**序 号 77**

**上 海 大 学**

**2021-2022冬季学期**

**《数据结构（1）》（08305009）**

**课程考核报告**

学 号： 20123101

姓 名： 李昀哲

课程考核评分表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 内容 | 分值 | 成绩 |
| 1 | 第一题 | 40 |  |
| 2 | 第二题 | 40 |  |
| 3 | 课程总结 | 10 |  |
| 4 | 报告规范性 | 10 |  |
| 考核成绩 | |  | |
| 评分人 | |  | |

计算机工程与科学学院

2022 年 4 月

**一、考核题目**

**（一）病毒株种类**

[问题描述]

2019年末一种从未出现过的新型病毒开始在全球迅速蔓延，并且不断变异，人类社会面临了极大的威胁。全球科学家开始共同抗疫。科学家通过研究发现，两种病毒之间可能存同类关系和相克关系。而且有以下二点是肯定的。第一，病毒A的同类病毒的同类病毒也一定是病毒A的同类；第二，病毒A的相克病毒的相克病毒也一定是病毒A的同类病毒。两种病毒是一个病毒株的当且仅当它们是同类。现在给你一些关于病毒关系的信息，问你至多有多少种不同病毒株。

[输入数据]

输入的第一行为N(2<=N<=1000)，表示病毒的种类数（从1编号到N）。

第二行M(1<=M<=100000)，表示病毒关系信息的条数。

以下M行，每行可能是S p q或是H p q，分别表示p和q是同类病毒，或是相克病毒。 假设输入不会产生矛盾。

[输出数据]

输出只有一行，表示最大可能的病毒株数。

[输入样例]

6

4

S 2 4

H 1 3

H 3 5

S 1 6

[输出样例]

3

[样例说明]

该样例最多有3个病毒株，分别是1、5、6号病毒为一个病毒株，2、4号病毒为一个病毒株，3号病毒为一个病毒株。

[测试数据要求]

输入数据在virus.in 文件中；输出数据在virus.out文件中。

**（二）核酸检测系统**

1. **系统功能**

2022年3月奥密克戎变异株迅速席卷东方帝国的魔都，攻破了魔都苦心构筑多年的精准防疫铜墙铁壁。魔都人民奋起抗击，开展了全民覆盖的核酸检测。为了更迅速开展检测，寻找确诊者、密接者和次密接者，需要开发一套核酸检测系统。系统需要有以下6个功能：

(1)排队：输入人员代码，并选择混合测试还是单人测试。将人员代码加入到相应队伍排队。人员代码是一个8位（xxxyyyyz）的数字，其中xxx表示楼栋号（000~999）；yyyy表示房间号，例如：0801、1801等；z代表一个房间中人员序号（1~9）。

(2)检测：选择混合测试还是单人测试。相应队伍中最前面的人员进行测试，混合测试每10人一个测试管；单人测试一人一管。每个测试管对应一个管号，管号是一个5位（kbbbb）的数字，其中k为0表示混合测试管，k为1表示单人测试管；bbbb是一个流水号，从0000开始自动生成。

(3)查看排队情况：按排队先后顺序，分别显示混合测试和单人测试排队人员的代码。

(4)登记测试管信息：输入测试管编号和测试结果。混合测试结果分为阴性、阳性和可疑三种。对阴性测试管对应的人员标记阴性状态；对阳性单人管测试人员标记确诊状态；对阳性混合管和可疑管人员标记可疑状态。并且对于确诊人员，其同一栋楼人员以及测试时排在他前面的10人和后面的人设置为密接者；密接者的同一栋楼人员为次密接者。

(5)各类人员查询：可以分类显示阴性、确诊、可疑、密接、次密接、待上传结果、在排队7种状态的人员代码。

(6)个人查询：输入人员代码，显示其当前状态。当前状态包括：阴性、确诊、可疑、密接、次密接、等待上传结果、在排队和未检测8种状态（未检测状态表示他没有参加排队检测）。

1. **初始化数据文件**

为了方便测试，系统启动时可以从文件读入初始排队信息和检测信息。初始排队信息文件第一行二个正整数n和m（1<n,m<10000），分别混合测试和单管测试的排队人数。接下来n行是混合测试人员的编号，在混合测试人员编号后面是m行单管测试人员信息。初始测试信息文件中有一行二个正整数x(0<x<=n)和y(0<y<=m)，分别表示已经完成混合测试和单人管测试的人数。

