项目说明文档

数据结构课程设计

——排课软件

同济大学

Tongji University

目 录

[1 分析 1](#_Toc21032)

[1.1 背景分析 1](#_Toc13854)

[1.2 功能分析 1](#_Toc26810)

[2 设计 1](#_Toc6240)

[2.1 数据结构设计 1](#_Toc18986)

[2.2 类结构设计 1](#_Toc27912)

[2.3 成员与操作设计 2](#_Toc11369)

[3 实现 3](#_Toc20704)

[3.1 构建拓扑功能的实现 3](#_Toc27631)

[3.1.1 构建拓扑功能流程图 3](#_Toc24536)

[3.1.2 输入功能核心代码 3](#_Toc24013)

[3.2 构建链表功能的实现 7](#_Toc7182)

[3.2.1 构建链表功能流程图 7](#_Toc10584)

[3.2.2 构建链表功能核心代码 7](#_Toc31811)

[3.3 筛选功能的实现 9](#_Toc18574)

[3.3.1 筛选功能流程图 9](#_Toc4260)

[3.3.2 筛选功能核心代码 9](#_Toc15347)

[3.4 输出功能的实现 11](#_Toc25214)

[3.4.1 输出功能流程图 11](#_Toc7144)

[3.4.2 输出功能核心代码 11](#_Toc13623)

[3.7总体系统实现 21](#_Toc7246)

[3.7.1 总体系统流程图 21](#_Toc19101)

[4 测试 21](#_Toc728)

[4.1 功能测试 22](#_Toc29156)

[4.2 边界测试 23](#_Toc31226)

[4.2.1 总课数与学期课数不匹配 23](#_Toc19640)

[4.2.2 学期课数错误 24](#_Toc4977)

# 1 分析

## 1.1 背景分析

排课系统是一个学校不可缺少的部分，它对于学校的管理者和学生来说都至关重要，所以一个良好的排课系统应该能够为用户提供充足的信息和功能。随着学生数量和考试数量的日益庞大，如何管理如此庞大的数据显得极为复杂，传统的手工管理工作量大且容易出错，于是我们可以考虑通过计算机辅助我们。

## 1.2 功能分析

作为一个简易的排课系统，首先应该有的功能就是输入学生应该学习的所有课程的总课表。其次，各个课程中有先后学习的关系，根据数据结构的对应相关知识，我们应该建立一个拓扑排序，还要一个链表用于记录该学期我们可以选的所有课程，一个选出该学期要选的课程函数，最后是一个输出到文本的输出函数。

综上所言，我们该程序需要一个构建拓扑功能，一个构建链表功能，一个筛选功能，一个输出功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求大量的增加、删除、修改操作，而链表进行增加、删除等操作十分简便，因此考虑使用链表数据结构。同时，为了实现简易，在第一个结点之前附加一个头结点，这样就使得增加或者删除头结点与处理其他结点方法相同，使得程序简洁。

## 2.2 类结构设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（date）与链表类（list），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。

## 2.3 成员与操作设计

**链表结点类（date）**

class date //记录先驱点所连接的所有后继点

{

public:

date \* link;

string classnumber;

string classname;

int put;

date() { link = NULL; put = 0; }

};

**链表类（lesson）**

class lesson //记录所有的课程的信息

{

public:

