

C++程序设计课程设计

课设报告

**2024/2025(2)**



实验题目 校友录管理系统

学生姓名 马浩俊

学生学号 302023033089

学生班级 软工2305

任课教师 毛国红

提交日期 2025年6月

**计算机科学与技术学院**

**校友录管理系统 实验报告**

# 大型实验的内容

校友录管理系统用于对校友信息进行管理。要求实现对校友信息的增加，对校友信息实现基于姓名的删除，对校友信息实现多维度的查询，对校友信息实现。要求使用学习过的C/C++程序设计的知识完成校友录管理系统的设计与实现。

# 运行环境

校友录管理系统在Clion 2024平台下开发。

操作系统：Windows 11

处理器：AMD Ryzen 7 7840HS w/ Radeon 780M Graphics @ 3.80 GHz

机带：RAM 16.0 GB (13.7 GB 可用)

系统类型：64 位操作系统, 基于 x64 的处理器实验课题分析

# 实验课题分析

## 校友录管理系统的主要功能

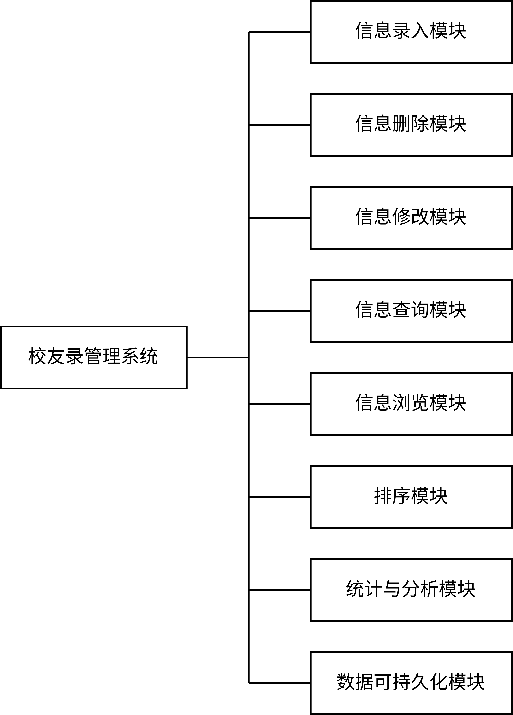
校友录管理系统主要功能为：信息录入、信息删除、信息修改，信息查询，信息浏览，对信息的排序，以及校友信息的统计与分析功能，基于文件存储的数据可持久化功能。

图 1 校友录管理系统主要模块

## 系统各模块的功能：

* + 1. 信息录入模块

用户在主菜单选择“1. 添加校友信息”后，系统会依次提示输入姓名、性别、年龄、年级、院系、班级、地址、电话、QQ、邮箱等信息。用户输入完毕后，系统将新校友信息添加到链表，并自动保存到文件，提示“添加成功”。

* + 1. 信息删除模块

用户选择“2. 删除校友信息”，系统提示输入要删除的校友姓名。输入后，系统查找并删除链表中对应的校友信息节点，操作完成后自动保存数据，并提示“删除完成”。

* + 1. 信息修改模块

用户选择“3. 修改校友信息”，系统提示输入要修改的校友姓名，然后依次输入新的性别、年龄、年级、院系、班级、地址、电话、QQ、邮箱等信息。系统查找该校友并更新其信息，修改后自动保存，并提示“修改成功”。

* + 1. 信息查询模块

用户选择“4. 查询校友信息”，可选择按姓名、年级、院系或班级查询。输入查询条件后，系统支持模糊或精确查找，并将所有匹配的校友信息以列表形式展示。

* + 1. 信息浏览模块

用户选择“5. 显示所有校友”，系统会以表格形式输出所有校友的详细信息，包括所有字段，便于用户整体浏览。

* + 1. 排序模块

用户选择“6. 排序校友信息”，可选择按姓名或年级排序。系统对链表中的校友信息进行归并排序，排序后自动保存。

* + 1. 统计与分析模块

用户选择“7. 统计信息”，可选择性别、院系、班级、年级、年龄等统计方式。系统统计各类别下的人数，并输出统计结果。

* + 1. 数据可持久化模块

系统启动时自动从文件加载校友信息，用户每次添加、删除、修改、排序后，系统都会自动保存数据到本地文件。用户也可通过“8. 从文件加载”手动重新加载数据。

## 系统分析及设计

**系统涉及对象有四个类：**校友类，校友列表类，节点类和菜单类。各种类设计的功能操作归纳为如下表所示：

|  |  |
| --- | --- |
| 对象 | 涉及的对象操作 |
| Alumni | 设置/获取包括姓名、性别、年龄、届级、系、班级、通讯地址、电话、qq、email在内的信息。 |
| AlumniList | 新增校友 |
| 根据姓名删除校友 |
| 更新校友信息 |
| 根据相关信息查精确查找或模糊查找校友信息 |
| 将数据保存至文件 |
| 从文件读取数据 |
| 按照姓名或届级对数据进行排序 |
| 获取基于某一维度的统计信息 |
| 获取链表首部 |
| 获取链表尾部 |
| 获取链表长度 |
| Node | 获取/设置节点校友数据 |
| 获取/设置下一个节点指针 |
| 获取/设置上一个结点指针 |
| Menu | 添加校友信息的菜单 |
| 删除校友信息的菜单 |
| 更新校友信息的菜单 |
| 搜索校友信息的菜单 |
| 显示校友信息统计的菜单 |

表 1 主要类

采用面向对象的方式实现校友录管理系统。核心类包括 Alumni (校友信息)、Node (链表节点)、AlumniList (校友数据管理) 和 Menu (用户界面交互)。所有校友数据通过双向链表组织和管理，并以文本文件进行持久化存储。AlumniList 类集中管理了数据的增删改查、排序、统计及文件操作等功能。

## 系统的实现

* + 1. 类的编写

系统工程名为：Alumni。包含了Alumni类（校友信息相关的类），Node类（链表的节点类），以及Menu类和AlumniList类分别对应菜单的显示和交互，以及数据的存储。

具体类结构声明如下：

* + - * Node类

class Node {

private:

  Alumni \*data;

  Node \*next;

  Node \*prev;

public:

  Node();

  Node(Alumni \*data, Node \*next, Node \*prev);

  ~Node();

  void setData(Alumni \*data);

  void setNext(Node \*next);

  void setPrev(Node \*prev);

  Alumni \*getData() const;

  Node \*getNext() const;

  Node \*getPrev() const;

};

* + - * Alumni类：

#ifndef ALUMNI\_H

#define ALUMNI\_H

#include <string>

using namespace std;

class Alumni {

private:

  string name;

  string gender;

  int age;

  int grade;

  string department;

  string className;

  string address;

  string phone;

  string qq;

  string email;

public:

  Alumni();

  Alumni(const string &name, const string &gender, const int &age, const int &grade, const string &department,

         const string &className, const string &address, const string &phone, const string &qq, const string &email);

  ~Alumni();

  void setName(const string &name);

  void setGender(const string &gender);

  void setAge(const int &age);

  void setGrade(const int &grade);

  void setDepartment(const string &department);

  void setClassName(const string &className);

  void setAddress(const string &address);

  void setPhone(const string &phone);

  void setQQ(const string &qq);

  void setEmail(const string &email);

  string getName() const;

  string getGender() const;

  int getAge() const;

  int getGrade() const;

  string getDepartment() const;

  string getClassName() const;

  string getAddress() const;

  string getPhone() const;

  string getQQ() const;

  string getEmail() const;

};

#endif // ALUMNI\_H

* + - * Menu类：

class Menu {

public:

  static void show();

  static void run(AlumniList &list);

private:

  static void addAlumni(AlumniList &list);

  static void deleteAlumni(AlumniList &list);

  static void updateAlumni(AlumniList &list);

  static void searchAlumni(AlumniList &list);

  static void sortAlumni(AlumniList &list);

  static void showStatistics(AlumniList &list);

};

* + 1. 链表的使用

为了便于数据的处理，采用了双向链表，除了尾节点，每一个节点都有一个指向下一个节点的指针。除了首节点，每个节点都有一个指向前一个节点的指针。同时每个节点都有自己的数据指针。并且将文件的读写也视为链表的一个方法，类似于将链表的所有信息都关于文件进行读写。同时统计的相关信息也被视为链表的相关方法，因为所有数据都是通过链表组织起来的。使用std::map进行数据的统计。同时实现了对链表的归并排序。和数据在交互界面的输出。

具体的声明如下：

class AlumniList {

private:

  Node \*head;

  Node \*tail;

  int len;

public:

  AlumniList();

  AlumniList(const AlumniList &other);

  ~AlumniList();

  AlumniList& operator=(const AlumniList &other);

  void add(const Alumni &alumni);

  void remove(const Alumni &alumni);

  void update(const string &name, const Alumni &alumni);

  AlumniList searchByName(const string &name, bool fuzzy = false);

  AlumniList searchByGrade(const int &grade, bool fuzzy = false);

  AlumniList searchByDepartment(const string &department, bool fuzzy = false);

  AlumniList searchByClassName(const string &className, bool fuzzy = false);

  void print();

  void saveToFile();

  void loadFromFile();

  void mergeSortByName(Node \*left, Node \*right, int len);

  void mergeSortByGrade(Node \*left, Node \*right, int len);

  map<string, int> getGenderStatistics();

  map<string, int> getDepartmentStatistics();

  map<string, int> getClassNameStatistics();

  map<int, int> getGradeStatistics();

  map<int, int> getAgeStatistics();

  Node \*getHead() const;

  Node \*getTail() const;

  int getLength() const;

};

