目录

1. **需求分析···········································2**
2. **概要设计···········································3**
3. **详细设计···········································5**
4. **调试分析··········································20**
5. **用户手册··········································21**
6. **测试数据··········································21**
7. **附录··············································27**
8. **需求分析**

（一）

个人简历录入程序

编写一个个人简历录入程序，可实现相关信息的添加、删除、修改。

要求：

1. 通过文本框录入姓名，通过单选按钮选择性别，通过组合框选择专业（计算机，电子，工商等）和文化程度（大专，本科，硕士，博士等），设置“提交”与“取消”两个按键，当用鼠标点击“提交”按键时，在文本框中显示所填写以及选择的信息。当点击“取消”按键退出系统。
2. 实现查询与统计功能（如35岁以下计算机专业的人数）
3. 实现存储与读取

（要求：1.数据用结构体方式实现。2.数据存储采用文件形式。3.标准的C输入输出。4.功能完善，适当的注释。5.关于文件的内容需要自学）

1. **概要设计**

（一）

设计使用结构体数组struct DATA作为成员信息单位，另用一个结构体指针struct DATA \*next 标记下一个节点的位置，使用typedef struct DATA data 将struct DATA代替成data，采用malloc函数开辟新节点并存入文件datas.myC中。

（二）

所有个人简历信息采用文本文件存储

生成的文件datas.myC用来存储所有的个人简历信息

（三）结构体中各成员的含义：

struct DATA {

char name[20], //成员的姓名

sex[20], //成员的性别

major[50], //成员的专业

eduStatus[50];//成员的学历

int age; //成员的年龄

struct DATA \*next; //指向下一个节点的结构体指针

};

（四）程序模块结构

Main

readFile

search

HowManyPeople

Switch

revise

displayAll

search

count

writeFileAll

delFunction

creatNew

deleteAll

addPeople

creatLink

deleteAll

search

delAStruct

**三、详细设计**

（一）成员简历信息结构体定义

struct DATA {

char name[20], //成员的姓名

sex[20], //成员的性别

major[50], //成员的专业

eduStatus[50]; //成员的学历

int age; //成员的年龄

struct DATA \*next; //指向下一个结构体的指针

};

（二）程序中引用的函数功能介绍

data \* readFile(data \*); //读取文件中的信息

void displayItem(int); //展示选项界面

data \*count(data \*); //统计功能

data \*displayAll(data \*); //展示链表成员

int HowManyPeople(data \*); //判断链表中有多少人

data \*creatNew(data \*); //添加信息功能

data \*creatLink(data \*); //创建链表

data \*addPeople(data \*); //在原链表里添加成员

data \*deleteAll(data \*); //删除所有信息

void writeAddFile(data \*); //重新写入新的数据

data \*delAStruct(data \*, data \*);//删除链表里的一个节点

void revise(data \*); //修改链表里的一个节点

data \*search(data \*, char \*); //在链表中查找名字相同的结构体

void delFunction(data \*); //删除信息功能的函数

void writeFileAll(data \*); //将链表里的所有信息写入文件

（三）各模块介绍

**（1）主函数模块**

int main() {

int choice;

data \*head, \*p;

char hisName[20];

head = (data \*)malloc(sizeof(data));

head -> next = NULL;

head = readFile(head); //这里是先读取保存的文件

displayItem(HowManyPeople(head)); //展示选项界面

while (scanf("%d", &choice)) { //当scanf成功接收到一个数字的时候会返回一个正值，满足while里面的条件。若接收到非数字则返回0，就不满足条件，直接退出

system("cls"); //清屏函数，保持界面整洁

switch (choice) { //判断用户的输入的数字

case 1:

getchar();

head = creatNew(head); //调用creatNew函数，创建新链表 或 添加一个节点

break;

case 2:

delFunction(head); //调用delFunction函数，删除整个链表 或 删除链表中指定节点

break;

case 3:

getchar();

revise(head); //调用revise函数，修改某个成员

break;

case 4:

head = displayAll(head);//调用displayAll函数，展示链表中所有数据

break;

case 5:

printf("输入你想查询的成员名字：\n");

getchar(); //在输入1~7选项的时候，还按了一个回车键，这个回车键会让下面的gets获取字符串出错，所以要用getchar()函数把回车键吃掉

gets(hisName);

p = search(head, hisName);//p返回的是查询到的那个人所在的节点的指针，如果返回NULL说明没找到

if (p == NULL)

printf("没有这个人！");

else

printf("姓名：%s\t性别：%s\t年龄：%d\t专业：%s\t文化程度：%s\n",

p -> name, p -> sex, p -> age, p -> major, p -> eduStatus);

break;

case 6:

head = count(head); //调用count函数，统计符合要求的成员

break;

case 7:

printf("正在更新数据，请稍等。");

writeFileAll(head); //调用writeFileAll函数，重新将链表中的数据写入文件

system("cls"); //清屏

printf("谢谢使用，再见！");

sleep(3); //延时函数，在这里是延时三秒

return 0;

