Tongji University

Data Structure Course Curriculum Design Document

Task9: Course scheduling software

Author:
嘉杰 李
Jiajie Li

Supervisor:
颖 张
Prof. Ying Zhang

A companion use documentation for a program submitted in fulfillment of the requirements for the Data Structure curriculum design

in the

School of Software Engineering, Tongji

December 27, 2019

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”

Dave Barry

Contents

1 Analysis 1
1.1 Background Analysis 1
1.2 Functional Analysis 1

2 Design 3
2.1 Data structure design 3
2.2 Member function design 3

3 Implementation 7
3.1 Read from file Function 7
3.1.1 Read from file core code 7
3.1.2 Read from file flowchart 8
3.2 TopoSearch Function 8
3.2.1 TopoSearch code 8
3.2.2 TopoSearch flowchart 10
3.3 Insert course Function 10
3.3.1 Insert course code 10
3.3.2 Insert course flowchart 12
3.4 File output Function 12
3.4.1 File output code 12
3.4.2 File output flowchart 14

4 Test 15
4.1 Function Tests 15
4.1.1 Normal 15

A Frequently Asked Questions 19
A.1 Why this document looks so sparse? 19
A.2 Why is there header files in this project? 19

Chapter 1

Analysis

1.1 Background Analysis

From elementary school to university, the curriculum accompanies students' academic life. Facing a variety of courses, scheduling has become a headache for teachers of the Academic Affairs Office. Especially in universities, there are general and professional courses, elective courses and compulsory courses. And many courses have a relationship of successive study. How to arrange the class time of each teacher to ensure that teachers' resources are allocated reasonably and not conflict between general courses and professional courses. At the same time, it is also necessary to enable each major student to reasonably arrange his or her schoolwork, so that students can arrange their compulsory and elective courses in a semester without conflict, and can also study the courses that require prerequisite courses. I have taken the pre-course before to ensure that I can understand. At the same time, a good class schedule can make teachers and classmates arrange their work and rest in the teaching process, which can optimize the quality of the teachers 'teaching courses and the efficiency of the students' learning courses. However, for Tongji University, there are about 6,500 courses. To allow reasonable time for each course, perfection cannot be achieved by hand alone. The use of computers can greatly assist the teacher's work, and the discharged schedule can greatly avoid conflicts in course resources. At the same time, by setting priority rules, teachers and students can arrange their own schedules. To achieve the effect of improving the quality of teaching. Therefore, it is very valuable to design a software that can schedule lessons. At the same time, such a project can be extended to the arrangement of various work tasks, so that employees can achieve the highest efficiency while reasonable work and rest.

1.2 Functional Analysis

As a course scheduling system, it should be possible to store and process the number of courses offered per semester, course number, course name, weekly hours, designated start semester, prerequisites, and produce a schedule for each semester.

This timetable can meet the requirements of the number of courses and input required for each semester. The arrangement of course hours is consistent with the input. There is no time conflict between courses. At the same time, for semesters, no courses can be started before the predecessor courses, and the prerequisite must be satisfied. At the same time, appropriate prompts should be given for unreasonable situations. At the same time, the project should have file input and output functions: input course list, and output the curriculum for each semester.

Chapter 2

Design

2.1 Data structure design

As described above, there should be an array with a customized data type to store all the courses read from the file. Also, some multidimensional arrays should be use to store results or middle results during the course inserting process.

