项目说明文档

数据结构课程设计

——排课软件

作 者 姓 名： 杨煜

学 号： 1850217

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](#_Toc26624396)

[1.1 项目内容 1](#_Toc26624397)

[1.2 项目要求 1](#_Toc26624398)

[2 设计 1](#_Toc26624399)

[2.1 数据结构设计 1](#_Toc26624400)

[2.2 队列结构设计 1](#_Toc26624401)

[2.3 成员与操作设计 2](#_Toc26624402)

[3 实现 3](#_Toc26624403)

[3.1 读取文件的实现 3](#_Toc26624404)

[3.1.1 读取文件 3](#_Toc26624405)

[3.1.2 读取文件核心代码 3](#_Toc26624406)

[3.2 第一种插入函数的实现 4](#_Toc26624407)

[3.2.1 第一种插入函数函数 4](#_Toc26624408)

[3.2.2 第一种插入函数核心代码 4](#_Toc26624409)

[3.3 第二种插入函数的实现 6](#_Toc26624410)

[3.3.1 第二种插入函数 6](#_Toc26624411)

[3.3.2 第二种插入函数核心代码 6](#_Toc26624412)

[3.4 填充函数的实现 8](#_Toc26624413)

[3.4.1 填充函数 8](#_Toc26624414)

[3.4.2 填充函数核心代码 8](#_Toc26624415)

[3.5 输出到文件的实现 9](#_Toc26624416)

[3.5.1 输出到文件 9](#_Toc26624417)

[3.5.2 输出到文件核心代码 9](#_Toc26624418)

[3.6 总体流程的实现 10](#_Toc26624419)

[3.6.1 总体流程 10](#_Toc26624420)

[3.6.2 总体流程核心代码 10](#_Toc26624421)

[4 测试 13](#_Toc26624422)

[4.1 正常测试 13](#_Toc26624423)

[4.1.1 第一学期 13](#_Toc26624424)

[4.1.2 第二学期 13](#_Toc26624425)

[4.1.3 第三学期 14](#_Toc26624426)

[4.1.4 第四学期 15](#_Toc26624427)

[4.1.5 第五学期 16](#_Toc26624428)

[4.1.6 第六学期 17](#_Toc26624429)

[4.1.7 第七学期 18](#_Toc26624430)

[4.1.8 第八学期 19](#_Toc26624431)

# 1 分析

## 1.1 项目内容

大学的每个专业都要进行排课。假设任何专业都有固定的学习年限，每学年含两学期，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必须满足先修关系。每门课程有哪些先修课程是确定的。每门课恰好占一个学期，假定每天上午与下午各有5节课。是在这样的前提下设计一个教学计划编制程序。

## 项目要求

1、输入数据包括：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。

2、如输入数据不合理，比如每学期所开的课程数值和与课程总数不相等，应显示适当的提示信息。

1. 用文本文件存储输入数据，并且读入计算机。
2. 用文本文件存储产生的各学期的课表。

# 2 设计

## 2.1 数据结构设计

如上述项目简介所述，每门课程都需要有前置课程的约束，是典型的AOV网络。当图中的顶点的入度为零时，则该节点可以移除，表明该节点已经没有其他前置要求了，可以进行下一步的此顶点的活动。本程序使用队列来模拟图取顶点的过程，来方便处理从文件中接收到的数据。

## 2.2 队列结构设计

使用数组作为队列的存储结构，使用一个头指针，一个尾指针做出控制。ben程序使用了循环队列来完成队列的实现。

## 2.3 成员与操作设计

struct Course

{

string id;

string name;

string span;

string semester;

string premise;

};

class SeqQueue

{

public:

