



太原理工大学  
TAIYUAN UNIVERSITY OF TECHNOLOGY

# 程序设计课程设计 指导书

软件学院 软件工程系

2019 年 6 月 1 日

## 太原理工大学课程设计任务书

学生姓名		专业班级	软件
课程名称	程序设计课程设计 (Programming Curriculum Design)		
设计名称	学生排队, 俄罗斯方块, 文本文件单词统计等	设计周数	2
设计任务主要设计参数	<p><b>1. 基本要求</b> 掌握 C 或 C++ 语言、结构化程序和面向对象程序设计方法、数据结构和离散数学理论知识, 熟悉 C 或 C++ 程序的开发环境及调试过程, 巩固和加深对理论课中知识的理解, 提高学生对所学知识的综合运用能力。</p> <p><b>2. 培养学生以下技能</b> ①培养学生查阅参考资料、手册的自学能力, 通过独立思考深入钻研问题, 学会自己分析、解决问题。 ②通过对所选题目分析, 找出解决方法, 设计算法, 编制程序与调试程序。 ③能熟练调试程序, 在教师的指导下, 完成课题任务。 ④按课程设计报告的要求撰写设计报告。</p>		
设计内容 设计要求	<p><b>1. 设计内容</b> ①学生排队; ②俄罗斯方块; ③文本文件单词统计; ④分数的加减乘除; ⑤送货; ⑥学生信息管理系统</p> <p><b>2. 设计要求</b> ①至少完成上述设计内容中的 4 个设计题目; ②对每个题目要给出设计方案、功能模块划分、算法思想; ③选择使用的数据结构; ④给出题目的程序实现; ⑤按要求撰写设计报告。</p>		
主要参考资料	<p>1. 《程序设计课程设计》指导书; 2. 《程序设计技术》、《数据结构》等课程教材; 3. 其他自选的相关资料。</p>		
学生提交 归档文件	<p>课程设计报告封面应给出专业、班级、姓名、学号、指导教师和完成日期。每个设计题目的内容包括以下几项: 设计题目、问题描述、问题分析、功能实现、测试实例及运行结果、源程序清单。</p>		

注:

1. 课程设计完成后, 学生提交的归档文件应按照: 封面—任务书—说明书—图纸的顺序进行装订上交 (大张图纸不必装订)。
2. 可根据实际内容需要续表, 但应保持原格式不变。

指导教师签名:

日期: 2019.6.10

# 前 言

《程序设计课程设计》是计算机科学与技术专业的重要实践性课程。目的在于培养学生分析问题和解决问题的能力，为学生提供了一个既动手又动脑，独立实践的机会。将数据结构、算法设计与分析、离散数学和 C（C++）语言等课本上的的理论知识与实际应用问题进行有机结合，提高学生问题分析、程序设计、程序调试及项目开发能力。为后续课程：操作系统、软件工程，编译原理等课程的学习奠定必要的实践基础。

本课程设计是利用数据结构、算法设计与分析、离散数学、C 语言理论知识和实验课中学到的编程知识和编程技巧，通过布置具有一定难度、一定编程量的课程设计题目，利用 C（C++）语言作为开发工具，使学生通过课程设计掌握高级编程语言的知识和编程技术，掌握程序设计的思想和方法，初步具备利用计算机求解实际问题的能力。

通过《程序设计课程设计》课程的学习，能够帮助学生加深理解数据结构、离散数学、C 语言基本概念，达到培养学生良好程序设计的习惯和运用 C 语言编写程序解决实际问题的能力。使学生学会把书本知识用于解决实际问题，起到深化理解和灵活掌握教学内容的目的。同时使学生在程序设计方法及上机操作等基本技能和科学作风方面受到比较系统和严格的训练。

通过该课程设计，学生应该掌握 C 或 C++语言程序设计、结构化程序和面向对象程序设计方法、数据结构和离散数学理论知识，熟悉 C 或 C++程序的开发环境及 C 或 C++程序的调试过程，巩固和加深对理论课中知识的理解，提高学生对所学知识的综合运用能力。