string classnumber; //课程编号

string classname; //课程名

int hours; //一周的课时

int term; //开课的时期

string beforeclass1; //需要所学的课程1

string beforeclass2; //需要所学的课程2

string beforeclass3; //需要所学的课程3

int used; //是否被先使用过

int out; //重要程度（越是越多的课程的先驱课程，重要性越高）

int must; //必须程度（若是有学期规定的课程的先行课，其先行课就拥有必须性）

int putintable;//有没有进入可选课表中

date \* link;//链接date类

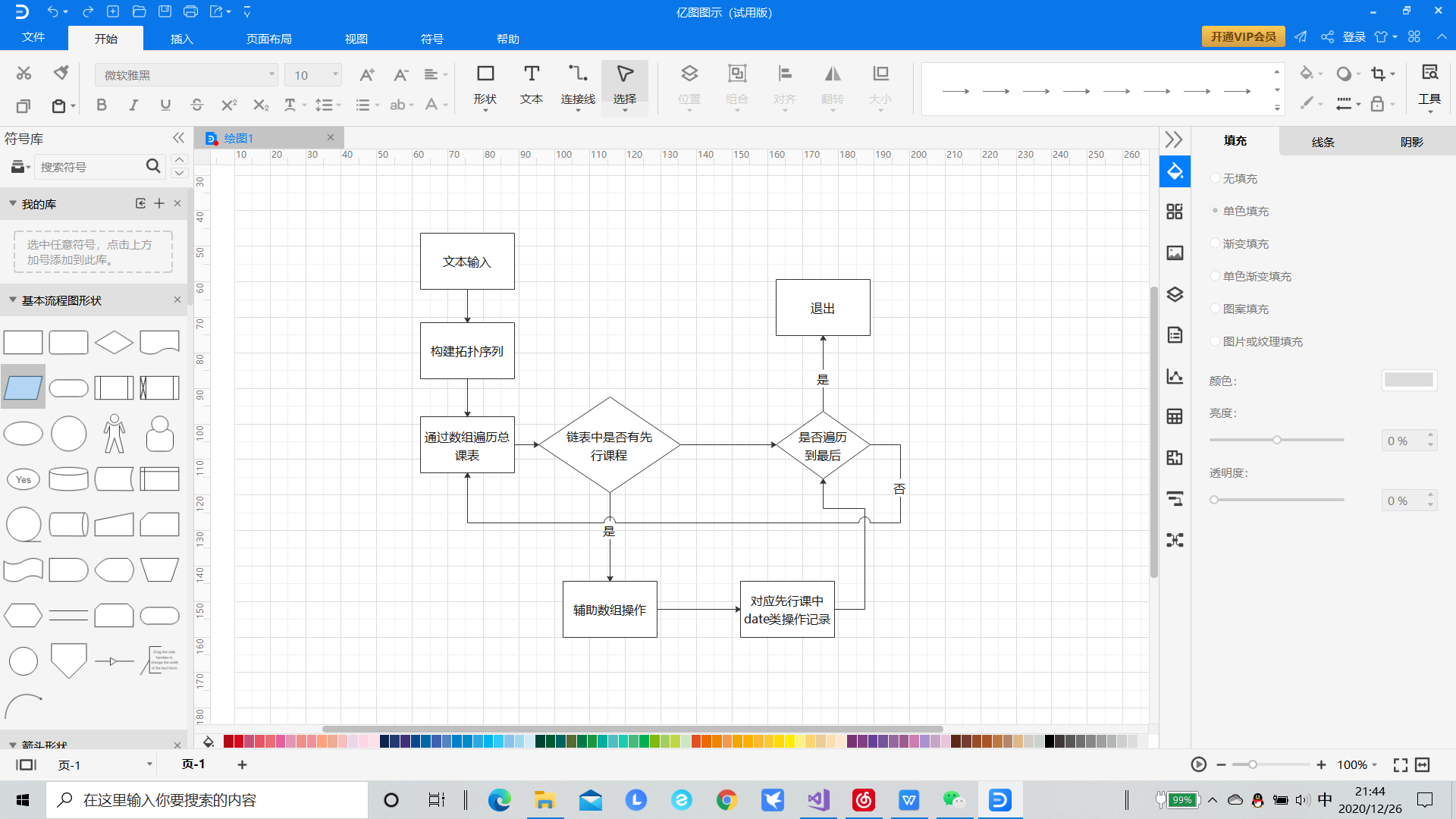
lesson \* tablelink;//链接自己类，用于构建可选课表

## };

# 3 实现

## 3.1 构建拓扑功能的实现

### 3.1.1 构建拓扑功能流程图



### 3.1.2 输入功能核心代码

文本输入阶段

if (infile.is\_open() != 1)

{

cout << "文件没有成功打开" << endl;

}

else

{

for (int n = 0; n < totalnumber; n++)

{

infile >> totalclass[n].classnumber;

infile >> totalclass[n].classname;

infile >> totalclass[n].hours;

infile >> totalclass[n].term;

infile >> totalclass[n].beforeclass1;

infile >> totalclass[n].beforeclass2;

infile >> totalclass[n].beforeclass3;

}

}

构建拓扑和完善辅助数组阶段

// 开始构建拓扑排序的链表

for (int n = 0; n < totalnumber; n++) //改变辅助数组和每个totalclass的date域链接

{

if (totalclass[n].beforeclass1 != "-1")

{

for (int m = 0; m < totalnumber; m++)

{

if (totalclass[m].classnumber == totalclass[n].beforeclass1)

{

count[n]++;

date \*p = new date;

p->classnumber = totalclass[n].classnumber;

p->link = NULL;

if (totalclass[m].link == NULL)

{

totalclass[m].link = p;

}

else

{

date\* head;

head = totalclass[m].link;

for (;;)

{

if (head->link == NULL) { break; }

head = head->link;

}

head->link = p;

}

totalclass[m].out++;

if (totalclass[n].term != 0) { totalclass[m].must++; }

break;

}

}

}

if (totalclass[n].beforeclass2 != "-1")

{

for (int m = 0; m < totalnumber; m++)

{

if (totalclass[m].classnumber == totalclass[n].beforeclass2)

{

count[n]++;

date \*p = new date;

p->classnumber = totalclass[n].classnumber;

p->link = NULL;

if (totalclass[m].link == NULL)

{

totalclass[m].link = p;

}

else

{

date\* head;

head = totalclass[m].link;

for (;;)

{

if (head->link == NULL) { break; }

head = head->link;

}

head->link = p;

}

totalclass[m].out++;

if (totalclass[n].term != 0) { totalclass[m].must++; }

break;

}

}

}

if (totalclass[n].beforeclass3 != "-1")

{

for (int m = 0; m < totalnumber; m++)

{

if (totalclass[m].classnumber == totalclass[n].beforeclass3)

{

count[n]++;

date \*p = new date;

p->classnumber = totalclass[n].classnumber;

p->link = NULL;

if (totalclass[m].link == NULL)

{

totalclass[m].link = p;

}

else

{

date\* head;

head = totalclass[m].link;

for (;;)

{

if (head->link == NULL) { break; }

head = head->link;

}

head->link = p;

}

totalclass[m].out++;

if (totalclass[n].term != 0) { totalclass[m].must++; }

break;

}

}

}

}

## 3.2 构建链表功能的实现

### 3.2.1 构建链表功能流程图

### 

### 3.2.2 构建链表功能核心代码

//开始构建一个链表用于存放可以优先输出的课程

lesson \* tablehead = new lesson;

tablehead->tablelink=NULL;

int tablenumber = 0;

### void putin(lesson \*&tablehead,lesson\*a, int &tablenumber);//放入函数，按out从大到小进行排序

void putin(lesson \*&tablehead, lesson\*a, int &tablenumber)

{

lesson \* q = a; q->putintable = 0;

if (tablehead->tablelink == NULL) { tablehead->tablelink = q; tablenumber++; }

else

{

if (tablenumber == 1)

{

if (tablehead->tablelink->out > q->out)

{

tablehead->tablelink->tablelink = q;

}

else

{

q->tablelink = tablehead->tablelink;

tablehead->tablelink = q;

}

tablenumber++;

}

//

else

{

lesson \* b = tablehead;

for (int n = 0; n < tablenumber - 1; n++)

{

if (q->out > b->tablelink->out) { q->tablelink = b->tablelink; b->tablelink = q; tablenumber++; break; }

if (b->tablelink->out >= q->out && b->tablelink->tablelink->out <= q->out)

{

q->tablelink = b->tablelink->tablelink;

b->tablelink->tablelink = q; tablenumber++; break;

}

if (n == tablenumber - 2 && q->out < b->tablelink->tablelink->out)

{

b->tablelink->tablelink->tablelink = q; tablenumber++; break;