2.2 Member function design

Vector Class:

```
template <class T>
class Vector {
public:
   Vector(int size=0):theSize(size),theCapacity(size+SPACE_CAPACITY){ data = new
       T[theCapacity];}
   ~Vector(void) { delete[] data;}
   Vector& operator=(Vector& other){
      if (this == &other)
         return *this;
      else {
         delete[]data;
         theSize = other.theSize;
         theCapacity = other.theCapacity;
         this->data = new T[theCapacity];

         for (int i = 0; i < theSize; ++i) {
            data[i] = other.data[i];
         }
         return *this;
      }
   }
   Vector(Vector& other) :theSize(0),theCapacity(0),data(NULL){ *this=other; }


   void reServe(int newCapacity) {
      if (newCapacity <= theCapacity)
```

```
            return;

        T *temp = data;
        data = new T[newCapacity];
        theCapacity=newCapacity;
        //std::cout << newCapacity << std::endl;
        for (int i = 0; i < theSize; ++i)
            data[i] = temp[i];
        delete[] temp;
    }

    void push_back(T val) {
        if (theSize == theCapacity)
            reServe(2 * theCapacity + 1);
        data[theSize++] = val;
    }
    bool pop_back(){
        if (!theSize) {
            std::cerr << "Fail to pop back, the vector is empty!\n";
            return false;
        }
        theSize--;
        return true;
    }
    bool Delete(int ind){
        if (ind<0||ind>=theSize){
            std::cerr << "Fail to delete, the ind is out of range!\n";
            return false;
        }
        for (int i=ind;i<theSize-1;++i) data[i]=data[i+1];
        return pop_back();
    }
    T& operator[] (int n) {return data[n];}
    T *data;
    int size() const {return theSize;}
private:
    const int SPACE_CAPACITY = 16;
    int theCapacity;
    int theSize;
};
```

Course struct

```
struct course{
    string code;
    string name;
    string s_hours, s_Semester;
    int hours;
    int Semester;
    Vector<string> preCourse;
    friend ostream& operator<< (ostream& out, course& c) {
        cout << c.code << " " << c.name << " " << c.hours << " " << c.Semester << "
             ";
        for (int i=0; i<c.preCourse.size(); ++i) cout << c.preCourse[i] << " ";
        cout << endl;
        return out;
    } //for test only
    friend ifstream& operator>> (ifstream& out, course& c) {
        string buffer;
        string temp;
        getline(inf, buffer);
        stringstream ss;
        ss << buffer;
        ss >> c.code >> c.name >> c.s_hours >> c.s_Semester;
        stringstream check1(c.s_hours), check2(c.s_Semester);
        if (!(check1>>c.hours)||!(check2>>c.Semester)) {
            cerr << "Invalid Input Format! Plz check again!" << endl;
            exit(1);
        }
        while (ss >> temp) c.preCourse.push_back(temp);
        //cout << buffer;
        return out;
    }
};
```

Chapter 3

Implementation

3.1 Read from file Function

3.1.1 Read from file core code

```
void info() {
    cout << endl;
    cout<<setfill('=')<<setw(70)<<" "<<endl;
    cout<<setfill(' ')<<"=="<<setw(15)<<""<<setw(50)<< left << "Welcome to the
         Curriculum software"<< setw(2)<< right <<"=="<<endl;
    cout<<setfill('=')<<setw(70)<<" "<<endl<<endl;
    cout << "Do you want to use the default input file path? (Y/n)";
    cin >> temp;
    if (temp == "n") {
        cout << "Please enter the input file path: " ;
        cin >> INPUT_PATH;
        inf.open(INPUT_PATH);
        if (!inf) {
            cerr << endl << "The current input file path is incorrect and the
                 default path is used instead. " <<endl;
            INPUT_PATH="../input.txt";
            inf.open(INPUT_PATH);
        }
    }
}
```