SeqQueue(int sz = 1000);//构造函数

~SeqQueue() { delete[] elements; };//析构函数，释放空间

bool EnQueue(const Course& x);//入队操作

Course DeQueue();//出队操作

bool isEmpty()const { return(front == rear) ? true : false; };//判断队列是否为空

bool isFull()const { return((rear + 1) % maxSize == front) ? true : false; };//判断队列是否为满

private:

int rear, front;//头指针，尾指针

Course\* elements;

int maxSize;//最大长度

};

SeqQueue::SeqQueue(int sz) :front(0), rear(0), maxSize(sz)

{

elements = new Course[maxSize];

}

bool SeqQueue::EnQueue(const Course& x)

{

if (isFull() == true)

return false;

elements[rear] = x;

rear = (rear + 1) % maxSize;

return true;

}

Course SeqQueue::DeQueue()

{

Course x;

x = elements[front];

front = (front + 1) % maxSize;

return x;

}

# 3 实现

## 3.1 读取文件的实现

### 3.1.1 读取文件

打开名为课程的txt文件，从中读取出所要排课的内容，判断排课内容是否有前置要求，随后分别存入到Course数组之中。

### 3.1.2 读取文件核心代码

bool input()

{

ifstream fin("课程.txt",ios::in);

if (!fin) {

cout << "error opening source file." << endl;

return 0;

}

fin >> A[i].id >> A[i].name;

while (!fin.eof())

{

fin >> A[i].span >> A[i].semester;

fin >> A[i].premise;

fin >> A[i + 1].id;

if (A[i + 1].id[0] != 'c')

{

A[i+1].name = A[i + 1].id;

A[i + 1].id = A[i].premise;

A[i].premise = '\0';

}

else

{

fin >> A[i + 1].name;

}

i++;

}

}

## 3.2 第一种插入函数的实现

### 3.2.1 第一种插入函数函数

该函数所对应的插入内容为，该课程的课时数为3的情况，因此只需要插入一次。所接受的两个参数分别为，所要插入内容的循环开始学期以及读取的课程。通过循环，找到课表之上为零的地方，表示空位，在此插入该课程对的课程代号。

### 3.2.2 第一种插入函数核心代码

void insert1(int a, Course b)

{

int flag = 0;

for (int k = a; k < 8; k++)

{

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

{

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

{

if (sem[k][m][n] == 0 && (m == 2 || m == 7))

{

for (int p = 0; p < 3; p++)

{

sem[k][m + p][n] = (b.id[1] - 48) \* 10 + (b.id[2] - 48);

isused[(b.id[1] - 48) \* 10 + (b.id[2] - 48) - 1] = k + 1;

}

flag = 1;

}

if (flag == 1)

{

break;

}

}

if (flag == 1)

{

break;

}

}

if (flag == 1)

{

break;

}

}

}

## 3.3 第二种插入函数的实现

### 3.3.1 第二种插入函数

该函数所对应的插入内容为，该课程的课时数为4，5，6的情况，因此需要插入两次内容。所接受的六个参数分别为，所要插入内容的循环开始学期以及读取的课程，后四位，分别是第一次插入的课程位置，第二次插入的课程位置，以及两次插入时所插入的课时个数。通过循环，找到课表之上为零的地方，表示空位，在此插入该课程对的课程代号。

### 3.3.2 第二种插入函数核心代码

void insert2(int a, Course b, int m1, int m2, int p1, int p2)

{

int flag = 0;

int r = 0;

int q = 0;

for (int k = a; k < 8; k++)

{

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

{

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

{

if (sem[k][m][n] == 0 && (m == m1 || m == m1 + 5) && flag == 0)

{

for (int p = 0; p < p1; p++)

{

sem[k][m + p][n] = (b.id[1] - 48) \* 10 + (b.id[2] - 48);

isused[(b.id[1] - 48) \* 10 + (b.id[2] - 48) - 1] = k + 1;

}

q = n;

r = m;

flag++;

}

if (sem[k][m][n] == 0 && (m == m2 || m == m2 + 5) && flag == 1 && abs(n - q) >= 2)