**二、考核要求**

题目一：

完成算法的设计和实现，准备好5组测试用例。在报告中介绍主要算法的思想和用到的主要数据结构；每一组测试用例需要有输入数据和输出数据的文件；并提供C++源程序和可执行代码。

题目二：

根据系统功能描述，采用模块化程序设计方法进行程序设计，要求程序结构清晰。上述各个功能模块要求分别用函数实现，在主函数中通过调用这些函数，完成系统功能的要求。代码书写要规范，有简要的注释，给出函数说明。

设计报告内容包括总体设计、详细设计、源代码和测试情况。

总体设计：对程序的整体设计思路进行描述，画出系统的总体功能模块图，说明系统使用的主要数据结构，列表给出需要用到的函数并描述其功能。

详细设计：画出函数调用关系图，分析并描述函数的功能。

测试情况：记录程序编写和调试过程中遇到的各种问题，以及解决这些问题的途径和方法。并提交适当运行的截屏。

课程总结：

在课程报告最后写一个课程总结。回顾整个《数据结构(1)》课程的学习过程，对课程学习进行总结，给出个人体会。

**《数据结构（1）》（08305009）课程考核报告**

#### 题目一：

##### 算法思想

从手算角度出发，不难发现解题的关键在于将一种病毒相克的相克作为同类。由此考虑算法设计：初始情况下每个病毒看作独立个体，对输入“病毒对”之间的关系进行观测，观测得到不同的关系，做出以下处理：

1. 观测到病毒关系为S时，直接将二者看作同类，将二者连通;

2. 观测到病毒关系为H时，两种病毒分别为(v1,v2)。固定一个病毒，假设固定v2，遍历和v1相克的病毒，作为v2的同类病毒，使二者连通；类似地，再固定v1，遍历和v2相克的病毒作为v1的同类，使二者连通。

3. 计算连通集合的个数，即毒株数量最大的个数。

##### 二、主要数据结构

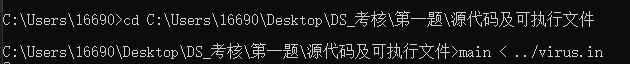
考虑到连通、合并以及等价类的问题，故采用并查集对病毒进行操作。初始化时，每一个病毒看作是一个独立的等价类。

对于遍历某一个病毒的相克病毒，采用图中的邻接矩阵存储相克的信息，称之为“相克病毒邻接矩阵”。需要注意的是，填入当前相克病毒的信息前，需要先对当前病毒进行遍历后再填入，就可以避免本条信息带来的干扰。

##### 三、测试数据

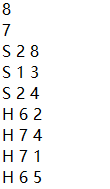
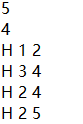
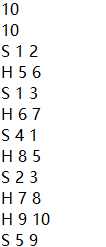
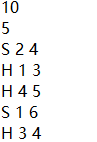
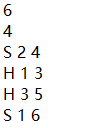
输入时，可以用标准输入重定向，从文件输入，如图1所示。具体步骤为：

1. 打开cmd。 cd 可执行程序main.exe所在文件夹
2. Main < ../virus.in

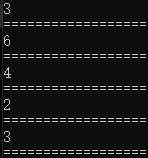


**图1 标准输入重定向**

输入数据：如图1所示，第一组为测试样例；第二组为有多个单独等价类；第三个为有多个关系；后两组为有较多相克信息。测试结果如图2所示。



**图2第一题输入数据**



**图3第一题输出结果**

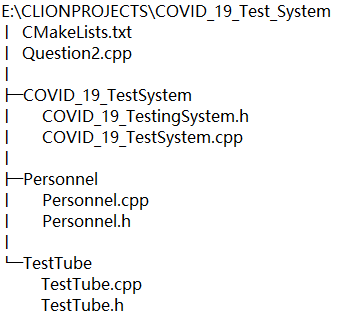
#### 题目二：

##### 总体设计

总体流程为：构造核酸检测系统（构造时从文件初始化已检测信息）；初始化队列人员信息；排队做核酸；试管检测；根据检测结果确定检测人员状态（密接、次密接、确诊、阴性等）；输出结果。

###### 类的设计

总体设计框架如图4所示，根据题目要求，共涉及了三个类，分别为Personnel（人员）类，TestTube（测试管）类，以及组合前二者综合设计的COVID\_19\_TestingSystem（核酸检测系统）类。编译执行的目标文件为Question2.cpp