3.1.2 Read from file flowchart

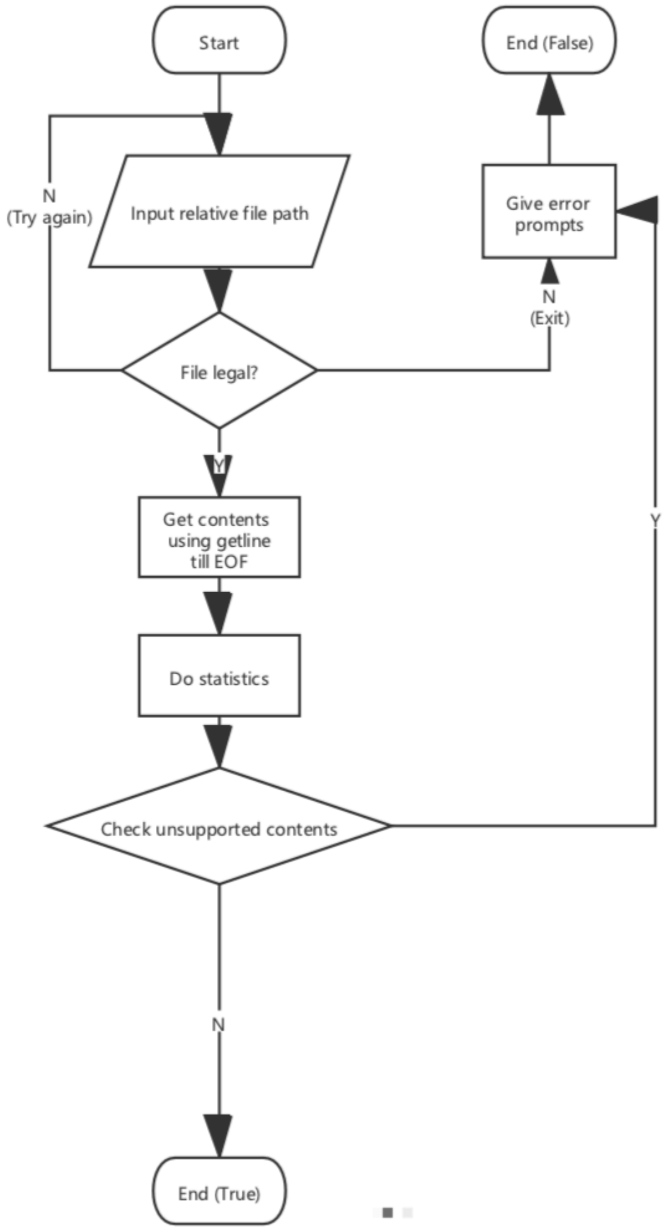


Figure 3.1: Read from file flowchart

3.2 TopoSearch Function

3.2.1 TopoSearch code

```
int topoSearch(int sem, int count){
    if (count == courses.size()) output();
    if (sem == maxSem +1) return 0;
    if (curTotalHours + (maxSem-sem+1)*50 < totalHours) return 0;
    if (hours[sem]>50) return 0;

    for (int i=0; i<courses.size(); ++i)
        if (courses[i].Semester==0 || courses[i].Semester==sem)
        if (ind[i]==0 && !used[i] && sem>=atleast[i]) {
            int tempLeast[1000];
            for (int j=0; j<nex[i].size(); ++j) --ind[nex[i][j]], tempLeast[nex[i][
                j]]=atleast[nex[i][j]], atleast[nex[i][j]]=sem+1;
            used[i] = true; curTotalHours+=courses[i].hours; hours[sem]+=courses[i
                ].hours;
            semCount[sem]-=(courses[i].Semester==sem)?1:0;
            topoOrder[sem].push_back(courses[i]);
            topoSearch(sem, count+1);
            used[i] = false; curTotalHours-=courses[i].hours; hours[sem]-=courses[i
                ].hours;
            semCount[sem]+=(courses[i].Semester==sem)?1:0;
            topoOrder[sem].Delete(topoOrder[sem].size() - 1);
            for (int j=0; j<nex[i].size(); ++j) ++ind[nex[i][j]], atleast[nex[i][j
                ]]=tempLeast[nex[i][j]];
        }
    if (semCount[sem]==0) topoSearch(sem+1, count);
    return 0;
}
```