}

displayItem(HowManyPeople(head));//展示功能面板

}

return 0;

}

**（2）展示可选项目函数模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：展示现有项目的列表和现有人数 \*

\*输入：现有人数 \*

\*返回：空 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void displayItem(int sum) {

printf("\n\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*欢迎来到个人简历录入程序\*\*\*\n");

printf("\*\*\*\*\*\*现在共有%d人的信息\*\*\*\*\*\*\n", sum);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\* 输入1：添加信息\n");

printf("\*\* 输入2：删除信息\n");

printf("\*\* 输入3：修改信息\n");

printf("\*\* 输入4：展示全部信息\n");

printf("\*\* 输入5：查询个人信息\n");

printf("\*\* 输入6：使用统计功能\n");

printf("\*\* 输入7：退出程序\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

**（3）添加信息功能模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：列表中添加信息功能 \*

\*输入：要写入的链表头指针 \*

\*返回：链表的头指针 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*creatNew(data \*head){

int choice;

printf("\*\* 输入1：在原有成员里添加新成员\n");

printf("\*\* 输入2：删除原有成员并创建新信息\n");

scanf("%d", &choice);

getchar(); //把输入的数字后面的回车键吃掉

switch(choice) { //通过用户输入的数字来调用不同的函数

case 1:

head = addPeople(head); //调用addPeople函数在原有的基础上添加成员

break;

case 2:

head = deleteAll(head); //删除整个链表

head = creatLink(head); //调用creatLink函数来创建新的链表

break;

}

return head;

}

**（4）删除信息功能模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：列表中删除信息功能 \*

\*输入：要写入的链表头指针 \*

\*返回：空 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void delFunction(data \*head) {

printf("\*\* 输入1：删除指定成员\n");

printf("\*\* 输入2：删除所有成员\n");

int choice;

char Fname[20];

data \*del;

scanf("%d", &choice);

getchar(); //吃掉回车

if (choice == 2)

head = deleteAll(head); //删除整个链表

else {

printf("输入你想删除的名字：\n");

gets(Fname);

del = search(head, Fname); //先找到要删除成员的位置，后删除

while (del == NULL) { //如果del指针指向了一个空的位置，则代表链表中没有要删除的这个成员

printf("没有找到这个人！请重新输入：\n");

gets(Fname);

del = search(head, Fname); //调用查找函数

}

head = delAStruct(head, del); //调用delAStruct函数来删除链表中的一个节点

}

}

**（5）删除链表中所有数据模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：删除链表中所有数据 \*

\*输入：链表的头指针 \*

\*返回：链表的头指针 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*deleteAll(data \*head) {

data \*p, \*q;

p = head -> next;

while (p != NULL) {

q = p -> next; //指针q用来暂时存放p -> next（也就是p的下一个节点），因为free(p)会把p释放掉，释放以后p -> next就找不到了

free(p);

p = q;

}

remove("datas.myC"); //remove函数的作用是删除电脑中的某个文件，在stdio.h这个头文件中，传入的参数是文件的路径

head -> next = NULL;

return head;

}

**（6）创建一个新的链表模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：创建一个新的链表 \*

\*输入：链表的头指针 \*

\*返回：链表的头指针 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*creatLink(data \*head) {

data \*p, \*q;

q = head;

p = (data \*) malloc(sizeof(data));

printf("请输入姓名：(输入非英文结束)\n");

gets(p -> name);

while ((\*p -> name >= 'a' && \*p ->= name < 'z') //\*p -> name中，p -> name是输入的字符串的指针，在前面加了一个\*代表取值，取的是字符数组第一个成员的值

|| (\*p -> name >= 'A' && \*p ->= name < 'A')) {

if (search(head, p -> name)){ //调用search函数查询链表中姓名相同的节点，如果没有找到，search函数会返回NULL，这时候if不会成立。

printf("已有相同的姓名，请重新输入姓名：(输入非英文结束)\n");

gets(p -> name);

continue;

