项目说明文档

数据结构课程设计

——排课软件

作 者 姓 名：翟晨昊

学　　号：1952216

指 导 教 师：张颖

学院、 专业：软件学院 软件工程

同济大学

Tongji University

目　　录

1 分析

1.1 背景分析

排课是每个学校的教务处都要面对的一项工作，教务老师需要给出全校课程表以及每个专业的课表来供学生们选课，大学的每个专业都要进行这样的排课过程，这也是教学过程中至关重要的一部分。

排课过程并不是随意进行的，它要考虑多方面的因素，首先要考虑这门课的前置课程是否已经学完，还要考虑每一个学期的学习压力，课与课之间联系的紧密程度，安排完一个学期所学的课后，还要考虑一些课程以外的因素，比如老师，上课的时间与教室，分班情况，如果一次课排不完周时，还要考虑分成两次课的间隔，单双周情况。学校教务处需要统筹规划，安排出最合适的课表。一个安排合适的课表不仅能促进教学质量的提高，也能够让学生们学习起来事半功倍。因此，一个好的排课系统对学校来说非常重要。

1.2 功能分析

系统假设共有八个学期，周一至周五上课，每天上 10 节课，上午与下午各有 5 节课。第 1 大节为第 1-2 节课，第二大节为第 3-5 节课，第 3 大节为第 6-7 节课，第 4 大节为 8-10 节课，在排课时，如一门课程有 3 节课，则优先安排 3 节课连续上；如 3 节课连续无法按排，再优先安排两节课连续上，最后再安排单节课上的情况；如果一门课程需要安排上两天，为教学效果较好，最好不安排在相邻的两天，比如优先安排相隔 2 天上课。输入数据会包括八个学期所开的课程数（必须使每学期所开的课程数之和与课程总数相等），课程编号，课程名称，周学时数，指定开课学期，先决条件。如指定开课学期为 0，表示有电脑自行指定开课学期。同时用文本文件存储输入数据与产生的各学期的课表。

因此，首先要有从文本文件输入与输出至文本文件的功能，随后还要能根据每学期的排课数，各个课程的信息以及课程与课程的关系来进行排课，这也是系统功能的重中之重。

2 设计

2.1 数据结构设计

可以把一个学生八个学期的学习当作一个工程，每一门课程的学习就是整个工程的一些活动。有的课程要求有先修课程，有的没有要求。因此有的课程之间有先后关系，有的课程可以在一个学期内学习。为了显示这种关系，可以用有向图来表示这个工程。在这个有向图中，用顶点来表示活动（课程），用有向边<Vi,Vj>来表示课程 Vi 必须要在课程 Vj 之前上完。这种有向图叫做顶点表示活动的网络，记作 AOV 网络。对于一个 AOV 网络，要先判断其中是否有有向环，因为一旦网络中出现有向环，说明某项活动（课程）应以自己作为先决条件，这是明显不对的，工程会无法进行。

在将所有课程以及它们之间的先修关系加入网络后，就要对网络构造拓扑排序序列，即将每个课程排列成为一个线性有序的排列，使得 AOV 网络中的所有前驱与后继关系都能满足。因此就要通过拓扑排序的算法来对课程进行排序。一个 AOV 网络的拓扑排序序列不唯一，但是如果排序失败，就说明该网络中存在有向环。

进行拓扑排序时，首先设置一个数组保存每个顶点的入度，即直接前驱的数目，入度为零的顶点就是没有直接前驱的顶点。在排序时，要将 AOV 网络输入，然后在 AOV 网络中找到一个没有直接前驱（入度为零）的结点 m 并将其输出，同时遍历所有顶点，如果存在顶点 i 的直接前驱是该输出顶点 m，就将 i 对应的入度减一。不断重复这一过程，直至所有的顶点都被输出，拓扑排序序列形成。在本系统中，课程的入度即为该课程的先修课程数。

在拓扑序列形成后，再根据每学期的排课数，各个课程的信息以及课程与课程的关系来进行排课操作。

如上分析所述，该排课系统需要一个有向图来保存课程与它们之间的联系，系统选用了用邻接表表示的有向图的数据结构来进行保存。在拓扑排序以及选择课程时，会有大量的课程出入容器的操作，为了使得课程满足“先入先出”的情况，采用了链式队列的数据结构。由于每学期的课程数目等数据是动态变化的，因此还设计了 Vector 向量类来保存数据。

2.2 类结构设计

如上分析所述，首先本系统有一个有向图类（Graph 类），以及图中所需的顶点类（Vertex 类）和边类（Edge 类）。其中 Graph 类与 Vertex 类通过友元来建立起联系，这样使得 Graph 类可以访问 Vertex 类。其次，为了使拓扑排序能保存入度为零的点并能够将其按“先入先出”的顺序输出，本系统设计了一个链式队列类（LinkedQueue 类）以及它的结点类（LinkedNode 类）。除此之外，系统还设计了课程类（Course 类）与课程系统类（CourseSystem 类）。前者用来保存一门课程的课程编号，课程名称，周学时数，指定开课学期，先决条件等信息，后者将排课的输入，过程，输出整合到一个类中，方便进行管理。系统运行过程中的数据主要是通过 Vector 向量类来保存。为了使数据结构更具有泛用性，本系统将 LinkedQueue 类，Vector 类，Graph 类等都设计为了模板类。

2.3 成员与操作设计

向量类（Vector）：

```
template <typename ElementType> class Vector {
public:
    ~Vector();
    Vector();
    ElementType& operator[](const int x);
    const ElementType& operator[](const int x)const;
    Vector<ElementType>(const Vector<ElementType>& rhs);
    Vector<ElementType>& operator=(const Vector<ElementType>& rhs);
    bool isFull();
    bool isEmpty();
    void pushBack(const ElementType& temp);
    void popBack();
    void clear();
    int getSize();
    void reSize(int newSize);
private:
    void extendSize();
    int size;
    int maxSize;
    ElementType* myVector;
};
```

私有成员：

int size;//Vector 中已经存储的元素数量
int maxSize;//Vector 中已经申请的空间大小，元素数量如果超过要再次申请
ElementType* myVector;//存储的元素序列的起始地址

私有操作：

void extendSize();
//在 Vector 申请的空间已经被占满时再次申请空间，每次扩充至 maxsize*2+1
是因为刚开始的 maxsize 为 0

公有操作：

Vector();
//构造函数，初始化指针并将 size 与 maxSize 置为 0

~Vector();
//析构函数，调用 clear()函数来删除元素，释放内存

ElementType& operator[](const int x);
//重载[]运算符，返回 ElementType&类型

```
const ElementType& operator[](const int x)const;
//重载[]运算符，返回 const ElementType&类型

Vector<ElementType>(const Vector<ElementType>& rhs);
//复制构造函数，将一个 Vector 复制给另一个 Vector

Vector<ElementType>& operator=(const Vector<ElementType>& rhs);
//重载=运算符，可以将一个 Vector 赋给另一个 Vector

bool isFull();
//判断是否 Vector 中申请的内存已经被占满

bool isEmpty();
//判断 Vector 是否为空

void pushBack(const ElementType& temp);
//在 Vector 末尾添加一个元素

void popBack();
//删除 Vector 最末尾的元素

void clear();
//删除 Vector 中的所有元素并释放内存

int getSize();
//返回 Vector 已经存储的元素数量

void reSize(int newSize);
//重设 Vector 的大小，若比之前小，则会抛弃多余的元素
```

链式队列结点类（LinkedNode）：

```
template<typename ElementType>
class LinkedNode {
public:
    friend class LinkedQueue<ElementType>;
    LinkedNode() = default;
    LinkedNode(const ElementType& inputNum) : number(inputNum) {};
private:
    LinkedNode<ElementType>* next = nullptr;
    ElementType number = 0;
};
```

私有成员：

LinkedNode<ElementType>* next;//指针域，指向该结点的后一个结点
ElementType number;//结点中顾客的编号

公有操作：

friend class LinkedQueue<ElementType>;
//将 LinkedQueue 类声明为友元

LinkedNode() = default;
//默认构造函数

LinkedNode(const ElementType& inputNum);
//含参构造函数

链式队列类（LinkedQueue）：

```
template<typename ElementType>
class LinkedQueue {
public:
    LinkedQueue();
    ~LinkedQueue();
    void makeEmpty();
    int getSize();
    bool isEmpty();
    void enQueue(const ElementType& inputData);
    bool deQueue();
    ElementType getFront();
private:
    int size;
    LinkedNode<ElementType>* front;
    LinkedNode<ElementType>* rear;
};
```

私有成员：

int size;//队列中元素个数
LinkedNode<ElementType>* front;//指针域，指向队首结点
LinkedNode<ElementType>* rear;//指针域，指向队尾结点

公有操作：

LinkedQueue();
//默认构造函数，开辟一个附加头结点并将队列结点个数设为 0

~LinkedQueue();
//析构函数，通过调用 makeEmpty()将队列中的结点删除，实现对内存的回收

```
void makeEmpty();
//删除队列中的所有元素并释放内存

int getSize();
//返回队列中已经存储的元素数量

bool isEmpty();
//判断队列是否为空

void enQueue(const ElementType& inputData);
//入队

bool deQueue();
//出队

ElementType getFront();
//获取队首结点的顾客编号
```

边类（Edge）：

```
class Edge {
public:
    Edge() = default;
    Edge(int vertex1, int vertex2) :
        current(vertex1), dest(vertex2) {}
    int current;
    int dest;
    Edge* next = nullptr;
};
```

公有成员：

```
int current;//边相连的当前顶点位置
int dest;//边相连的另一顶点位置
Edge* next;//当前顶点的下一条边
```

公有操作：

```
Edge() = default;
//默认构造函数

Edge(int vertex1, int vertex2);
//含参构造函数
```

顶点类（Vertex）：

```
template<typename NameType>
class Vertex {
public:
    friend class Graph<NameType>;
    Vertex() = default;
    Vertex(int inputOrder, NameType inputName) :
        order(inputOrder), name(inputName) {}
private:
    int order;
    NameType name;
    Edge* head = nullptr;
};
```

