山西工程技术学院

《程序设计基础》课程设计报告

（ 2023 - 2024 学年第 二 学期）

系　　部：大数据与智能工程系

课程名称：程序设计基础课程设计

专业班级：

学　　号：

学生姓名：

任课教师：

2024 年 6 月

目　录

摘　要

本系统依据开发要求主要应用于教育系统，完成对日常的教育工作中学生成绩档案的数字化管理。开发本系统可使学院教职员工减轻工作压力，此较系统地对教务、教学上的各项服务和信息进行管理，同时，可以减少劳动力的使用，加快查询速度、加强管理，以及国家各部门关于信息化的步伐，使各项管理更加规范化。

本系统在开发过程中，注意使其符合操作的业务流程，并力求系统的全面性、通用性，使得本系统不只适用于一家学校。在开发时，我们进行了系统调查研究、系统分析、系统设计和系统实施四个主要阶段进行设计，而在具体的设计上，采取了演化式原型法，随着用户的使用及对系统了解的不断加深，对某一部分或几部分进行重新分析、设计、实施。

关键词：C/C++；学生信息管理系统；多功能；自动化管理，链表；

第 1 章　开发背景

1.1 设计学生信息管理系统的背景和意义

1.1.1 设计学生信息管理系统的背景

随着时代发展，高等教育也逐步普及开来，越来越多学生步入高等教育的门槛，进入高等院校进行深造，据不完全统计 2023 年进入本科院校进行学习的学生约有 500 万人，如此之多的学生进入高等院校，必然需要有效的学生管理系统才能对学生的成绩进行有效且科学管理。

传统上，学生成绩管理采用手工方式，这种方式存在许多不足，例如：效率低下、容易出错、数据难以维护和更新等。因此，开发一个自动化的学生成绩管理系统成为一种迫切的需求。通过学生成绩管理系统，教师可以更加方便地查看学生的学习情况，及时发现学生的学习问题，并采取相应的措施进行辅导和帮助，从而提高教学质量。学生和家长可以通过学生成绩管理系统，及时查看学生的成绩和其他相关信息，方便快捷地了解学生的学习情况，更好地参与学生的学业管理，随着教育改革的深入，对于学生的各项信息管理也提出了更高的要求。学生成绩管理系统可以满足教育改革对于学生信息管理的需求，提供更加全面、准确的学生成绩信息，方便学校、老师、学生和家长的使用。近年来，计算机技术和网络技术得到了飞速的发展，这为开发学生成绩管理系统提供了可能。通过使用计算机和网络技术，可以实现学生成绩的自动化管理和维护，提高管理效率和准确性。

1.1.2 设计学生信息管理系统的意义

1.提高管理效率：传统的学生成绩管理方式通常需要人工记录和整理，不仅容易出错，而且效率低下。通过设计一个自动化的学生成绩管理系统，可以大大提高管理效率，减少错误率，并且可以节省大量的人力物力。

2.方便学生和教师查询：学生和教师可以随时通过学生成绩管理系统查询自己的成绩，无需到教务处或者教学科进行查询，方便快捷。

3.方便统计分析：学生成绩管理系统可以方便地进行统计分析，例如计算平均分、及格率、优秀率等指标，为教学评估和改进提供数据支持。

4.节省人力物力：通过自动化的管理系统，可以减少学校在人工成绩管理方面的投入，节省人力物力。

5.促进学校信息化建设：学生成绩管理系统是学校信息化建设的重要组成部分，可以提高学校信息化水平

综上所述，设计学生信息管理系统具有重要的意义和目的，可以提高工作效率、方便快捷的查询、确保数据准确性、统计分析功能、提高教学质量、加强学生管理、节省人力物力、促进学校信息化建设、保密性和安全性等。

1.2 设计内容

1.学生信息录入：学生基本信息（如姓名、班级、学号等）通过链表和新建节点等操作，在新节点的数据域完成对学生基本信息的录入，而指针域指向下一结点，以建立与其他数据的联系。

2 学生信息浏览：通过链表的建立和对链表的操作，完成“按学号查询”“按班级查询”“按照姓名查询”“统计班级人数”“统计省份人数”等操作

3.学生信息修改，通过在链表上查找到相应的数据位置，再修改数据域的相关数据。

4.学生信息删除，通过链表操作，找到想要删除的数据，先改变其前后节点的关系，再删除该结点。

5.学生信息排序：对结构体的元素进行升序和降序排列，通过调用函数 void sortUp(struct Node* headNode) 和 void sortDrop (struct Node* headNode)，其中使用算法对数据进行排序，然后输出。

6.学生信息统计：对结构体元素 data.addr 的内容进行检查和统计，实现“按省份统计”和“按班级统计”的统计结果。

7.文件的保存：使用函数 void writeInFromFile(struct Node* headNode,char* filename),进行了文件读写操作，让文件保存在特定 txt 文档里，让结果得以长时间保存。

第 2 章　设计学生成绩管理系统的开发环境

Visual Studio Code（简称“VS Code”）是 Microsoft 在 2015 年 4 月 30 日 Build 开发者大会上正式宣布一个运行于 Mac OS X、Windows 和 Linux 之上的，针对于编写现代 Web 和云应用的跨平台源代码编辑器，可在桌面上运行，并且可用于 Windows，macOS 和 Linux。它具有对 JavaScript，TypeScript 和 Node.js 的内置支持，并具有丰富的其他语言（例如 C++，C＃，Java，Python，PHP，Go）和运行时（例如.NET 和 Unity）扩展的生态系统。

这标志着微软公司第一次向开发者们提供了一款真正的跨平台编辑器。虽然完整版的 Visual Studio 仍然是只能运行在 Windows 和 macOS（Mac OS X ）之上，但是这一次的声明展示了微软公司对于支持其他计算机平台的承诺

该编辑器集成了所有一款现代编辑器所应该具备的特性，包括语法高亮（syntax high lighting），可定制的热键绑定（customizable keyboard bindings），括号匹配（bracket matching）以及代码片段收集（snippets）。Somasegar 也告诉笔者这款编辑器也拥有对 Git 的开箱即用的支持。Microsoft Docs（微软文档）提供了相应的学习教程帮助用户在 Visual Studio Code 中登陆 GitHub。

Visual Studio Code 提供了丰富的快捷键。用户可通过快捷键 [Ctrl]＋[K]＋[S] （按住 Ctrl 键不放，再按字母 K 键和 S 键）调出快捷键面板，查看全部的快捷键定义。也可在面板中双击任一快捷键，为某项功能指定新的快捷键。一些预定义的常用快捷键包括：格式化文档（整理当前视图中的全部代码），[Shift]＋[Alt]＋[F]；格式化选定内容（整理当前视图中被选定部分代码），[Ctrl] + [K] + [F]；放大视图，[Ctrl] + [Shift] + [=]；缩小视图，[Ctrl]＋[Shift]＋[-]；打开新的外部终端（打开新的命令行提示符）：[Ctrl]＋[Shift] + [C]。

该编辑器支持多种语言和文件格式的编写，截止 2019 年 9 月，已经支持了如下 37 种语言或文件：F#、HandleBars、Markdown、Python、Java、PHP、Haxe、Ruby、Sass、Rust、PowerShell、Groovy、R、Makefile、HTML、JSON、TypeScript、Batch、Visual Basic、Swift、Less、SQL、XML、Lua、Go、C++、Ini、Razor、Clojure、C#、Objective-C、CSS、JavaScript、Perl、Coffee Script、Dockerfile、Dart。