3.2.2 TopoSearch flowchart

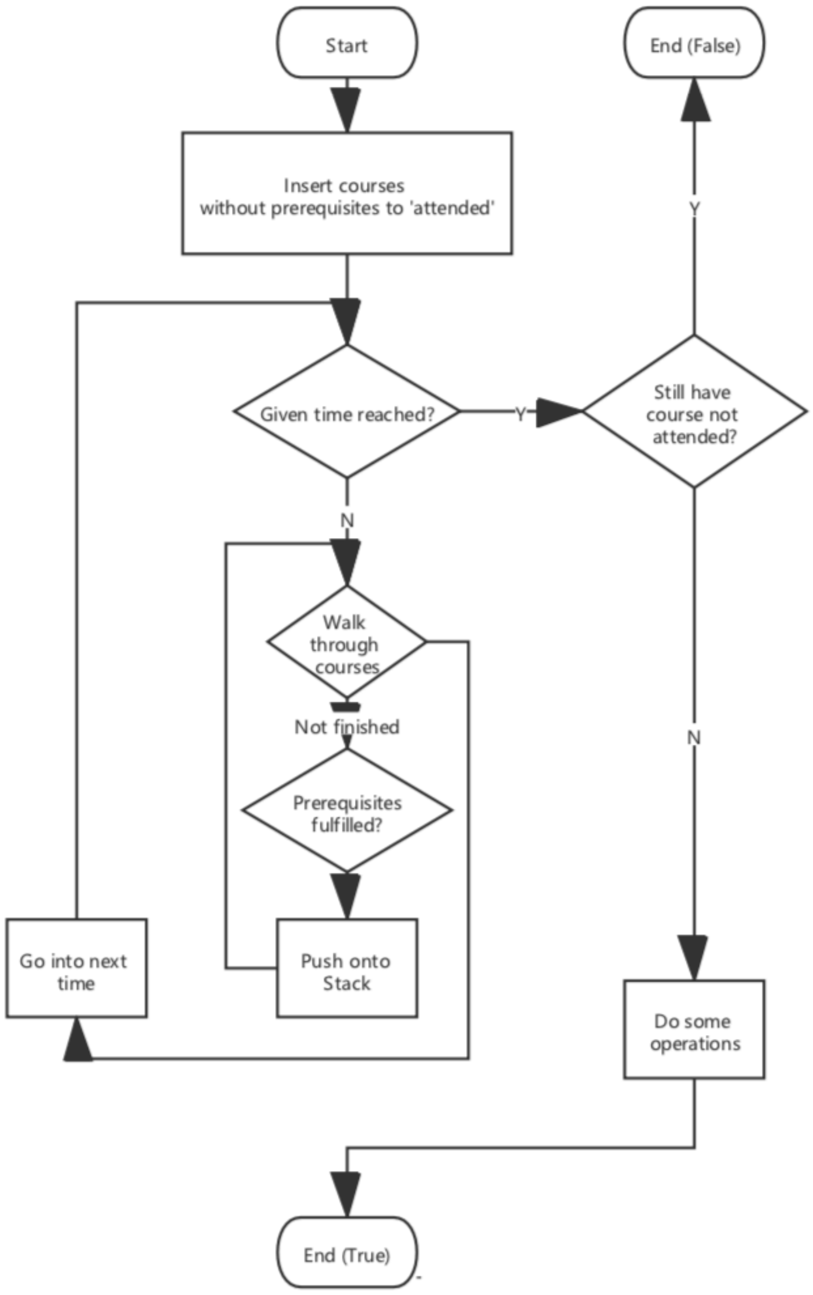


Figure 3.2: TopoSearch flowchart

3.3 Insert course Function

3.3.1 Insert course code

```
or (int i=1; i<=maxSem; ++i) {
    memset(&table[0][0],0,sizeof(table));
    memset(avai,0,sizeof(avai));
```

```
        if (i<=3) outf <<endl<< "The " << semName[i] << " Semester: "<<endl;
        else outf <<endl<< "The " << i << "th Semester: "<<endl ;
        if (i==6) {

        }
        for (int j = 0; j < topoOrder[i].size(); ++j) {
                for (int d=1; d<=11; ++d)
                for (int t=4; t>=1; --t) {
                    if (!avai[jumpDay[d]][t] && weektime[jumpDay[d]][t] == 3 &&
                         topoOrder[i][j].hours >= 3) {
                        avai[jumpDay[d]][t] = true;
                        topoOrder[i][j].hours -= 3;
                        for (int k = 0; k < weektime[jumpDay[d]][t]; ++k) table[jumpDay[
                            d]][jumpTime[t] + k] = j + 1;
                        break;
                    }

                    if (!avai[jumpDay[d]][t] && weektime[jumpDay[d]][t] == 2 &&