私有成员：

int order;//该顶点在图中的位置
NameType name;//该顶点的名称
Edge* head;//与该顶点相连的第一条边

公有操作：

```
friend class Graph<NameType>;
//将 Graph 声明为友元

Vertex() = default;
//默认构造函数

Vertex(int inputOrder, NameType inputName);
//含参构造函数
```

图类（Graph）：

```
template<typename NameType>
class Graph {
public:
    friend class CourseSystem;
    Graph() = default;
    ~Graph();
    void clear();
    NameType getName(int order);
    Edge* getFirstEdge(int order);
    Edge* getNextEdge(Edge* curEdge);
    void insertVertex(NameType inputVertex);
    void insertEdge(int order1, int order2);
private:
```

```
    bool checkEdge(int order1, int order2);
    Vector<Vertex<NameType>*> vertexTable;
};
```

私有成员：

Vector<Vertex<NameType, ElementType>*> vertexTable;
//存储图中的所有顶点

私有操作：

bool checkEdge(int order1, int order2);
//检查两个位置所对应的顶点之间是否已经存在边<order1,order2>

公有操作：

friend class CourseSystem;
//将 CourseSystem 声明为友元

Graph() = default;
//默认构造函数

~Graph();
//析构函数，调用 clear()函数来删除元素，释放内存

void clear();
//删除图中的所有元素并释放内存

NameType getName(int order);
//获得该位置所对应的顶点的名称

Edge* getFirstEdge(int order);
//返回该位置所对应的顶点的第一条边

Edge* getNextEdge(Edge* curEdge);
//返回该边相连的下一条边

void insertVertex(NameType inputVertex);
//向图中加入顶点

void insertEdge(int order1, int order2);
//向两个位置所对应的顶点之间加入边<order1,order2>

课程类（Course）：

```
class Course {
public:
    friend class CourseSystem;
    Course() = default;
    Course(string inputNumber, string inputName, int inputDegree, int i
nputTime, int inputSemester) :
        number(inputNumber), name(inputName), degree(inputDegree), time
(inputTime), semester(inputSemester) {}
    Course& operator =(const Course& temp)
    {
        number = temp.number;
        name = temp.name;
        degree = temp.degree;
        time = temp.time;
        semester = temp.semester;
        orderInGraph = temp.orderInGraph;
        preCourse = temp.preCourse;
        return *this;
    }
    void clear();
private:
    string number;
    string name = "null";
    int degree = 0;
    int time = 0;
    int semester = 0;
    int orderInGraph = -1;
    Vector<int> preCourse;
};
```

私有成员：

```
string number;//课程的编号
string name = "null";//课程的名称
int degree;//课程的先修课程数
int time;//课程的周学时数
int semester;//课程的指定开课学期
int orderInGraph;//课程在图中的位置
Vector<int> preCourse;//课程的先修课程在图中的位置
```

公有操作：

```
friend class CourseSystem;
//将 CourseSystem 声明为友元
```

```
Course() = default;
//默认构造函数

Course(string inputNumber, string inputName, int inputDegree, int
inputTime, int inputSemester);
//含参构造函数

Course& operator =(const Course& temp);
//重载=运算符，可以把一个 Course 赋值给另一个 Course

void clear();
//将课程信息清空并初始化
```

课程系统类（CourseSystem）：

```
class CourseSystem {
public:
    CourseSystem() = default;
    ~CourseSystem();
    int stringToInt(string& str);
    bool inputData(fstream& in);
    bool outputData(fstream& out);
    void topologicalSort();
    bool selectCourse();
    void arrangeCourse(int semester);
private:
    bool judgeFixCourse(int semester, int& curScheduleTime);
    void arrangeThreeClass(int day, int first, Course& data);
    void arrangeTwoClass(int day, int first, Course& data);
    bool arrangeFiveClass(int day, int first, Course& data);
    bool arrangeFourClass(int day, int first, Course& data);
    int maxCourseNum[8] = { 0 };
    int scheduleMaxTime = 50;
    Vector<Vector<Course> > courseSchedule;
    Course courseTable[5][10];
    Vector<Course> allCourse;
    Vector<Course> fixCourse;
    Vector<Course> autoCourse;
    Graph<string> system;
    bool* isInSchedule = nullptr;
    int* inDegree = nullptr;
};
```

私有成员：

```
int maxCourseNum[8];//每个学期的课程数
int scheduleMaxTime = 50;//一个学期的最大课时数（10*5）
Vector<Vector<Course> > courseSchedule;
//保存每个学期安排的课程
Course courseTable[5][10];//一个学期的课程表
Vector<Course> allCourse;//保存输入的所有课程
Vector<Course> fixCourse;//保存指定开课学期的课程
Vector<Course> autoCourse; //保存未指定开课学期的课程
Graph<string> system;//用来储存课程与课程之间关系的图
bool* isInSchedule;//用来判断课程是否已经被安排
int* inDegree;//每个课程当前未被安排的先修课程数
```

私有操作：

```
bool judgeFixCourse(int semester, int& curScheduleTime);
//判断指定开课学期的课程中学期为 semester 的课程，并进行安排

void arrangeThreeClass(int day, int first, Course& data);
//在周 day，从第 first 节课开始安排三课时的课程 data

void arrangeTwoClass(int day, int first, Course& data);
//在周 day，从第 first 节课开始安排两课时的课程 data

bool arrangeFiveClass(int day, int first, Course& data);
//在周 day，从第 first 节课开始安排三课时的课程 data，再根据题目要求安排剩下的两课时

bool arrangeFourClass(int day, int first, Course& data);
//在周 day，从第 first 节课开始安排两课时的课程 data，再根据题目要求安排剩下的两课时
```

公有操作：

```
CourseSystem() = default;
//默认构造函数

~CourseSystem();
//构造函数，用来删除元素，释放内存

int stringToInt(string& str);
//将 string 类型的课程编号转化为 int 类型

bool inputData(fstream& in);
//将保存在文本文件中的课程信息与每学期排课数要求输入到系统中
```

```
bool outputData(fstream& out);
//将排好的课程表输出到文本文件中

void topologicalSort();
//将未指定开课学期的课程进行拓扑排序

bool selectCourse();
//选择每学期的课程

void arrangeCourse(int semester);
//将每学期选择的课程排入该学期的课程表中
```

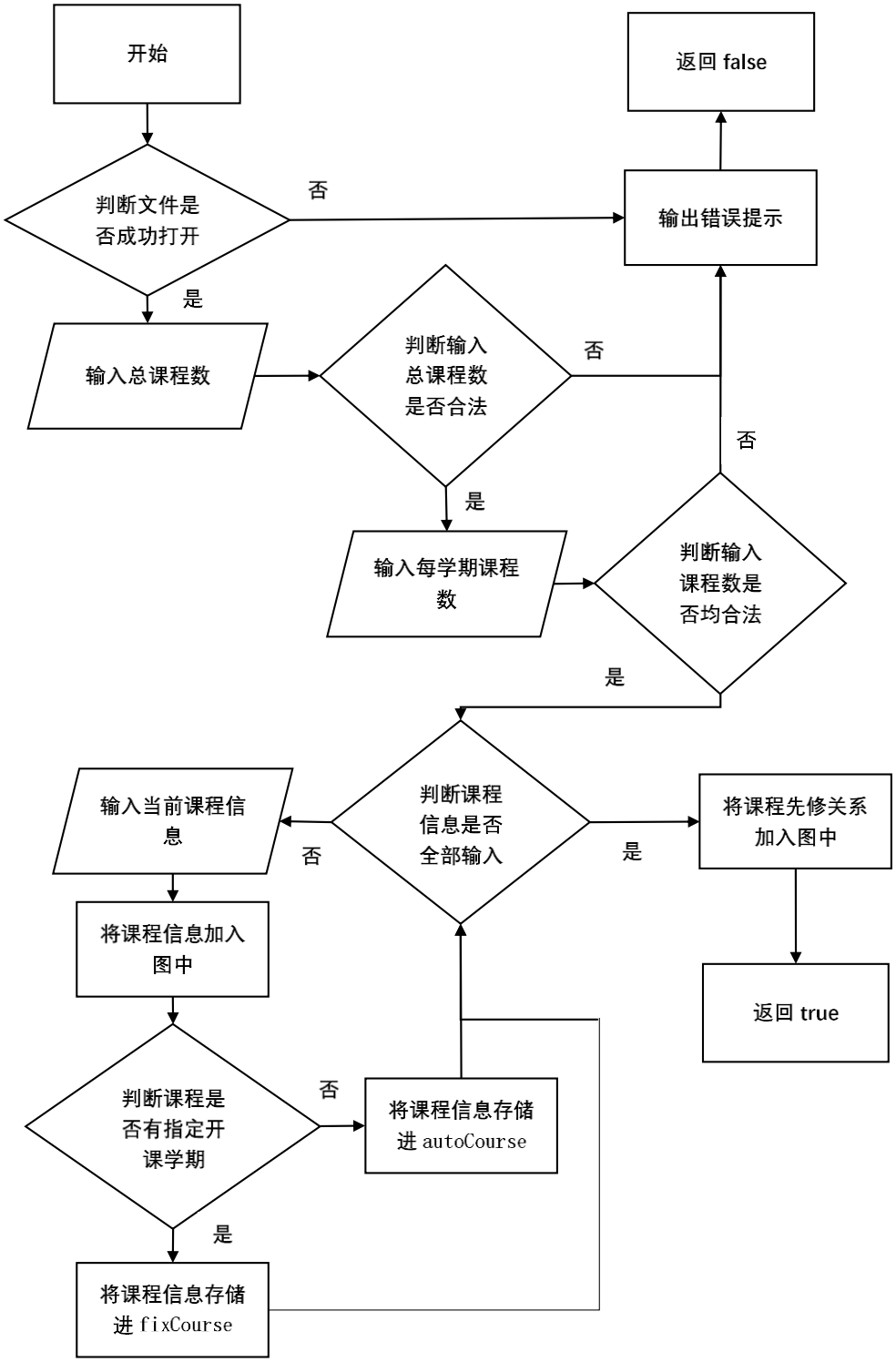
2.4 系统设计

系统首先会接收文本文件的输入，将课程信息与课程之间的关系存入课程网络中，并同时将指定开课学期的课程与未指定开课学期的课程分开储存。随后对未指定开课学期的课程进行拓扑排序，用排序生成的课程序列来进行每学期的课程安排，最后将生成的课程表输出到文本文档中并显示。功能有从文本文件输入与输出至文本文件，根据每学期的排课数、各个课程的信息以及课程与课程的关系来进行排课等等。