****

**图4 核酸检测系统文件框架**

Personnel(人员)作为组成核酸检测系统的核心，类的成员定义如下表所示：

|  |  |  |
| --- | --- | --- |
| **数据成员类型** | **名称** | **用途** |
| **Protected** | vector<Personnel> close\_contact\_in\_queue\_ | **维护排队时的密接者群体** |
| **Protected** | std::string personnel\_code\_ | **存储人员代码** |
| **Protected** | STATUS status\_ | **存储人员状态** |
| **成员函数类型** | **名称** | **用途** |
| **Public** | void AddCloseContactPersonnel(const Personnel& personnel) | **添加排队时密接人员** |
| **Public** | vector<Personnel> GetCloseContactInQueue() const | **获取排队密接人员** |
| **Public** | STATUS GetStatus() | **获取当前人员状态** |
| **Public** | std::string GetCode() | **获取当前人员代码** |
| **Public** | void SetStatus(STATUS status) | **设置人员状态** |
| **Public** | Personnel& operator=(const Personnel& personnel) | **重载赋值运算符** |

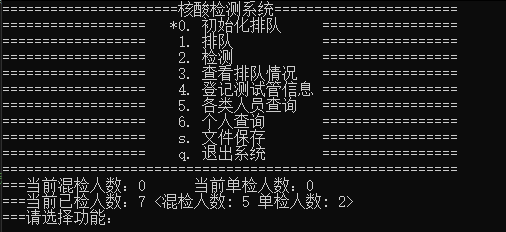
TestTube（测试管）为保存检测结果的容器和反映指标，其类的成员定义如下表所示：

|  |  |  |
| --- | --- | --- |
| **数据成员类型** | **名称** | **用途** |
| **Protected** | bool is\_tested | **标记该管是否已被检测** |
| **Protected** | TEST\_RESULT test\_result\_ | **检测结果** |
| **Protected** | int test\_mode\_ | **检测模式（混检/单检）** |
| **Protected** | static int total\_mixed\_test\_tube\_amount\_ | **混合管总数（用于生成流水号）** |
| **Protected** | static int total\_single\_test\_tube\_amount\_ | **单检管总数（用于生成流水号）** |
| **Protected** | std::string tube\_number\_ | **测试管编号** |
| **Protected** | vector<Personnel> tested\_personnels | **该管检测组人数（上限为10）** |
| **成员函数类型** | **名称** | **用途** |
| **Protected** | std::string ProduceTubeNum(int test\_tube\_number); | **生成该管编号** |
| **Public** | bool IsTested() | **判断是否该管被检测** |
| **Public** | void SetTested | **将该管设置为已检测** |
| **Public** | vector<Personnel> GetPersonnelInThisTube() | **获取该管的所有人员** |
| **Public** | void SetTestResult(TEST\_RESULT test\_result) | **设置检测结果** |
| **Public** | void OpenOneTube(int test\_mode, vector<Personnel> personnels) | **启封一个新的测试管** |
| **Public** | TEST\_RESULT GetTestResult() | **获取检测结果** |
| **Public** | std::string GetTubeNum() | **获取测试管编号** |

核酸检测系统类的设计较为复杂不作为“类”来介绍，将在函数功能模块列表介绍。

###### 功能模块设计

功能模块在囊括了题目要求的基础上，添加了保存文件等功能，功能菜单如图5所示。在主函数中，每个功能模块在类内封装为各个单独的函数，使用者仅能调用这个功能接口，而不能操作该检测系统中其他任何函数。



**图5 核酸检测系统功能菜单**

考虑实际因素，菜单的设计在下方会实时显示队列人数和已完成的情况。

下面将对每个功能简要介绍：

1. 初始化排队，即，从文件中初始化排队人员信息。功能选择后，将提示初始化的文件信息。
2. 排队，即，自主添加人员加入队列进行核酸检测。因时间仓促暂不支持输入重复的人员代码。
3. 检测，即，将队列中的所有人进行检测，出队，并将状态设置为“待上传结果”。
4. 查看排队情况，即，输出各个队列的顺次信息、人员代码、队列人数。
5. 登记测试管信息，即，对功能２中检测出队的人员的测试管进行结果设置，设置比例为１．２％为阳性。
6. 各类人员查询，即，输出每个状态的人员代码。
7. 个人查询，即，根据人员代码输出其状态。