                         topoOrder[i][j].hours >= 2) {
                        avai[jumpDay[d]][t] = true;
                        topoOrder[i][j].hours -= 2;
                        for (int k = 0; k < weektime[jumpDay[d]][t]; ++k) table[jumpDay[
                            d]][jumpTime[t] + k] = j + 1;
                        break;
                    }
                }
        }
        for (int j = 0; j < topoOrder[i].size(); ++j)
            while (topoOrder[i][j].hours>0)
                for (int k = 1; k<=50; ++k)
                    if (table[(k/10)+1][k%10+1]==0) {
                        table[(k/10)+1][k%10+1]=j;
                        --topoOrder[i][j].hours;
                        break;
                    }
```

3.3.2 Insert course flowchart

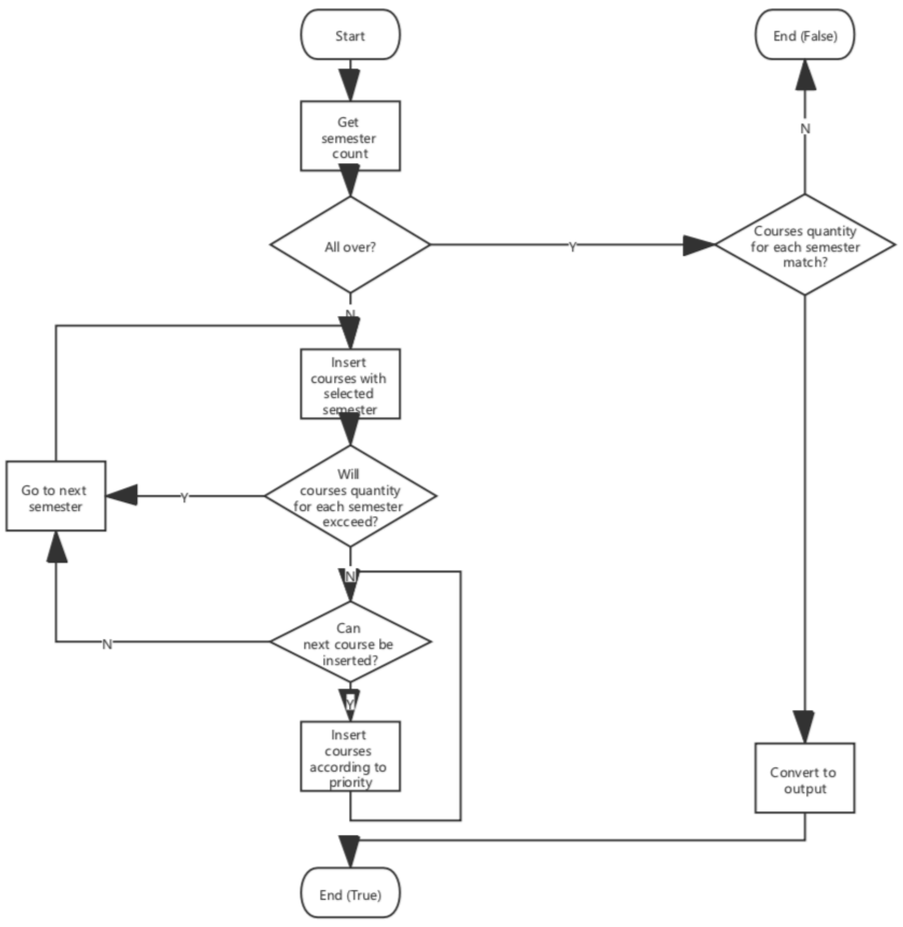

Start
Get semester count
All over?
Y
N
Courses quantity for each semester match?
End (False)
Insert courses with selected semester
Will courses quantity for each semester excceed?
Go to next semester
Can next course be inserted?
Insert courses according to priority
Convert to output
End (True)

