指向不用(空)元素, 与定义不同
    int stacksize;
public:
    SqStack();
    ~SqStack();
    Status Push(T e);
    Status Pop(T &e);
    Status GetTop(T &e);
    int StackLength();
    Status IsEmpty();
    void DispStack();
};

```

图 2 栈的定义参考图

## 3、基于邻接表存储结构的图结构的定义与实现

设计并实现基于邻接表的图存储结构，需要包括创建图、展现图、统计图中各个顶点的入度等方法。图的定义参考如图 3，图 4 所示。

```

//弧信息
template <class T>
struct ArcInfo
{
    T From; //起点
    T To; //终点
    int weight;
};

//弧结点
template <class T> <T>
struct ArcNode
{
    int adjvex; //临接点位置
    int weight; //权值
    struct ArcNode *nextarc;
};

//顶点节点
template <class T>
struct VNode
{
    T data;
    int in;
    ArcNode *firstarc;
};

```

图 3 结构体定义

```

#pragma once
#include "GraphInfo.h"
#include "SqList.h"
#include "SqStack.h"

template <class T> <T>
class ALGraph
{
public:
    int vexnum; //顶点数目
    int arcnum; //弧数目
    VNode<T> vertices[Max]; //邻接表
public:
    ALGraph();
    ~ALGraph();
    void CreateGraph(int vnum, int anum,
        VNode<T> data[], ArcInfo arcList[]); //创建图
    void DispGraph(); //展示图
    int TopOrder();
    void IndegreeCal(); //统计每个顶点的入度
private:
    int LocateVex(VNode<T> v); //根据顶点信息, 返回顶点的坐标
};

```

图 4 类定义

#### 4、拓扑结构算法的实现与应用

拓扑排序的算法

- (1) 在 AOV 网中选一个入度为 0 的顶点（没有前驱）且输出之；
- (2) 从 AOV 网中删除此顶点及该顶点发出来的所有有向边；
- (3) 重复（1）、（2）两步，直到 AOV 网中所有顶点都被输出或网中不存在入度为 0 的顶点。

#### 5、主函数的实现

主函数要求控制良好的界面操作，提示用户进行各种不同功能操作的选择。

### 四、实验注意事项

1. 使用 c++语言的模板类实现；
2. 设计具有通用性、可复用性, 代码可读性强；
3. 实验分析和设计要有详尽的描述；
4. 在完成基本功能的基本上可以自己扩展其他功能。