第 3 章　总体设计

3.1 菜单界面设计

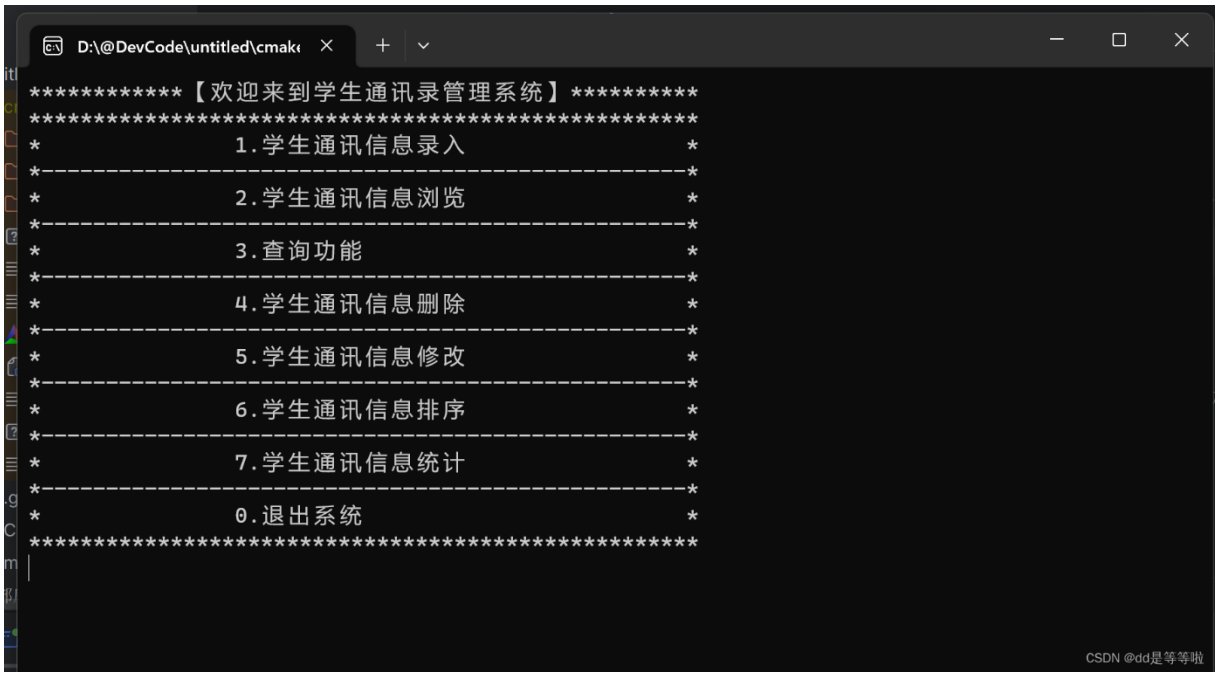

图 3-1 菜单运行界面

程序是一个学生通讯录管理系统的简单示例，基于命令行界面。通过该系统，用户可以进行学生信息的录入、浏览、查询、删除、修改、排序和统计等操作。

程序主要包括两个函数：menu()和 keyDown()。

menu()函数用于显示系统的主菜单，通过 printf 语句打印菜单选项供用户选择。

keyDown()函数用于处理用户选择的菜单选项。它首先读取用户输入的选项，然后使用 switch 语句根据选项的值执行相应的功能。

根据用户选择的不同选项，执行相应的操作：

选择 1：录入学生信息，并通过 insertNodeByHead()函数将信息插入链表中。

选择 2：打印所有学生信息，通过 printList()函数遍历链表并输出学生信息。

选择 3：进入查询功能子菜单，通过调用 keydown1()函数实现按学号、姓名、班级进行查询。

选择 4：进入删除功能子菜单，通过调用 keydown2()函数实现按学号、姓名进行删除。

选择 5：进入修改功能子菜单，通过调用 keydown3()函数实现按学号、姓名进行修改。

选择 6：进入排序功能子菜单，通过调用 keydown4()函数实现按学号升序、降序排序。

选择 7：进入统计功能子菜单，通过调用 keydown5()函数实现统计各班人数、各省份人数。

选择 0：退出系统，并将链表中的数据写入文件。

程序中还包含一些输出语句用于显示相关提示信息。

3.2 录入和浏览

insertNodeByHead()函数用于将新的学生信息插入到链表的头部。它接受两个参数：

headNode 表示链表的头节点，data 表示要插入的学生信息。函数首先创建一个新节点 newNode，通过调用 createNode()函数创建一个包含指定学生信息的节点。然后，将新节点的 next 指针指向原先头节点的下一个节点，将头节点的 next 指针指向新节点，从而完成插入操作。

图 3-2 信息录入

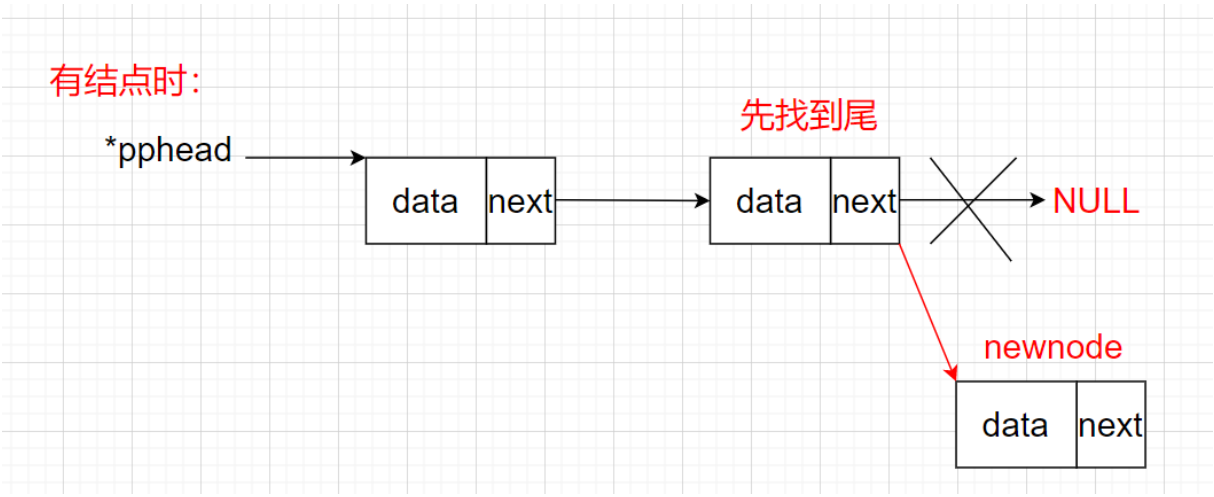

图 3-3 录入信息链表示意图

printList()函数用于打印整个学生通讯录链表中的学生信息。它接受一个参数 headNode，表示链表的头节点。函数通过使用一个指针 pMove 从链表的第一个节点开始，逐个遍历链表节点并输出学生信息，直到遍历到链表末尾（即指针为 NULL）。在输出学生信息时，使用 printf()函数按照指定的格式输出学号、姓名、班级、地址、电话和专业等信息。