Figure 3.3: Insert course flowchart

3.4 File output Function

3.4.1 File output code

```
    for (int p=1; p<=10; ++p) {
        for (int j = 1; j <= 5; ++j)
            if (table[j][p]==0) outf << setw(5) << setiosflags(ios::left) << "无  ";
            else outf << setw(5) << setiosflags(ios::left) << topoOrder[i][table[j][p
                ]-1].name << "  ";
        outf << endl;
    }
    outf << endl;
}
cout << endl << "The Curriculum has been saved! " <<endl;
inf.close();
outf.close();
```

```
exit(0);
```

3.4.2 File output flowchart

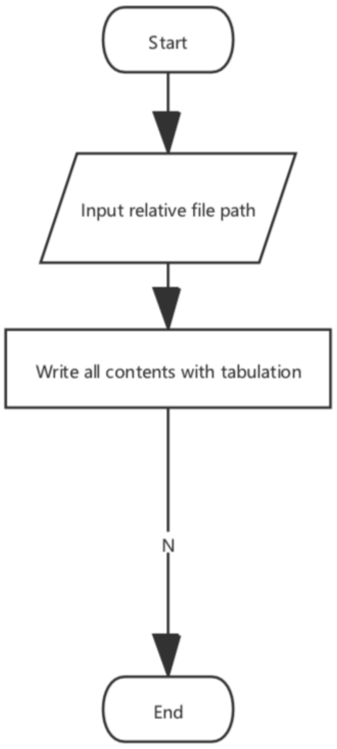


Figure 3.4: File output flowchart

Chapter 4

Test

4.1 Function Tests

4.1.1 Normal

Program Input: 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

c221 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

c31 英语 5 2

c32 英语 5 3

c33 英语 5 4

c34 英语 5 5

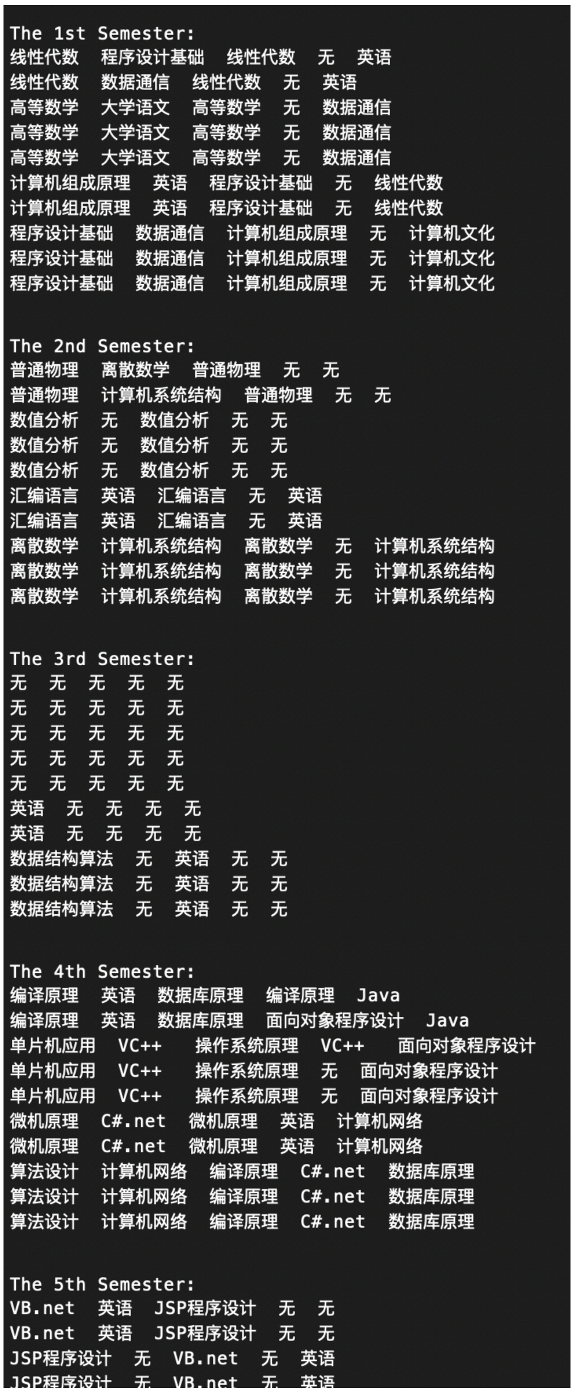
c35 英语 5 6

c36 英语 5 7

c37 英语 5 8

c38 大学语文 3 1

Program Result:



```
The 1st Semester:
线性代数  程序设计基础  线性代数  无  英语
线性代数  数据通信  线性代数  无  英语
高等数学  大学语文  高等数学  无  数据通信
高等数学  大学语文  高等数学  无  数据通信
高等数学  大学语文  高等数学  无  数据通信
计算机组成原理  英语  程序设计基础  无  线性代数
计算机组成原理  英语  程序设计基础  无  线性代数
程序设计基础  数据通信  计算机组成原理  无  计算机文化
程序设计基础  数据通信  计算机组成原理  无  计算机文化
程序设计基础  数据通信  计算机组成原理  无  计算机文化

The 2nd Semester:
普通物理  离散数学  普通物理  无  无