{

if (isused[(b.id[1] - 48) \* 10 + (b.id[2] - 48) - 1] == k + 1)

{

for (int p = 0; p < p2; p++)

{

sem[k][m + p][n] = (b.id[1] - 48) \* 10 + (b.id[2] - 48);

isused[(b.id[1] - 48) \* 10 + (b.id[2] - 48) - 1] = k + 1;

}

flag++;

}

else

{

for (int p = 0; p < p1; p++)

{

sem[isused[(b.id[1] - 48) \* 10 + (b.id[2] - 48) - 1] - 1][r + p][q] = 0;

}

isused[(b.id[1] - 48) \* 10 + (b.id[2] - 48) - 1] = 0;

flag--;

continue;

}

}

if (flag == 2)

{

break;

}

}

if (flag == 2)

{

break;

}

}

if (flag == 2)

{

break;

}

}

}

## 3.4 填充函数的实现

### 3.4.1 填充函数

填充函数所执行的功能为根据课时数的内容，分别调用不同的插入函数。

### 3.4.2 填充函数核心代码

void blank(int a,char f,Course b)

{

switch (f)

{

case '3':

insert1(a, b);

break;

case '4':

insert2(a, b, 0, 0, 2, 2);

break;

case '5':

insert2(a, b, 2, 0, 3, 2);

break;

case '6':

insert2(a, b, 2, 2, 3, 3);

break;

default:

break;

}

}

## 3.5 输出到文件的实现

### 3.5.1 输出到文件

创建8个文件，将sem数组之中存储的课程信息，转化为课程名字放入到课表之中，将课表放入到txt文件之中。

### 3.5.2 输出到文件核心代码

void output()

{

ofstream fout;

for (int i = 0; i < 8; i++)

{

string str = "第" + to\_string(i + 1) + "学期.txt";

fout.open(str);

fout <<setw(17)<<" "<< "周一" << setw(30) << " " << "周二" << setw(30) << " " << "周三" << setw(30) << " " << "周四" << setw(30) << " " << "周五" << setw(30) << " " << endl;

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

{

fout << m;

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

{

if (sem[i][m][n] != 0)

{

fout << setw(15) << " " << A[sem[i][m][n]-1].name << setw(16 - A[sem[i][m][n]-1].name.size()) << " ";

}

else

{

fout <<setw(36)<<" ";

}

}

fout << endl;

}

fout.close();

}

}

## 3.6 总体流程的实现

### 3.6.1 总体流程

创建一个队列，当队列元素不为零时，循环，使用一个辅助数组来记录课程是否已被填入，如果被填入，记录下数组填入的学期。如果没有则将其重新放入队列中，这是一个类似广度优先搜素的想法。

### 3.6.2 总体流程核心代码

int main()

{

int x, y;

input();

SeqQueue course;

for (int k = 0; k < i ;k++)

{

if (A[k].semester[0]-48 >= 1 && A[k].semester[0]-48 <= 8)

{

blank(A[k].semester[0] - 49, A[k].span[0],A[k]);

}

else

{

course.EnQueue(A[k]);

}

}

while (!course.isEmpty())

{

Course b = course.DeQueue();

if (b.premise.size() == 1)

{

blank(0, b.span[0], b);

}

else

{

if (b.premise.size() == 3)

{

x = isused[(b.premise[1] - 48) \* 10 + (b.premise[2] - 48) - 1];

if (x != 0)

{

blank(x, b.span[0], b);

}

else

{

course.EnQueue(b);

continue;

}

}

else if (b.premise.size() == 6)

{

x = isused[(b.premise[1] - 48) \* 10 + (b.premise[2] - 48) - 1];

y = isused[(b.premise[4] - 48) \* 10 + (b.premise[5] - 48) - 1];

if (x != 0 && y != 0)

{

blank(x > y ? x:y , b.span[0], b);

}

else

{

course.EnQueue(b);

continue;