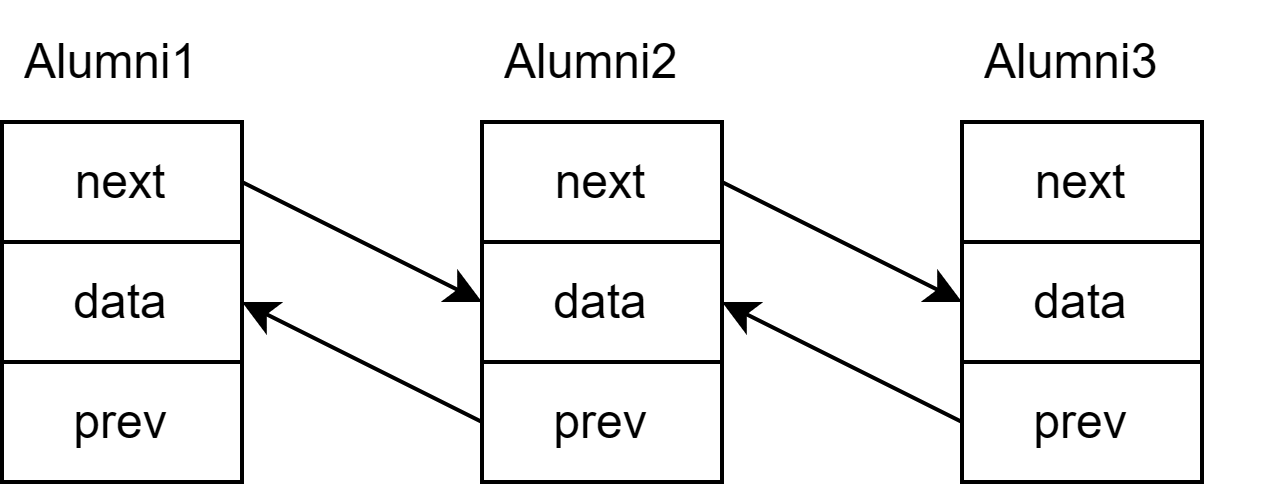
在运用时，令当前节点的next结点指向下一个校友结点，即结点的指针next保存新的校友结点的地址（如下图2所示），以此类推，所有校友信息就通过链表的形式串联起来了。

图 2 校友链表的建立

* + - * 校友的查询：

在用户在查询校友信息时，有四种不同的模式，每种模式都可以选择是否使用模糊查询（需要被包含或者全等于查询字符串）。

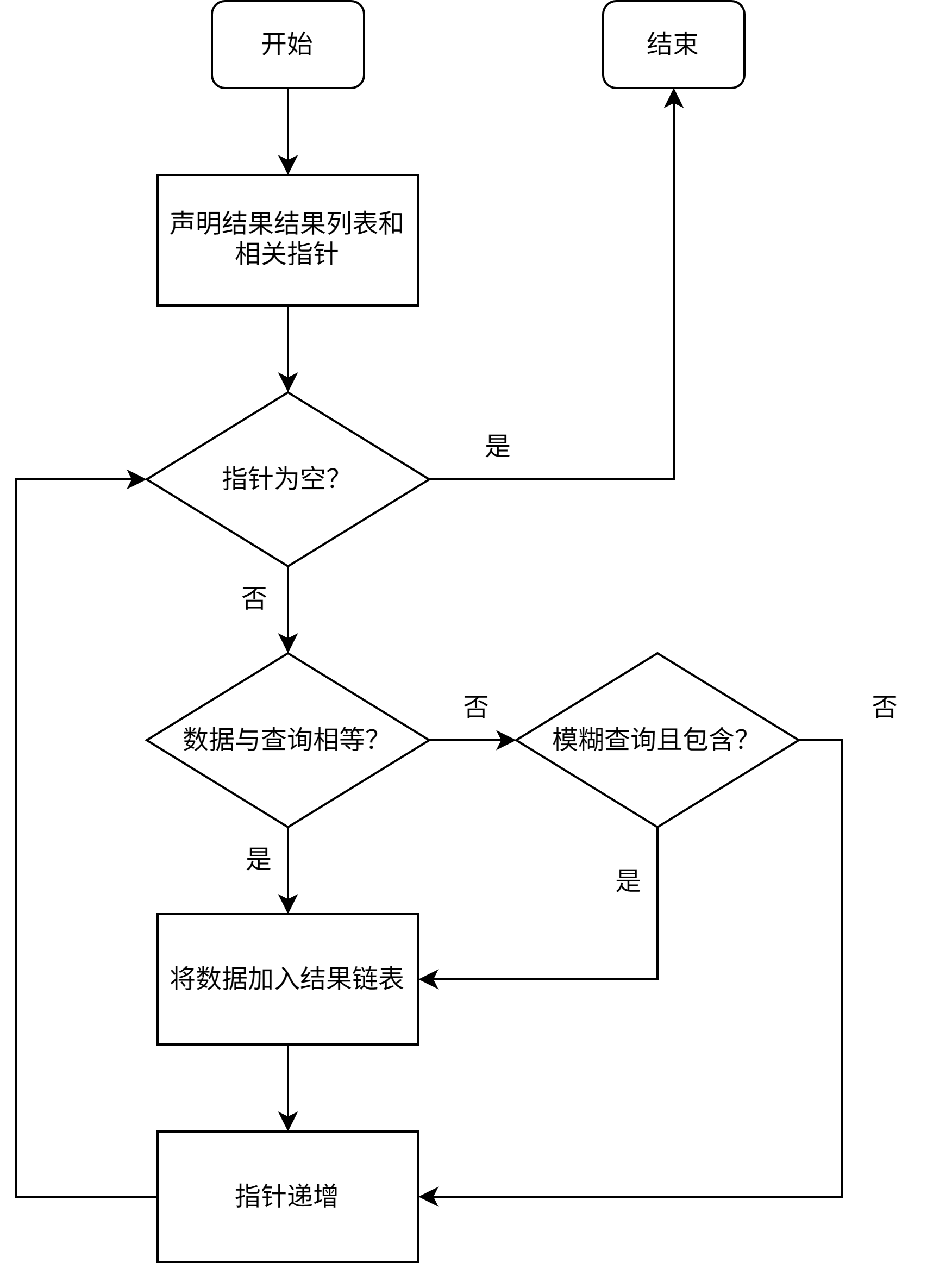
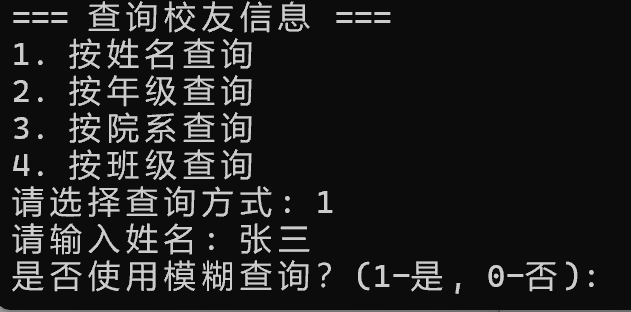
这三种情况都需要对链表中的所有数据进行顺序的搜索。

图 3 控制台查询校友信息

图 4 校友数据查询流程图

例如校友姓名的查找，定义一个指针，每次根据校友姓名是否包含查询字符串且是模糊查询或者全等于查询字符串且不是模糊查询来判断是否需要将当前数据加入结果链表中。

然后回调用链表的print函数输出结果。如果没有找到则会进行相关提示。

* + - * 校友的修改：

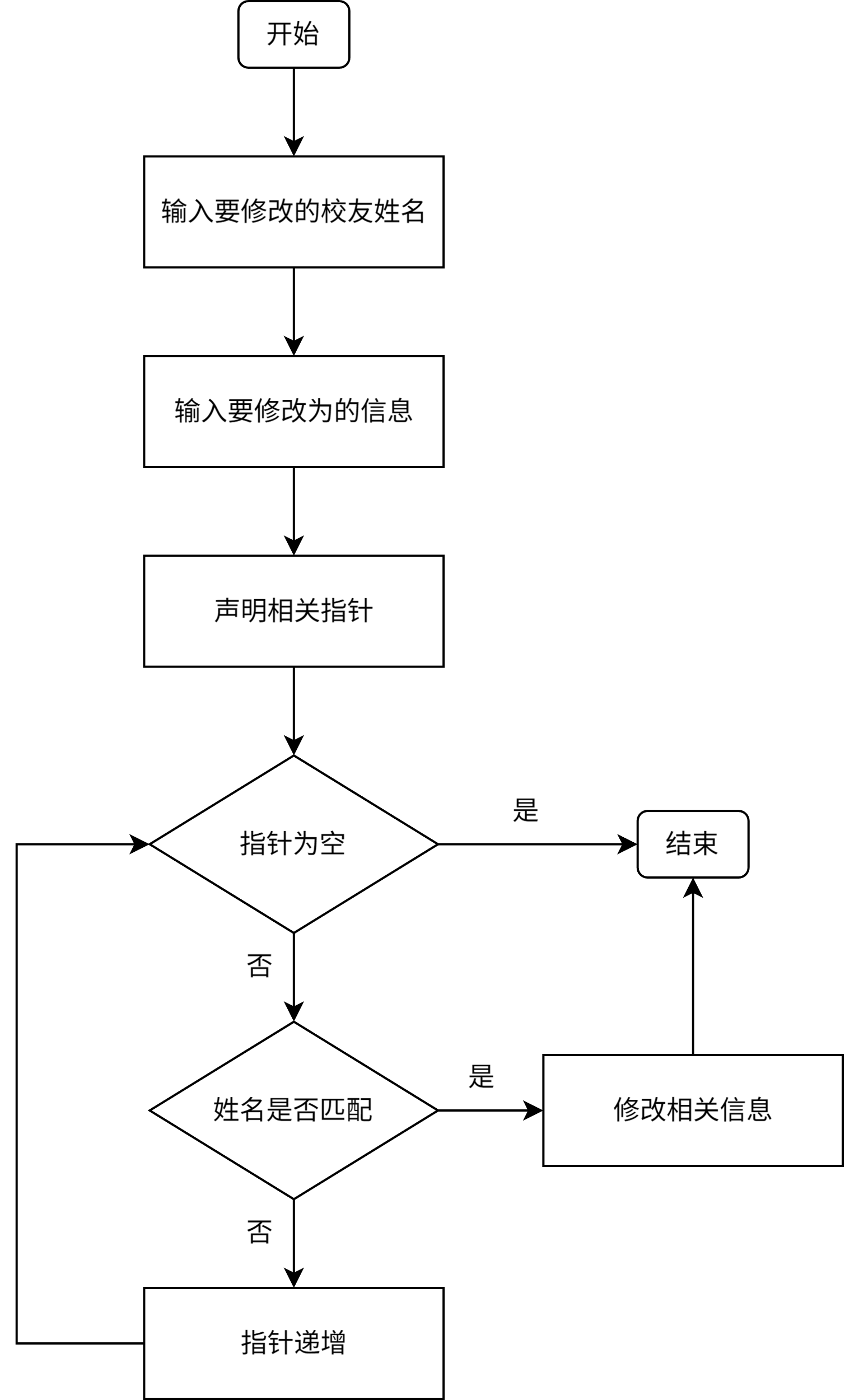
首先读入需要修改的校友的信息，然后读入要将对应修改为的信息，包括性别，年龄，届级，系，班级名，地址，电话，qq，电子邮箱在内的所有信息。然后开始遍历整个校友链表，如果找到姓名完全的校友，则进行修改并结束。我们假设不存在两名校友姓名相同。