普通物理  计算机系统结构  普通物理  无  无
数值分析  无  数值分析  无  无
数值分析  无  数值分析  无  无
数值分析  无  数值分析  无  无
汇编语言  英语  汇编语言  无  英语
汇编语言  英语  汇编语言  无  英语
离散数学  计算机系统结构  离散数学  无  计算机系统结构
离散数学  计算机系统结构  离散数学  无  计算机系统结构
离散数学  计算机系统结构  离散数学  无  计算机系统结构

The 3rd Semester:
无  无  无  无  无
无  无  无  无  无
无  无  无  无  无
无  无  无  无  无
无  无  无  无  无
英语  无  无  无  无
英语  无  无  无  无
数据结构算法  无  英语  无  无
数据结构算法  无  英语  无  无
数据结构算法  无  英语  无  无

The 4th Semester:
编译原理  英语  数据库原理  编译原理  Java
编译原理  英语  数据库原理  面向对象程序设计  Java
单片机应用  VC++   操作系统原理  VC++   面向对象程序设计
单片机应用  VC++   操作系统原理  无  面向对象程序设计
单片机应用  VC++   操作系统原理  无  面向对象程序设计
微机原理  C#.net  微机原理  英语  计算机网络
微机原理  C#.net  微机原理  英语  计算机网络
算法设计  计算机网络  编译原理  C#.net  数据库原理
算法设计  计算机网络  编译原理  C#.net  数据库原理
算法设计  计算机网络  编译原理  C#.net  数据库原理

The 5th Semester:
VB.net  英语  JSP程序设计  无  无
VB.net  英语  JSP程序设计  无  无
JSP程序设计  无  VB.net  无  英语
JSP程序设计  无  VB.net  无  英语
```

Figure 4.1: Screenshots

Appendix A

Frequently Asked Questions

A.1 Why this document looks so sparse?

This document is built using LaTeX and is arranged in book format. There may be blank pages before and after each chapter.

A.2 Why is there header files in this project?

I wrote basic stl header files, such as vector.h, etc. Except for the header files I wrote, the only header files used in all projects are iostream and string.h