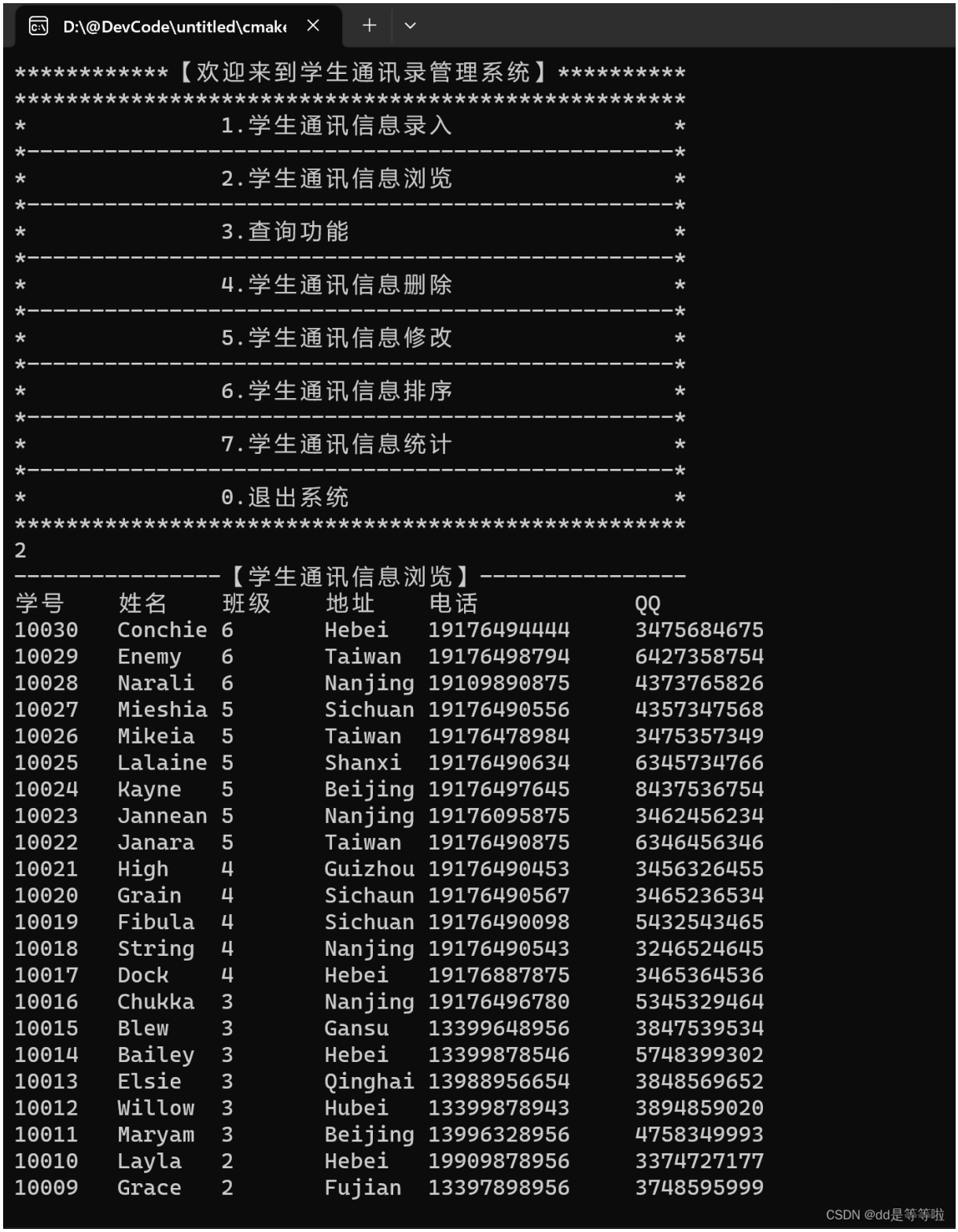

图 3-4 浏览信息

3.3 删除和修改

3.3.1 删除

deleteNumNode()函数用于按学号删除节点。它接受两个参数：headNode 表示链表

的头节点，num 表示要删除的学生的学号。函数通过遍历链表找到学号匹配的节点，并将其从链表中删除。具体步骤如下：

初始化两个指针：posNode 指向链表的第一个节点，posFrontNode 指向头节点。
使用循环遍历链表，判断当前节点的学号是否与目标学号 num 匹配。
如果找到匹配的节点，将前一个节点 posFrontNode 的 next 指针指向当前节点的下一个节点，从而跳过当前节点。
释放当前节点的内存空间，避免内存泄漏。
输出删除成功的提示信息。

图 3-5 按学号删除

deleteNameNode()函数用于按姓名删除节点。它接受两个参数：headNode 表示链表的头节点，name 表示要删除的学生的姓名。函数通过遍历链表找到姓名匹配的节点，并将其从链表中删除。具体步骤如下：
初始化两个指针：posNode 指向链表的第一个节点，posFrontNode 指向头节点。
使用循环遍历链表，判断当前节点的姓名是否与目标姓名 name 匹配。
如果找到匹配的节点，将前一个节点 posFrontNode 的 next 指针指向当前节点的下一个节点，从而跳过当前节点。
释放当前节点的内存空间，避免内存泄漏。
输出删除成功的提示信息。

3.3.2 修改

alterNumNode()函数用于按学号修改节点。它接受两个参数：headNode 表示链表的头节点，num 表示要修改的学生的学号。函数通过遍历链表找到学号匹配的节点，并将其从链表中删除。然后，函数要求用户重新输入学生的学号、姓名、班级、地址、电话和专业，并调用 insertNodeByHead()函数将修改后的学生信息插入到链表的头部。具体步骤如下：
初始化两个指针：posNode 指向链表的第一个节点，posFrontNode 指向头节点。
使用循环遍历链表，判断当前节点的学号是否与目标学号 num 匹配。
如果找到匹配的节点，将前一个节点 posFrontNode 的 next 指针指向当前节点的下一个节点，从而跳过当前节点。
释放当前节点的内存空间，避免内存泄漏。
提示用户重新输入学生的学号、姓名、班级、地址、电话和专业。
创建一个包含新学生信息的结构体变量 data，并将用户输入的信息赋值给 data。
调用 insertNodeByHead()函数，将新的学生信息插入到链表的头部。
输出修改成功的提示信息。