通过本课程设计训练，学生应该具备如下基本技能：

①培养学生查阅参考资料、手册的自学能力，通过独立思考深入钻研问题，学会自己分析、解决问题。

②通过对所选题目方案分析比较，确立方案，编制程序与调试程序。

③能熟练调试程序，在教师的指导下，完成课题任务。

④根据个人的设计调试过程，按课程设计报告的要求撰写设计报告。

## 选用教材及主要参考书：

### 1 教材

呼克佑. C 语言程序设计 电子工业出版社, 2019

严蔚敏. 数据结构(C 语言版) 清华大学出版社, 2012

# 设计题目（6 选 5）

## 1. 学生排队

### 1.1 【问题描述】

体育老师小明要将自己班上的学生按顺序排队。他首先让学生按学号从小到大的顺序排成一排，学号小的排在前面，然后进行多次调整。一次调整小明可能让一位同学出队，向前或者向后移动一段距离后再插入队列。

例如，下面给出了一组移动的例子，例子中学生的人数为 8 人。

0) 初始队列中学生的学号依次为 1, 2, 3, 4, 5, 6, 7, 8;

1) 第一次调整，命令为“3 号同学向后移动 2”，表示 3 号同学出队，向后移动 2 名同学的距离，再插入到队列中，新队列中学生的学号依次为 1, 2, 4, 5, 3, 6, 7, 8;

2) 第二次调整，命令为“8 号同学向前移动 3”，表示 8 号同学出队，向前移动 3 名同学的距离，再插入到队列中，新队列中学生的学号依次为 1, 2, 4, 5, 8, 3, 6, 7;

3) 第三次调整，命令为“3 号同学向前移动 2”，表示 3 号同学出队，向前移动 2 名同学的距离，再插入到队列中，新队列中学生的学号依次为 1, 2, 4, 3, 5, 8, 6, 7。

小明记录了所有调整的过程，请问，最终从前向后所有学生的学号依次是多少？

请特别注意，上述移动过程中所涉及的号码指的是学号，而不是在队伍中的位置。在向后移动时，移动的距离不超过对应同学后面的人数，如果向后移动的距离正好等于对应同学后面的人数则该同学会移动到队列的最后面。在向前移动时，移动的距离不超过对应同学前面的人数，如果向前移动的距离正好等于对应同学前面的人数则该同学会移动到队列的最前面。

### 1.2 【输入输出数据格式】

**输入格式：**输入的第一行包含一个整数  $n$ ，表示学生的数量，学生的学号由 1 到  $n$  编号。第二行包含一个整数  $m$ ，表示调整的次数。接下来  $m$  行，每行两个整数  $p, q$ ，如果  $q$  为正，表示学号为  $p$  的同学向后移动  $q$ ，如果  $q$  为负，表示学号为  $p$  的同学向前移动  $-q$ 。  $1 \leq n, m \leq 1000$

**输出格式：**输出一行，包含  $n$  个整数，相邻两个整数之间由一个空格分隔，表示最终从前向后所有学生的学号。

**样例数据输入：**

```
8
3
3 2
8 -3
3 -2
```

**样例数据输出：**

```
1 2 4 3 5 8 6 7
```

## 2. 俄罗斯方块

### 2.1 【问题描述】

俄罗斯方块是俄罗斯人阿列克谢·帕基特诺夫发明的一款休闲游戏。游戏在一个 15 行 10 列的方格图上进行，方格图上的每一个格子可能已经放置了方块，或者没有放置方块。每一轮，都会有一个新的由 4 个小方块组成的板块从方格图的上方落下，玩家可以操作板块左右移动放到合适的位置，当板块中某一个方块的下边缘与方格图上的方块上边缘重合或者达到下边界时，板块不再移动，如果此时方格图的某一行全放满了方块，则该行被消除并得分。在这个问题中，你需要写一个程序来模拟板块下落，你不需要处理玩家的操作，也不需要处理消行和得分。

### 2.2 【基本要求】

给定一个初始的方格图，以及一个板块的形状和它下落的初始位置，你要给出最终的方格图。

**输入格式：**（提示：把数据写入文件中，从文件读取数据，避免调试时繁琐的输入）

输入的前 15 行包含初始的方格图，每行包含 10 个数字，相邻的数字用空格分隔。如果一个数字是 0，表示对应的方格中没有方块，如果数字是 1，则表示初始的时候有方块。输入保证前 4 行中的数字都是 0。

输入的第 16 至第 19 行包含新加入的板块的形状，每行包含 4 个数字，组成了板块图案，同样 0 表示没方块，1 表示有方块。输入保证板块的图案中正好包含 4 个方块，且 4 个方块是连在一起的（准确的说，4 个方块是四连通的，即给定的板块是俄罗斯方块的标准板块）。