}

}

}

}

output();

for (int m = 0; m < 8; m++)

{

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

{

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

{

cout << sem[m][n][j] << " ";

}

cout << endl;

}

cout << endl;

}

}

# 4 测试

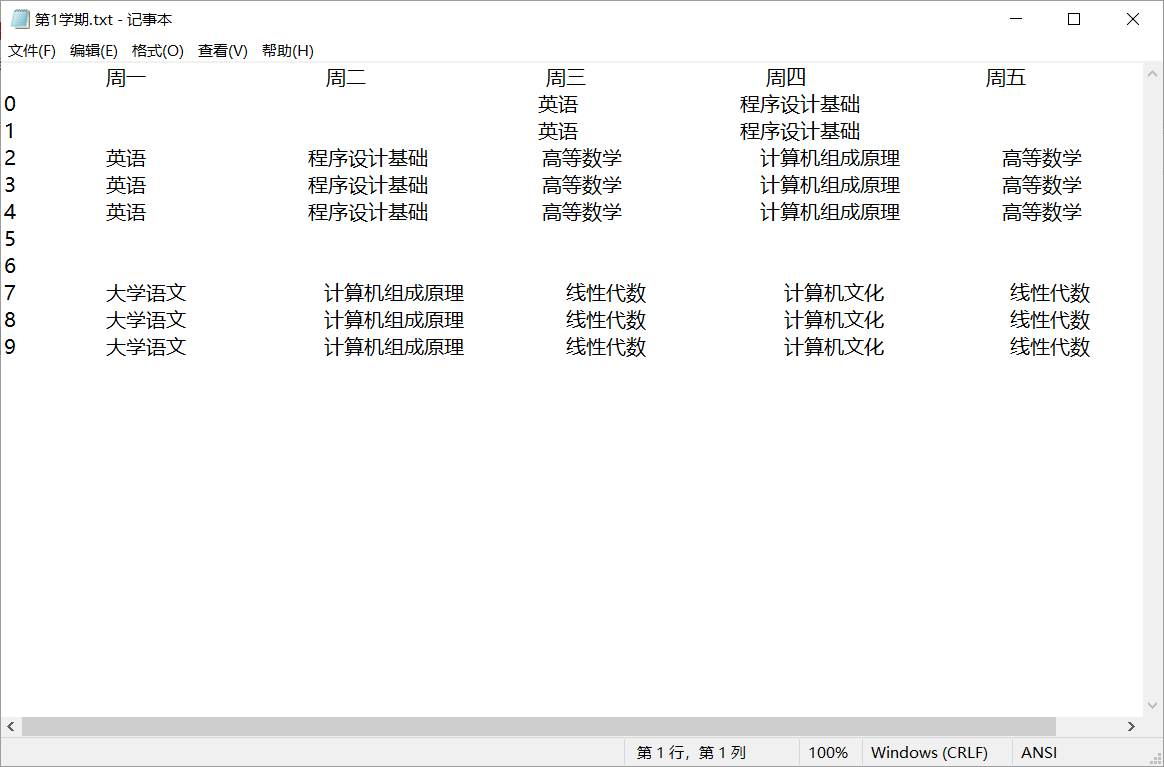
## 4.1 正常测试

测试数据：7 6 3 10 6 4 1 1



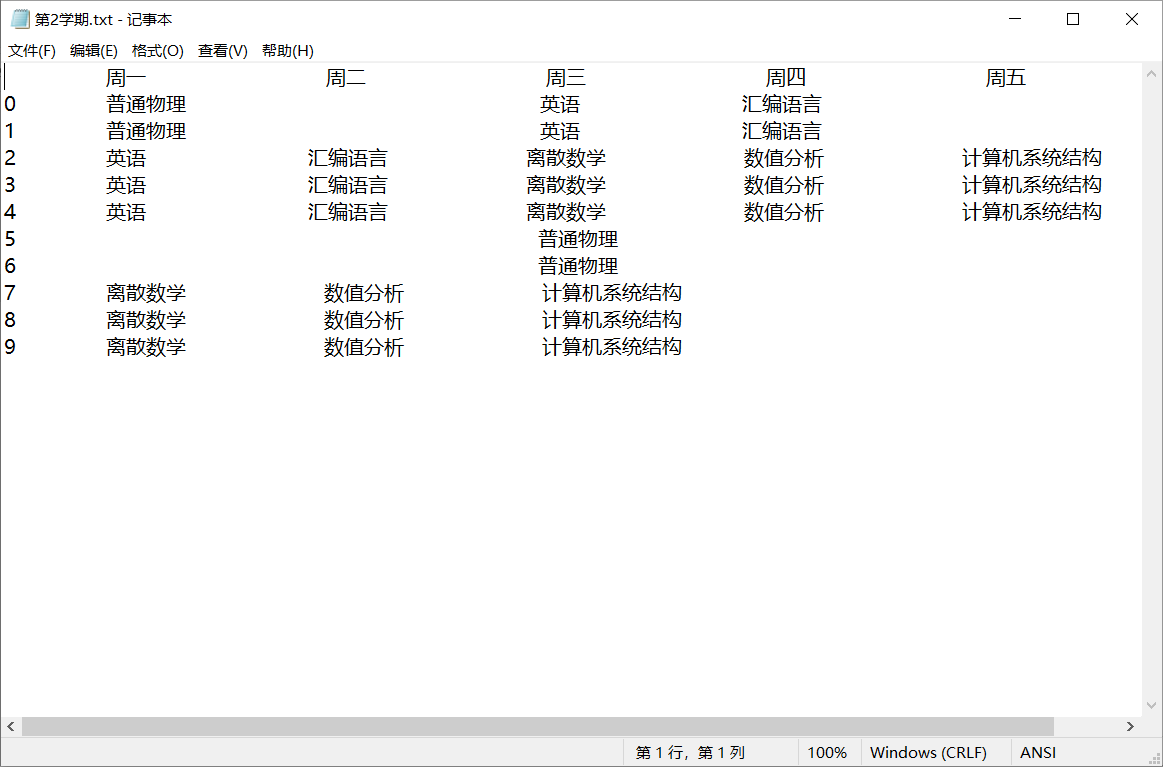