s．文件保存，即，将当前的人员信息（代码＋状态）保存入指定文档。

q．退出系统。

###### 数据结构设计

　本题的核心为排队做核酸，故使用队列作为核心数据结构，在出队和入队时加入该系统特定的操作，完成相应的功能。

　同时使用线性表，实现随机存取各类人员的信息，操作便捷。

###### 各个函数功能描述

４.１私有成员函数

私有成员函数为公有函数的功能接口提供功能支持，但外界不能直接调用；

下一节会介绍各个函数的调用关系。（注：图中相同色块表明被同一个功能函数调用）

|  |  |
| --- | --- |
| **私有成员函数** | **功能** |
| void UpdateHistoryInfo() | 更新历史信息 |
| void UpdateAmount(std::ifstream& inFile) | 更新已检测人数 |
| void UpdateTestedResult(std::ifstream& inFile) | 更新测试管信息 |
| void AddTestedPersonnel(const Personnel& personnel) | 添加已检测人员 |
| int GetTestedMixedAmount() const | 获取完成混检人数 |
| int GetTestedSingleAmount() const | 获取完成单检人数 |
| int GetTestedPersonnelAmount() const | 获取完成检测总人数 |
| int GetQueueAmount(QUEUE\_TYPE queue\_type) | 获取各队伍人数 |
| void LineUpFromFile(const std::string& file\_name) | 从文件中获取队列 |
| void EnQueue(QUEUE\_TYPE queue\_type, Personnel personnel) | 入队 |
| void DelQueue(QUEUE\_TYPE queue\_type) | 出队检测 |
| void TestNuclearAcid(QUEUE\_TYPE queue\_type, vector<Personnel>& testing\_personnels) | 检测台接受检测 |
| void DisplayQueueing() | 展示队伍情况 |
| int GetTestedTubeAmount() | 获取已检测测试管数量 |
| void RandomizeTubeInfo() | 随机选择阳性管 |
| void SetTubeResult(TestTube& tube, TEST\_RESULT result) | 设置测试管结果 |
| void DisplayTube() | 展示测试管信息 |
| void DecidePersonnelStatus() | 决定各人员状态 |
| vector<std::string> DecideCloseContact(vector<std::string> confirmed\_blocks) | 确定密接人员 |
| void DecideSubCloseContact(vector<std::string> close\_contact\_blocks) | 确定次密接人员 |
| void DecideNegative() | 确定阴性人员 |
| void UpdateAllTypeInfo() | 更新所有人员信息 |
| void DisplayType(STATUS status) | 根据类型展示人员信息 |
| void DisplayStatus(const std::string& personnel\_code) | 根据人员代码展示信息 |
| STATUS GetPersonnelStatus(const std::string& personnel\_code) | 根据人员代码人员状态 |
| void SaveToFile(const std::string& file\_name) | 保存入文件 |
| void DisplayPersonnel() | 展示所有人员信息 |
| void DisplayPersonnelCloser() | 展示排队密接者信息 |
| bool IsEmpty(QUEUE\_TYPE queue\_type) | 队列判空 |

４.２公有成员函数

如上所述，检测系统的公有函数只有功能接口，不能对任何其他成员进行操作。

|  |  |
| --- | --- |
| **公有成员函数** | **用途** |
| void DisplayMenu(); | 展示菜单 |
| void InitQueue(); | 根据文件初始化队列 |
| void AddQueuePersonnel(); | 添加队列人员 |
| void Testing(); | 检测 |
| void ShowQueueStatus(); | 查看排队情况 |
| void RegisterPersonnelInfo(); | 登记人员信息 |
| void DisplayAllTypePersonnel(); | 展示各类型人员 |
| void DisplaySpecificPersonnel(); | 根据人员代码展示人员信息 |
| void Save(); | 保存 |

##### 二、详细设计

1．各个函数的调用关系图

|  |
| --- |
| **函数关系图** |
| **DisplayMenu()**  |――－int GetTestedMixedAmount() const  |―――int GetTestedPersonnelAmount() const  |―――int GetQueueAmount(QUEUE\_TYPE queue\_type)  |―――int GetTestedSingleAmount() const |
| **void InitQueue()**  ｜―――void LineUpFromFile(const std::string& file\_name) |
| **void AddQueuePersonnel()**  ｜―――void EnQueue(QUEUE\_TYPE queue\_type, Personnel personnel) |
| **void Testing()**  ｜―――void DelQueue(QUEUE\_TYPE queue\_type)  ｜―――void TestNuclearAcid(QUEUE\_TYPE queue\_type, vector<Personnel>& testing\_personnels) |
| **void ShowQueueStatus()**  ｜―――void DisplayQueueing() |
| **void RegisterPersonnelInfo()**  ｜―――int GetTestedTubeAmount()  ｜―――void RandomizeTubeInfo()  ｜―――void SetTubeResult(TestTube& tube, TEST\_RESULT result)  ｜―――void DisplayTube()  ｜―――void DecidePersonnelStatus()  ｜―――vector<std::string> DecideCloseContact(vector<std::string> confirmed\_blocks)  ｜―――void DecideSubCloseContact(vector<std::string> close\_contact\_blocks)  ｜―――void DecideNegative() |
| **void DisplayAllTypePersonnel()**  ｜―――void UpdateAllTypeInfo()  ｜―――void DisplayType(STATUS status) |
| **void DisplaySpecificPersonnel()**  ｜―――void DisplayStatus(const std::string& personnel\_code)  ｜―――STATUS GetPersonnelStatus(const std::string& personnel\_code) |
| **void Save()**  ｜―――void SaveToFile(const std::string& file\_name) |