第 20 行包含一个 1 到 7 之间的整数，表示板块图案最左边开始的时候是在方格图的哪一列中。注意，这里的板块图案指的是 16 至 19 行所输入的板块图案，如果板块图案的最左边一列全是 0，则它的左边和实际所表示的板块的左边是不一致的（见样例）

**输出格式：**输出 15 行，每行 10 个数字，相邻的数字之间用一个空格分隔，表示板块下落后的方格图。注意，你不需要处理最终的消行。

**样例数据输入：**

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0 0
1 1 1 0 0 0 1 1 1 1
0 0 0 0 1 0 0 0 0 0
```

```
0 0 0 0
0 1 1 1
0 0 0 1
0 0 0 0
3
```

样例数据输出：

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 1 1 0 0 0 0
```

### 3. 文本文件单词统计

#### 3.1 【问题描述】

假设有如下的英文文本文件：（文件名是：Happiness.txt）

#### What Is Happiness

Most of us probably don't believe we need a formal definition of happiness; we know it when we feel it, and we often use the term to describe a range of positive emotions, including joy, pride, contentment, and gratitude.

But to understand the causes and effects of happiness, researchers first need to define it. Many of them use the term interchangeably with "subjective well-being," which they measure by simply asking people to report how satisfied they feel with their own lives and how much positive and negative emotion they're experiencing. In her 2007 book *The How of Happiness*, positive psychology researcher Sonja Lyubomirsky elaborates, describing happiness as "the experience of joy, contentment, or positive well being, combined with a sense that one's life is good, meaningful, and worthwhile."

That definition resonates with us here at *Greater Good*: It captures the fleeting positive emotions that come with happiness, along with a deeper sense of meaning and purpose in life and suggests how these emotions and sense of meaning reinforce one another.

设计 C 或 C++ 程序，统计在这样的英文文本文件中，出现了多少个单词（不区分大小写），每个单词出现了几次。连续的英文字符都认为是单词(不包括数字)，单词之间用空格或标点符号分隔。

### 3.2 【设计需求及分析】

要统计英文文本文件中出现了哪些单词，就要从文件中读取字符，读取出来的连续英文字符认为是一个单词，遇空格或标点符号单词结束。

使用线性表记录单词以及每个单词出现的次数。线性表中的单词按字典顺序存储。

**线性表的顺序存储结构如下：（必须使用如下定义的存储结构，否则无效）**

```
#define LIST_INIT_SIZE 100    //线性表存储空间的初始分配量
#define LISTINCREMENT 10     //线性表存储空间的分配增量
typedef struct{
    char word[21]              //存储单词，不超过 20 个字符
    int count;                  //单词出现的次数
} ElemType;
typedef struct{
    ElemType *elem;             //存储空间基址
    int length;                 //当前长度
    int listsize;               //当前分配的存储容量
} SqList;
```

### 3.3 【功能设计】

#### 3.3.1 实现顺序表的基本操作（必须使用下面给定的函数名和参数表，否则无效）

(1)顺序表的初始化：InitList(SqList &L)

(2)顺序表上查找指定的单词：LocateElem(SqList &L, char \*s)

若找到，单词的出现次数增 1，返回 0，否则返回该单词的插入位置。

(3)在顺序表上插入新的单词：InsertList(SqList &L, int i, char \*s)

要求按字典顺序有序。新单词的出现次数为 1。

(4)输出顺序表上存储的单词统计信息：PrintList(SqList &L)

输出文件中每个单词出现的次数以及文件中总的单词数(可输出到文件中)。

#### 3.3.2 统计单词数

统计过程如下：

(1) 输入要统计单词的文本文件名，打开相应的文件；

(2) 初始化顺序表；

(3) 从文本文件中读取字符，直到文件结束。具体描述如下：

```
while (读文件没有结束)
{
    过滤单词前的非字母字符;
```

读取一个单词，以字符串形式存储在一个字符数组中；  
在线性表中查找该单词，若找到，单词的出现次数加 1，否则返回其插入位置；  
上一步中，若没找到，则进行插入操作；  
处理下一个单词。