}

printf("请输入他/她的性别：\n"); //这里是在输入要填写的内容

gets(p -> sex);

printf("请输入他/她的年龄：\n");

scanf("%d", &p -> age);

getchar();

printf("请输入他/她的专业：\n");

gets(p -> major);

printf("请输入他/她的文化程度：\n");

gets(p -> eduStatus);

q -> next = p; //脸上链表

q = p;

p -> next = NULL;

p = (data \*) malloc(sizeof(data)); //申请一块新的内存

printf("请输入姓名：(输入非英文结束)\n");

gets(p -> name);

}

free(p); //如果用户输入了非英文，代表输入结束，则上面申请的新空间不需要录入新信息。这里就把这块不需要用的空间释放掉了

q -> next = NULL; //把链表尾设为NULL

writeFileAll(head); //调用writeFileAll函数，把链表里的所有内容写入文件

return head;

}

**（7）在现有链表中添加一个成员模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：在现有链表中添加一个成员 \*

\*输入：链表的头指针 \*

\*返回：链表的头指针 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*addPeople(data \* head) {

data \*p, \*q;

q = head -> next;

p = (data \*) malloc(sizeof(data));

printf("请输入姓名：(输入非英文结束)\n"); //输入的这一段和上面的creatLink()函数代码一样，有不懂的可以参照上面的注释

gets(p -> name);

while ((\*p -> name >= 'a' && \*p -> name <= 'z')

|| (\*p -> name >= 'A' && \*p -> name <= 'Z')) {

if (search(head, p -> name)){

printf("已有相同的姓名，请重新输入姓名：(输入非英文结束)\n");

gets(p -> name);

continue;

}

printf("请输入他/她的性别：\n");

gets(p -> sex);

printf("请输入他/她的年龄：\n");

scanf("%d", &p -> age);

getchar();

printf("请输入他/她的专业：\n");

gets(p -> major);

printf("请输入他/她的文化程度：\n");

gets(p -> eduStatus);

head -> next = p;

p -> next = q; //从链表的头插入，而不是从它的尾插入。在这里从头插入避免了检索整个链表，会更快一些

writeAddFile(p); //这里是把新的数据写入文件的末尾，防止数据丢失

q = p;

p = (data \*) malloc(sizeof(data));

printf("请输入姓名：(输入非英文结束)\n");

gets(p -> name);

}

free(p);

return head;

}

**（8）统计链表中的人数模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：统计链表中的人数 \*

\*输入：链表的头指针 \*

\*返回：链表中的人数 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

int HowManyPeople(data \*head) {

data \*p;

int sum = 0;

for (p = head -> next; p != NULL; p = p -> next) //每有一个链表，总数+1，每个链表代表一个人的信息

sum++;

return sum;

}

**（9）输出链表中所有成员的信息模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：输出链表中所有成员的信息 \*

\*输入：链表的头指针 \*

\*返回：链表的头指针 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*displayAll(data \*head) {

data \*p;

p = head -> next;

while (p != NULL) {

printf("姓名：%s\t性别：%s\t年龄：%d\t专业：%s\t文化程度：%s\n",

p -> name, p -> sex, p -> age, p -> major, p -> eduStatus);

p = p -> next;

}

return head;

}

**（10）统计功能模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：统计功能 \*

\*输入：链表的头指针 \*

\*返回：链表的头指针 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*count(data \*head) {

int choice, Fage, a[5] = {0}; //a[5]代表4个选项，1为需要筛选，0为不需要筛选

char Fsex[20], Fmajor[50], FeduStatus[50];

/\*设置筛选项目\*/

while (1) {

printf("\n\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\* 输入1：设置年龄最大值\n");

printf("\*\* 输入2：设置性别\n");

printf("\*\* 输入3：设置专业\n");

printf("\*\* 输入4：设置文化程度\n");

printf("\*\* 输入5：开始\n");

printf("\*\* 输入6：取消\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d", &choice);

switch(choice) { //判断用户的选择

case 1:

printf("请输入最大的年龄:\n");

scanf("%d", &Fage);

a[1] = 1;

printf("设置成功！\n");

break;

case 2:

printf("请输入要求的性别:\n");

getchar(); //这里的getchar()是为了吃掉输入数字时的回车键

gets(Fsex);

a[2] = 1;

printf("设置成功！\n");

break;

case 3:

printf("请输入要求的专业:\n");

getchar();

gets(Fmajor);

a[3] = 1;

printf("设置成功！\n");

break;

case 4:

printf("请输入要求的文化程度:\n");

getchar();

gets(FeduStatus);

a[4] = 1;

printf("设置成功！\n");

break;

case 5:

break;

case 6:

return head;

}

if(choice == 5)

break;

}

/\*开始筛选\*/

printf("\n\n");

int i, key, sum = 0; //key的初始值为1，代表满足筛选的条件，如果有一个条件不符，则被修改成0.