### 4.1.1 第一学期

**实验结果：**



### 4.1.2 第二学期

**实验结果：**



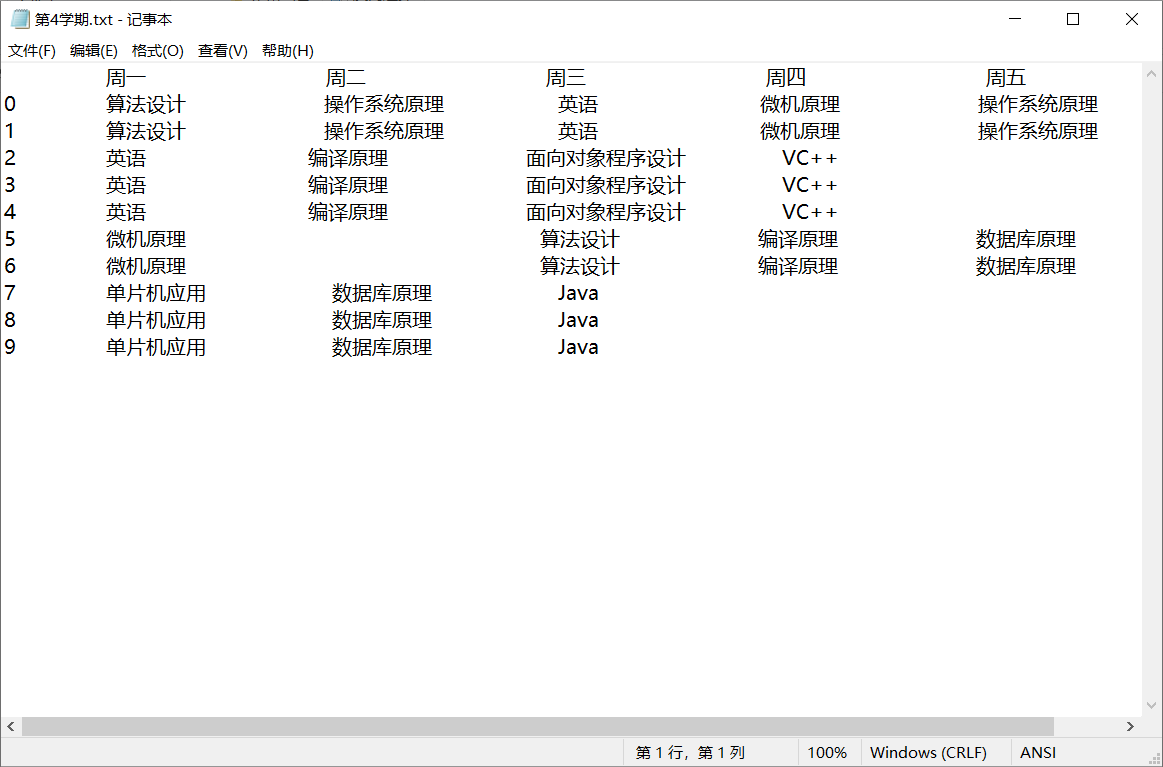
### 4.1.3 第三学期

**实验结果：**



### 4.1.4 第四学期

**实验结果：**



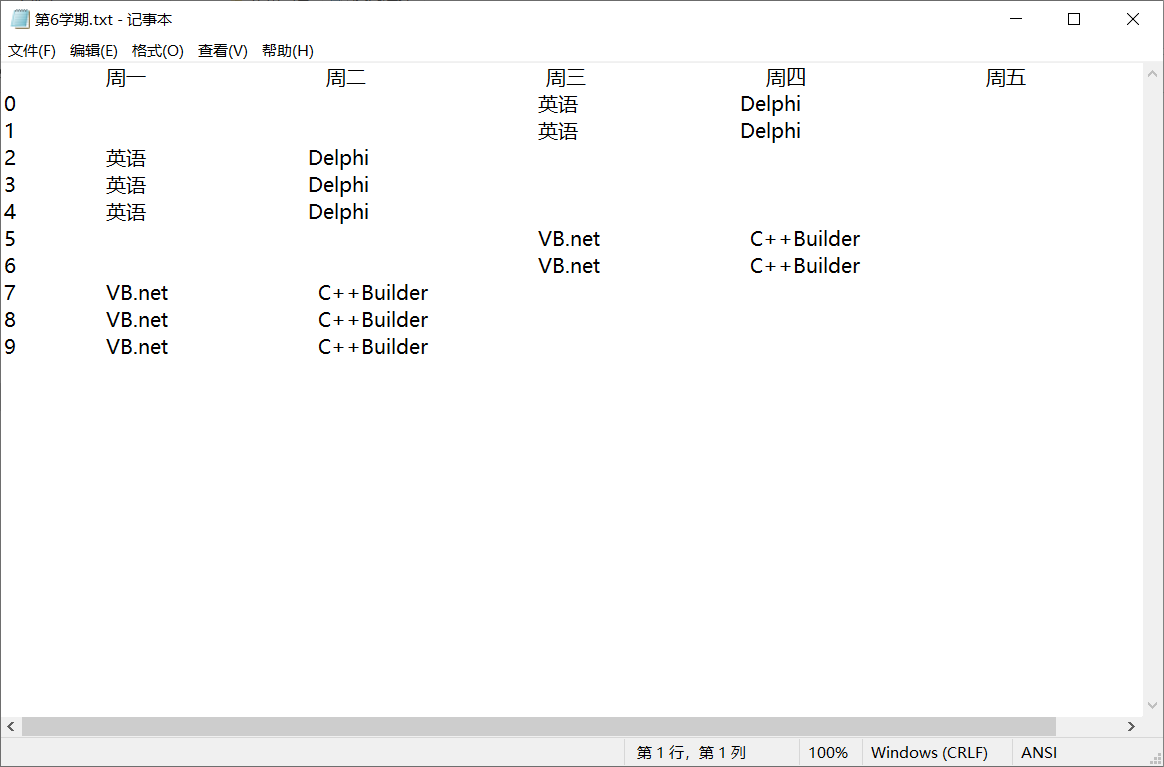