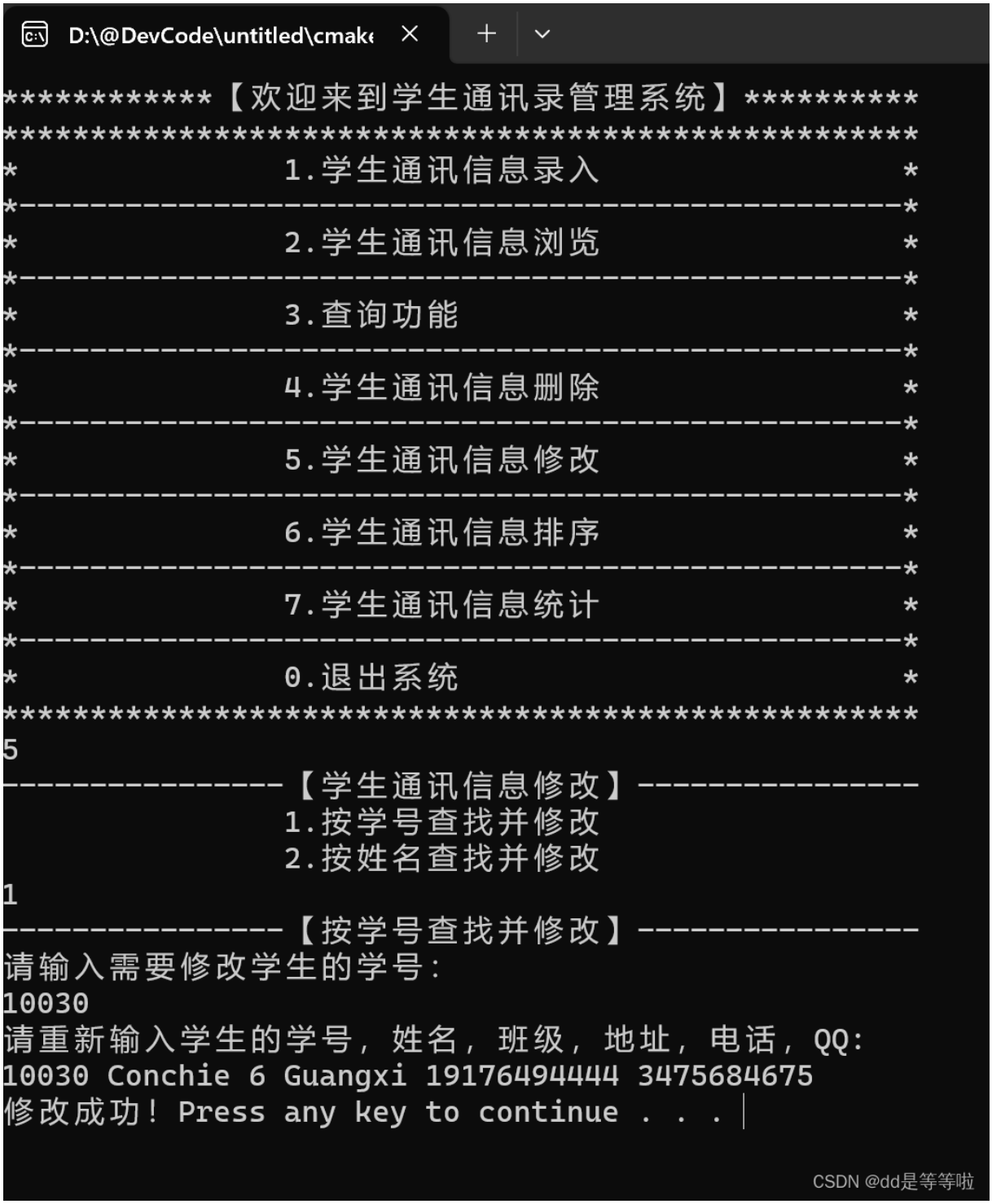
D:\@DevCode\untitled\cmake
**********【欢迎来到学生通讯录管理系统】**********
1.学生通讯信息录入
2.学生通讯信息浏览
3.查询功能
4.学生通讯信息删除
5.学生通讯信息修改
6.学生通讯信息排序
7.学生通讯信息统计
0.退出系统
5
----------【学生通讯信息修改】----------
1.按学号查找并修改
2.按姓名查找并修改
1
----------【按学号查找并修改】----------
请输入需要修改学生的学号:
10030
请重新输入学生的学号，姓名，班级，地址，电话，QQ:
10030 Conchie 6 Guangxi 19176494444 3475684675
修改成功! Press any key to continue . . .
CSDN @dd是等等啦

图 3-6 按学号修改信息

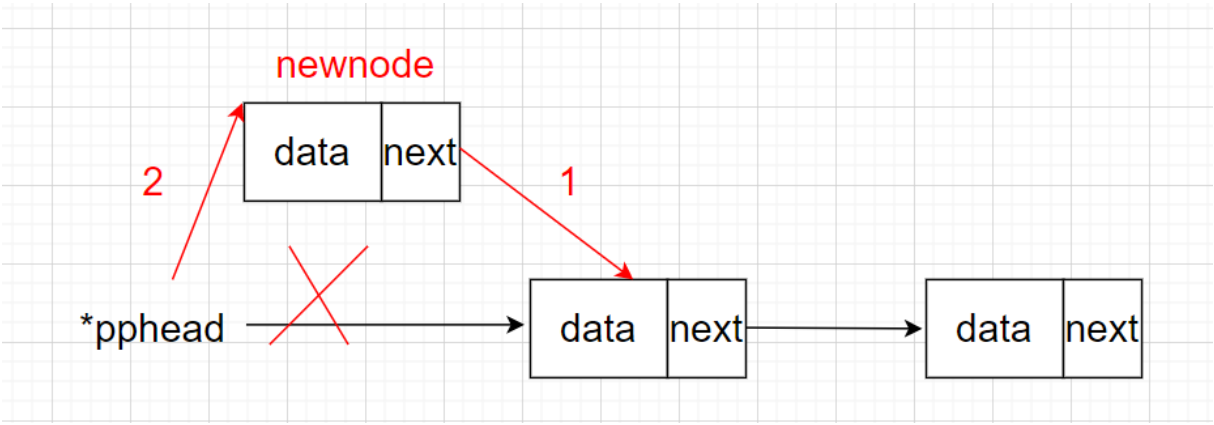

图 3-6 头插法

alterNameNode()函数用于按姓名修改节点。它接受两个参数：headNode 表示链表的头节点，name 表示要修改的学生的姓名。函数通过遍历链表找到姓名匹配的节点，并将其从链表中删除。然后，函数要求用户重新输入学生的学号、姓名、班级、地址、电话和专业，并调用 insertNodeByHead()函数将修改后的学生信息插入到链表的头部。具体步骤与 alterNumNode()函数类似。