3 实现

3.1 从文本文件中输入功能的实现

3.1.1 从文本文件中输入功能流程图



3.1.2 从文本文件中输入功能核心代码

CourseSystem 类中：

```
bool CourseSystem::inputData(fstream& in)
{
    int allCourseNum = 0;
    int sumCourseNum = 0;
    in.open("in.txt", ios_base::in | ios_base::binary);
    if (in.is_open() == false)
    {
        cout << "文件打开失败，请检查文件是否存在！" << endl;
        return false;
    }
    in >> allCourseNum;
    if (allCourseNum <= 0 || in.fail())
    {
        cout << "总课程数不合法，排课软件将退出！" << endl;
        return false;
    }
    for (int i = 0; i < 8; ++i)
    {
        in >> maxCourseNum[i];
        if (maxCourseNum[i] <= 0 || in.fail())
        {
            cout << "存在学期课程数不合法，排课软件将退出！" << endl;
            return false;
        }
        sumCourseNum += maxCourseNum[i];
    }
    while (in.get() != '\n') continue;
    if (sumCourseNum != allCourseNum)
    {
        cout << "每学期所开的课程数值和与课程总数不相等，排课软件将退出！
" << endl;
        return false;
    }
    for (int i = 0; i < allCourseNum; ++i)
    {
        Course curCourse;
        string s;
        getline(in, s);
        stringstream ss(s);
        ss >> curCourse.number;
        ss >> curCourse.name;
```

```
            ss >> curCourse.time;
            ss >> curCourse.semester;
            while (ss >> s)
            {
                curCourse.preCourse.pushBack(stringToInt(s) - 1);
                curCourse.degree++;
            }
            curCourse.orderInGraph = i;
            system.insertVertex(curCourse.number);
            allCourse.pushBack(curCourse);
            if (curCourse.semester == 0)
            {
                autoCourse.pushBack(curCourse);
            }
            else
            {
                fixCourse.pushBack(curCourse);
            }
        }
        in.close();
        for (int i = 0; i < allCourse.getSize(); i++)
        {
            for (int j = 0; j < allCourse[i].preCourse.getSize(); j++)
            {
                system.insertEdge(allCourse[i].preCourse[j], allCourse[i].o
rderInGraph);
            }
        }
        isInSchedule = new bool[allCourseNum];
        for (int i = 0; i < allCourseNum; i++)
        {
            isInSchedule[i] = false;
        }
        return true;
}

int CourseSystem::stringToInt(string& str)
{
        int cur = 0;
        int sum = 0;
        int multiple = 1;
        for (int i = str.size() - 1; i >= 0; i--)
        {
            if (str[i] == 'c')
```

```
        {
            break;
        }
        cur = 0 + str[i] - '0';
        sum += (cur * multiple);
        multiple *= 10;
        cur = 0;
    }
    return sum;
}
```

Graph 类中：

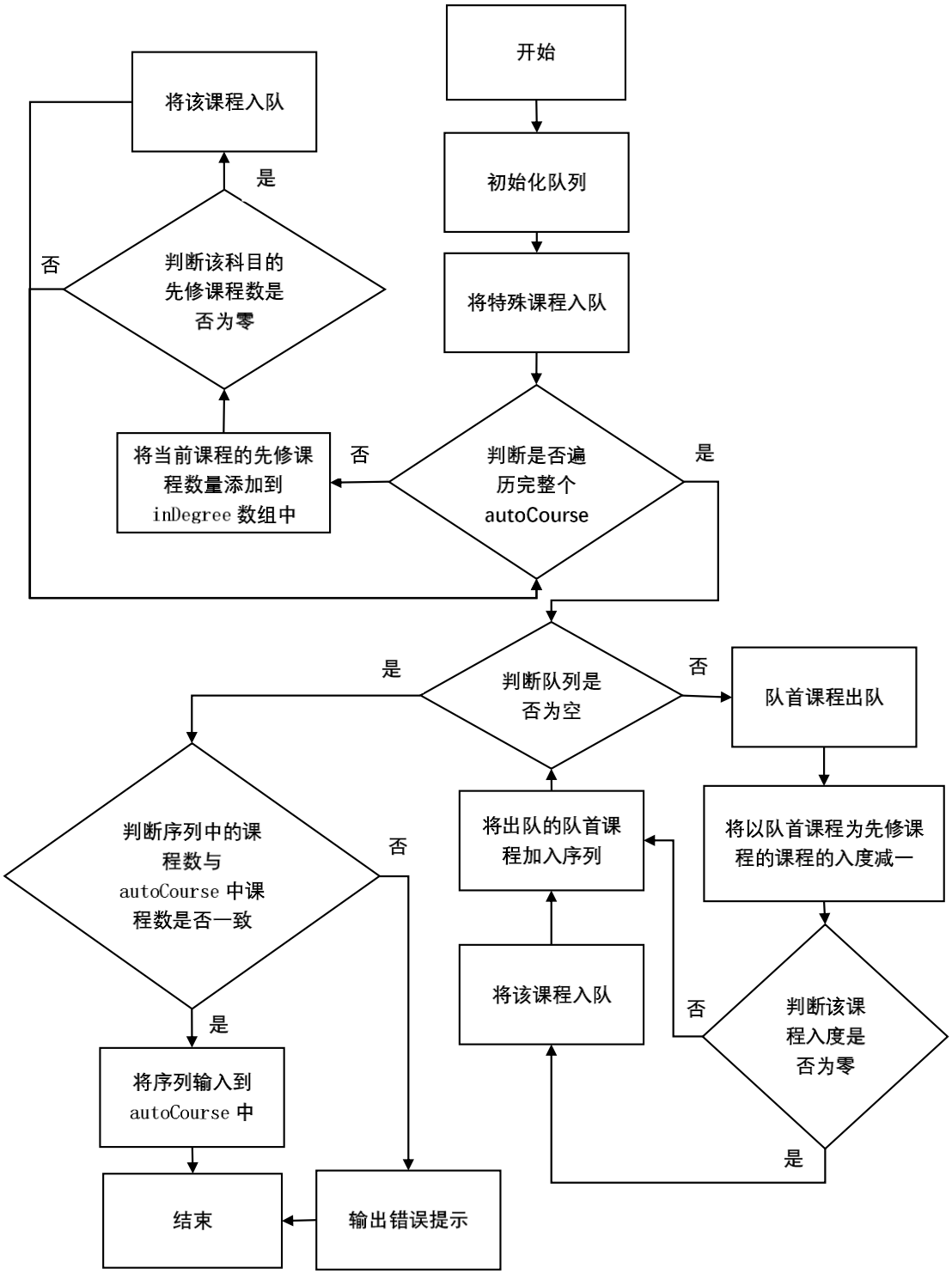
```
template<typename NameType>
void Graph<NameType>::insertVertex(NameType inputName)
{
    Vertex<NameType>* newVertex = new Vertex<NameType>(vertexTable.getS
ize(), inputName);
    vertexTable.pushBack(newVertex);
}

template<typename NameType>
void Graph<NameType>::insertEdge(int order1, int order2)
{
    Vertex<NameType>* pFirst = vertexTable[order1];
    Edge* pTemp = pFirst->head;
    if (pTemp == nullptr)
    {
        pFirst->head = new Edge(order1, order2);
    }
    else
    {
        while (pTemp->next != nullptr)
        {
            pTemp = pTemp->next;
        }
        pTemp->next = new Edge(order1, order2);
    }
}
```

3.1.3 从文本文件中输入功能描述

首先，尝试打开输入的文件，如果打不开文件，就输出错误提示并停止输入。如果能够打开，就向排课系统中输入总课程数与每学期的课程数。此时需要检验每学期所开的课程数值和与课程总数是否相等。如果不相等，会提示错误并停止输入，如果两者相等，采用 stringstream 来进行流的输入操作，将每一个课程的课程编号，课程名称，周学时数，指定开课学期，先决条件等信息输入进系统网络中，并通过判断课程是否指定开课学期来将课程分成两类。将所有信息输入后，再根据课程之间的先决关系在图中加入有向边。

3.2 拓扑排序功能的实现

3.2.1 拓扑排序功能流程图



3.2.2 拓扑排序功能核心代码

CourseSystem 类中：

```
void CourseSystem::topologicalSort()
{
    int current = 0;
    LinkedQueue<int> temp;
    Vector<Course> sortVec;
    int num = autoCourse.getSize();
    inDegree = new int[num + 4];
    for (int i = 0; i < num; i++)
    {
        inDegree[i] = 0;
    }
    for (int i = 0; i < 3; i++)//将 c01，c02，c03 提前排进去
    {
        temp.enQueue(autoCourse[i].orderInGraph);
    }
    for (int i = 3; i < autoCourse.getSize(); i++)
    {
        inDegree[i] = autoCourse[i].degree;
        if (inDegree[i] == 0)
        {
            temp.enQueue(autoCourse[i].orderInGraph);
        }
    }
    while (!temp.isEmpty())
    {
        current = temp.getFront();
        temp.deQueue();
        for (int i = 0; i < autoCourse.getSize(); i++)
        {
            if (system.checkEdge(current, autoCourse[i].orderInGraph))
            {
                inDegree[i]--;
                if (inDegree[i] == 0)
                {
                    temp.enQueue(autoCourse[i].orderInGraph);
                }
            }
        }
        sortVec.pushBack(allCourse[current]);
    }
    if (sortVec.getSize() != autoCourse.getSize())
```

```
    {
        cout << "课程中存在有向循环!" << endl;
        return;
    }
    else
    {
        autoCourse = sortVec;
    }
}
```