### 4.1.5 第五学期

**实验结果：**



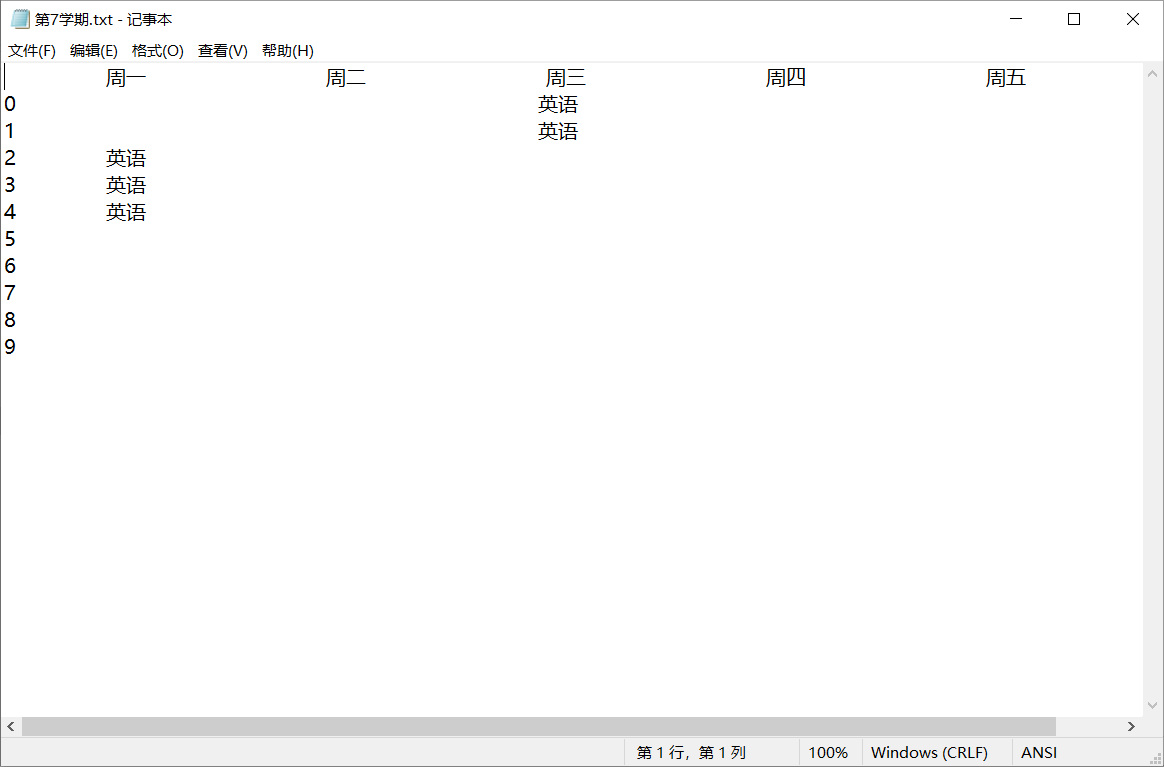
### 4.1.6 第六学期

**实验结果：**



### 4.1.7 第七学期

**实验结果：**



### 4.1.8 第八学期

**实验结果：**