}

(4) 关闭文件，输出统计结果。

### 3.4 【测试数据】

将上述给定的英文文档写入文本文件：Happiness.txt 作为测试数据文件。

## 4. 分数的加减乘除

### 4.1 【问题描述】

用分数形式表示的有理数类如下：

```
class Rational{
private:
    int x,y;                //成员变量 x 和 y，分别存放分子和分母
public:
    Rational(int a=1,int b=1);    //具有默认参数的构造函数，默认值为 1
    Rational Add(Rational &r);    //求两个分数的和
    Rational Sub(Rational &r);    //求两个分数的差
    Rational Mul(Rational &r);    //求两个分数的积
    Rational Div(Rational &r);    //求两个分数的商
    Rational operator+(Rational &r); //重载 “+” 运算符，求两个分数的和
    Rational operator-(Rational &r); //重载 “-” 运算符，求两个分数的差
    Rational operator*(Rational &r); //重载 “*” 运算符，求两个分数的积
    Rational operator/(Rational &r); //重载 “/” 运算符，求两个分数的商
    int Divisor(int a,int b);      //求最大公约数
    friend ostream& operator<<(ostream &output,Rational &r);    //以 x/y 的形式输出分数
};
```

### 4.2 【基本要求】

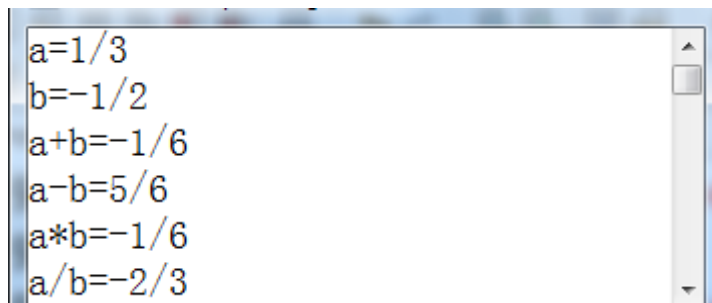
1. 成员变量 x 和 y，分别存放分子和分母，要求分数以最简形式存放。例如：分数 2/4 应简化为 1/2。
2. 定义成员函数 Add、Sub、Mul 和 Div，求两个分数的和差积商。计算结果仍以最简形式存放。
3. 重载运算符 “+、-、\*、/”，求两个分数的和差积商。计算结果仍以最简形式存放。
4. 定义友元函数，重载 “<<” 运算符，以 x/y 的形式输出分数。例如，有如下的主函数：



```

int main() {
    Rational a(5, 15), b(1, -2), c;
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
    c=a.Add(b);    //c=a+b;
    cout<<"a+b="<<c<<endl;
    c=a.Sub(b);    //c=a-b;
    cout<<"a-b="<<c<<endl;
    c=a.Mul(b);    //c=a*b;
    cout<<"a*b="<<c<<endl;
    c=a.Div(b);    //c=a/b;
    cout<<"a/b="<<c<<endl;
}

```



```

a=1/3
b=-1/2
a+b=-1/6
a-b=5/6
a*b=-1/6
a/b=-2/3

```

运行结果如图 4-1 所示：

图 4-1 运行结果

## 5. 送货

### 5.1 【问题描述】

为了增加公司收入，F 公司新开设了物流业务。由于 F 公司在业界的良好口碑，物流业务一开通即受到了消费者的欢迎，物流业务马上遍及了城市的每条街道。然而，F 公司现在只安排了小明一个人负责所有街道的服务。

任务虽然繁重，但是小明有足够的信心，他拿到了城市的地图，准备研究最好的方案。城市中有  $n$  个交叉路口， $m$  条街道连接在这些交叉路口之间，每条街道的首尾都正好连接着一个交叉路口。除开街道的首尾端点，街道不会在其他位置与其他街道相交。每个交叉路口都至少连接着一条街道，有的交叉路口可能只连接着一条或两条街道。

### 5.2 【基本要求】

小明希望设计一个方案，从编号为 1 的交叉路口出发，每次必须沿街道去往街道另一端的路口，再从新的路口出发去往下一个路口，直到所有的街道都经过了正好一次。

#### 输入数据格式：

输入的第一行包含两个整数  $n, m$  ( $1 \leq n \leq 10, n-1 \leq m \leq 20$ )，表示交叉路口的数量和街道的数量，交叉路口从 1 到  $n$  标号。

接下来  $m$  行，每行两个整数  $a, b$ ，表示和标号为  $a$  的交叉路口和标号为  $b$  的交叉路口之间有一条街道，街道是双向的，小明可以从任意一端走向另一端。两个路口之间最多有一条街道。