##### 2. 函数设计

对于一些语句、逻辑简单的函数，就不过多赘述其设计。本节将简要描述一些有特殊功能的函数设计。

###### 2.1生成测试管编号

生成测试管编号为TestTube类的私有成员函数，其中使用到了静态成员变量用于记录测试管使用的数量以生成测试管的流水号。静态成员变量是面向对象程序设计中一个设计十分巧妙的语法，使得所有实例化得类对象共有一个成员变量。

|  |
| --- |
| std::string TestTube::ProduceTubeNum(int test\_tube\_number)  {  std::string tube\_number = std::to\_string(test\_mode\_); // 强制转换为string  if (test\_tube\_number == 0) // 0时单独确定编号  tube\_number += "000";  while (test\_tube\_number < 1000 && test\_tube\_number != 0) // 确定一般数字的编号  {  test\_tube\_number \*= 10; // 确定位数  tube\_number += "0";  }  if(test\_mode\_ == 1)  tube\_number += std::to\_string(total\_single\_test\_tube\_amount\_); // 操作静态变量  if (test\_mode\_ == 0)  tube\_number += std::to\_string(total\_mixed\_test\_tube\_amount\_);  return tube\_number;  } |

###### 2.2入队、出队时确定排队时得密接者

由于题目规定：对于阳性患者，其排队时的前十人和他后面的所有人都会是密接者，故，无法将确定排队密接者的过程解耦，只能在入队、出队时添加耦合地操作进行维护。

|  |
| --- |
| void COVID\_19\_TestingSystem::EnQueue(QUEUE\_TYPE queue\_type, Personnel personnel) // 入队  {  personnel.SetStatus(STATUS::QUEUING);  queue<Personnel> temp\_queue; // 临时辅助队列  vector<Personnel> temp\_close\_contact\_personnels;  if (queue\_type == QUEUE\_TYPE::SINGLE)  {  temp\_queue = single\_testing\_queue\_;  while (!temp\_queue.empty() && personnel.GetCloseContactInQueue().size() < 10)  {  personnel.AddCloseContactPersonnel(temp\_queue.front()); // 当前入队人前面的10人密接  temp\_queue.pop();  }  single\_testing\_queue\_.push(personnel);  } else if (queue\_type == QUEUE\_TYPE::MIXED) // 操作类似，此处省略  } |

|  |
| --- |
| void COVID\_19\_TestingSystem::DelQueue(QUEUE\_TYPE queue\_type) // 出队  {  vector<Personnel> temporary\_testing\_group;  queue<Personnel> temp\_queue;  if (queue\_type == QUEUE\_TYPE::SINGLE)  {  temp\_queue = single\_testing\_queue\_;  temp\_queue.pop();  if (single\_testing\_queue\_.empty())  std::cout << "无人排队" << std::endl;  temporary\_testing\_group.push\_back(single\_testing\_queue\_.front()); // 获取队首的人  while (!temp\_queue.empty()) // 其后的人都加入临近群体中  {  temporary\_testing\_group[0].AddCloseContactPersonnel(temp\_queue.front());  temp\_queue.pop();  }  TestNuclearAcid(queue\_type, temporary\_testing\_group); // 大白核酸检测  tested\_single\_amount\_++;  single\_testing\_queue\_.pop(); // 检测完毕，出队  } else if ... // 操作类似  } |