图 5 校友信息修改流程图

* + - * 校友的添加：

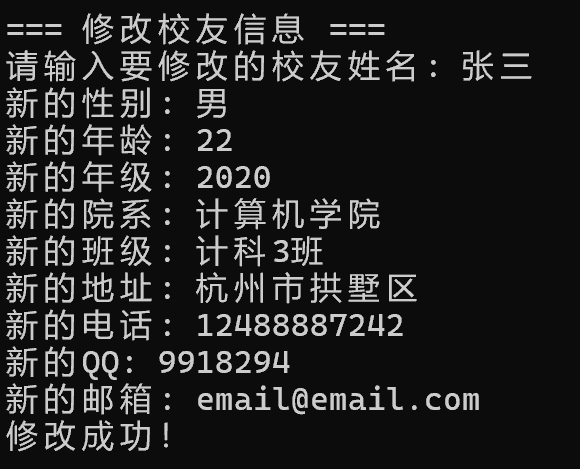
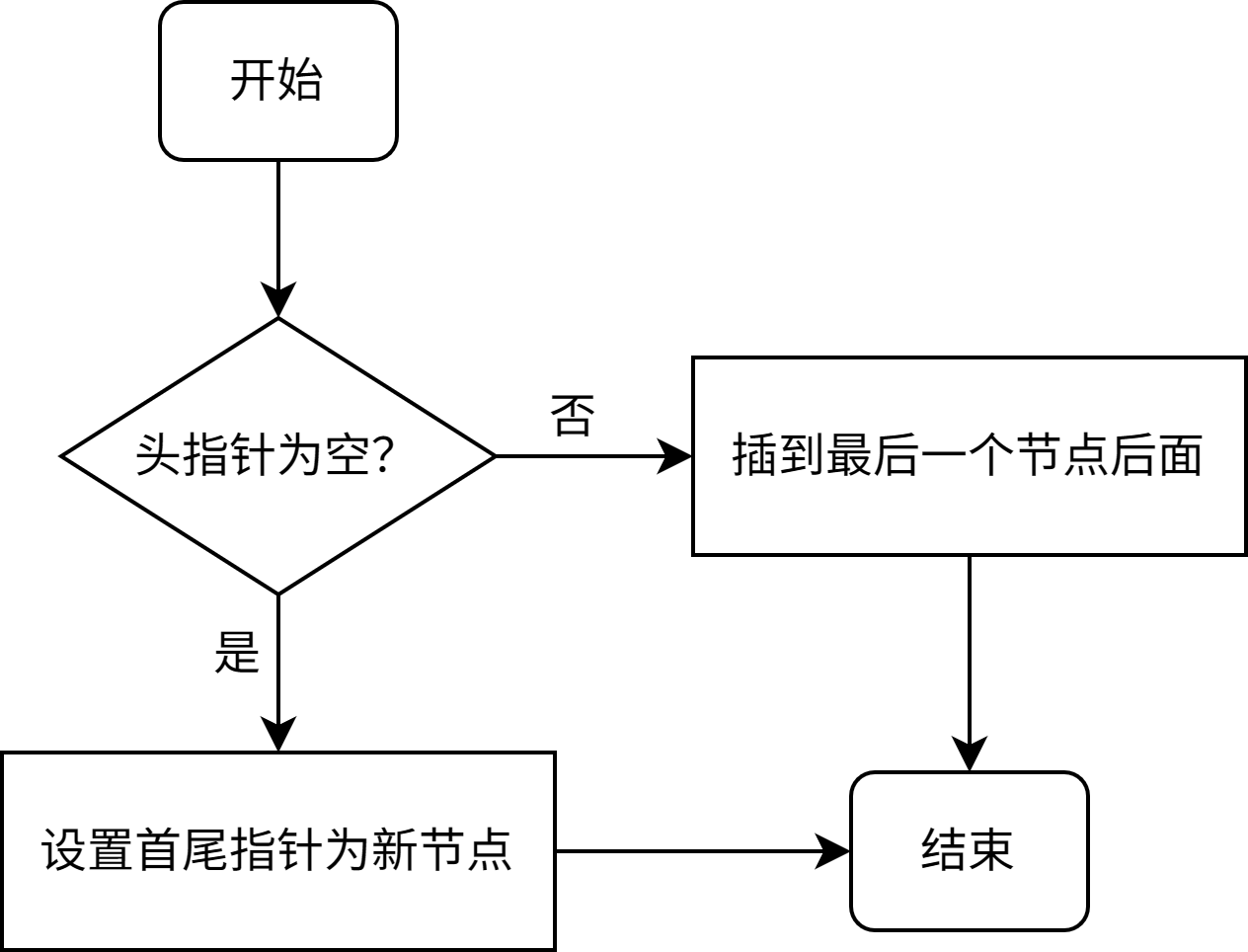
采用尾插法，插入到最后一个节点后面。同时修改相关指针。虽然实际上可以插入到链表的任何一个位置，但我们这里使新添加的校友节点为链表的尾节点。同时需要特判链表是否为空。

图 6 校友信息修改控制台

* + - * 校友的删除：

图 7 校友信息添加控制台

由于使用了双向链表，需要维护两个指针。因此需要对两个指针，和前后的指针分别进行判断。

图 8 校友信息添加流程图

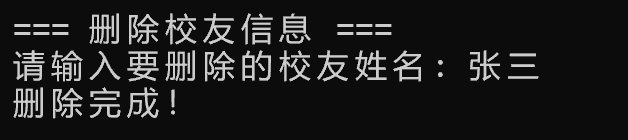
如果当前是最后一个节点，则不需要维护后一个节点并设置链表的尾节点。如果当前是第一个节点，则不需要维护前一个节点，并设置首节点。否则需要依次蛇者前一个节点的后一个节点的指针和后一个节点的前一个节点的指针。然后删除当前节点即可。