#### 输出输出格式：

如果小明可以经过每条街道正好一次，则输出一行包含  $m+1$  个整数  $p_1, p_2, p_3, \dots, p_{m+1}$ ，表示小明经过的路口的顺序，相邻两个整数之间用一个空格分隔。如果有多种方案满足条件，则输出字典序最小的一种方案，即首先保证  $p_1$  最小， $p_1$  最小的前提下再保证  $p_2$  最小，依此类推。

如果不存在方案使得小明经过每条街道正好一次，则输出一个整数-1。

5.3 【测试数据】

测试数据一

输入:	输出:
4 5 1 2 1 3 1 4 2 4 3 4	1 2 4 1 3 4

输出说明：城市的地图和小明的路径如图 5-1 所示。

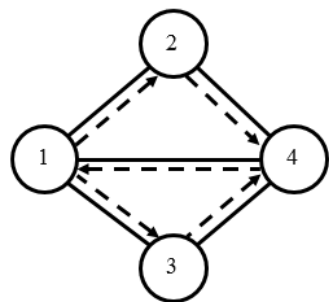


图 5-1 测试数据一对应的地图

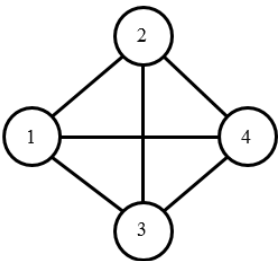


图 5-2 测试数据二对应的地图

测试数据二

输入:	输出:
4 6 1 2 1 3 1 4 2 4 3 4 2 3	-1

输出说明：城市的地图如图 5-2 所示，不存在满足条件的路径。

## 6. 学生信息管理系统

### 6.1 【问题描述】

大学里有各种类型的学生，校方需要对这些学生的信息进行计算机管理。所开发的软件应包括各类学生的添加、修改、删除和查找等功能。考虑到软件的可重用性、可扩展性和可维护性，校方决定采用面向对象的程序设计方法来开发系统。学生信息需要以文件方式保存到计算机硬盘中。另外，系统的用户界面应该尽可能友好，方便用户使用。

### 6.2 【设计需求及分析】

- (1) 使用 C++ 语言开发，充分利用面向对象程序设计的类、对象、继承、封装和多态性等概念来设计和实现该管理系统。
- (2) 设计一个 Person（人员）类，考虑到通用性，只抽象出所有类型人员都具有的属性：name（姓名），id（身份证号），gender（性别），birthday（出生日期）等等。其中“出生日期”为内嵌子对象，是一个 Date（日期）类型，Date 类具有属性：year（年），month（月），day（日）。用成员函数实现对人员信息的录入和显示等必要功能操作。
- (3) 从 Person 类派生出 Student（学生）类，添加属性：studentNo（学号），schoolName（学校），classIn（班级）。从 Person 类派生出 Teacher（教师）类，添加属性：teacherNo（教师编号），schoolName（学校），department（部门）。
- (4) 从 Student 类中派生出 UnderGraduate（本科生）类，添加属性：major（专业）。从 Student 类中派生出 Graduate（研究生）类，添加属性：direction（研究方向），adviserName（导师姓名）。
- (5) 从 Graduate 类和 Teacher 类派生出 TA（助教博士生）类。
- (6) 写程序测试上述各类，看能否正常运行。
- (7) 构建必要的辅助类，实现对本科生、研究生和助教博士生的添加、修改、删除、查询管理。
- (8) 根据需要定义类的构造函数、析构函数、拷贝构造函数、成员函数。必要时重载函数。
- (9) 要求将 Person 类设置为虚基类，以消除其派生类成员访问的二义性问题（注意在虚基类各级派生类的构造函数实现时调用虚基类的构造函数）。
- (10) 要求在 Person 类中定义虚函数 displayDetails（），用于显示当前对象的信息；同时定义虚函数 inputData（），用于从键盘获取当前对象的信息。Person 类所有派生类也要定义同名虚函数，使程序可以实现动态多态性。
- (11) 用菜单方式设计主控模块程序。
- (12) 对程序源代码要给出各部分的详细注释，这也是该题目的考核重点之一。
- (13) 用 UML 语言描述系统用到的类及其关系。