LinkedQueue 类中：

```
template<typename ElementType>
void LinkedQueue<ElementType>::enQueue(const ElementType& inputData)
{
    LinkedNode<ElementType>* temp = new LinkedNode<ElementType>(inputDa
ta);
    rear->next = temp;
    rear = rear->next;
    size++;
}

template<typename ElementType>
void LinkedQueue<ElementType>::deQueue()
{
    if (isEmpty())
    {
        return;
    }
    LinkedNode<ElementType>* temp = front->next;
    front->next = temp->next;
    delete temp;
    temp = nullptr;
    size--;
    if (size == 0)
    {
        rear = front;
    }
}

template<typename ElementType>
ElementType LinkedQueue<ElementType>::getFront()
{
    return front->next->number;
}
```

Graph 类中：

```
template<typename NameType>
bool Graph<NameType>::checkEdge(int order1, int order2)
{
    if (order1 == order2)
    {
        return false;
    }
    Vertex<NameType>* pFirst = vertexTable[order1];
    Edge* pTemp = pFirst->head;
    while (pTemp != nullptr)
    {
        if (pTemp->dest == order2)
        {
            return true;
        }
        pTemp = pTemp->next;
    }
    return false;
}
```

3.2.3 拓扑排序功能描述

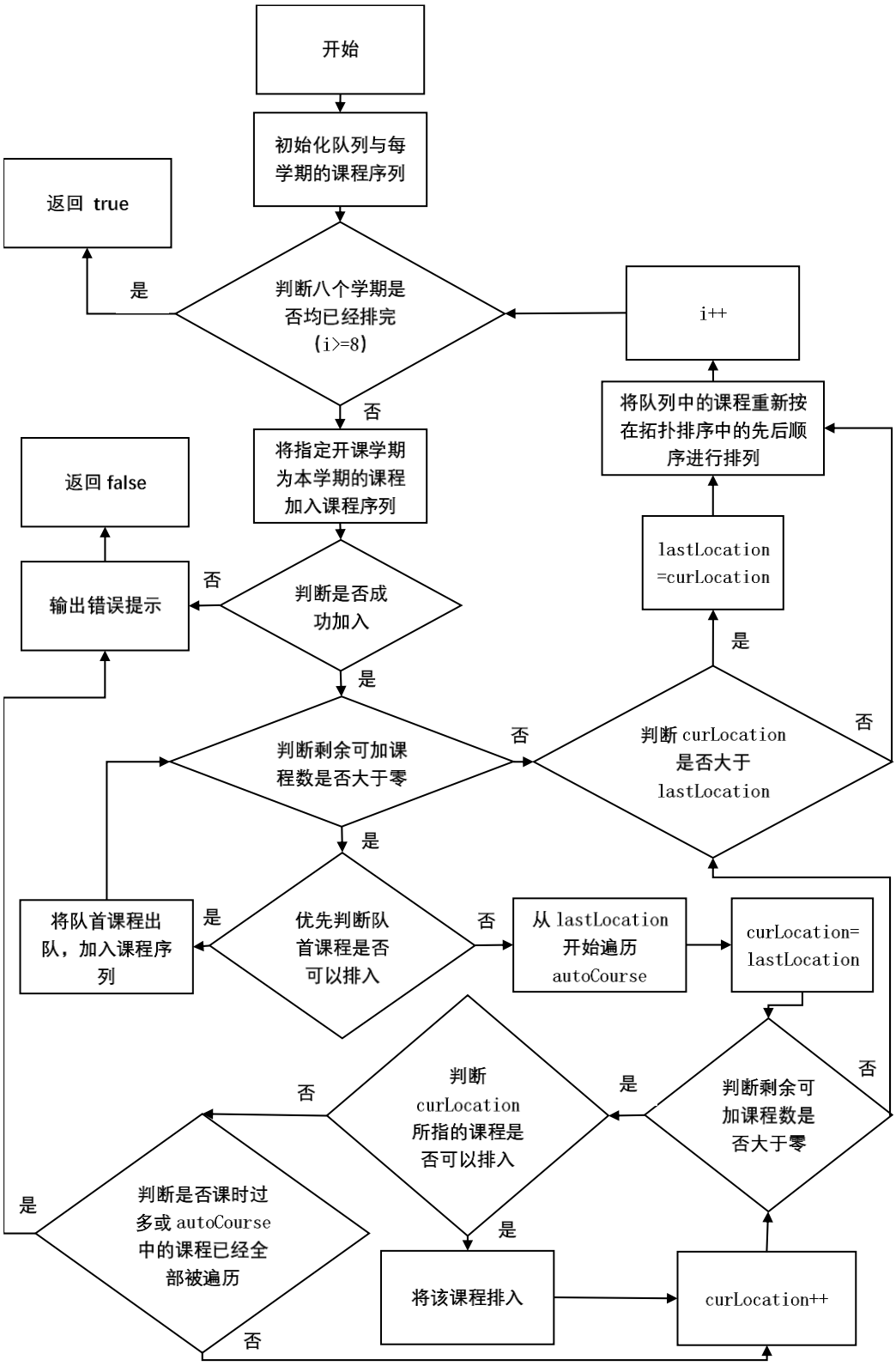
为什么要提前将特殊课程 c01，c02，c03 入队？
课表中学期为 0 且前置课程不要求 c03 的一共有 12 门课，而 c03 最早也只能在第三学期才能学，前三个学期的指定课程有三节英语和一节语文，共四节，因此可以得出前三学期安排的最多课程数目为 12+4=16 节，超过这个数目就无法进行排课。
拓展可以得到：
c03 课所在学期 n 与之前学期课程总和不能超过 13+n 节，否则排课失败。
为了使该限制对课表的影响最小，在拓扑排序时将它们三个优先加入队列，将 c01，c02，c03 分别优先安排在第一学期第二学期与第三学期。这样就使得只有前三学期有影响。

排序结果：

c01	程序设计基础	5 0	
c02	离散数学	6 0	c01
c03	数据结构算法	4 0	c01 c02
c06	计算机组成原理	6 0	
c12	高等数学	6 0	
c13	线性代数	6 0	
c16	计算机文化	3 0	
c19	数据通信	6 0	
c04	汇编语言	5 0	c01
c07	微机原理	4 0	c03
c08	单片机应用	3 0	c03
c09	编译原理	5 0	c03
c10	操作系统原理	4 0	c03
c11	数据库原理	5 0	c03
c18	计算机网络	5 0	c03
c20	面向对象程序设计	3 0	c01 c03
c21	Java	3 0	c01 c03
c22	C#.net	5 0	c01 c03
c23	PowerBuilder	5 0	c01 c03
c24	VC++	3 0	c01 c03
c25	ASP 程序设计	5 0	c01 c03
c26	JSP 程序设计	5 0	c01 c03
c27	VB.net	5 0	c01 c03
c28	Delphi	5 0	c01 c03
c29	C++Builder	5 0	c01 c03
c17	计算机系统结构	6 0	c06
c14	数值分析	6 0	c12
c15	普通物理	4 0	c12
c05	算法设计	4 0	c03 c04

3.3 选择课程功能的实现

3.3.1 选择课程功能流程图



3.3.2 选择课程功能核心代码

CourseSystem 类中：

```
bool CourseSystem::selectCourse()
{
    bool canBeAdd = true;
    int lastLocation = 0;
    int curLocation = 0;
    int addCourseNum = 0;
    int curScheduleTime = 0;
    LinkedQueue<int> remainCourse;
    int remainNum = 0;
    courseSchedule.reSize(8);
    for (int i = 0; i < 8; i++)
    {
        remainNum = remainCourse.getSize();
        if (!judgeFixCourse(i + 1, curScheduleTime))
        {
            return false;
        }
        addCourseNum = maxCourseNum[i] - courseSchedule[i].getSize();
        while (addCourseNum > 0)
        {
            if (remainNum > 0)
            {
                curLocation = remainCourse.getFront();
                remainCourse.deQueue();
                remainNum--;
            }
            else if (remainNum == 0)
            {
                curLocation = lastLocation;
                remainNum--;
            }
            if (curLocation >= autoCourse.getSize())
            {
                cout << "无法成功按要求排课!" << endl;
                return false;
            }
            if ((curScheduleTime + autoCourse[curLocation].time)
                <= scheduleMaxTime)
            {
                for (int j = 0; j < autoCourse[curLocation].preCourse.
                    getSize(); j++)
```

```
                {
                    if (isInSchedule[autoCourse[curLocation].
                        preCourse[j]] == false)
                    {
                        remainCourse.enQueue(curLocation);
                        canBeAdd = false;
                        break;
                    }
                }
                if (canBeAdd)
                {
                   courseSchedule[i].pushBack(autoCourse[curLocation]);
                    curScheduleTime += autoCourse[curLocation].time;
                    addCourseNum--;
                }
            }
            else
            {
                cout << "一学期安排的学时过多，请妥善安排课程!" << endl;
                return false;
            }
            canBeAdd = true;
            curLocation++;
        }
        if (curLocation > lastLocation)
        {
            lastLocation = curLocation;
        }
        while (remainNum > 0)
        {
            int temp = 0;
            temp = remainCourse.getFront();
            remainCourse.deQueue();
            remainCourse.enQueue(temp);
            remainNum--;
        }
        for (int j = 0; j < courseSchedule[i].getSize(); j++)
        {
            isInSchedule[courseSchedule[i][j].orderInGraph] = true;
        }
        curScheduleTime = 0;
        addCourseNum = 0;
    }
    return true;
```

```
}

bool CourseSystem::judgeFixCourse(int semester, int& curScheduleTime)
{
    for (int i = 0; i < fixCourse.getSize(); i++)
    {
        if (fixCourse[i].semester == semester)
        {
            courseSchedule[semester - 1].pushBack(fixCourse[i]);
            curScheduleTime += fixCourse[i].time;
        }
    }
    if (courseSchedule[semester - 1].getSize() > maxCourseNum[semester
- 1])
    {
        cout << "输入的学期课程数不合理!" << endl;
        return false;
    }
    else if (curScheduleTime > scheduleMaxTime)
    {
        cout << "课程设计存在问题" << endl;
        return false;
    }
    return true;
}
```