图 9 校友信息删除控制台

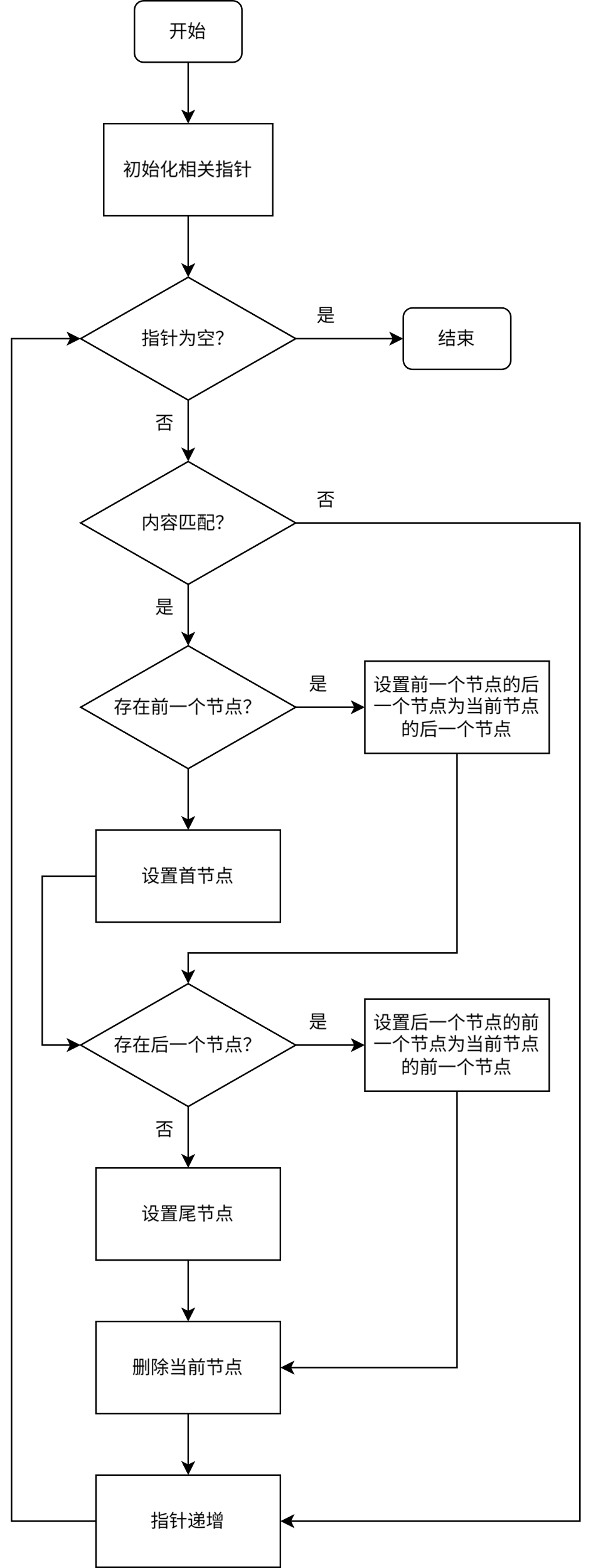


图 10 校友信息删除流程图

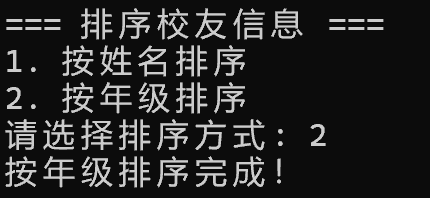
* + - * 校友链表的归并排序

图 11 校友信息排序控制台

首先特判是否只有一个元素，如果是则可以直接退出。

否则需要递归调用自身。分别排序左半边和右半边。然后循环，只要两侧至少存在一个未合并的元素，则对两侧进行判断，并将较小的那一个加入，且对那一侧的指针递增。

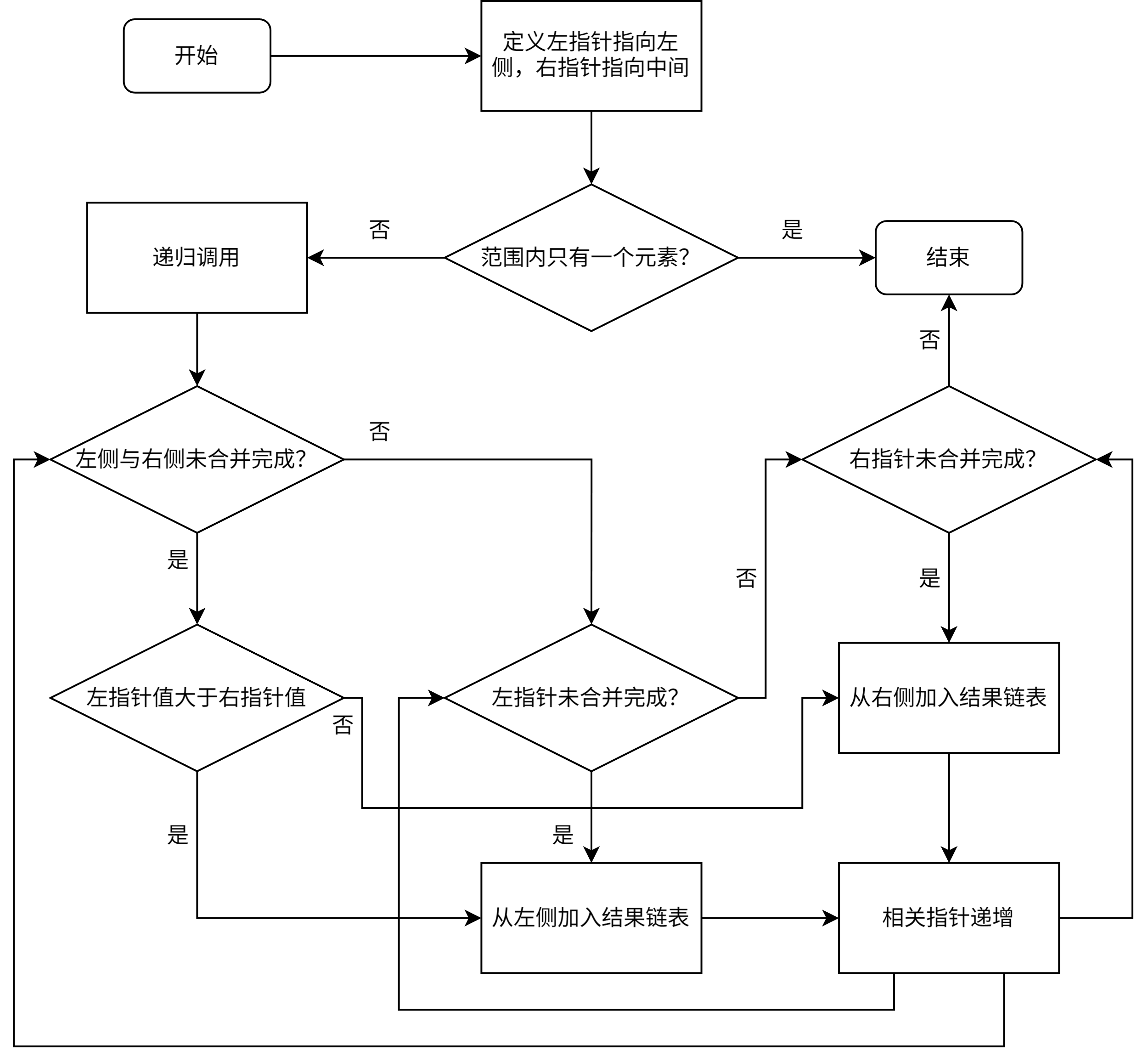
然后至少有一侧已经合并完毕，只需要合并另一侧即可。

图 12 校友信息排序流程图

* 校友信息各维度的统计

校友信息各维度的统计，包括性别，系，届级，年龄，班级名。可以统计所有校友在某个维度下为某个值的有多少人。

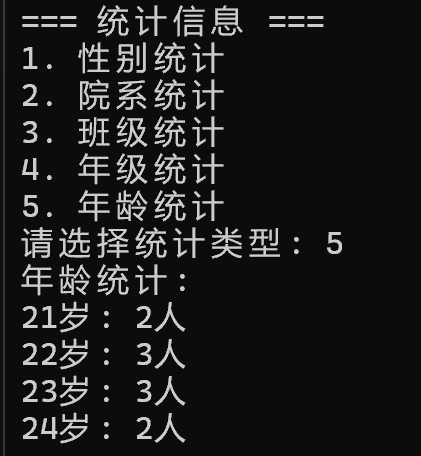
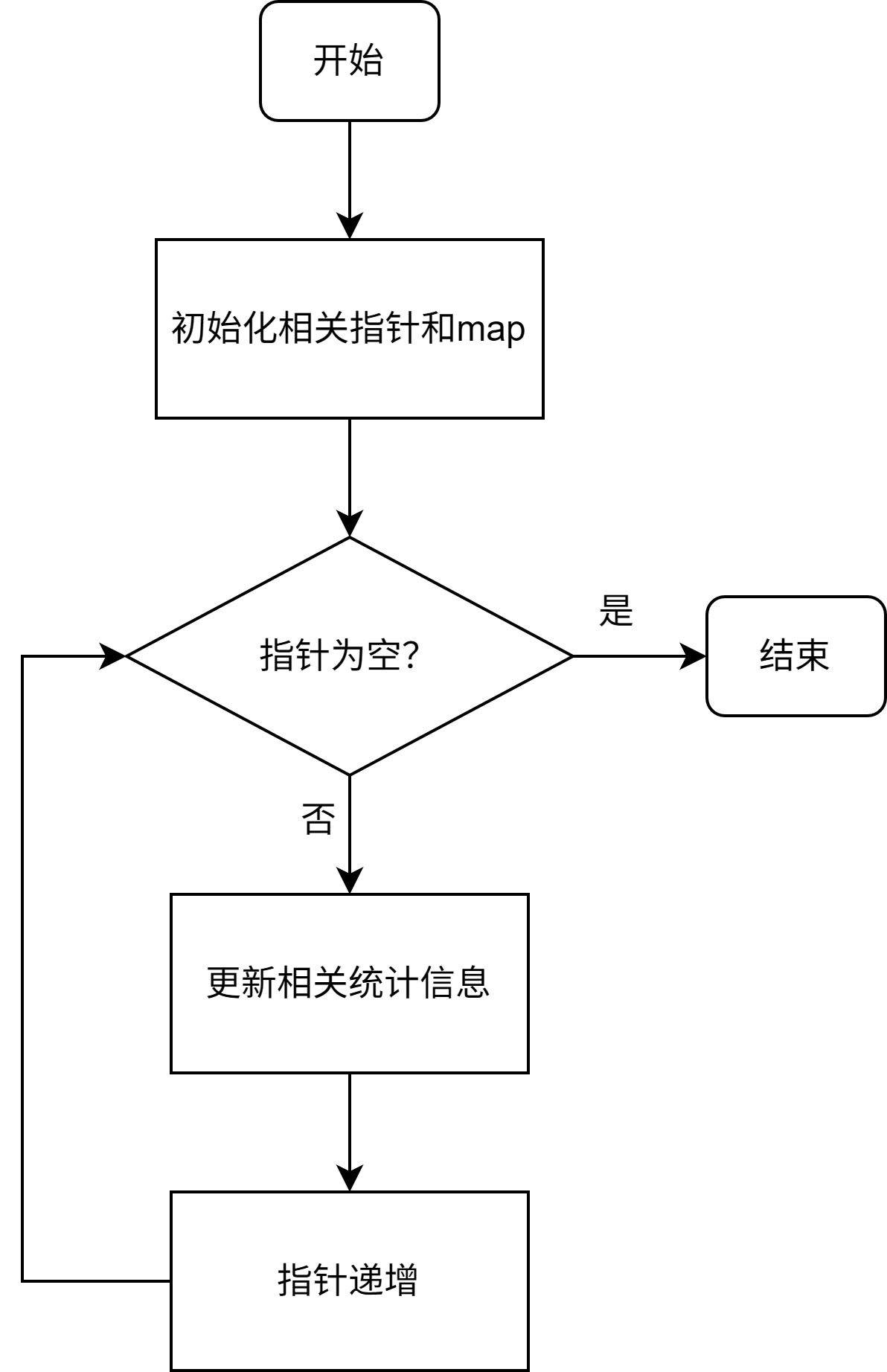
具体实现，即遍历整个链表，并使用std::map来统计，然后返回一个std::map。