}

b = b->tablelink;

}

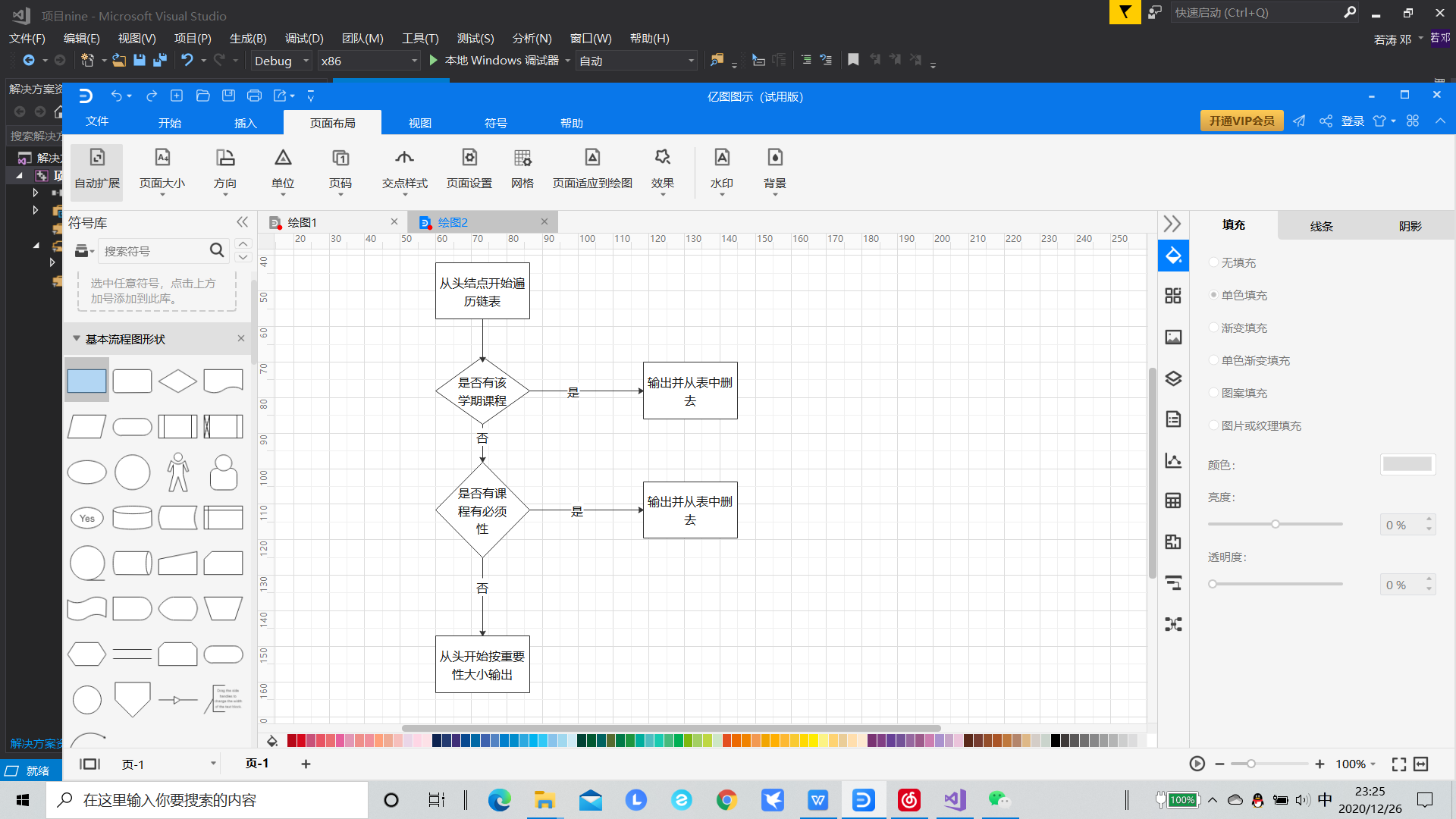
}

}

}

## 3.3 筛选功能的实现

### 3.3.1 筛选功能流程图



### 3.3.2 筛选功能核心代码

int putout(lesson \*&tablehead, int time, int&tablenumber, lesson \*&outlink)//先进行学期判断，再进行must判断，最后按out的大小顺序进行输出

{

lesson \* b = tablehead;

for (int n = 0; n < tablenumber; n++)

{

if (b->tablelink->term == time)

{

outlink = b->tablelink;

outlink->used = 1;

b->tablelink = b->tablelink->tablelink;

outlink->tablelink = NULL;

tablenumber--; return 1;

}

else { b = b->tablelink; }

}

b = tablehead;

for (int n = 0; n < tablenumber; n++)

{

if (b->tablelink->must != -1 )

{

outlink = b->tablelink;

outlink->used = 1;

b->tablelink = b->tablelink->tablelink;

outlink->tablelink = NULL;

tablenumber--; return 2;

}

else { b = b->tablelink; }

}

b = tablehead;

for (int n = 0; n < tablenumber ; n++)

{

if (b->tablelink->term == 0)

{

outlink = b->tablelink;

outlink->used = 1;

b->tablelink = b->tablelink->tablelink;

outlink->tablelink = NULL;

tablenumber--; return 3;