LinkedQueue 类中：

```
template<typename ElementType>
void LinkedQueue<ElementType>::enQueue(const ElementType& inputData)
{
    LinkedNode<ElementType>* temp = new LinkedNode<ElementType>(inputDa
ta);
    rear->next = temp;
    rear = rear->next;
    size++;
}

template<typename ElementType>
void LinkedQueue<ElementType>::deQueue()
{
    if (isEmpty())
    {
        return;
    }
```

```
    LinkedNode<ElementType>* temp = front->next;
    front->next = temp->next;
    delete temp;
    temp = nullptr;
    size--;
    if (size == 0)
    {
        rear = front;
    }
}

template<typename ElementType>
ElementType LinkedQueue<ElementType>::getFront()
{
    return front->next->number;
}
```

3.3.3 选择课程功能描述

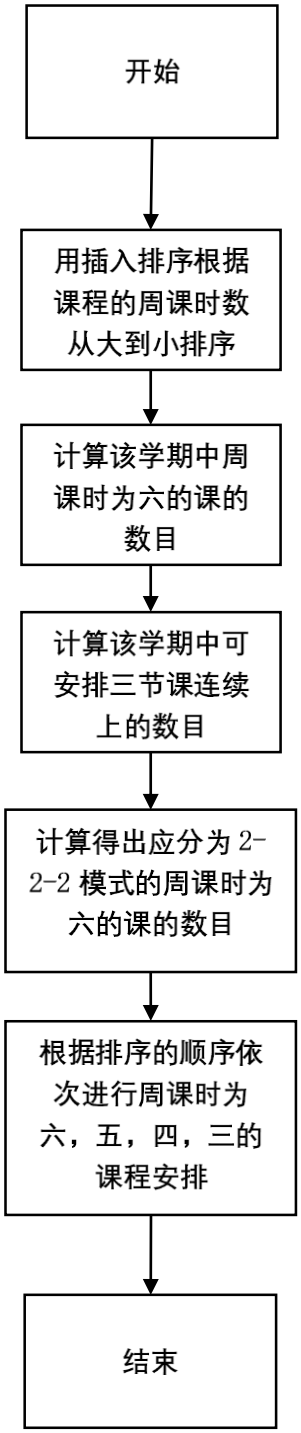
之前已经输入了每个学期的课程安排数。对于每个学期，先将指定课程为该学期的课程排入，如果这些课程排入后总数量超过了之前输入的该学期课程数或者总课时超过了一周的最大课时数，就说明输入的课程信息存在问题。如果没有该问题，且剩余可加课程数大于零，那么就在拓扑序列中寻找剩下的可加课程。

拓扑课程序列从第一个课程开始遍历，如果遇到课程无法排入，就将其加入到队列中。直到本学期课程安排已满，此时保存遍历到的位置。到下一个学期时，要先将队列中的课程遍历一遍，寻找到能够排入的课程就将其出队。此时分为两种情况：如果遍历完整个队列，剩余可加课程数大于零，就将位置更新为上一次遍历到的位置继续进行遍历拓扑序列过程；如果在遍历队列时就已经满足了本学期的课程安排，就将此时队列中的课程重新按在拓扑排序中的先后顺序进行排列，随后进入下一个学期的课程选择过程。一旦在拓扑课程序列中进行了遍历，就要更新保存最后遍历的位置。

在选择课程的过程中，如果出现已经遍历完所有的课程，但本学期的课程安排还没有被满足，或者在选择课程过程中，出现本学期选择课程的总周学时大于周最大允许学时时，都说明学期安排课程数存在有一定的问题。

3.4 安排课程表功能的实现

3.4.1 安排课程表功能流程图



3.4.2 安排课程表功能核心代码

CourseSystem 类中：

```
void CourseSystem::arrangeCourse(int semester)
{
    Course temp;
    Vector<Course> notInSchedule;
    int threeClassNum = 0;
    int threeToTwo = 0;
    int sixTimeClass = 0;
    bool isInTable = false;
    int i, j;
    if (courseSchedule[semester][0].time != 4)
    {
        threeClassNum += (courseSchedule[semester][0].time / 3);
    }
    for (i = 1; i < courseSchedule[semester].getSize(); i++)
    {
        temp = courseSchedule[semester][i];
        if (courseSchedule[semester][i].time != 4)
        {
            threeClassNum += (courseSchedule[semester][i].time / 3);
            sixTimeClass += (courseSchedule[semester][i].time / 3 - 1);
        }
        for (j = i; j > 0 && courseSchedule[semester][j - 1].time > tem
p.time; j--)
        {
            courseSchedule[semester][j] = courseSchedule[semester][j -
1];
        }
        courseSchedule[semester][j] = temp;
    }
    if (threeClassNum > 10)
    {
        threeToTwo = (threeClassNum - 9) / 2;//threeToTwo 不会超过三
    }
    while (sixTimeClass - threeToTwo > 4)
    {
        threeToTwo++;
    }
    for (i; i > 0; i--)
    {
        temp = courseSchedule[semester][i - 1];
        if (temp.time == 6)//七门六学时课
```

```
{
    if (threeToTwo > 0)
    {
        if (threeToTwo == 3)
        {
            arrangeTwoClass(1, 0, temp);
            arrangeTwoClass(3, 0, temp);
            arrangeTwoClass(3, 5, temp);
        }
        else if (threeToTwo == 2)
        {
            arrangeTwoClass(0, 0, temp);
            arrangeTwoClass(2, 0, temp);
            arrangeTwoClass(4, 0, temp);
        }
        else if (threeToTwo == 1)
        {
            arrangeTwoClass(0, 5, temp);
            arrangeTwoClass(2, 5, temp);
            arrangeTwoClass(4, 5, temp);
        }
        threeToTwo--;
    }
    else
    {
        for (int k = 0; k < 3; k++)
        {
            if (courseTable[k][2].time == 0)
            {
                arrangeThreeClass(k, 2, temp);
                arrangeThreeClass(k + 2, 2, temp);
                break;
            }
            else if (courseTable[k][7].time == 0)
            {
                arrangeThreeClass(k, 7, temp);
                arrangeThreeClass(k + 2, 7, temp);
                break;
            }
        }
    }
}
else if (temp.time == 5)
{
```

```
        for (int k = 0; k < 5; k++)
        {
            if (courseTable[k][2].time == 0)
            {
                if (arrangeFiveClass(k, 2, temp))
                {
                    break;
                }
            }
            else if (courseTable[k][7].time == 0)
            {
                if (arrangeFiveClass(k, 7, temp))
                {
                    break;
                }
            }
            if (k == 4)
            {
                notInSchedule.pushBack(temp);
            }
        }
    }
    else if (temp.time == 4)//5门四学时课
    {
        for (int k = 0; k < 5; k++)
        {
            if (courseTable[k][0].time == 0)
            {
                if (arrangeFourClass(k, 0, temp))
                {
                    break;
                }
            }
            else if (courseTable[k][5].time == 0)
            {
                if (arrangeFourClass(k, 5, temp))
                {
                    break;
                }
            }
            if (k == 4)
            {
                notInSchedule.pushBack(temp);
            }
```

```
            }
        }
        else if (temp.time == 3)
        {
            for (int k = 0; k < 5; k++)
            {
                if (courseTable[k][2].time == 0)
                {
                    arrangeThreeClass(k, 2, temp);
                    break;
                }
                else if (courseTable[k][7].time == 0)
                {
                    arrangeThreeClass(k, 7, temp);
                    break;
                }
                if (k == 4)
                {
                    notInSchedule.pushBack(temp);
                }
            }
        }
    }
    int num = 0;
    i = 0;
    if (notInSchedule.getSize() > 0)
    {
        temp = notInSchedule[i];
        for (j = 0; j < 5; j++)
        {
            for (int k = 0; k < 10; k++)
            {
                if (courseTable[j][k].time == 0)
                {
                    courseTable[j][k] = temp;
                    num++;
                }
                if (num == temp.time)
                {
                    i++;
                    if (i == notInSchedule.getSize())
                    {
                        return;
                    }
```

```
                    temp = notInSchedule[i];
                    num = 0;
                }
            }
        }
    }
}

void CourseSystem::arrangeThreeClass(int day, int first, Course& data)
{
    courseTable[day][first] = data;
    courseTable[day][first + 1] = data;
    courseTable[day][first + 2] = data;
}

void CourseSystem::arrangeTwoClass(int day, int first, Course& data)
{
    courseTable[day][first] = data;
    courseTable[day][first + 1] = data;
}

bool CourseSystem::arrangeFiveClass(int day, int first, Course& data)
{
    for (int j = day + 2; j < 5; j++)
    {
        if (courseTable[j][0].time == 0)
        {
            arrangeThreeClass(day, first, data);
            arrangeTwoClass(j, 0, data);
            return true;
        }
        else if (courseTable[j][5].time == 0)
        {
            arrangeThreeClass(day, first, data);
            arrangeTwoClass(j, 5, data);
            return true;
        }
    }
    for (int j = day - 2; j >= 0; j--)
    {
        if (courseTable[j][0].time == 0)
        {
            arrangeThreeClass(day, first, data);
            arrangeTwoClass(j, 0, data);
```

```
            return true;
        }
        else if (courseTable[j][5].time == 0)
        {
            arrangeThreeClass(day, first, data);
            arrangeTwoClass(j, 5, data);
            return true;
        }
    }
    return false;
}

bool CourseSystem::arrangeFourClass(int day, int first, Course& data)
{
    for (int j = day + 2; j < 5; j++)
    {
        if (courseTable[j][0].time == 0)
        {
            arrangeTwoClass(day, first, data);
            arrangeTwoClass(j, 0, data);
            return true;
        }
        else if (courseTable[j][5].time == 0)
        {
            arrangeTwoClass(day, first, data);
            arrangeTwoClass(j, 5, data);
            return true;
        }
    }
    for (int j = day - 2; j >= 0; j--)
    {
        if (courseTable[j][0].time == 0)
        {
            arrangeTwoClass(day, first, data);
            arrangeTwoClass(j, 0, data);
            return true;
        }
        else if (courseTable[j][5].time == 0)
        {
            arrangeTwoClass(day, first, data);
            arrangeTwoClass(j, 5, data);
            return true;
        }
    }
```