### 6.3 【设计功能的实现】（用 C 或 C++ 语言描述）

//说明：此内容由学生自己设计完成。

//以下代码仅供参考。

程序框架:

```
/*  
Copyright (C), 2019, Tyut  
File name:      main.cpp  
Author: Tyut      Version: 1.0      Date: 2019.5.28  
Description:    应用程序主函数  
*/
```



```

        cout<<" *****"<<endl;
        cout<<"*|*|          感谢使用学生管理系统          |*|*"<<endl;
        cout<<" *****\a"<<endl;
    }

```

```

/*****

```

Copyright (C), 2019, Tyut

File name:        undergraduateManager.h

Author: Tyut        Version: 1.0        Date: 2019.5.28

Description:    本科生管理类

```

*****/

```

```

#ifndef UNDERGRADUATE_MANAGER_H

```

```

#define UNDERGRADUATE_MANAGER_H

```

```

#include <iostream>

```

```

#include <string>

```

```

#include <fstream>

```

```

#include "undergraduate.h"

```

```

using namespace std;

```

```

/* Define a Class : UndergraduateManager 本科生管理类*/

```

```

class UndergraduateManager

```

```

{

```

```

    private:

```

```

        int top; //记录指针

```

```

        Undergraduate undergraduates[100]; //本科生记录

```

```

    public:

```

```

        UndergraduateManager();//构造函数,将 Undergraduate.txt 读到 undergraduates[]中

```

```

        int queryByNo(string sno);//按本科生号查找 //找到: 返回数组下标//没找到: 返回-1

```

```

        void clearStudent(); //删除所有本科生信息

```

```

        int addStudent(Undergraduate s); //添加本科生,需要先查找是否存在

```

```

        int modifyStudent(string sno); //修改学生信息 ,需要先查找是否存在

```

```

        int deleteStudent(string sno);//删除本科生, 删除前先查找其是否存在

```

```

        int queryStudent(string sno);//查找本科生,查到则显示,否则提示未查到

```

```

        void displayAll();//输出所有本科生信息

```

```

        void dataManage(); //本科生库维护

```

```

        void dataSave();

```

```

        void dataRead();

```

```

        ~UndergraduateManager();//析构函数,将 undergraduates[]写入 Undergraduate.txt 文件中

```

```

};

```

```

//构造函数,将 Undergraduate.txt 读到 undergraduates[]中

```

```

UndergraduateManager::UndergraduateManager()

```

```

{

```

```

    dataRead();

```

```

}

```

```

//按本科生号查找

```

```

//找到：返回数组下标
//没找到：返回-1
int UndergraduateManager::queryByNo(string sno)
{
    for(int i=0;i<=top;i++)
        if (undergraduates[i].getStudentNo()==sno)
            return i;
    return -1;
}
//删除所有本科生信息
void UndergraduateManager::clearStudent()
{
    top=-1;
}
//添加本科生,需要先查找是否存在
int UndergraduateManager::addStudent(Undergraduate s)
{
    int p=queryByNo(s.getStudentNo());
    if (p==-1)
    {
        top++;
        undergraduates[top]= s;
        dataSave();//保存
        return 1;
    }
    else
    {
        cout<<"----->此学生已经存在 !<-----"<<endl<<endl;
        return 0;
    }
}
//修改科生，删除前先查找其是否存在
int UndergraduateManager::modifyStudent(string sno)
{
    int p=queryByNo(sno);
    if (p==-1)
    {
        cout<<"----->此学生不存在 !<-----"<<endl<<endl;
        return 0;
    }
    else
    {
        cout << "请输入该生的新信息: " << endl<<endl;
        undergraduates[p].inputData();
    }
}

```

```

        dataSave();//保存
        return 1;
    }

}

//删除本科生，删除前先查找其是否存在
int UndergraduateManager::deleteStudent(string sno)
{
    int p=queryByNo(sno);
    if (p==-1)
    {
        cout<<"----->此学生不存在 !<-----"<<endl<<endl;
        return 0;
    }
    else
    {
        for(int i = p; i < top ; i++)
            undergraduates[i]=undergraduates[i+1];
        top--;
        cout << "----->删除完成!<-----" << endl<<endl;
        dataSave();//保存
        return 1;
    }
}

//查找科生
int UndergraduateManager::queryStudent(string sno)
{
    int p=queryByNo(sno);
    if (p==-1)
    {
        cout<<"----->此学生不存在 !<-----"<<endl<<endl;
        return 0;
    }
    else
    {
        cout<<"----->此学生存在:<-----"<<endl<<endl;
        undergraduates[p].displayDetails();
        return 1;
    }
}

//输出所有本科生信息
void UndergraduateManager::displayAll()
{
    for (int i=0;i<=top;i++)