}

else { b = b->tablelink; }

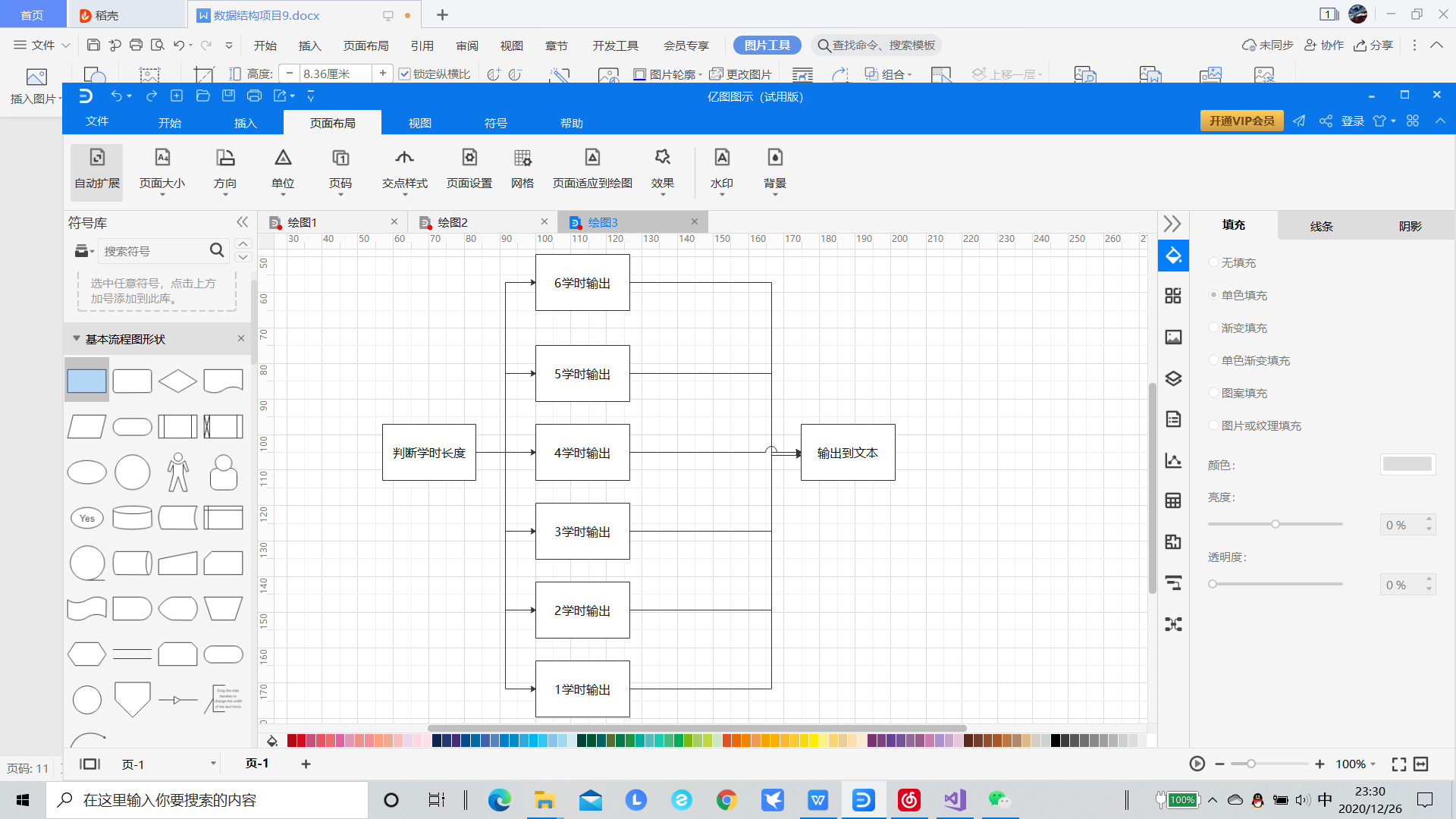
}

return 0;

}

## 3.4 输出功能的实现

### 3.4.1 输出功能流程图



### 3.4.2 输出功能核心代码

void classtablesetup(lesson\* head,int number)

{

ofstream outfile;

outfile.open("2.txt",ios::app);

int remainder, result;

remainder = result = 0;

date timetable[5][10];

for (int n = 0; n < 5; n++)

{

for (int m = 0; m < 10; m++)

{

timetable[n][m].put = 0;

}

}

for (int y = 0; y < number; y++)

{

switch (head->tablelink->hours)

{

case 6:

{

int last = 0;

for (int n = 0; n < 5; n++)

{

if (timetable[n][2].put == 0)

{

last = n;

timetable[n][2].put = timetable[n][3].put = timetable[n][4].put = 1;

timetable[n][2].classname = timetable[n][3].classname = timetable[n][4].classname = head->tablelink->classname;

break;

}

if (timetable[n][7].put == 0)

{

last = n;

timetable[n][7].put = timetable[n][8].put = timetable[n][9].put = 1;

timetable[n][7].classname = timetable[n][8].classname = timetable[n][9].classname = head->tablelink->classname;

break;

}

}

if (last < 2)

{

for (int n = last + 2; n < 5; n++)

{

if (timetable[n][2].put == 0)

{

timetable[n][2].put = timetable[n][3].put = timetable[n][4].put = 1;

timetable[n][2].classname = timetable[n][3].classname = timetable[n][4].classname = head->tablelink->classname;

break;

}

if (timetable[n][7].put == 0)

{

timetable[n][7].put = timetable[n][8].put = timetable[n][9].put = 1;

timetable[n][7].classname = timetable[n][8].classname = timetable[n][9].classname = head->tablelink->classname;

break;

}

}

}

else

{

if (last == 2)

{

for (int n = last - 2; n < 5; n = n + 4)

{

if (timetable[n][2].put == 0)

{

timetable[n][2].put = timetable[n][3].put = timetable[n][4].put = 1;

timetable[n][2].classname = timetable[n][3].classname = timetable[n][4].classname = head->tablelink->classname;

break;

}

if (timetable[n][7].put == 0)

{

timetable[n][7].put = timetable[n][8].put = timetable[n][9].put = 1;

timetable[n][7].classname = timetable[n][8].classname = timetable[n][9].classname = head->tablelink->classname;

break;

}

}

}

else

{

for (int n = last - 2; n >= 0; n = n--)

{

if (timetable[n][2].put == 0)

{

timetable[n][2].put = timetable[n][3].put = timetable[n][4].put = 1;

timetable[n][2].classname = timetable[n][3].classname = timetable[n][4].classname = head->tablelink->classname;

break;

}

if (timetable[n][7].put == 0)

{

timetable[n][7].put = timetable[n][8].put = timetable[n][9].put = 1;

timetable[n][7].classname = timetable[n][8].classname = timetable[n][9].classname = head->tablelink->classname;

break;