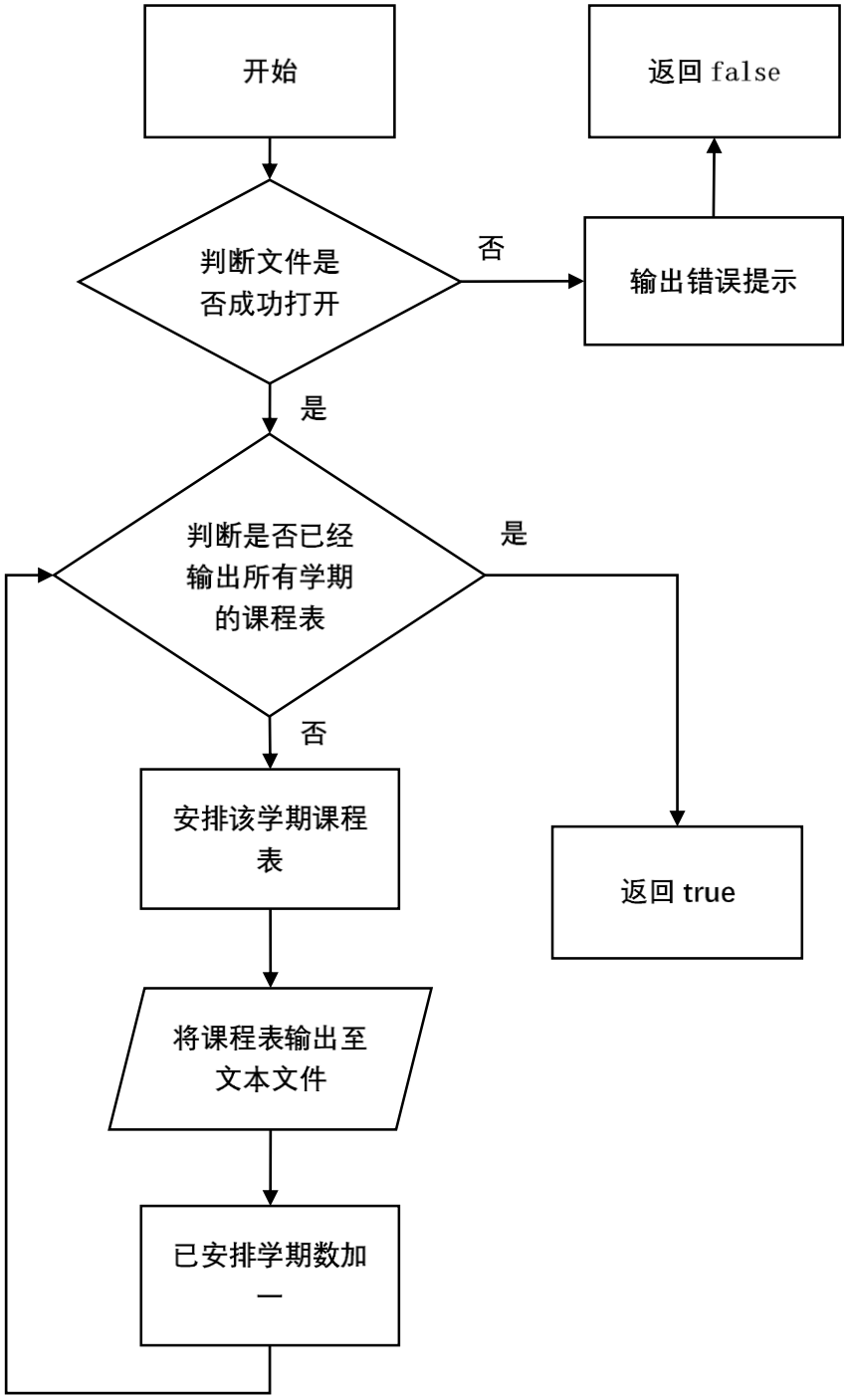
```
    return false;
}
```

3.4.3 安排课程表功能描述

首先将本学期的课程用插入排序根据课程的周课时数从大到小排序，并同时计算周课时数为六的课程数以及该学期中可安排三节课连续上的数目（例如六课时可以安排为两个三课时，数目加二；五课时可以安排为一个三课时加一个两课时，数目加一；三课时数目加一；四课时由于优先安排为两个两课时，因此数目加零）。由于一周中包含三个课时的课节数只有十节，因此如果可安排三节课连续上的数目超过十的话，需要把一些安排为三课时的转到两课时中。由于六课时的课可以从 3-3 排课方式变为 2-2-2 排课方式，因此优先选择将六课时的进行转化。threeToTwo 变量就记录了要进行转化的六课时数目。如果三课时数目为 threeClassNum（大于 10），那么 threeToTwo=(threeClassNum-9)/2。例如，如果 threeClassNum 为 13，由于一节六课时的课占两个三课时，因此只需要将两门六课时的课程进行转化，就能让 threeClassNum 变为 9，因此 threeToTwo 计算出来的结果即为 2。由于 50（一周最大学时数）除以 3 的商为 16，因此 threeToTwo 的最大值只能取 3，只需要准备三种 2-2-2 的排课方式即可。准备工作安排好后，接下来就按照插入排序生成的序列依次加入课程表中。最后将实在无法严格按要求的课程依次加入到课程表中。

3.5 输出到文本文件功能的实现

3.5.1 输出到文本文件功能流程图



3.5.2 输出到文本文件功能核心代码

CourseSystem 类中：

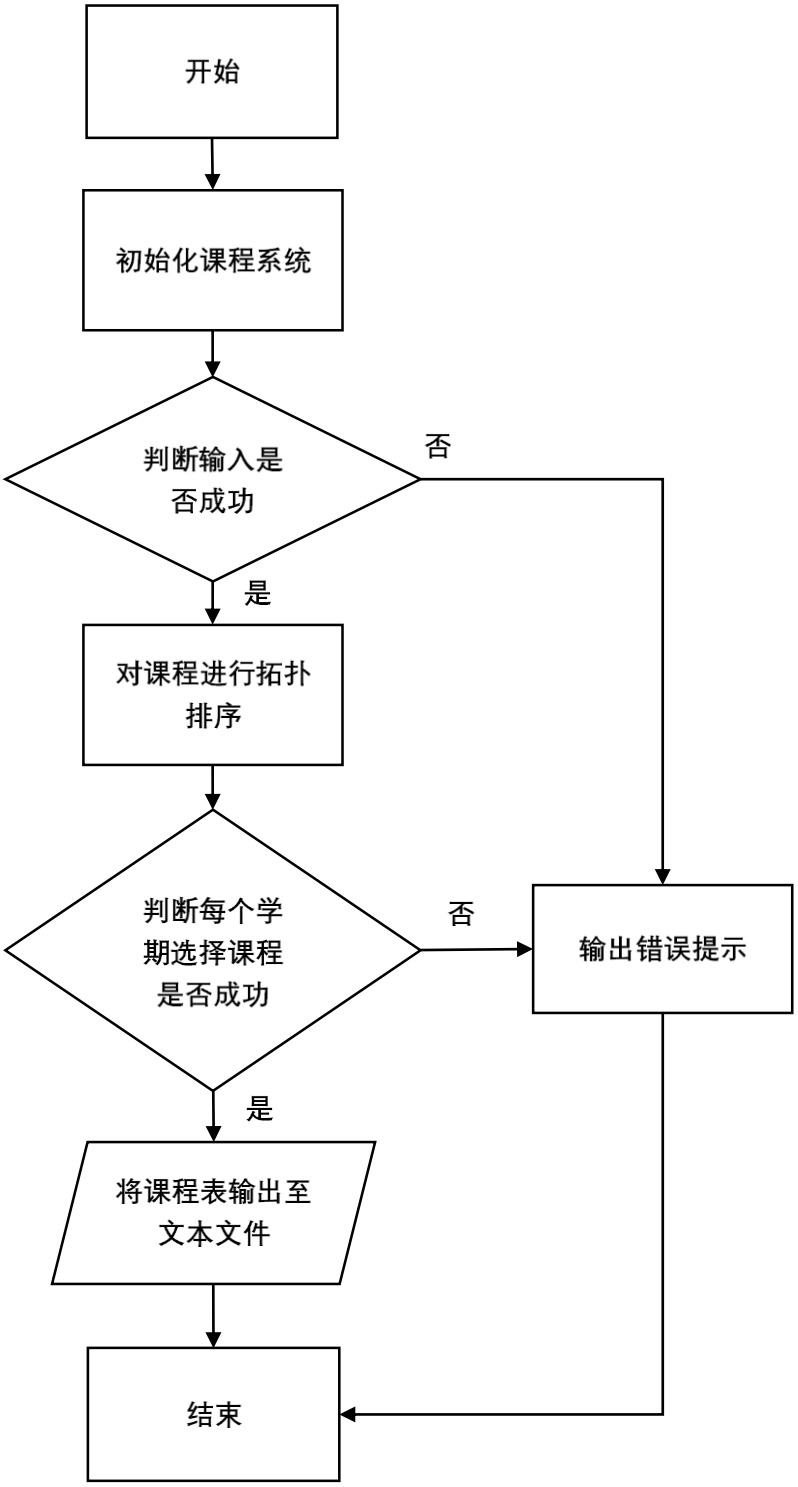
```
bool CourseSystem::outputData(fstream& out)
{
    out.open("out.txt", ios_base::out | ios_base::binary);
    if (out.is_open() == false)
    {
        cout << "文件打开失败，请检查文件是否存在！" << endl;
        return false;
    }
    for (int i = 0; i < 8; i++)
    {
        arrangeCourse(i);
        out << "第" << i + 1 << "学期课表如下:" << endl;
        for (int hour = 0; hour < 10; hour++)
        {
            for (int day = 0; day < 5; day++)
            {
                out << courseTable[day][hour].name << '\t';
                if (courseTable[day][hour].name.size() <= 8)
                {
                    out << '\t';
                }
                courseTable[day][hour].clear();
            }
            out << endl;
        }
        out << endl;
    }
    out.close();
    return true;
}
```