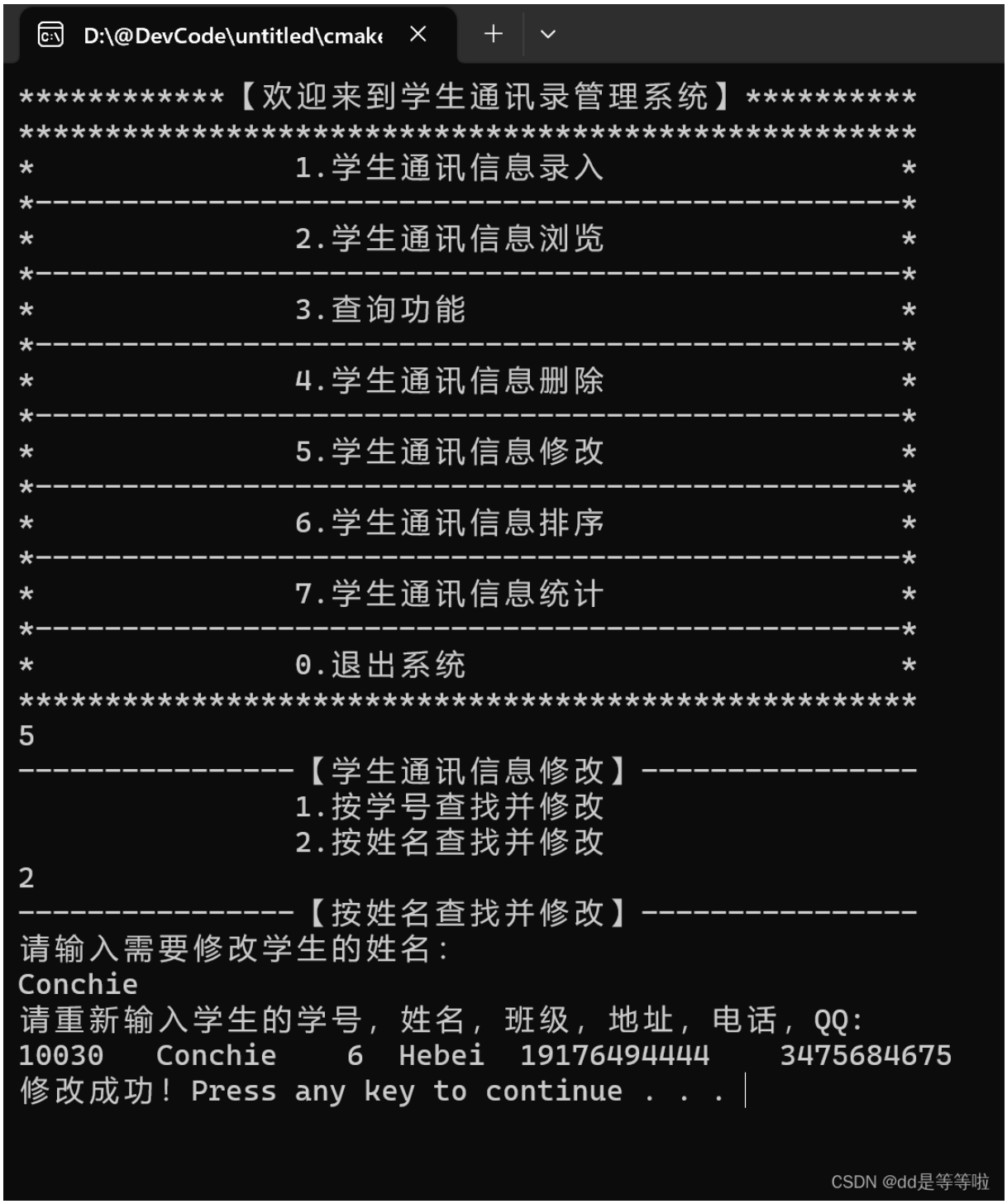

图 3-7 按姓名修改信息

3.4 排序

3.4.1 排序操作

sortUp()函数用于对链表中的学生信息按学号进行升序排序。它接受一个参数headNode，表示链表的头节点。函数使用冒泡排序算法对链表进行排序。具体步骤如下：

（1）定义两个指针 turn 和 move，分别用于遍历链表。
（2）使用外层循环 turn 遍历链表中的节点，从头节点的下一个节点开始。
（3）使用内层循环 move 遍历链表中的节点，从头节点的下一个节点开始。
（4）比较当前节点 move 的学号与下一个节点的学号，如果当前节点的学号大于下一个节点的学号，则交换两个节点的数据。
（5）重复步骤 3 和步骤 4，直到内层循环结束。
（6）重复步骤 2 到步骤 5，直到外层循环结束。
（7）输出排序后的链表。
sortDrop()函数用于对链表中的学生信息按学号进行降序排序。它的实现与 sortUp()函数类似，只是在比较学号大小时使用了不同的判断条件。具体步骤与 sortUp()函数相同

3.4.2 排序算法

Bubble_sort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order. (Introduction to Algorithms)[1]