}

}

}

}

break;

}

case 5:

{

int last = 0;

for (int n = 0; n < 5; n++)

{

if (timetable[n][2].put == 0)

{

last = n;

timetable[n][2].put = timetable[n][3].put = timetable[n][4].put = 1;

timetable[n][2].classname = timetable[n][3].classname = timetable[n][4].classname = head->tablelink->classname;

break;

}

if (timetable[n][7].put == 0)

{

last = n;

timetable[n][7].put = timetable[n][8].put = timetable[n][9].put = 1;

timetable[n][7].classname = timetable[n][8].classname = timetable[n][9].classname = head->tablelink->classname;

break;

}

}

if (last < 2)

{

for (int n = last + 2; n < 5; n++)

{

if (timetable[n][0].put == 0)

{

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

}

else

{

if (last == 2)

{

for (int n = last - 2; n < 5; n = n + 4)

{

if (timetable[n][0].put == 0)

{

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

}

else

{

for (int n = last - 2; n >= 0; n = n--)

{

if (timetable[n][0].put == 0)

{

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

}

}

break;

}

case 4:

{

int last = 0;

for (int n = 0; n < 5; n++)

{

if (timetable[n][0].put == 0)

{

last = n;

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

last = n;

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

if (last < 2)

{

for (int n = last + 2; n < 5; n++)

{

if (timetable[n][0].put == 0)

{

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

}

else

{

if (last == 2)

{

for (int n = last - 2; n < 5; n = n + 4)

{

if (timetable[n][0].put == 0)

{

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

}

else

{

for (int n = last - 2; n >= 0; n = n--)

{

if (timetable[n][0].put == 0)

{

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

}

}

break;

}

case 3:

{

for (int n = 0; n < 5; n++)

{

if (timetable[n][2].put == 0)

{

timetable[n][2].put = timetable[n][3].put = timetable[n][4].put = 1;

timetable[n][2].classname = timetable[n][3].classname = timetable[n][4].classname = head->tablelink->classname;

break;

}

if (timetable[n][7].put == 0)

{

timetable[n][7].put = timetable[n][8].put = timetable[n][9].put = 1;

timetable[n][7].classname = timetable[n][8].classname = timetable[n][9].classname = head->tablelink->classname;

break;

}

}

break;

}

case 2:

{

for (int n = 0; n < 5; n++)

{

if (timetable[n][0].put == 0)

{

timetable[n][0].put = timetable[n][1].put = 1;

timetable[n][0].classname = timetable[n][1].classname = head->tablelink->classname;

break;

}

if (timetable[n][5].put == 0)

{

timetable[n][5].put = timetable[n][6].put = 1;

timetable[n][5].classname = timetable[n][6].classname = head->tablelink->classname;

break;

}

}

break;

}

case 1:

{

int ending = 0;

for (int n = 0; n < 5; n++)

{

for (int m = 0; m < 10; m++)

{

if (timetable[n][m].put == 0)

{

timetable[n][m].put = 1;

timetable[n][m].classname = head->tablelink->classname; ending = 1; break;

}

}

if (ending == 0) { break; }

}

break;

}

default:

{cout << "wrong" << " "; break; }

}

head = head->tablelink;

}

outfile << setw(32) << setiosflags(ios::right)<<setfill(' ') << "星期一";

outfile << setw(32) << setiosflags(ios::right) << setfill(' ') << "星期二";

outfile << setw(32) << setiosflags(ios::right) << setfill(' ') << "星期三";

outfile << setw(32) << setiosflags(ios::right) << setfill(' ') << "星期四";

outfile << setw(32) << setiosflags(ios::right) << setfill(' ') << "星期五";

outfile << endl;

for (int m = 0; m < 10; m++)

{

for (int n = 0; n < 5; n++)

{

if (timetable[n][m].put == 0)

{

outfile << setw(32) << setiosflags(ios::right) << setfill(' ')<< "无";

}

else

{

outfile << setw(32) << setiosflags(ios::right) << setfill(' ')<< timetable[n][m].classname;

}

}

outfile << endl;