```





```

        cout << " 请输入学号:";
        cin >> sstudentNo;
        deleteStudent(sstudentNo);
        break;
    case 4:
        cout << " 请输入学号:";
        cin >> sstudentNo;
        queryStudent(sstudentNo);
        break;
    case 5:
        displayAll();
        break;
    case 6:
        clearStudent();
        break;
    default:
        break;
    }
}

void UndergraduateManager::dataSave()//存储资料函数,将 read[]写入 Undergraduate.txt 文件中
{
    fstream file("Undergraduate.dat",ios::out);
    for (int i=0;i<=top;i++)
        file.write((char *)&undergraduates[i],sizeof(undergraduates[i]));
    file.close();
}

void UndergraduateManager::dataRead() //构造函数,将 Undergraduate.txt 读到 read[]中
{
    Undergraduate s;
    top=-1;
    fstream file("Undergraduate.dat",ios::in);
    while (1)
    {
        file.read((char *)&s,sizeof(s));
        if (!file) break;
        top++;
        undergraduates[top]=s;
    }
    file.close();
}

#endif //UNDERGRADUATE_MANAGER_H

```

```
/******
```

Copyright (C), 2019, Tyut

File name: date.h

Author:Tyut Version: 1.0 Date: 2019.5.28

Description: 日期类

```
*****/
```

```
#ifndef DATE_H
```

```
#define DATE_H
```

```
#include <iostream>
```

```
using namespace std;
```

```
/* Define a Class : Date*/
```

```
/* with attributes: year, month, and day, and */
```

```
/* operations: getYear, getMonth,... */
```

```
class Date
```

```
{
```

```
private:
```

```
int year;
```

```
int month;
```

```
int day;
```

```
public:
```

```
Date(){year = 0; month = 0; day = 0;}
```

```
Date(int yy,int mm,int dd){year = yy; month = mm; day = dd;}
```

```
Date(Date& d){year = d.year; month = d.month; day = d.day;}
```

```
~Date(){} 
```

```
void setYear(int yy){ year = yy;}
```

```
void setMonth(int mm){ month = mm;}
```

```
void setDay(int dd){ day = dd;}
```

```
int getYear(){return year;}
```

```
int getMonth(){return month;}
```

```
int getDay(){return day;}
```

```
void inputDate()
```

```
{
```

```
cout << "年: "; cin >> year;
```

```
cout << "月: "; cin >> month;
```

```
cout << "日: "; cin >> day;
```

```
}
```

```
void displayDate(){ cout << year << "/" << month << "/" << day << endl; }
```

```
};
```

```
#endif //DATE_H
```

```
/******
```

Copyright (C), 2010, Tyut

File name: person.h

Author: Tyut      Version: 1.0      Date: 2019.5.28

Description:    人员类

\*\*\*\*\*/

```
#ifndef PERSON_H
#define PERSON_H
#include <iostream>
#include <cstring>
#include <string>
using namespace std;
/* Define a Class : Person */
/* with attributes: name, id,gender, birthday */
/* operations: getName, getId, and setId. */
/* person is the base class */
//该类为虚基类
class Person
{   protected:
    char name[20];//姓名
    char id[18];  //身份证号
    char gender[2];//性别
    Date birthday;//出生日期
public:
    Person();
    Person(char* sname, char* sid, char* sgender,int yy,int mm ,int dd);
    ~Person(){}
    void setName(char* sname){ strcpy(name,sname); }
    char* getName(){ return name; }
    void setId(char* sid){ strcpy(id,sid); }
    char* getId(){ return id;}
    void setGender(char* sgender){ strcpy(gender,sgender); }
    char* getGender(){ return gender; }
    void setBirthday(Date d){ birthday = d; }
    Date getBirthday() { return birthday; }
    virtual void inputData();//输入数据
    virtual void displayDetails();    //显示数据

};
Person::Person()
{
    strcpy(name,"NA");
    strcpy(gender,"男");
    strcpy(id,"000");
    //注意： 这里没有给 birthday 赋值,系统会调用其默认构造函数
    //也可以
    //Date d(1980,1,1);
```

```

        //birthday = d;
    }
Person::Person(char* sname, char* sid, char* sgender,int yy,int mm ,int dd):birthday(yy,mm,dd)
{
    strcpy(name,sname);
    strcpy(id,sid);
    strcpy(gender,sgender);
}
void Person::inputData()
{
    cout << "姓名: ";
    cin >> name ;
    cout << "身份证号: ";
    cin >> id ;
    cout << "性别: ";
    cin >> gender ;
    cout << "出生日期: " << endl;
    birthday.inputDate();
}

void Person::displayDetails()
{
    cout << "姓名: " <<name <<endl;
    cout << "身份证号: " <<id <<endl;
    cout << "性别: " <<gender<<endl;
    cout << "出生日期: ";
    birthday.displayDate();
}
#endif //PERSON_H

/*****
Copyright (C), 2010, Tyut
File name:      teacher.h
Author: Tyut      Version: 1.0      Date: 2019.5.28
Description:    教师类
*****/

#ifndef TEACHER_H
#define TEACHER_H
#include <iostream>
#include <string>
#include <cstring>
#include "person.h"
using namespace std;
/* Define a Class : Teacher */