图 13 校友信息统计控制台

图 14 校友信息统计流程图

* + 1. 交互界面的实现

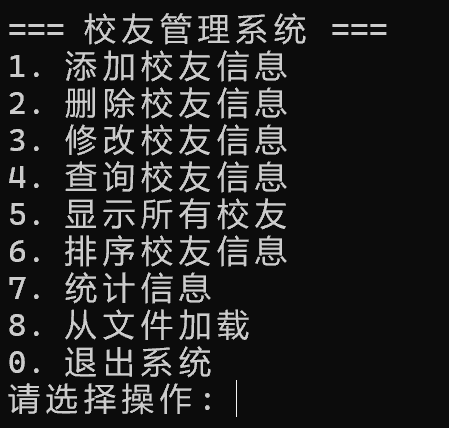
系统运行开始的界面如下所示：

图 15 交互界面控制台

主要通过选择结构和循环结构实现界面的切换。利用Menu类实现分别实现主菜单界面，添加校友，删除校友，修改校友信息，校友信息查询，显示所有校友信息，排序校友信息，对信息进行统计的功能。

# 实验调试、测试、运行记录及分析

系统在调试测试过程中遇到若干问题，不过经过仔细反复的检查已经消除各种bug。

主要的测试经过如下：

添加校友信息功能的测试：

图 16 添加校友信息测试

根据提示输入相关信息后，经验证确实有新增该校友。且位于最后一个位置，符合从最后一个节点后面插入的特点。

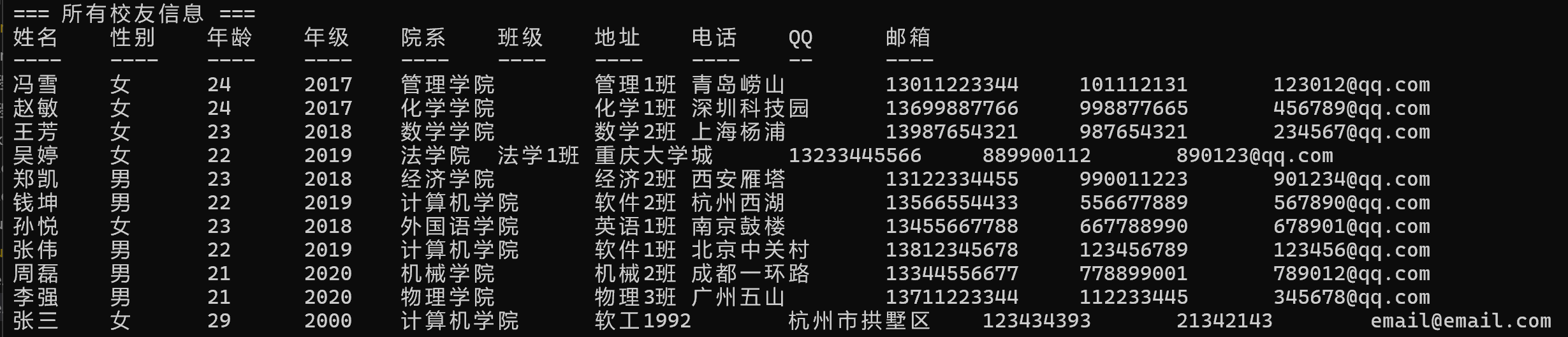


图 17 添加校友信息测试结果

然后进行删除校友信息的测试。

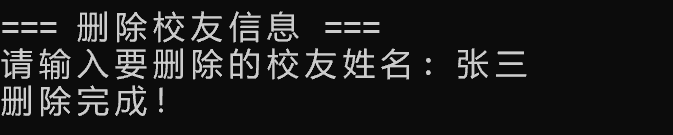


图 18 删除校友信息测试

经检验删除成功。



图 19 删除校友信息测试结果

然后测试修改校友信息的功能。修改名为张伟的校友的相关信息。

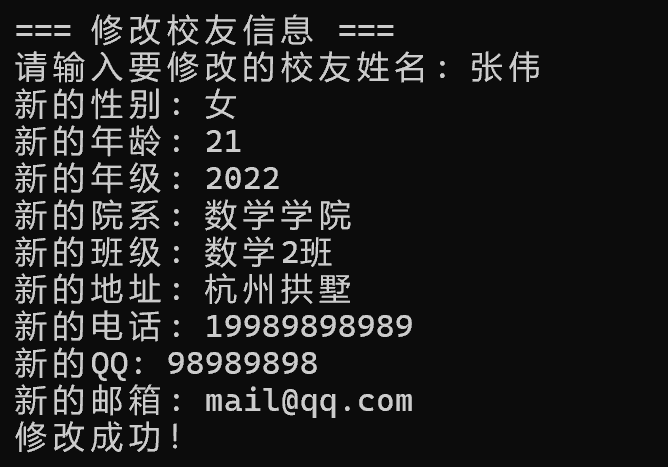


图 20 修改校友信息测试

经检验相关校友信息确实被修改成功。

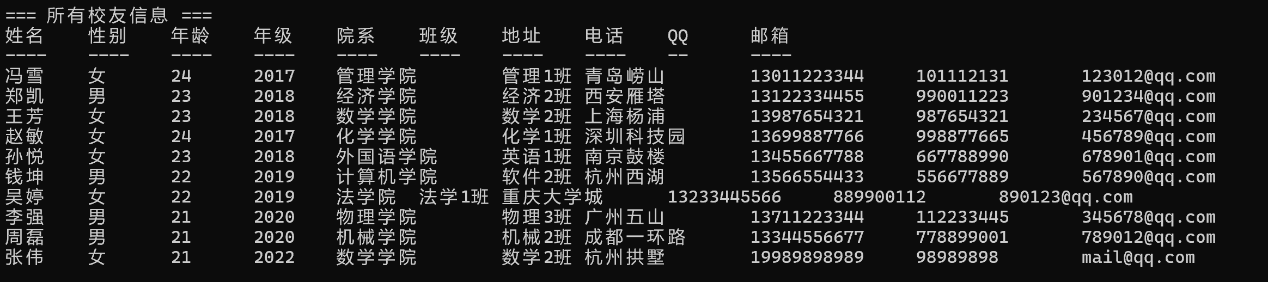


图 21 修改校友信息测试

然后测试信息查询功能，模糊查询按年级查询功能。

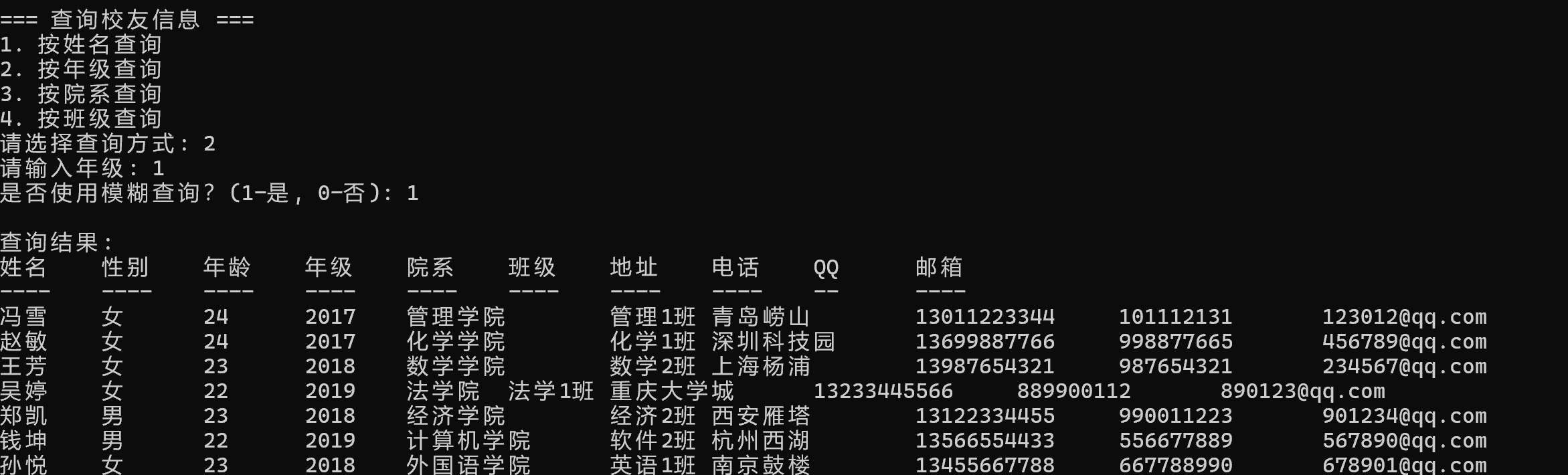


图 22 模糊查询校友信息测试结果

然后测试校友信息排序功能：

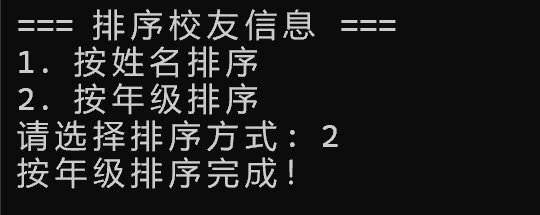


图 23 测试校友信息排序功能

可以看到校友信息已经按照年龄排序成功：

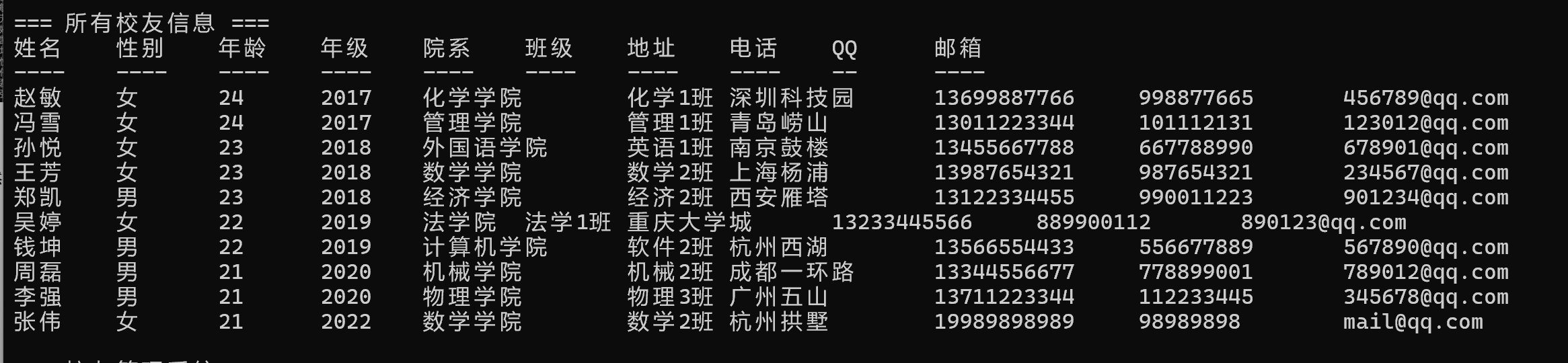


图 24 校友信息排序测试结果

* + - * 问题1：

问题描述：测试按年级模糊查询时程序闪退。

图 25 问题1

解决方法：搜索相关部分的AlumniList需要拷贝函数，但是没有写深拷贝，用的是默认的。于是重写了拷贝函数和赋值函数。

问题2：

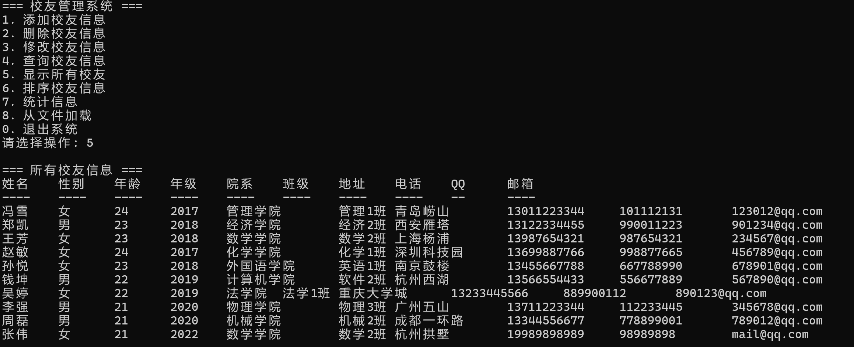
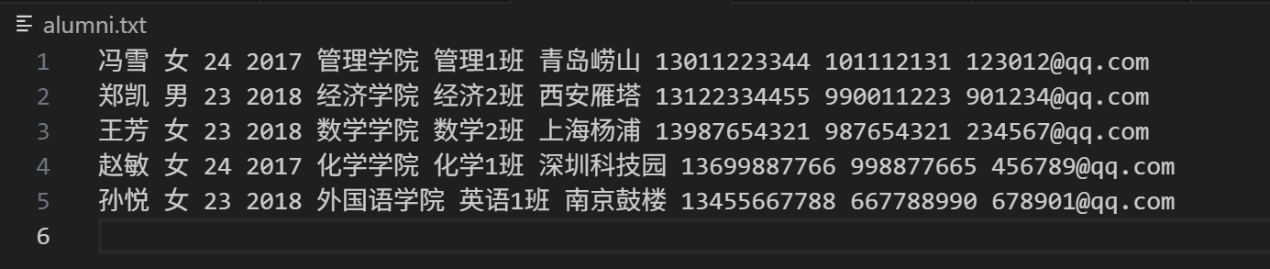
问题描述：从文件读取数据后，显示的数据没有更新

图 27 问题2 文件

图 27 问题2 文件

图 26 问题2 控制台

图 26 问题2 控制台

问题解决：重写了 AlumniList::loadFromFile() 函数，每次读取时，应该把数据全部清空。

# 实验总结（优点、不足、收获及体会）

本次校友录管理系统项目使我将C++面向对象编程的理论知识与实际应用相结合，深刻体会了从需求分析、系统设计到代码实现、测试调试的完整软件开发流程。

* + - * 优点：

模块化设计：系统采用了面向对象的思想，将校友信息、链表操作、菜单界面等功能分别封装到独立的类中（Alumni, Node, AlumniList, Menu），使得代码结构清晰、逻辑分明，易于理解和维护。

功能完善性：系统实现了任务书中要求的各项核心功能，包括校友信息的增、删、改、查、显示所有、排序、统计分析以及数据的持久化存储，基本满足了任务书的功能要求。

* + - * 不足与改进：

代码复用性：在对校友信息进行排序和统计时，针对不同属性（如姓名、年级、性别等）的代码逻辑存在相似的重复。未来可以考虑设计通用的模板函数或接受函数对象的接口，以实现更高级的代码复用，减少重复代码量。

错误处理：虽然在“查询不到”时提供了相关提示，但这算是比较基础的错误处理。目前缺少或未能充分覆盖对用户输入的详细校验（如年龄、年级的非数字输入，或字符串类型字段的格式合法性）、以及文件操作的错误捕获与处理机制。增强这些方面的处理将显著提高系统的健壮性和用户体验。

* + - * 收获及体会

通过此次实践，我不仅巩固了C++基础知识和面向对象思想，更提升了问题分析、调试排错和代码优化能力。深刻体会到理论与实践结合的重要性，正如“实践是检验真理的唯一标准”，为未来的学习奠定了坚实基础。

# 附录：源代码

* + 1. Alumni.h

#ifndef ALUMNI\_H

#define ALUMNI\_H

#include <string>

using namespace std;

class Alumni {

private:

  string name;

  string gender;

  int age;

  int grade;

  string department;

  string className;

  string address;

  string phone;

  string qq;

  string email;

public:

  Alumni();

  Alumni(const string &name, const string &gender, const int &age, const int &grade, const string &department,

         const string &className, const string &address, const string &phone, const string &qq, const string &email);

  ~Alumni();

  void setName(const string &name);

  void setGender(const string &gender);

  void setAge(const int &age);

  void setGrade(const int &grade);