###### 2.3确定各人员状态

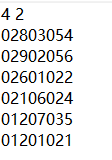
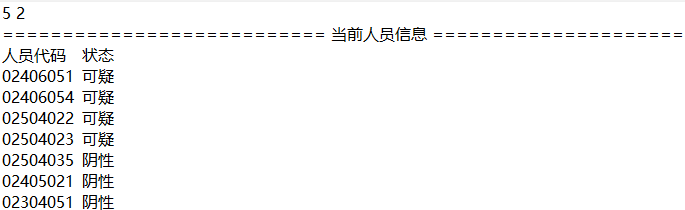
本题另一个难点在于每个人状态的确定，这势必要花费极大的时间复杂度去查询每一个可能的密接、次密接者。联系实际也不难发现，疫情之下相关医护保障人员工作的艰辛。

此处对于确诊者的代码中的楼号作为线索搜寻密接者，再采用类似的做法确定次密接。上述确定的排队密接者也会在确定密接者函数中协同操作。

|  |
| --- |
| // 决定人员状态  void COVID\_19\_TestingSystem::DecidePersonnelStatus()  {  // 确定确诊楼号，标记确诊、可疑人员  vector<std::string> confirmed\_blocks;  for (auto& tube : tested\_tube\_box\_)  {  if (tube.GetTestResult() == TEST\_RESULT::POSITIVE && tube.GetPersonnelInThisTube().size() == 1) // 单人管阳性  {  Personnel personnel = tube.GetPersonnelInThisTube()[0];  for (auto& target\_personnel : tested\_personnels\_)  {  if (target\_personnel.GetCode() == personnel.GetCode())  {  target\_personnel.SetStatus(STATUS::COMFIRMED); // 单人管确诊  }  if (target\_personnel.GetStatus() == STATUS::COMFIRMED)  {  confirmed\_personnels\_.push\_back(target\_personnel);  confirmed\_blocks.push\_back(target\_personnel.GetCode()); // 确诊楼号  }  }  } else if ... // 混检类似  vector<std::string> close\_contact\_blocks = DecideCloseContact(confirmed\_blocks);// 根据确诊楼号确定密接  DecideSubCloseContact(close\_contact\_blocks); // 根据密接楼号确定次密接  DecideNegative(); // 以上的补集为阴性  } |

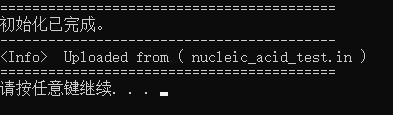
##### 三、功能测试

初始条件：有两个文件line\_up.in和Nucleic\_acid\_test.in，二者信息如图6所示

**图6a line\_up.in信息**  **图6 b Nucleic\_acid\_test.in信息**

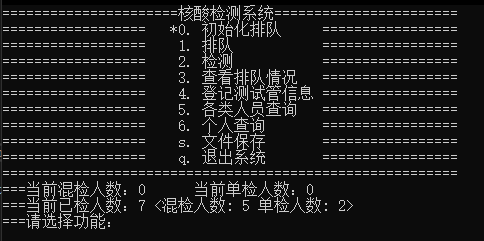
###### 1 初始化



初始化将图6 b中已检测过的信息自动读入。读入完成后显示信息。

注意：这里初始化并不提供自动初始化队列，作为可选项在启动后菜单栏选择。

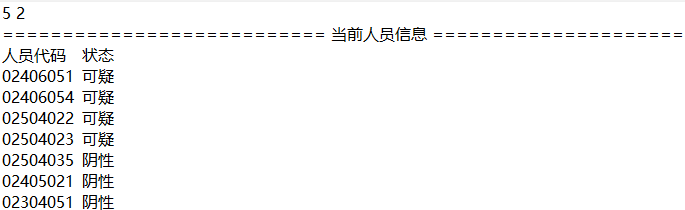
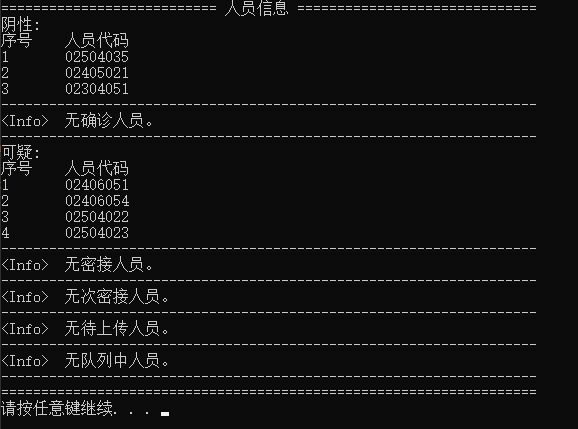
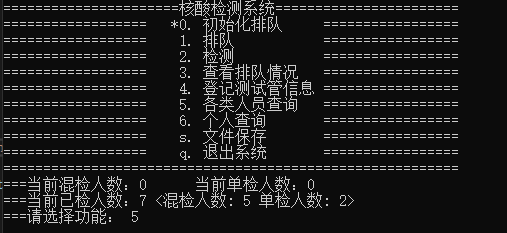
###### 2 进入系统界面