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

图 3-8 冒泡排序算法伪代码

第 4 章　学生成绩管理系统结构图和流程图

系统函数调用模型↵

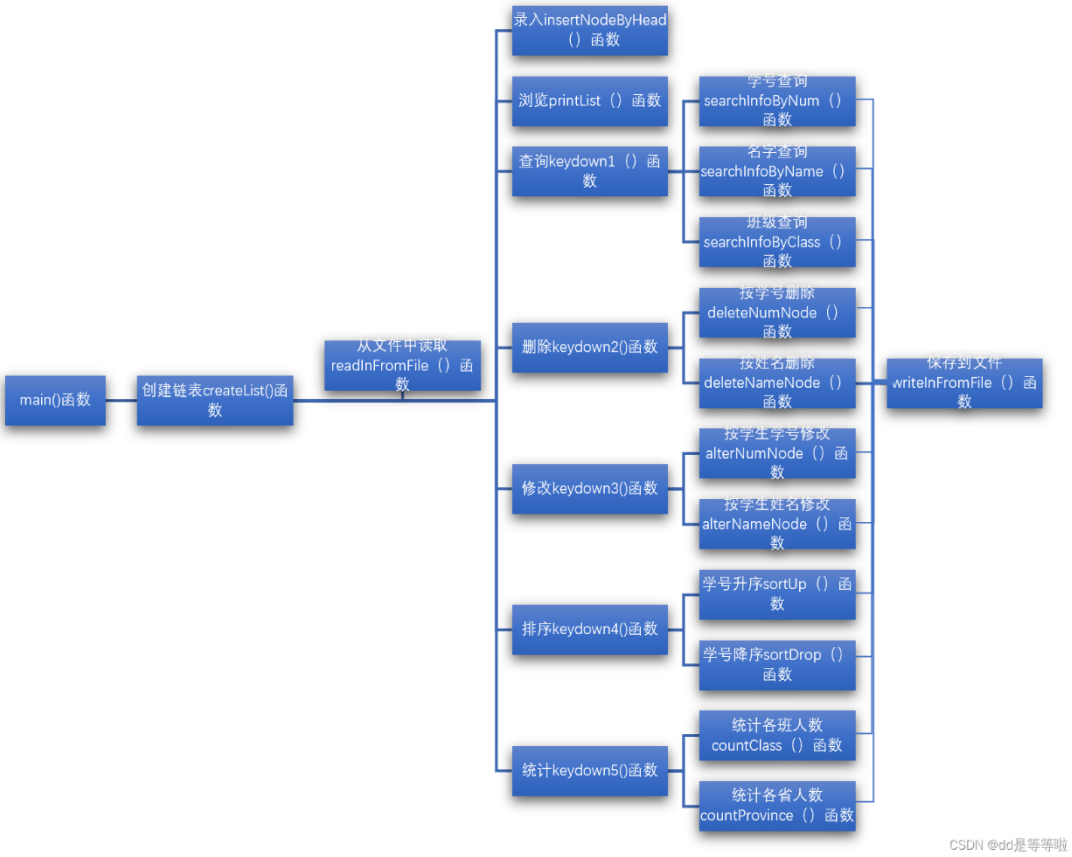

图 4-1　流程图

第 5 章　调试结果

5.1 管理系统界面

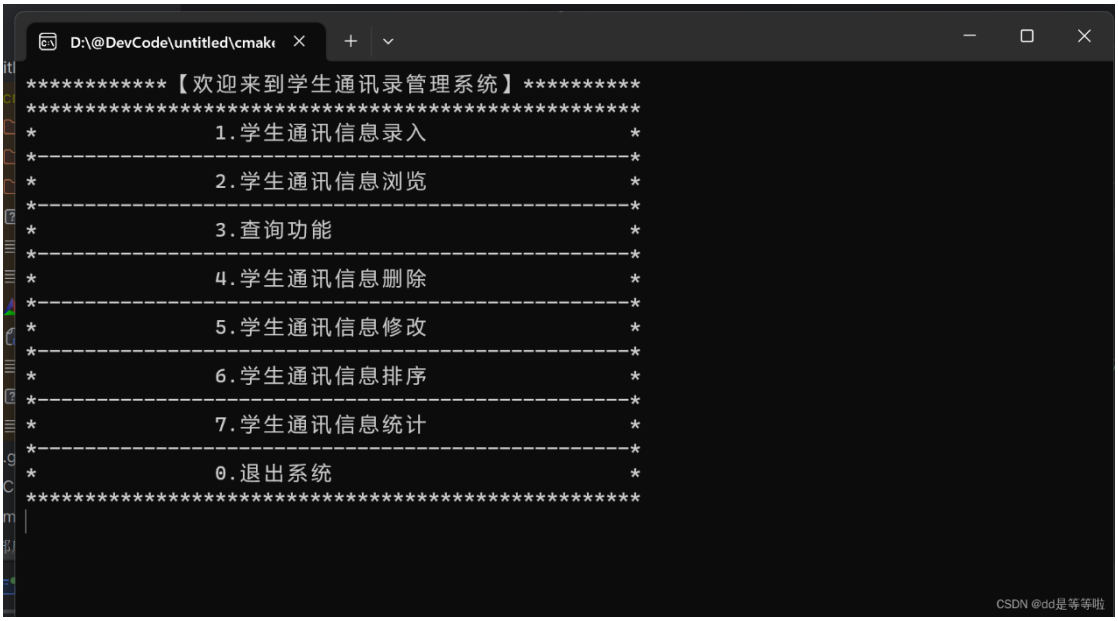

图 5-1 学生信息管理系统引导界面

5.2 管理人员控制

根据提示输入数字（1.2.3.4.5.6.7.8.0）来进行操作（学生通讯信息录入，学生通讯信息浏览，查询，学生通讯信息删除，学生通讯信息修改，学生通讯信息排序，学生通讯信息统计）

5.3 操作

根据提示进行输入后就会调取相应的函数进行录入/浏览/查询/删除/修改/排序/统计。。

也调取了 student.txt 文件进行保存。

第 6 章　设计总结

这个程序是一个学生通讯录管理系统，通过命令行界面提供了一系列功能，包括录入学生信息、浏览信息、查询信息、删除信息、修改信息、排序信息和统计信息。下面是对程序设计的总结：

• 数据结构：

程序中定义了两个结构体：struct student 和 struct Node。

struct student 用于存储学生的学号、姓名、班级、地址、电话和专业等信息。

struct Node 是链表节点结构，包含一个学生信息的数据成员 data，以及指向下一个节点的指针 next。

主要函数：

createList()函数用于创建一个空的链表，并返回链表的头节点。

createNode()函数用于创建一个新的节点，并将给定的学生信息存储在节点的数据成员中。

menu()函数用于打印主菜单，展示程序提供的功能选项。

keyDown()函数是程序的主控函数，根据用户输入的选项执行相应的操作。

insertNodeByHead()函数用于在链表的头部插入一个新节点，将学生信息存储在节点中。

printList()函数用于遍历链表并打印所有学生信息。

searchInfoByNum()函数通过学号在链表中查找并打印对应的学生信息。

searchInfoByName()函数通过姓名在链表中查找并打印对应的学生信息。

deleteNumNode()函数通过学号在链表中删除对应的学生节点。

deleteNameNode()函数通过姓名在链表中删除对应的学生节点。

alterNumNode()函数通过学号在链表中查找并修改对应的学生信息。

alterNameNode()函数通过姓名在链表中查找并修改对应的学生信息。

sortUp()函数按学号升序对链表中的学生信息进行排序。

sortDrop()函数按学号降序对链表中的学生信息进行排序。

countClass()函数统计各个班级的学生人数并打印结果。

countProvince()函数统计各个省份的学生人数并打印结果。

readInFromFile()函数从文件中读取学生信息并存储到链表中。

writeInFromFile()函数将链表中的学生信息写入文件中。

程序流程：

main()函数首先创建一个空的链表，并调用 readInFromFile()函数从文件中读取学生信息。

然后进入一个无限循环，循环中调用 menu()函数打印菜单，接收用户输入的选项，并调用 keyDown()函数执行相应的操作。

在 keyDown()函数中，根据用户选择的选项执行相应的功能，如录入信息、浏览信

息、查询信息等。

每次操作结束后，调用 writeInFromFile()函数将更新后的学生信息写入文件中。

循环继续，直到用户选择退出系统。

文件操作：

程序通过调用 readInFromFile()函数从文件中读取学生信息，在程序开始时进行初始化。

在每次对学生信息进行修改后，通过调用 writeInFromFile()函数将更新后的信息写入文件中，以便下次程序运行时读取最新的数据。

总体而言，该程序通过链表数据结构实现了学生通讯录的管理功能，提供了多种操作选项，并支持文件的读写操作，使得学生信息的录入、浏览、查询、修改、排序和统计变得方便和高效。

参考文献

[1]《Introduction to Algorithm》Thomas H. Cormen .Charles E. Leiserson . Ronald L. Rivest

[2] Kernighan, B.W.,& Ritchie, D.M(1988). The C programming language. Prentice-Hall.

[3] Prate, S.(2013). C primer plus. Pearson Education.