  void setDepartment(const string &department);

  void setClassName(const string &className);

  void setAddress(const string &address);

  void setPhone(const string &phone);

  void setQQ(const string &qq);

  void setEmail(const string &email);

  string getName() const;

  string getGender() const;

  int getAge() const;

  int getGrade() const;

  string getDepartment() const;

  string getClassName() const;

  string getAddress() const;

  string getPhone() const;

  string getQQ() const;

  string getEmail() const;

};

#endif // ALUMNI\_H

* + 1. Alumni.cpp

#include "Alumni.h"

void Alumni::setName(const string &name) {

  this->name = name;

}

void Alumni::setGender(const string &gender) {

  this->gender = gender;

}

void Alumni::setAge(const int &age) {

  this->age = age;

}

void Alumni::setGrade(const int &grade) {

  this->grade = grade;

}

void Alumni::setDepartment(const string &department) {

  this->department = department;

}

void Alumni::setClassName(const string &className) {

  this->className = className;

}

void Alumni::setAddress(const string &address) {

  this->address = address;

}

void Alumni::setPhone(const string &phone) {

  this->phone = phone;

}

void Alumni::setQQ(const string &qq) {

  this->qq = qq;

}

void Alumni::setEmail(const string &email) {

  this->email = email;

}

string Alumni::getName() const {

  return name;

}

string Alumni::getGender() const {

  return gender;

}

int Alumni::getAge() const {

  return age;

}

int Alumni::getGrade() const {

  return grade;

}

string Alumni::getDepartment() const {

  return department;

}

string Alumni::getClassName() const {

  return className;

}

string Alumni::getAddress() const {

  return address;

}

string Alumni::getPhone() const {

  return phone;

}

string Alumni::getQQ() const {

  return qq;

}

string Alumni::getEmail() const {

  return email;

}

Alumni::Alumni() : name(""), gender(""), age(0), grade(0), department(""), className(""), address(""), phone(""), qq(""), email("") {}

Alumni::Alumni(const string &name, const string &gender, const int &age, const int &grade, const string &department,

               const string &className, const string &address, const string &phone, const string &qq, const string &email)

    : name(name), gender(gender), age(age), grade(grade), department(department), className(className), address(address), phone(phone), qq(qq), email(email) {}

Alumni::~Alumni() {}

* + 1. AlumniList.h

#ifndef ALUMNI\_LIST\_H

#define ALUMNI\_LIST\_H

#include "Alumni.h"

#include "Node.h"

#include <map>

class AlumniList {

private:

  Node \*head;

  Node \*tail;

  int len;

public:

  AlumniList();