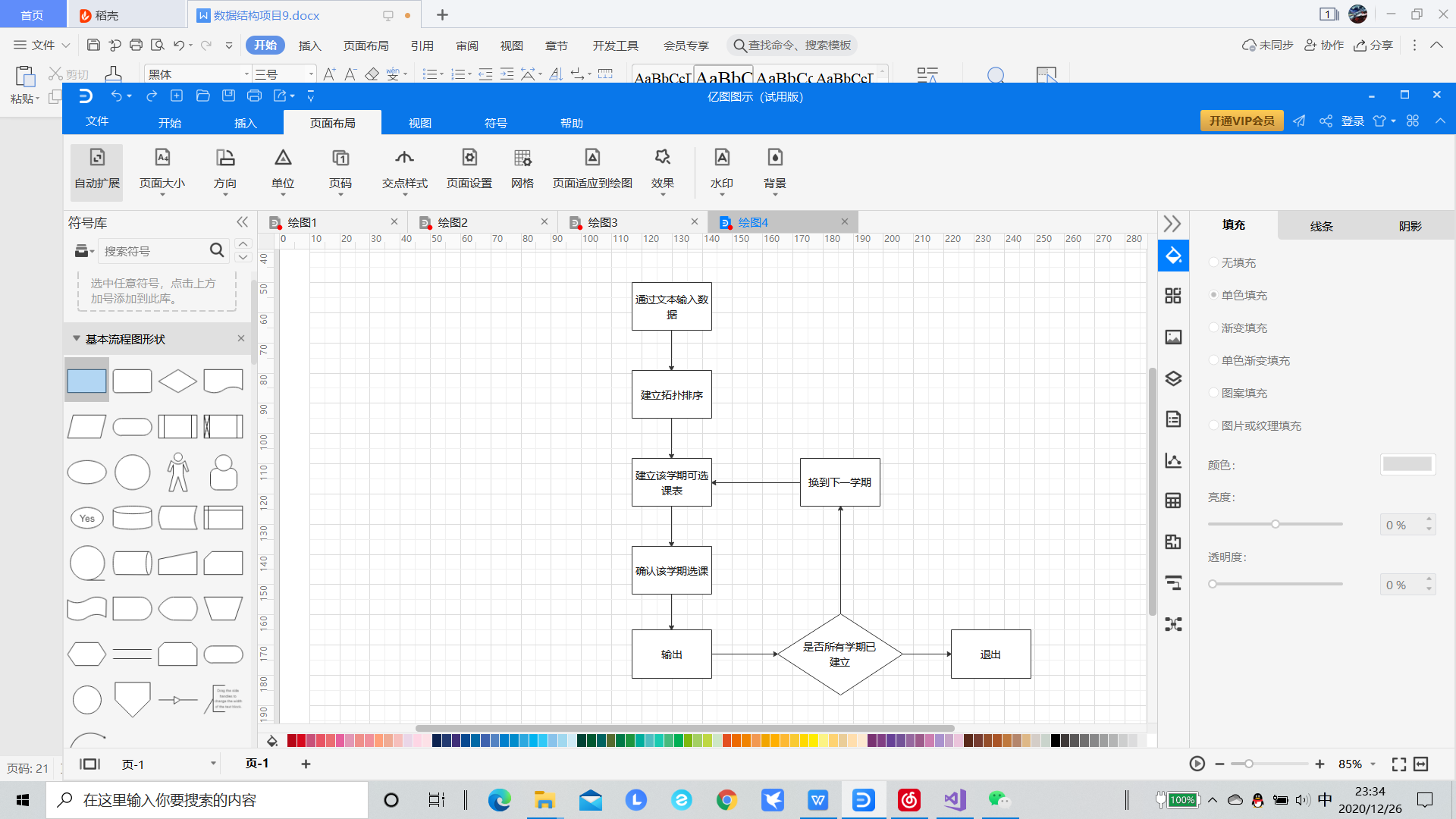
}

outfile.close();