3.5.3 输出到文本文件功能描述

首先，尝试打开要输出到的文件，如果打不开文件，就输出错误提示并停止输入。如果能够打开，就依次对八个学期先安排课程表后再输出，输出结束后关闭文件，功能完成。

3.6 总体功能的实现

3.6.1 总体功能流程图



3.6.2 总体功能核心代码

```
int main()
{
    fstream in, out;
    CourseSystem myCourseSystem;
    if (myCourseSystem.inputData(in))
    {
        myCourseSystem.topologicalSort();
        if (myCourseSystem.selectCourse())
        {
            myCourseSystem.outputData(out);
        }
    }
    return 0;
}
```

3.6.3 总体功能截屏示例

在命令行窗口中：

```
38
5 5 3 6 5 4 3 7
第1学期课表如下:
null            null            null            程序设计基础    null
null            null            null            程序设计基础    null
高等数学        程序设计基础    高等数学        大学语文        null
高等数学        程序设计基础    高等数学        大学语文        null
高等数学        程序设计基础    高等数学        大学语文        null
null            null            null            英语            null
null            null            null            英语            null
计算机组成原理  英语            计算机组成原理  null            null
计算机组成原理  英语            计算机组成原理  null            null
计算机组成原理  英语            计算机组成原理  null            null
```

```
第2学期课表如下:
null            null            null            英语            null
null            null            null            英语            null
数据通信        离散数学        数据通信        离散数学        null
数据通信        离散数学        数据通信        离散数学        null
数据通信        离散数学        数据通信        离散数学        null
null            null            null            null            null
null            null            null            null            null
线性代数        英语            线性代数        计算机文化      null
线性代数        英语            线性代数        计算机文化      null
线性代数        英语            线性代数        计算机文化      null
```

```
第3学期课表如下:
数据结构算法    null            汇编语言        数据结构算法    null
数据结构算法    null            汇编语言        数据结构算法    null
汇编语言        null            null            null            null
汇编语言        null            null            null            null
汇编语言        null            null            null            null
null            null            英语            null            null
null            null            英语            null            null
英语            null            null            null            null
英语            null            null            null            null
英语            null            null            null            null
```

```
第4学期课表如下:
操作系统原理    null            数据库原理      英语            微机原理
操作系统原理    null            数据库原理      英语            微机原理
数据库原理      英语            null            null            null
数据库原理      英语            null            null            null
数据库原理      英语            null            null            null
微机原理        null            编译原理        操作系统原理    null
微机原理        null            编译原理        操作系统原理    null
编译原理        单片机应用      null            null            null
编译原理        单片机应用      null            null            null
编译原理        单片机应用      null            null            null
```

```
第5学期课表如下:
null            null            C#.net          英语            null
null            null            C#.net          英语            null
C#.net          英语            面向对象程序设计 null            null
C#.net          英语            面向对象程序设计 null            null
C#.net          英语            面向对象程序设计 null            null
null            null            计算机网络      null            null
null            null            计算机网络      null            null
计算机网络      Java            null            null            null
计算机网络      Java            null            null            null
计算机网络      Java            null            null            null
```

```
第6学期课表如下:
null            null            ASP程序设计     英语            null
null            null            ASP程序设计     英语            null
ASP程序设计     英语            null            null            null
ASP程序设计     英语            null            null            null
ASP程序设计     英语            null            null            null
null            null            PowerBuilder    null            null
null            null            PowerBuilder    null            null
PowerBuilder    VC++            null            null            null
PowerBuilder    VC++            null            null            null
PowerBuilder    VC++            null            null            null
```

```
第7学期课表如下:
null            null            VB.net          英语            null
null            null            VB.net          英语            null
VB.net          英语            null            null            null
VB.net          英语            null            null            null
VB.net          英语            null            null            null
null            null            JSP程序设计     null            null
null            null            JSP程序设计     null            null
JSP程序设计     null            null            null            null
JSP程序设计     null            null            null            null
JSP程序设计     null            null            null            null
```

```
第8学期课表如下:
算法设计        英语            算法设计        C++Builder      null
算法设计        英语            算法设计        C++Builder      null
数值分析        C++Builder      数值分析        英语            null
数值分析        C++Builder      数值分析        英语            null
数值分析        C++Builder      数值分析        英语            null
普通物理        null            普通物理        Delphi          null
普通物理        null            普通物理        Delphi          null
计算机系统结构  Delphi          计算机系统结构  null            null
计算机系统结构  Delphi          计算机系统结构  null            null
计算机系统结构  Delphi          计算机系统结构  null            null
```

4 测试

4.1 功能测试

4.1.1 基本功能测试

测试用例：

```
38
5 5 5 5 5 5 5 3
c01 程序设计基础 5 0
c02 离散数学 6 0 c01
c03 数据结构算法 4 0 c01 c02
c04 汇编语言 5 0 c01
c05 算法设计 4 0 c03 c04
c06 计算机组成原理 6 0
c07 微机原理 4 0 c03
c08 单片机应用 3 0 c03
c09 编译原理 5 0 c03
c10 操作系统原理 4 0 c03
c11 数据库原理 5 0 c03
c12 高等数学 6 0
c13 线性代数 6 0
c14 数值分析 6 0 c12
c15 普通物理 4 0 c12
c16 计算机文化 3 0
c17 计算机系统结构 6 0 c06
c18 计算机网络 5 0 c03
c19 数据通信 6 0
c20 面向对象程序设计 3 0 c01 c03
c21 Java 3 0 c01 c03
c22 C#.net 5 0 c01 c03
c23 PowerBuilder 5 0 c01 c03
c24 VC++ 3 0 c01 c03
c25 ASP 程序设计 5 0 c01 c03
c26 JSP 程序设计 5 0 c01 c03
c27 VB.net 5 0 c01 c03
c28 Delphi 5 0 c01 c03
c29 C++Builder 5 0 c01 c03
c30 英语 5 1
c30 英语 5 2
c32 英语 5 3
```

c33 英语 5 4
c34 英语 5 5
c35 英语 5 6
c36 英语 5 7
c37 英语 5 8
c38 大学语文 3 1

预期结果：输出课程表，程序正常运行不崩溃。

实验结果：

第1学期课表如下:

null	null	null	程序设计基础	null
null	null	null	程序设计基础	null
高等数学	程序设计基础	高等数学	大学语文	null
高等数学	程序设计基础	高等数学	大学语文	null
高等数学	程序设计基础	高等数学	大学语文	null
null	null	null	英语	null
null	null	null	英语	null
计算机组成原理	英语	计算机组成原理	null	null
计算机组成原理	英语	计算机组成原理	null	null
计算机组成原理	英语	计算机组成原理	null	null

第2学期课表如下:

null	null	null	英语	null
null	null	null	英语	null
数据通信	离散数学	数据通信	离散数学	null
数据通信	离散数学	数据通信	离散数学	null
数据通信	离散数学	数据通信	离散数学	null
null	null	null	null	null
null	null	null	null	null
线性代数	英语	线性代数	计算机文化	null
线性代数	英语	线性代数	计算机文化	null
线性代数	英语	线性代数	计算机文化	null

第3学期课表如下:

数据结构算法	null	数据结构算法	汇编语言	null
数据结构算法	null	数据结构算法	汇编语言	null
数值分析	汇编语言	数值分析	null	null
数值分析	汇编语言	数值分析	null	null
数值分析	汇编语言	数值分析	null	null
null	null	null	英语	null
null	null	null	英语	null
计算机系统结构	英语	计算机系统结构	null	null
计算机系统结构	英语	计算机系统结构	null	null
计算机系统结构	英语	计算机系统结构	null	null

第4学期课表如下:

操作系统原理	null	编译原理	操作系统原理	null
操作系统原理	null	编译原理	操作系统原理	null
编译原理	单片机应用	null	null	null
编译原理	单片机应用	null	null	null
编译原理	单片机应用	null	null	null
微机原理	null	英语	微机原理	null
微机原理	null	英语	微机原理	null
英语	null	null	null	null
英语	null	null	null	null
英语	null	null	null	null

第5学期课表如下:

null	null	计算机网络	英语	null
null	null	计算机网络	英语	null
计算机网络	英语	面向对象程序设计	null	null
计算机网络	英语	面向对象程序设计	null	null
计算机网络	英语	面向对象程序设计	null	null
null	null	数据库原理	null	null
null	null	数据库原理	null	null
数据库原理	Java	null	null	null
数据库原理	Java	null	null	null
数据库原理	Java	null	null	null

第6学期课表如下:

null	null	ASP程序设计	C#.net	null
null	null	ASP程序设计	C#.net	null
ASP程序设计	C#.net	VC++	null	null
ASP程序设计	C#.net	VC++	null	null
ASP程序设计	C#.net	VC++	null	null
null	null	PowerBuilder	英语	null
null	null	PowerBuilder	英语	null
PowerBuilder	英语	null	null	null
PowerBuilder	英语	null	null	null
PowerBuilder	英语	null	null	null

第7学期课表如下:

null	null	C++Builder	VB.net	英语
null	null	C++Builder	VB.net	英语
C++Builder	VB.net	英语	null	null
C++Builder	VB.net	英语	null	null
C++Builder	VB.net	英语	null	null
null	null	Delphi	JSP程序设计	null
null	null	Delphi	JSP程序设计	null
Delphi	JSP程序设计	null	null	null
Delphi	JSP程序设计	null	null	null
Delphi	JSP程序设计	null	null	null

第8学期课表如下:

算法设计	null	英语	普通物理	null
算法设计	null	英语	普通物理	null
英语	null	null	null	null
英语	null	null	null	null
英语	null	null	null	null
普通物理	null	算法设计	null	null
普通物理	null	算法设计	null	null
null	null	null	null	null
null	null	null	null	null
null	null	null	null	null