[4]管纪文，刘大有。数据结构，高等教育出版社，1985

[5]Robert L K, Alexander J R.数据结构与程序设计——C++语言描述，高等教育出版社，2001

附录：源程序代码

```
#include<stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 6

struct student{
	int num;
	char name[20];
	int class;
	char addr[50];
	char tel[20];
	char major[50];
};

struct Node{
	struct student data;
	struct Node *next;
};

struct Node *createList(){
	struct Node* headNode = (struct Node*)malloc(sizeof(struct Node));
	headNode->next = NULL;
	return headNode;
}

struct Node *createNode(struct student data){
	struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
	newNode->data = data;
	newNode->next = NULL;
	return newNode;
}

void menu();
void keyDown();
void keydown1();
void keydown2();
void keydown3();
void keydown4();
void keydown5();
void insertNodeByHead(struct Node* headNode,struct student data);
void printList(struct Node* headNode);
```

```
void searchInfoByNum(struct Node* headNode,int num);
void searchInfoByName(struct Node* headNode,char *name);
void searchInfoByClass(struct Node* headNode,int class);
void deleteNumNode(struct Node* headNode,int num);
void deleteNameNode(struct Node* headNode,char *name);
void alterNumNode(struct Node* headNode,int num);
void alterNameNode(struct Node* headNode,char* name);
void sortUp(struct Node* headNode);
void sortDrop(struct Node* headNode);
void countClass(struct Node* headNode);
void countProvince(struct Node* headNode);
void readInFromFile(struct Node* headNode,char* filename);
void writeInFromFile(struct Node* headNode,char* filename);

struct Node* list = NULL;

int main()
{
	list = createList();
	readInFromFile(list,"student.txt");
	while (1)
	{
		menu();
		keyDown();
		system("pause");
		system("cls");
	}
}

void menu(){
	printf("*****************【学生通讯录管理系统】**************\n");
	printf("*\t\t1.学生通讯信息录入                          *\n");
	printf("*-------------------------------------------------*\n");
	printf("*\t\t2.学生信息浏览                              *\n");
	printf("*-------------------------------------------------*\n");
	printf("*\t\t3.查询功能                                  *\n");
	printf("*-------------------------------------------------*\n");
	printf("*\t\t4.学生信息删除                              *\n");
	printf("*-------------------------------------------------*\n");
	printf("*\t\t5.学生信息修改                              *\n");
	printf("*-------------------------------------------------*\n");
	printf("*\t\t6.学生信息排序                              *\n");
	printf("*-------------------------------------------------*\n");
	printf("*\t\t7.学生信息统计                              *\n");
```

```
    printf("*--------------------------------------------------*\n");
    printf("*\t\t0.退出系统                                      *\n");
    printf("****************************************************\n");
}

void keyDown(){
    struct student data;
    int choice = 0;
    scanf("%d",&choice);
    switch(choice){
        case 0:
            printf("正常退出。");
            system("pause");
            exit(0);
            break;
        case 1:
            printf("----------【学生通讯信息录入】----------\n");
            printf("请输入学生的学号，姓名，班级，地址，电话，专业:\n");
            fflush(stdin);

scanf("%d %s %d %s %s %s",&data.num,data.name,&data.class,data.addr,data.tel,data.major);
            insertNodeByHead(list,data);
            printf("录入成功！");
            break;
        case 2:
            printf("----------【学生通讯信息浏览】----------\n");
            printList(list);
            break;
        case 3:
            printf("----------【查询功能】----------\n");
            printf("\t\t1.按学号查询\n");
            printf("\t\t2.按姓名查询\n");
            printf("\t\t3.按班级查询\n");
            keydown1();
            break;
        case 4:
            printf("----------【学生通讯信息删除】----------\n");
            printf("\t\t1.按学号删除\n");
            printf("\t\t2.按姓名删除\n");
            keydown2();
            break;
        case 5:
            printf("----------【学生通讯信息修改】----------\n");
            printf("\t\t1.按学号查找并修改\n");
```

```
            printf("\t\t2.按姓名查找并修改\n");
            keydown3();
            break;
        case 6:
            printf("----------【学生通讯信息排序】----------\n");
            printf("\t\t1.按学号升序排序\n");
            printf("\t\t2.按学号降序排序\n");
            keydown4();
            break;
        case 7:
            printf("----------【学生通讯信息统计】----------\n");
            printf("\t\t1.统计各班人数\n");
            printf("\t\t2.统计各省份人数\n");
            keydown5();
            break;
        default:
            printf("选择错误，请重新输入.....\n");
            system("pause");
    }
    writeInFromFile(list,"student.txt");
}
void keydown1(){
    int choice = 0;
    scanf("%d",&choice);
    struct student data;
    switch(choice){
        case 1:
            printf("----------【按学号查询】----------\n");
            printf("请输入学号：");
            scanf("%d",&data.num);
            searchInfoByNum(list,data.num);
            break;
        case 2:
            printf("----------【按姓名查询】----------\n");
            printf("请输入姓名：");
            scanf("%s",data.name);
            searchInfoByName(list,data.name);
            break;
        case 3:
            printf("----------【按班级查询】----------\n");
            printf("请输入班级：");
            scanf("%d",&data.class);
            searchInfoByClass(list,data.class);
            break;
```

```
        default:
            printf("选择错误，请重新输入.....\n");
            system("pause");
            break;
    }
}
void keydown2(){
    struct student data;
    int choice = 0;
    scanf("%d",&choice);
    switch(choice){
        case 1:
            printf("----------【按学号删除】----------\n");
            printf("请输入删除学生的学号：\n");
            scanf("%d",&data.num);
            deleteNumNode(list,data.num);
            break;
        case 2:
            printf("----------【按姓名删除】----------\n");
            printf("请输入删除学生的姓名：\n");
            scanf("%s",data.name);
            deleteNameNode(list,data.name);
            break;
        default:
            printf("选择错误，请重新输入.....\n");
            system("pause");
            break;
    }
}
void keydown3() {
    struct student data;
    int choice = 0;
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            printf("----------【按学号查找并修改】----------\n");
            printf("请输入需要修改学生的学号：\n");
            scanf("%d", &data.num);
            alterNumNode(list, data.num);
            break;
        case 2:
            printf("----------【按姓名查找并修改】----------\n");
            printf("请输入需要修改学生的姓名：\n");
            scanf("%s", data.name);
```

```
                alterNameNode(list,data.name);
                break;
            default:
                printf("选择错误，请重新输入.....\n");
                system("pause");
                break;
        }
}
void keydown4(){
    int choice = 0;
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            printf("----------【按学号升序排序】----------\n");
            sortUp(list);
            break;
        case 2:
            printf("----------【按学号降序排序】----------\n");
            sortDrop(list);
            break;
        default:
            printf("选择错误，请重新输入.....\n");
            system("pause");
            break;
    }
}
void keydown5(){

    int choice = 0;
    scanf("%d", &choice);
    switch (choice) {
        case 1:
            printf("----------【统计各班人数】----------\n");
            countClass(list);
            break;
        case 2:
            printf("----------【统计各省份人数】----------\n");
            countProvince(list);
            break;
        default:
            printf("选择错误，请重新输入.....\n");
            system("pause");
            break;
    }
```

```
}

void insertNodeByHead(struct Node* headNode,struct student data)
{