### }

# 3.7总体系统实现

### 3.7.1 总体系统流程图

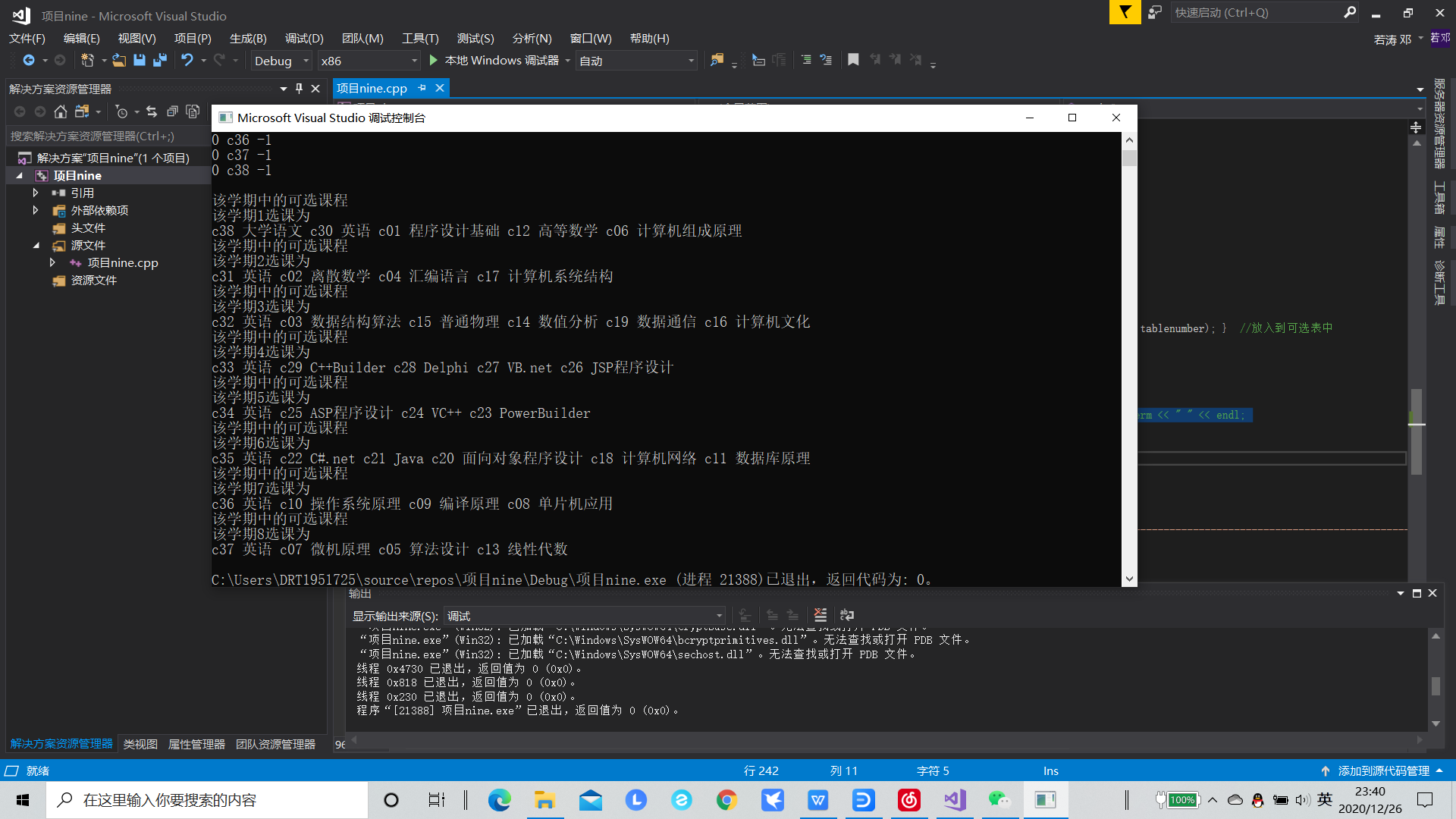


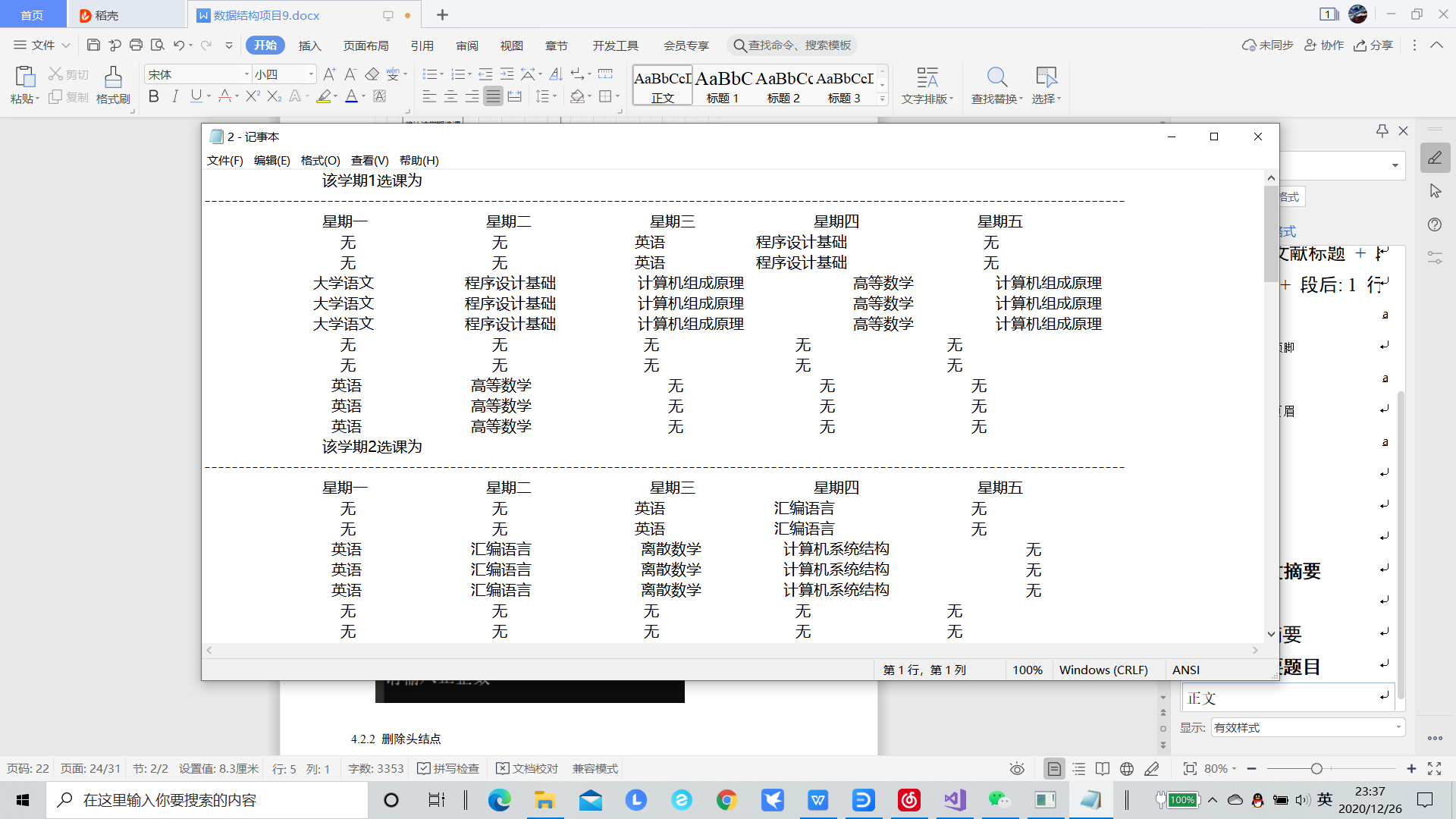
# 4 测试

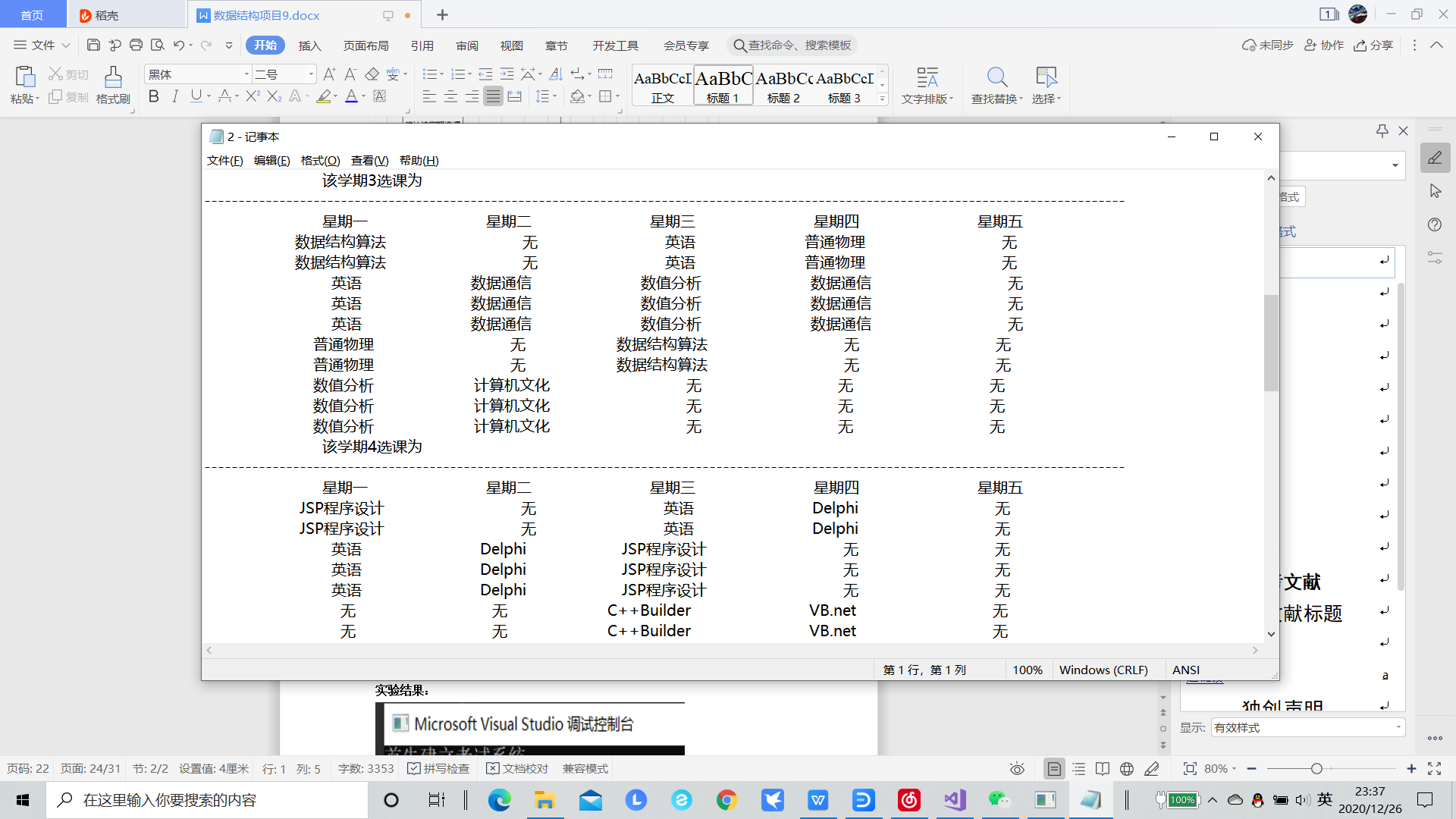
## 4.1 功能测试

总课程数 38