4.1.2 存在学期课程数较多功能测试

测试用例：

```
38
5 2 3 6 6 4 2 10
c01 程序设计基础 5 0
c02 离散数学 6 0 c01
c03 数据结构算法 4 0 c01 c02
c04 汇编语言 5 0 c01
c05 算法设计 4 0 c03 c04
c06 计算机组成原理 6 0
c07 微机原理 4 0 c03
c08 单片机应用 3 0 c03
c09 编译原理 5 0 c03
c10 操作系统原理 4 0 c03
c11 数据库原理 5 0 c03
c12 高等数学 6 0
c13 线性代数 6 0
c14 数值分析 6 0 c12
c15 普通物理 4 0 c12
c16 计算机文化 3 0
c17 计算机系统结构 6 0 c06
c18 计算机网络 5 0 c03
c19 数据通信 6 0
c20 面向对象程序设计 3 0 c01 c03
c21 Java 3 0 c01 c03
c22 C#.net 5 0 c01 c03
c23 PowerBuilder 5 0 c01 c03
c24 VC++ 3 0 c01 c03
c25 ASP 程序设计 5 0 c01 c03
c26 JSP 程序设计 5 0 c01 c03
c27 VB.net 5 0 c01 c03
c28 Delphi 5 0 c01 c03
c29 C++Builder 5 0 c01 c03
c30 英语 5 1
c30 英语 5 2
c32 英语 5 3
c33 英语 5 4
c34 英语 5 5
c35 英语 5 6
c36 英语 5 7
c37 英语 5 8
c38 大学语文 3 1
```

预期结果：输出课程表，程序正常运行不崩溃。

实验结果：

第1学期课表如下:

null	null	null	程序设计基础	null
null	null	null	程序设计基础	null
高等数学	程序设计基础	高等数学	大学语文	null
高等数学	程序设计基础	高等数学	大学语文	null
高等数学	程序设计基础	高等数学	大学语文	null
null	null	null	英语	null
null	null	null	英语	null
计算机组成原理	英语	计算机组成原理	null	null
计算机组成原理	英语	计算机组成原理	null	null
计算机组成原理	英语	计算机组成原理	null	null

第2学期课表如下:

null	null	英语	null	null
null	null	英语	null	null
离散数学	null	离散数学	null	null
离散数学	null	离散数学	null	null
离散数学	null	离散数学	null	null
null	null	null	null	null
null	null	null	null	null
英语	null	null	null	null
英语	null	null	null	null
英语	null	null	null	null

第3学期课表如下:

数据结构算法	null	英语	null	null
数据结构算法	null	英语	null	null
线性代数	null	线性代数	null	null
线性代数	null	线性代数	null	null
线性代数	null	线性代数	null	null
null	null	数据结构算法	null	null
null	null	数据结构算法	null	null
英语	null	null	null	null
英语	null	null	null	null
英语	null	null	null	null

第4学期课表如下:

微机原理	null	汇编语言	英语	null
微机原理	null	汇编语言	英语	null
数据通信	英语	数据通信	null	null
数据通信	英语	数据通信	null	null
数据通信	英语	数据通信	null	null
null	null	微机原理	null	null
null	null	微机原理	null	null
汇编语言	单片机应用	计算机文化	null	null
汇编语言	单片机应用	计算机文化	null	null
汇编语言	单片机应用	计算机文化	null	null

第5学期课表如下:

操作系统原理	null	计算机网络	编译原理	操作系统原理
操作系统原理	null	计算机网络	编译原理	操作系统原理
计算机网络	编译原理	面向对象程序设计	null	null
计算机网络	编译原理	面向对象程序设计	null	null
计算机网络	编译原理	面向对象程序设计	null	null
null	null	数据库原理	英语	null
null	null	数据库原理	英语	null
数据库原理	英语	null	null	null
数据库原理	英语	null	null	null
数据库原理	英语	null	null	null

第6学期课表如下:

null	null	PowerBuilder	英语	null
null	null	PowerBuilder	英语	null
PowerBuilder	英语	null	null	null
PowerBuilder	英语	null	null	null
PowerBuilder	英语	null	null	null
null	null	C#.net	null	null
null	null	C#.net	null	null
C#.net	Java	null	null	null
C#.net	Java	null	null	null
C#.net	Java	null	null	null

第7学期课表如下:

null	null	英语	null	null
null	null	英语	null	null
英语	null	null	null	null
英语	null	null	null	null
英语	null	null	null	null
null	null	null	null	null
null	null	null	null	null
VC++	null	null	null	null
VC++	null	null	null	null
VC++	null	null	null	null

第8学期课表如下:

算法设计	VB.net	ASP程序设计	C++Builder	算法设计
算法设计	VB.net	ASP程序设计	C++Builder	算法设计
数值分析	C++Builder	数值分析	VB.net	ASP程序设计
数值分析	C++Builder	数值分析	VB.net	ASP程序设计
数值分析	C++Builder	数值分析	VB.net	ASP程序设计
普通物理	JSP程序设计	英语	Delphi	普通物理
普通物理	JSP程序设计	英语	Delphi	普通物理
计算机系统结构	Delphi	计算机系统结构	JSP程序设计	英语
计算机系统结构	Delphi	计算机系统结构	JSP程序设计	英语
计算机系统结构	Delphi	计算机系统结构	JSP程序设计	英语

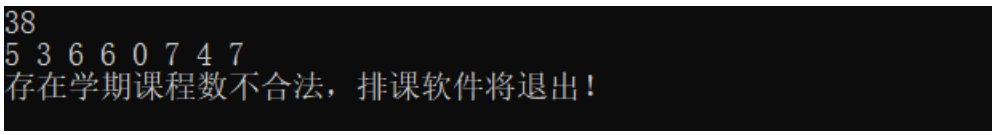
4.2 边界测试

4.2.1 输入总课程数为零

测试用例：0
预期结果：程序给出提示信息，程序运行正常不崩溃。
实验结果：

4.2.2 输入某学期课程数为零

测试用例：
38
5 3 6 6 0 7 4 7
预期结果：程序给出提示信息，程序运行正常不崩溃。
实验结果：



4.3 出错测试

4.3.1 每学期所开的课程数值和与课程总数不相等

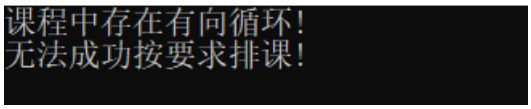
测试用例：
38
5 5 5 5 5 5 5 5
预期结果：程序给出提示信息，程序运行正常不崩溃。
实验结果：

```
38
5 5 5 5 5 5 5 5
每学期所开的课程数值和与课程总数不相等，排课软件将退出！
```

4.3.2 课程中存在有向循环

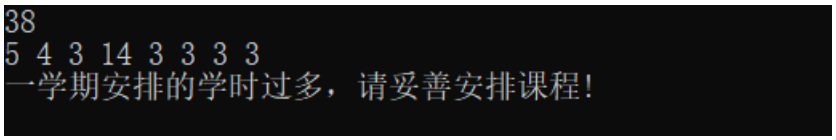
测试用例：
38
5 5 5 5 5 5 5 3
c01 程序设计基础 5 0
c02 离散数学 6 0 c01
c03 数据结构算法 4 0 c01 c02
c04 汇编语言 5 0 c01
c05 算法设计 4 0 c03 c04
c06 计算机组成原理 6 0
c07 微机原理 4 0 c03
c08 单片机应用 3 0 c03
c09 编译原理 5 0 c03
c10 操作系统原理 4 0 c03
c11 数据库原理 5 0 c03
c12 高等数学 6 0 c14//在这里修改添加回路
c13 线性代数 6 0
c14 数值分析 6 0 c12
c15 普通物理 4 0 c12
c16 计算机文化 3 0
c17 计算机系统结构 6 0 c06
c18 计算机网络 5 0 c03
c19 数据通信 6 0
c20 面向对象程序设计 3 0 c01 c03
c21 Java 3 0 c01 c03
c22 C#.net 5 0 c01 c03
c23 PowerBuilder 5 0 c01 c03

c24 VC++ 3 0 c01 c03
c25 ASP 程序设计 5 0 c01 c03
c26 JSP 程序设计 5 0 c01 c03
c27 VB.net 5 0 c01 c03
c28 Delphi 5 0 c01 c03
c29 C++Builder 5 0 c01 c03
c30 英语 5 1
c30 英语 5 2
c32 英语 5 3
c33 英语 5 4
c34 英语 5 5
c35 英语 5 6
c36 英语 5 7
c37 英语 5 8
c38 大学语文 3 1
预期结果：程序给出提示信息，程序正常运行不崩溃。
实验结果：



4.3.3 一学期安排的学时大于最高要求学时

测试用例：
38
5 4 3 14 3 3 3 3
第四学期的课时总数超过 50
预期结果：程序给出提示信息，程序正常运行不崩溃。
实验结果：



4.3.4 输入总课程数不正确

测试用例：测试
预期结果：程序给出提示信息，程序正常运行不崩溃。

实验结果：

```
测试
总课程数不合法，排课软件将退出！
```

测试用例：-3
预期结果：程序给出提示信息，程序正常运行不崩溃。
实验结果：

```
-3
总课程数不合法，排课软件将退出！
```

测试用例：as
预期结果：程序给出提示信息，程序正常运行不崩溃。
实验结果：

```
as
总课程数不合法，排课软件将退出！
```

4.3.5 输入某学期课程数不正确

测试用例：
38
5 5 5 测试 5 5 5 3
预期结果：程序给出提示信息，程序正常运行不崩溃。
实验结果：

```
38
5 5 5 测试 5 5 5 3
存在学期课程数不合法，排课软件将退出！
```

测试用例：
38
5 5 -6 5 5 5 5 3
预期结果：程序给出提示信息，程序正常运行不崩溃。
实验结果：

```
38
5 5 -6 5 5 5 5 3
存在学期课程数不合法，排课软件将退出！
```

测试用例：

38

5 5 7 3 wasd 5 5 3

预期结果：程序给出提示信息，程序正常运行不崩溃。

实验结果：