    struct Node* newNode = createNode(data);
    newNode->next = headNode->next;
    headNode->next = newNode;
}

void printList(struct Node* headNode){
    struct Node* pMove = headNode->next;
    printf("学号\t 姓名\t 班级\t 地址\t 电话\t\t 专业\n");
    while (pMove){

printf("%d\t%s\t%d\t%s\t%s\t%s\n",pMove->data.num,pMove->data.name,pMove->data.class,pMove->da
ta.addr,pMove->data.tel,pMove->data.major);
        pMove = pMove->next;
    }
    printf("\n");
}

void searchInfoByNum(struct Node* headNode,int num){
    struct Node* pMove = headNode;
    while (pMove){
        if(pMove -> data.num == num){
            printf("学号\t 姓名\t 班级\t 地址\t 电话\t\t 专业\n");
            printf("%d\t%s\t%d\t%s\t%s\t%s\n",        pMove->data.num,        pMove->data.name,
pMove->data.class, pMove->data.addr,
                    pMove->data.tel, pMove->data.major);
            break;
        }
        pMove = pMove -> next;
    }
    if(pMove == NULL){
        printf("学号不存在，无法查找！ \n ");
        return ;
    }

}

void searchInfoByName(struct Node* headNode,char *name){
    struct Node* pMove = headNode;
    while (pMove){
        if(strcmp(pMove -> data.name,name) == 0){
```

```
                printf("学号\t 姓名\t 班级\t 地址\t 电话\t\t 专业\n");
                printf("%d\t%s\t%d\t%s\t%s\t%s\n",          pMove->data.num,          pMove->data.name,
pMove->data.class, pMove->data.addr,
                        pMove->data.tel, pMove->data.major);
                break;
            }
            pMove = pMove -> next;
        }
        if(pMove == NULL){
            printf("名字不存在，无法查找！\n ");
            return ;
        }
}

void searchInfoByClass(struct Node* headNode,int class){
        struct Node* pMove = headNode;
        printf("学号\t 姓名\t 班级\t 地址\t 电话\t\t 专业\n");
        while (pMove){
            if(pMove -> data.class == class){
                printf("%d\t%s\t%d\t%s\t%s\t%s\n",          pMove->data.num,          pMove->data.name,
pMove->data.class, pMove->data.addr,
                        pMove->data.tel, pMove->data.major);
            }
            pMove = pMove -> next;
        }

        if(pMove == NULL){
            printf("班级不存在，无法查找！\n ");
            return ;
        }
}

void deleteNumNode(struct Node* headNode,int num){
        struct Node* posNode = headNode -> next;
        struct Node* posFrontNode = headNode;
        while(posNode -> data.num != num)
        {
            posFrontNode = posNode;
            posNode = posNode -> next;
        }
        if(posNode == NULL){
            printf("未找学号位置，无法删除！");
            return;
        }
```

```
    else{
        posFrontNode -> next = posNode -> next;
        free(posNode);
        printf("删除成功！");
    }

}

void deleteNameNode(struct Node* headNode,char *name){
    struct Node* posNode = headNode -> next;
    struct Node* posFrontNode = headNode;
    if(posNode == NULL){
        printf("名字不存在，无法删除！\n ");
        return;
    }
    while(strcmp(posNode -> data.name,name))
    {
        posFrontNode = posNode;
        posNode = posNode -> next;
    }
    if(posNode == NULL){
        printf("未找名字位置，无法删除！");
        return;
    }
    else{
        posFrontNode -> next = posNode -> next;
        free(posNode);
        printf("删除成功!");
    }

}

void alterNumNode(struct Node* headNode,int num){
    struct Node* posNode = headNode -> next;
    struct Node* posFrontNode = headNode;
    while(posNode -> data.num != num)
    {
        posFrontNode = posNode;
        posNode = posNode -> next;
    }
    if(posNode == NULL){
        printf("未找该学生，无法修改！");
        return;
    }
```

```
    else{
        struct student data;
        posFrontNode -> next = posNode -> next;
        free(posNode);
        printf("请重新输入学生的学号，姓名，班级，地址，电话，专业:\n");
        fflush(stdin);
        scanf("%d %s %d %s %s %s",&data.num,data.name,&data.class,data.addr,data.tel,data.major);
        insertNodeByHead(list,data);
        printf("修改成功！");
    }
}

void alterNameNode(struct Node* headNode,char* name){
    struct Node* posNode = headNode -> next;
    struct Node* posFrontNode = headNode;
    while(strcmp(posNode -> data.name,name))
    {
        posFrontNode = posNode;
        posNode = posNode -> next;
    }
    if(posNode == NULL){
        printf("未找该学生，无法修改！");
        return;
    }
    else{
        struct student data;
        posFrontNode -> next = posNode -> next;
        free(posNode);
        printf("请重新输入学生的学号，姓名，班级，地址，电话，专业:\n");
        fflush(stdin);
        scanf("%d %s %d %s %s %s",&data.num,data.name,&data.class,data.addr,data.tel,data.major);
        insertNodeByHead(list,data);
        printf("修改成功！");
    }
}

void sortUp(struct Node* headNode){
    struct Node* turn;
    struct Node* move;
    struct student temp;
    for(turn = headNode -> next;turn -> next != NULL;turn = turn -> next){
        for(move = headNode -> next;move -> next != NULL;move = move -> next){
            if(move -> data.num > move -> next -> data.num){
                temp = move -> data;
```

```
                move -> data = move -> next -> data;
                move -> next -> data = temp;
            }
        }
    }
    printList(list);
}

void sortDrop(struct Node* headNode){
    struct Node* turn;
    struct Node* move;
    struct student temp;
    for(turn = headNode -> next;turn -> next != NULL;turn = turn -> next){
        for(move = headNode -> next;move -> next != NULL;move = move -> next){
            if(move -> data.num < move -> next -> data.num){
                temp = move -> data;
                move -> data = move -> next -> data;
                move -> next -> data = temp;
            }
        }
    }
    printList(list);
}

void countClass(struct Node* headNode){
    int i;
    int count[N + 1] = {0};
    struct Node* pMove = headNode;
    while (pMove){
        for( i = 1;i <= N; i++){
            if (pMove->data.class == i ) count[i]++;
        }
        pMove = pMove -> next;
    }
    for (int i = 1;i <= N; i++) {
        printf("%d 班的人数为：%d\n",i,count[i]);
    }
}

void countProvince(struct Node* headNode){
    char province[34][100] =
{"Hebei","Shanxi","Liaoning","Jilin","Heilongjiang","Jiangsu","Zhejiang","Anhui","Fujian","Jiangxi","Sh
andong","Henan","Hubei","Hunan","Guangdong","Hannan","Sichuan","Guizhou","Yunnan","Shanxi","Ga
nsu","Qinghai","Taiwan","Neimenggu","Guangxi","Xizang","Ningxia","Xinjiang","Beijiang","Tianjing","
```

```
Shanghai","Chongqing","Xianggang","Aomen"};
    int i;
    int count[34] ={0};
    struct Node* pMove = headNode;
    while (pMove){
        for( i = 0;i < 34; i++){
            if (strcmp(pMove->data.addr,province[i]) == 0 ) count[i]++;
        }
        pMove = pMove -> next;
    }
    for (int i = 0;i < 34; i++) {
        if(count[i] != 0)
            printf("地址为%s 的人数为%d\n",province[i],count[i]);
    }
}

void readInFromFile(struct Node* headNode,char* filename){
    struct student data;
    FILE *fp;
    fp = fopen(filename,"r");
    if(fp == NULL){
        printf("文件打开失败！ ");
    }
    while
(fscanf(fp,"%d\t%s\t%d\t%s\t%s\t%s\n",&data.num,data.name,&data.class,data.addr,data.tel,data.major) !=
EOF){
        insertNodeByHead(headNode,data);
    }

    fclose(fp);
}

void writeInFromFile(struct Node* headNode,char* filename){
    FILE *fp = fopen(filename,"w");
    struct Node* pMove = headNode -> next;
    while(pMove){

fprintf(fp,"%d\t%s\t%d\t%s\t%s\t%s\n",pMove->data.num,pMove->data.name,pMove->data.class,pMove-
>data.addr,pMove->data.tel,pMove->data.major);
        pMove = pMove -> next;
    }
    fclose(fp);
}
```

评阅意见

评分标准

1、按时完成规定任务。（10 分）

2、格式规范，页面排版严格按照规范标准设计。（20 分）

3、观点正确，结构合理，包含项目规定的内容。（20 分）

4、掌握知识程度。（20 分）

5、综合应用知识能力，综合分析处理问题能力。（30 分）

评阅意见

评分

指导教师：　　日期：