各个学期课程数 5 4 6 5 4 6 4 4







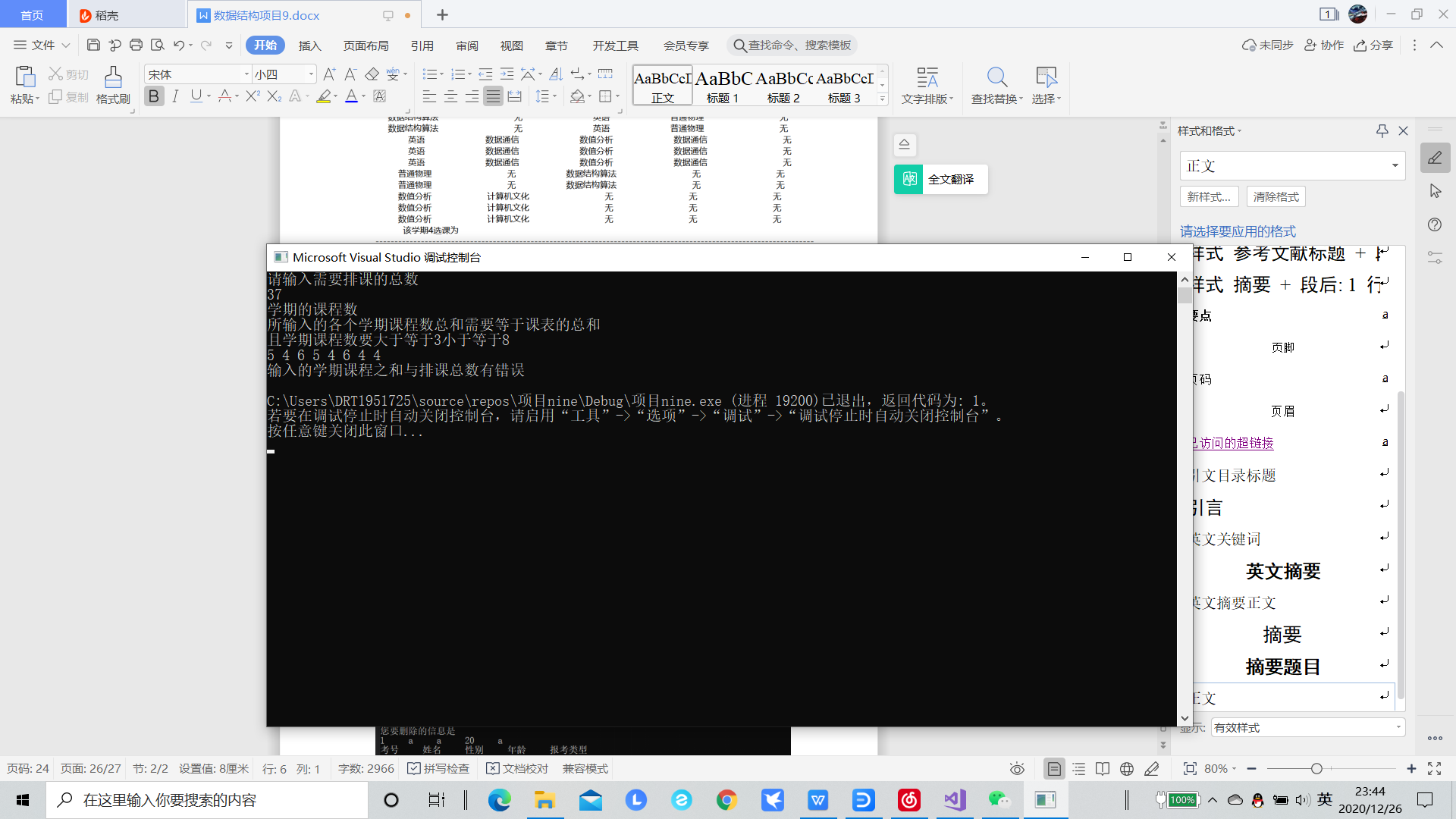
## 4.2 边界测试

### 4.2.1 总课数与学期课数不匹配

总课程数 37

各个学期课程数 5 4 6 5 4 6 4 4

**实验结果：**



### 4.2.2 学期课数错误

总课程数 38

各个学期课程数 9 0 6 5 4 6 4 4

**实验结果：**