```

```

/* with attributes:    */
/* operations:        */
class Teacher: virtual public Person
{
    protected:
        char teacherNo[5];    //教师编号
        char schoolName[20];  //学校名称
        char department[20];  //部门
    public:
        Teacher();
        Teacher(char* sname, char* sid, char* sgender,int yy,int mm ,int dd
                ,char* steacherNo,char* sschool,char* sdepartment);
        ~Teacher(){}
        void setTeacherNo(char* sno){ strcpy(teacherNo,sno); }
        char* getTeacherNo(){ return teacherNo;}
        void setSchoolName(char* sschool){ strcpy(schoolName,sschool); }
        char* getSchoolName(){ return schoolName; }
        void setDepartment(char* sdepartment){ strcpy(department,sdepartment); }
        char* getDepartment(){ return department; }
        virtual void inputData();//输入数据
        virtual void displayDetails();    //显示数据
};

Teacher::Teacher():Person()
{
    strcpy(teacherNo,"001");
    strcpy(schoolName,"defaultSchoolName");
    strcpy(department,"defaultDepartment");
}

Teacher::Teacher(char* sname, char* sid, char* sgender,int yy,int mm ,int dd
                ,char* steacherNo,char* sschool,char* sdepartment)
    :Person(sname,sid,sgender,yy,mm,dd)
{
    strcpy(teacherNo,steacherNo);
    strcpy(schoolName,sschool);
    strcpy(department,sdepartment);
}

void Teacher::inputData()
{
    Person::inputData();
    cout << "教师编号: " ;cin >> teacherNo ;
    cout << "学校: " ;cin >> schoolName ;
    cout << "部门: " ;cin >> department ;
}

void Teacher::displayDetails()
{

```

```

        Person::displayDetails();
        cout << "教师编号: " << teacherNo << endl;
        cout << "学校: " << schoolName << endl;
        cout << "部门: " << department << endl;
    }
#endif //TEACHER_H

/*****

Copyright (C), 2019, Tyut
File name:      student.h
Author: Tyut      Version: 1.0      Date: 2019.5.28
Description:     学生类
*****/

/*****

Copyright (C), 2019, Tyut
File name:      graduate.h
Author: Tyut      Version: 1.0      Date: 2019.5.28
Description:     研究生类
*****/

/*****

Copyright (C), 2019, Tyut
File name:      undergraduate.h
Author: Tyut      Version: 1.0      Date: 2019.5.28
Description:     本科生类
*****/

/*****

Copyright (C), 2019, Tyut
File name:      ta.h
Author: Tyut      Version: 1.0      Date: 2019.5.28
Description:     助教博士生类
*****/

```

#### 6.4 【实例测试及运行结果】

输入多个本科生、研究生和助教博士生的数据。

//说明：此内容由学生自己设计完成。

#### 6.5 【实现提示】

//说明：学生自己补充。



太原理工大学  
TAIYUAN UNIVERSITY OF TECHNOLOGY

## 课程设计

课程名称： 程序设计课程设计

设计名称： \_\_\_\_\_

专业班级： \_\_\_\_\_ 学号： \_\_\_\_\_

学生姓名： \_\_\_\_\_

指导教师： \_\_\_\_\_

2019 年 6 月 21 日