初始化完成后进入系统界面，提示可选菜单内容。

* 1. 直接查看由文件初始化的信息

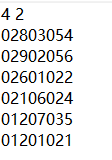
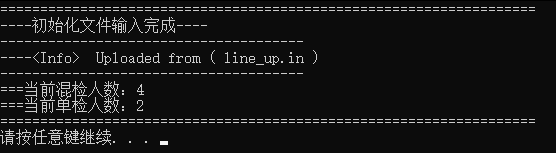
选择功能5，查看初始化是否正确。对比输入文件，发现完全一致。



###### 3 各功能

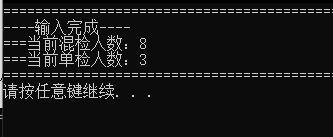
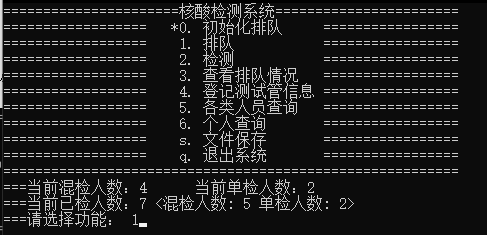
3.1 功能0：由文件初始化队列

选择功能0后，系统自动读入line\_up.in的信息，读入完成后提示。对比输入文件，发现正确。



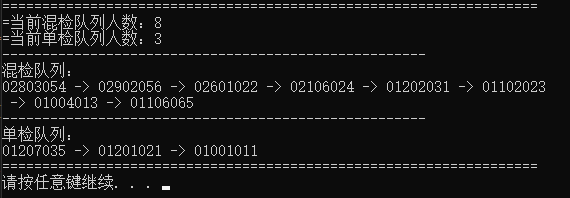
3.2 功能1：自由输入人员信息加入队列（此处输入5人（4+1））

输入前为7（4+2），输入了5（4+1），当前队列结果为11（8+3）



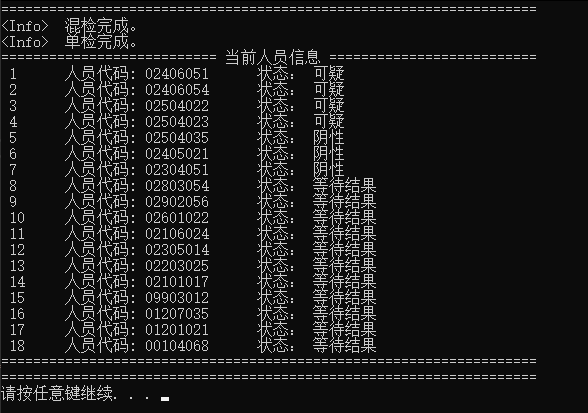
3.3 功能3：查看队列信息（注意！！这是**功能3建议在功能2前执行）**

功能2执行后会清空队列，故直接执行功能3可以显示队列信息。



3.4 功能2：检测

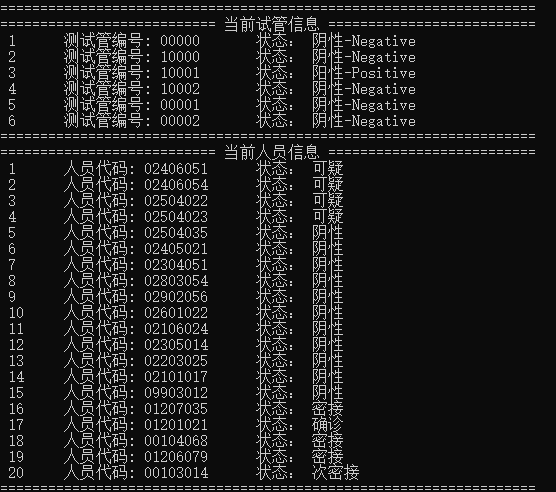
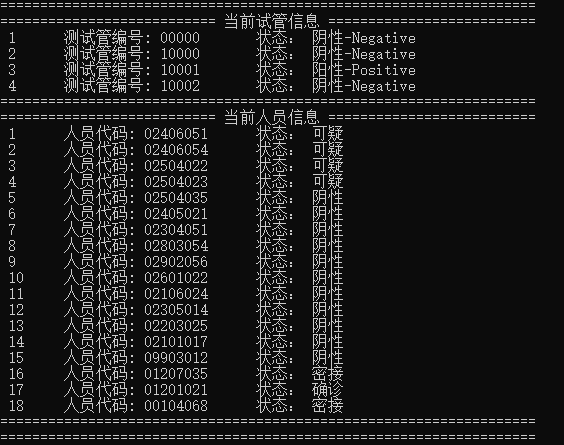
输入2后，完成混检、单检两个队列的检测任务，并显示人员信息。检测后队列会清空，因此功能2执行后直接执行功能3会显示没有队列信息。