  AlumniList(const AlumniList &other);

  ~AlumniList();

  AlumniList& operator=(const AlumniList &other);

  void add(const Alumni &alumni);

  void remove(const Alumni &alumni);

  void update(const string &name, const Alumni &alumni);

  AlumniList searchByName(const string &name, bool fuzzy = false);

  AlumniList searchByGrade(const int &grade, bool fuzzy = false);

  AlumniList searchByDepartment(const string &department, bool fuzzy = false);

  AlumniList searchByClassName(const string &className, bool fuzzy = false);

  void print();

  void saveToFile();

  void loadFromFile();

  void mergeSortByName(Node \*left, Node \*right, int len);

  void mergeSortByGrade(Node \*left, Node \*right, int len);

  map<string, int> getGenderStatistics();

  map<string, int> getDepartmentStatistics();

  map<string, int> getClassNameStatistics();

  map<int, int> getGradeStatistics();

  map<int, int> getAgeStatistics();

  Node \*getHead() const;

  Node \*getTail() const;

  int getLength() const;

};

#endif

* + 1. AluminList.cpp

#include "AlumniList.h"

#include <fstream>

#include <iostream>

using namespace std;

AlumniList::AlumniList() : head(nullptr), tail(nullptr), len(0) {}

AlumniList::AlumniList(const AlumniList &other) : head(nullptr), tail(nullptr), len(0) {

  Node \*cur = other.head;

  while (cur) {

    add(\*(cur->getData()));

    cur = cur->getNext();

  }

}

AlumniList::~AlumniList() {

  Node \*cur = head;

  while (cur) {

    Node \*next = cur->getNext();

    delete cur;

    cur = next;

  }

}

AlumniList& AlumniList::operator=(const AlumniList &other) {

  if (this != &other) {

    Node \*cur = head;

    while (cur) {

      Node \*next = cur->getNext();

      delete cur;

      cur = next;

    }

    head = tail = nullptr;

    len = 0;

    cur = other.head;

    while (cur) {

      add(\*(cur->getData()));

      cur = cur->getNext();

    }

  }

  return \*this;

}

AlumniList AlumniList::searchByName(const string &name, bool fuzzy) {

  AlumniList results;

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    if (!fuzzy && data->getName() == name) {

      results.add(\*data);

    } else if (fuzzy && data->getName().find(name) != string::npos) {

      results.add(\*data);

    }

    cur = cur->getNext();

  }

  return results;

}

AlumniList AlumniList::searchByGrade(const int &grade, bool fuzzy) {

  AlumniList results;

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    if (!fuzzy && data->getGrade() == grade) {

      results.add(\*data);

    } else if (fuzzy && to\_string(data->getGrade()).find(to\_string(grade)) != string::npos) {

      results.add(\*data);

    }

    cur = cur->getNext();

  }

  return results;

}

AlumniList AlumniList::searchByDepartment(const string &department, bool fuzzy) {

  AlumniList results;

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    if (!fuzzy && data->getDepartment() == department) {

      results.add(\*data);

    } else if (fuzzy && data->getDepartment().find(department) != string::npos) {

      results.add(\*data);

    }

    cur = cur->getNext();

  }

  return results;

}

AlumniList AlumniList::searchByClassName(const string &className, bool fuzzy) {

  AlumniList results;

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    if (data) {

      if (!fuzzy && data->getClassName() == className) {

        results.add(\*data);

      } else if (fuzzy && data->getClassName().find(className) != string::npos) {

        results.add(\*data);

      }

    }

    cur = cur->getNext();

  }

  return results;

}

void AlumniList::add(const Alumni &alumni) {

  Node \*node = new Node(new Alumni(alumni), nullptr, tail);

  if (!head) {

    head = tail = node;

  } else {

    tail->setNext(node);

    node->setPrev(tail);

    tail = node;

  }

  len++;

}

void AlumniList::remove(const Alumni &alumni) {

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    if (data->getName() == alumni.getName()) {

      if (cur->getPrev())

        cur->getPrev()->setNext(cur->getNext());

      else

        head = cur->getNext();

      if (cur->getNext())

        cur->getNext()->setPrev(cur->getPrev());

      else

        tail = cur->getPrev();

      delete cur;

      len--;

      return;

    }

    cur = cur->getNext();

  }

}

void AlumniList::update(const string &name, const Alumni &alumni) {

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    if (data->getName() == name) {

      \*data = alumni;

      return;

    }

    cur = cur->getNext();

  }

}

void AlumniList::print() {

  cout << "姓名\t性别\t年龄\t年级\t院系\t班级\t地址\t电话\tQQ\t邮箱" << endl;

  cout << "----\t----\t----\t----\t----\t----\t----\t----\t--\t----" << endl;

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    cout << data->getName() << "\t" << data->getGender() << "\t" << data->getAge() << "\t" << data->getGrade() << "\t"

         << data->getDepartment() << "\t" << data->getClassName() << "\t" << data->getAddress() << "\t"

         << data->getPhone() << "\t" << data->getQQ() << "\t" << data->getEmail() << endl;

    cur = cur->getNext();

  }

}

void AlumniList::saveToFile() {

  ofstream os("alumni.txt");

  Node \*cur = head;

  while (cur) {

    Alumni \*data = cur->getData();

    os << data->getName() << ' ' << data->getGender() << ' ' << data->getAge() << ' ' << data->getGrade() << ' '

       << data->getDepartment() << ' ' << data->getClassName() << ' ' << data->getAddress() << ' ' << data->getPhone()

       << ' ' << data->getQQ() << ' ' << data->getEmail() << '\n';

    cur = cur->getNext();

  }

}

void AlumniList::loadFromFile() {

  this->~AlumniList();

  head = tail = nullptr;

  len = 0;

  ifstream is("alumni.txt");

  string name, gender, department, className, address, phone, qq, email;

  int age, grade;

  while (is >> name >> gender >> age >> grade >> department >> className >> address >> phone >> qq >> email) {

    add(Alumni(name, gender, age, grade, department, className, address, phone, qq, email));

  }

}

void AlumniList::mergeSortByName(Node \*left, Node \*right, int len) {

  if (len == 1) return;

  Node \*mid = left;

  for (int i = 0; i < len / 2; i++) mid = mid->getNext();

  mergeSortByName(left, mid, len / 2);

  mergeSortByName(mid, right, len - len / 2);

  AlumniList temp;

  Node \*p1 = left, \*p2 = mid;

  while (p1 != mid && p2) {

    if (p1->getData()->getName() < p2->getData()->getName()) {

      temp.add(\*(p1->getData()));

      p1 = p1->getNext();

    } else {

      temp.add(\*(p2->getData()));

      p2 = p2->getNext();

    }

  }

  while (p1 != mid) {

    temp.add(\*(p1->getData()));

    p1 = p1->getNext();

  }

  while (p2) {

    temp.add(\*(p2->getData()));

    p2 = p2->getNext();

  }

  Node \*cur = left;

  Node \*tempCur = temp.head;

  while (true) {

    if (!cur) break;

    \*(cur->getData()) = \*(tempCur->getData());

    cur = cur->getNext();

    tempCur = tempCur->getNext();

  }

}

void AlumniList::mergeSortByGrade(Node \*left, Node \*right, int len) {

  if (len == 1) return;

  Node \*mid = left;

  for (int i = 0; i < len / 2; i++) mid = mid->getNext();

  mergeSortByGrade(left, mid, len / 2);

  mergeSortByGrade(mid, right, len - len / 2);

  AlumniList temp;

  Node \*p1 = left, \*p2 = mid;

  while (p1 != mid && p2) {

    if (p1->getData()->getGrade() < p2->getData()->getGrade()) {

      temp.add(\*(p1->getData()));

      p1 = p1->getNext();

    } else {

      temp.add(\*(p2->getData()));

      p2 = p2->getNext();

    }

  }

  while (p1 != mid) {

    temp.add(\*(p1->getData()));

    p1 = p1->getNext();

  }

  while (p2) {

    temp.add(\*(p2->getData()));

    p2 = p2->getNext();

  }

  Node \*cur = left;

  Node \*tempCur = temp.head;

  while (true) {

    if (!cur) break;

    \*(cur->getData()) = \*(tempCur->getData());

    cur = cur->getNext();

    tempCur = tempCur->getNext();

  }

}

map<string, int> AlumniList::getGenderStatistics() {

  map<string, int> mp;

  Node \*cur = head;

  while (cur) {

    mp[cur->getData()->getGender()]++;

    cur = cur->getNext();

  }

  return mp;

}

map<string, int> AlumniList::getDepartmentStatistics() {

  map<string, int> mp;

  Node \*cur = head;

  while (cur) {

    mp[cur->getData()->getDepartment()]++;

    cur = cur->getNext();

  }

  return mp;

}

map<string, int> AlumniList::getClassNameStatistics() {

  map<string, int> mp;

  Node \*cur = head;

  while (cur) {

    mp[cur->getData()->getClassName()]++;

    cur = cur->getNext();

  }

  return mp;

}

map<int, int> AlumniList::getGradeStatistics() {

  map<int, int> mp;

  Node \*cur = head;

  while (cur) {

    mp[cur->getData()->getGrade()]++;

    cur = cur->getNext();

  }

  return mp;

}

map<int, int> AlumniList::getAgeStatistics() {

  map<int, int> mp;

  Node \*cur = head;

  while (cur) {

    mp[cur->getData()->getAge()]++;

    cur = cur->getNext();

  }

  return mp;

}

Node \*AlumniList::getHead() const {

  return head;

}

Node \*AlumniList::getTail() const {

  return tail;

}

int AlumniList::getLength() const {

  return len;