3.5 功能4：登记检测信息

此处单人管10001号为阳性，则01201021转变为确诊，同楼宇的01207035转变为密接，队列中较为紧密的00104068也转变为密接。手动添加一名001楼的人员，查看结果，发现为次密接。则检测逻辑正确。

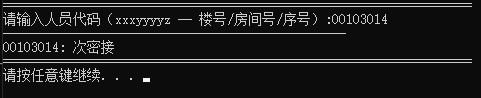
测试中，曾遇到已经检测过的人员检测结果改变的情况，排查后发现，测试管测试后没有被标记，就放回了容器中。故在TestTube类中多加了一个is\_tested属性用于判别，解决问题。



3.5 功能5：各类人员查看

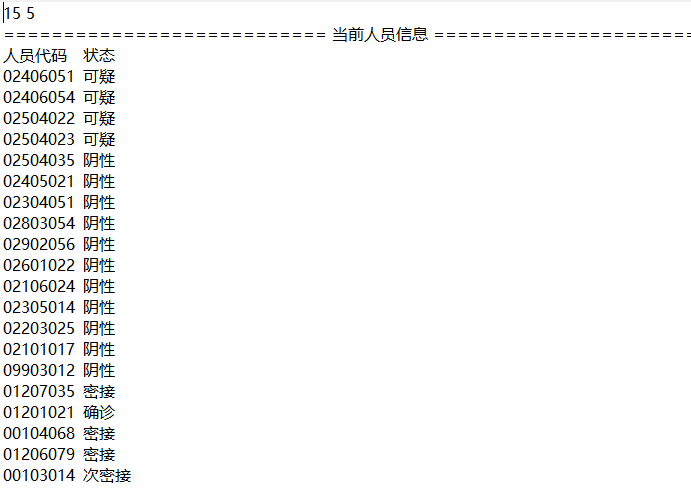


3.5 功能6：指定人员查看



3.5 功能7：保存

查看文件，发现保存可用。



##### 课程总结：

本次课程在两道以疫情为大背景的题目中结束，总体而言，不仅在前期学到了知识，在最后的考核中也得到了应用，提升了实际使用的水平和能力，也对项目有了初步的认识和理解。虽然两道题并不算十分困难，但从各个维度上，对每个人的综合能力都是一次极大的考验。

数据结构是计算机专业的核心课程，相较于课程，我更认为是一种技能、一种理念的引领、一种和实际结合的载体，学期中突然有一节课后就好像茅塞顿开，不应该把数据结构看成是一门让人苦恼的课，而是一种值得反复欣赏把玩的艺术品。上海大学的短学期制，使我们必然要花大量的时间在课程老师的指导之余自学很多内容，虽然强度不小，但对于能力的提升是显而易见的。在课前通过网上资料预习相关内容，课上许老师进行针对性讲解、点拨，我认为效果还是相当不错的。

冬季寒假回来的研讨同样值得延续，这让每个人都有了表达自己观点的权利，教学相长，在和同学的研讨中也能激励我们去探索、发现问题。好奇心是学习过程中非常重要的驱动力。但研讨或许能给每个组多一些时间，对于某些感兴趣的话题，可能在研讨中难免有些讨论和思考，增加一定的时间我认为可以优化这个过程。

在设计核酸检测系统的过程中，结合这几周的核酸检测，这又何尝不是一个简易版的“健康云”呢？其实实际的项目和产品离我们并不遥远，只要善于发现，有很多创意和点子我们都能尝试思考实现。

同时，在设计“决定人员状态”的算法时，发现无论用何种方式，时间、空间复杂度都非常大，需要大量的资源来查找、标记。联系实际，这何尝不是这几周医护人员们的日常工作呢？面对上海日益严重的疫情，全国上下都已赶来支援，庞大的工作量背后，是中华儿女团结一心的民族精神，感谢为了疫情献出自己的医护人员，希望疫情能早日过去，希望校园也能尽早重获活力！