}

* + 1. Menu.h

#ifndef MENU\_H

#define MENU\_H

#include "AlumniList.h"

class Menu {

public:

  static void show();

  static void run(AlumniList &list);

private:

  static void addAlumni(AlumniList &list);

  static void deleteAlumni(AlumniList &list);

  static void updateAlumni(AlumniList &list);

  static void searchAlumni(AlumniList &list);

  static void sortAlumni(AlumniList &list);

  static void showStatistics(AlumniList &list);

};

#endif // MENU\_H

* + 1. Menu.cpp

#include "Menu.h"

#include <iostream>

using namespace std;

void Menu::show() {

  cout << "\n=== 校友管理系统 ===" << endl;

  cout << "1. 添加校友信息" << endl;

  cout << "2. 删除校友信息" << endl;

  cout << "3. 修改校友信息" << endl;

  cout << "4. 查询校友信息" << endl;

  cout << "5. 显示所有校友" << endl;

  cout << "6. 排序校友信息" << endl;

  cout << "7. 统计信息" << endl;

  cout << "8. 从文件加载" << endl;

  cout << "0. 退出系统" << endl;

  cout << "请选择操作: ";

}

void Menu::run(AlumniList &list) {

  int choice;

  while (true) {

    show();

    cin >> choice;

    switch (choice) {

      case 1:

        addAlumni(list);

        break;

      case 2:

        deleteAlumni(list);

        break;

      case 3:

        updateAlumni(list);

        break;

      case 4:

        searchAlumni(list);

        break;

      case 5:

        cout << "\n=== 所有校友信息 ===" << endl;

        list.print();

        break;

      case 6:

        sortAlumni(list);

        break;

      case 7:

        showStatistics(list);

        break;

      case 8:

        list.loadFromFile();

        cout << "数据已从文件重新加载！" << endl;

        break;

      case 0:

        cout << "正在保存数据..." << endl;

        list.saveToFile();

        cout << "数据已保存，感谢使用，再见！" << endl;

        return;

      default:

        cout << "无效选择，请重新输入！" << endl;

    }

  }

}

void Menu::addAlumni(AlumniList &list) {

  string name, gender, department, className, address, phone, qq, email;

  int age, grade;

  cout << "\n=== 添加校友信息 ===" << endl;

  cout << "姓名: ";

  cin >> name;

  cout << "性别: ";

  cin >> gender;

  cout << "年龄: ";

  cin >> age;

  cout << "年级: ";

  cin >> grade;

  cout << "院系: ";

  cin >> department;

  cout << "班级: ";

  cin >> className;

  cout << "地址: ";

  cin >> address;

  cout << "电话: ";

  cin >> phone;

  cout << "QQ: ";

  cin >> qq;

  cout << "邮箱: ";

  cin >> email;

  Alumni alumni(name, gender, age, grade, department, className, address, phone, qq, email);

  list.add(alumni);

  cout << "添加成功！" << endl;

  list.saveToFile();

}

void Menu::deleteAlumni(AlumniList &list) {

  string name;

  cout << "\n=== 删除校友信息 ===" << endl;

  cout << "请输入要删除的校友姓名: ";

  cin >> name;

  Alumni temp(name, "", 0, 0, "", "", "", "", "", "");

  list.remove(temp);

  cout << "删除完成！" << endl;

  list.saveToFile();

}

void Menu::updateAlumni(AlumniList &list) {

  string name, gender, department, className, address, phone, qq, email;

  int age, grade;

  cout << "\n=== 修改校友信息 ===" << endl;

  cout << "请输入要修改的校友姓名: ";

  cin >> name;

  cout << "新的性别: ";

  cin >> gender;

  cout << "新的年龄: ";

  cin >> age;

  cout << "新的年级: ";

  cin >> grade;

  cout << "新的院系: ";

  cin >> department;

  cout << "新的班级: ";

  cin >> className;

  cout << "新的地址: ";

  cin >> address;

  cout << "新的电话: ";

  cin >> phone;

  cout << "新的QQ: ";

  cin >> qq;

  cout << "新的邮箱: ";

  cin >> email;

  Alumni newAlumni(name, gender, age, grade, department, className, address, phone, qq, email);

  list.update(name, newAlumni);

  cout << "修改成功！" << endl;

  list.saveToFile();

}

void Menu::searchAlumni(AlumniList &list) {

  cout << "\n=== 查询校友信息 ===" << endl;

  cout << "1. 按姓名查询" << endl;

  cout << "2. 按年级查询" << endl;

  cout << "3. 按院系查询" << endl;

  cout << "4. 按班级查询" << endl;

  cout << "请选择查询方式: ";

  int choice;

  cin >> choice;

  string key;

  AlumniList results;

  bool fuzzy;

  switch (choice) {

    case 1:

      cout << "请输入姓名: ";

      cin >> key;

      cout << "是否使用模糊查询？(1-是, 0-否): ";

      cin >> fuzzy;

      results = list.searchByName(key, fuzzy);

      break;

    case 2:

      cout << "请输入年级: ";

      cin >> key;

      cout << "是否使用模糊查询？(1-是, 0-否): ";

      cin >> fuzzy;

      results = list.searchByGrade(stoi(key), fuzzy);

      break;

    case 3:

      cout << "请输入院系: ";

      cin >> key;

      cout << "是否使用模糊查询？(1-是, 0-否): ";

      cin >> fuzzy;

      results = list.searchByDepartment(key, fuzzy);

      break;

    case 4:

      cout << "请输入班级: ";

      cin >> key;

      cout << "是否使用模糊查询？(1-是, 0-否): ";

      cin >> fuzzy;

      results = list.searchByClassName(key, fuzzy);

      break;

    default:

      cout << "无效选择！" << endl;

      return;

  }

  if (results.getLength() == 0) {

    cout << "未找到匹配的校友信息！" << endl;

  } else {

    cout << "\n查询结果:" << endl;

    results.print();

  }

}

void Menu::sortAlumni(AlumniList &list) {

  cout << "\n=== 排序校友信息 ===" << endl;

  cout << "1. 按姓名排序" << endl;

  cout << "2. 按年级排序" << endl;

  cout << "请选择排序方式: ";

  int choice;

  cin >> choice;

  switch (choice) {

    case 1:

      list.mergeSortByName(list.getHead(), list.getTail(), list.getLength());

      cout << "按姓名排序完成！" << endl;

      break;

    case 2:

      list.mergeSortByGrade(list.getHead(), list.getTail(), list.getLength());

      cout << "按年级排序完成！" << endl;

      break;

    default:

      cout << "无效选择！" << endl;

      return;

  }

  list.saveToFile();

}

void Menu::showStatistics(AlumniList &list) {

  cout << "\n=== 统计信息 ===" << endl;

  cout << "1. 性别统计" << endl;

  cout << "2. 院系统计" << endl;

  cout << "3. 班级统计" << endl;

  cout << "4. 年级统计" << endl;

  cout << "5. 年龄统计" << endl;

  cout << "请选择统计类型: ";

  int choice;

  cin >> choice;

  switch (choice) {

    case 1: {

      auto mp = list.getGenderStatistics();

      cout << "性别统计:" << endl;

      for (auto &[k, v]: mp) {

        cout << k << ": " << v << "人" << endl;

      }

      break;

    }

    case 2: {

      auto mp = list.getDepartmentStatistics();

      cout << "院系统计:" << endl;

      for (auto &[k, v]: mp) {

        cout << k << ": " << v << "人" << endl;

      }

      break;

    }

    case 3: {

      auto mp = list.getClassNameStatistics();

      cout << "班级统计:" << endl;

      for (auto &[k, v]: mp) {

        cout << k << ": " << v << "人" << endl;

      }

      break;

    }

    case 4: {

      auto mp = list.getGradeStatistics();

      cout << "年级统计:" << endl;

      for (auto &[k, v]: mp) {

        cout << k << "级: " << v << "人" << endl;

      }

      break;

    }

    case 5: {

      auto mp = list.getAgeStatistics();

      cout << "年龄统计:" << endl;

      for (auto &[k, v]: mp) {

        cout << k << "岁: " << v << "人" << endl;

      }

      break;

    }

    default:

      cout << "无效选择！" << endl;

  }

}

* + 1. Node.h

#ifndef NODE\_H

#define NODE\_H

#include "Alumni.h"

class Node {

private:

  Alumni \*data;

  Node \*next;

  Node \*prev;

public:

  Node();

  Node(Alumni \*data, Node \*next, Node \*prev);

  ~Node();

  void setData(Alumni \*data);

  void setNext(Node \*next);

  void setPrev(Node \*prev);

  Alumni \*getData() const;

  Node \*getNext() const;

  Node \*getPrev() const;

};

#endif

* + 1. Node.cpp

#include "Node.h"

Node::Node() : data(nullptr), next(nullptr), prev(nullptr) {}

Node::Node(Alumni \*data, Node \*next, Node \*prev) : data(data), next(next), prev(prev) {}

Node::~Node() {

  delete data;

}

void Node::setData(Alumni \*data) {

  this->data = data;

}

void Node::setNext(Node \*next) {

  this->next = next;

}

void Node::setPrev(Node \*prev) {

  this->prev = prev;

}

Alumni \*Node::getData() const {

  return data;

}

Node \*Node::getNext() const {

  return next;

}

Node \*Node::getPrev() const {

  return prev;

}

* + 1. Main.cpp

#include <iostream>

#include "AlumniList.h"

#include "Menu.h"

using namespace std;

int main() {

  AlumniList list;

  cout << "欢迎使用校友管理系统！" << endl;

  cout << "正在从文件加载数据..." << endl;

  list.loadFromFile();

  cout << "数据加载完成！" << endl;

  Menu::run(list);

  return 0;

}