data \*p;

p = head -> next;

while (p != NULL) { //筛选的原理是，遍历整个链表，在每个成员处依次判断a[1~5]（数组a代表是否需要筛选），如需要筛选则判断是否与输入的数据相符，不需要筛选则跳过

key = 1;

for (i = 1; i <= 4 && key == 1; i++){

if (a[i] != 1)

continue;

else {

switch (i) { //i 会从 1加到 4，这里会循环 4次，来判断是否需要筛选

case 1:

if (p -> age >= Fage)

key = 0;

break;

case 2:

if (strcmp(p -> sex, Fsex)) //strcmp中的两个指针指向的字符串如果相同，则返回 0.则不执行if中的语句

key = 0;

break;

case 3:

if (strcmp(p -> major, Fmajor))

key = 0;

break;

case 4:

if (strcmp(p -> eduStatus, FeduStatus))

key = 0;

break;

}

}

}

if (key == 1){

printf("姓名：%s\t性别：%s\t年龄：%d\t专业：%s\t文化程度：%s\n",

p -> name, p -> sex, p -> age, p -> major, p -> eduStatus);

sum++; //sum用来统计满足条件的人数

}

else

key = 1;

p = p -> next;

}

printf("符合条件的共有%d人。\n", sum);

return head;

}

**（10）读取文件中的信息模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：读取文件中的信息 \*

\*输入：链表的头指针 \*

\*返回：链表的头指针 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \* readFile(data \*head) {

FILE \* fp;

data \*p, \*last;

last = head;

if ((fp = fopen("datas.myC", "r")) == NULL) { //r 是以只读的方式打开

head -> next = NULL;

return head;

}

else {

while (!feof(fp)) { //判断fp指针是否指向了文件的末尾

p = (data \*) malloc(sizeof(data));

last->next = p;

fscanf(fp, "%s%s%d%s%s\n", p->name, p->sex, &p ->age, p->major, p->eduStatus);

last = p;

}

last->next = NULL;

}

if (fclose(fp) != 0) {

printf("Cannot close ""datas.myC!""");

exit(0);

}

return head;

}

**（11）在文件的末尾写入新数据模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：在文件的末尾写入新数据 \*

\*输入：要写入的数据链表指针 \*

\*返回：空 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void writeAddFile(data \*p) {

FILE \*fp;

if ((fp = fopen("datas.myC", "a+")) == NULL) { //a+ 是读和写的方式打开，不清空原有的文件

printf("写入失败！");

exit(0);

}

fseek(fp, 0, SEEK\_END); //将指针指向文件的末尾

fprintf(fp,"%s %s %d %s %s\n", p -> name, p -> sex, p ->age, p -> major, p -> eduStatus);

if (fclose(fp) != 0) {

printf("Cannot close the file !");

exit(0);

}

}

**（12）将链表里的所有数据写入文件模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：将链表里的所有数据写入文件 \*

\*输入：要写入的数据链表指针 \*

\*返回：空 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void writeFileAll(data \*head) {

FILE \*fp;

data \*p;

p = head -> next;

if (head -> next == NULL)

return ;

if ((fp = fopen("datas.myC", "w")) == NULL) { //w 是以只读的方式打开，并且清空原来的文件

printf("写入失败");

exit(0);

}

while (p != NULL) {

fprintf(fp, "%s %s %d %s %s\n", p->name, p->sex, p -> age, p->major, p->eduStatus);

p = p -> next;

}

if (fclose(fp) != 0) {

printf("Cannot close the file !");

exit(0);

}

}

**（13）删除链表里的一个节点模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：删除链表里的一个节点 \*

\*输入：1.链表的头；2.要删除位置的指针 \*

\*返回：链表的头 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*delAStruct(data \*head, data \*del ) {

data \*p, \*last;

last = head;

p = head -> next;

for (; p != del; p = p -> next) //找到del指针在链表中的位置，并用last指针记录del指针上一个节点的位置

last = p;

last -> next = p -> next; //删除 p节点，并释放它

free(p);

printf("删除成功！\n");

return head;

}

**（13）修改信息功能模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：列表中修改信息功能 \*

\*输入：要写入的链表头指针 \*

\*返回：空 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void revise(data \*head) {

int choice;

char hisname[20];

data \*string;

printf("\*\* 输入你想修改的成员的姓名\n");

gets(hisname);

string = search(head, hisname); //调用search函数查找节点的位置，并用指针string接受这个位置

if (string == NULL) { //如果指针string为空的话，说明没有这个节点

printf("在数据中没有找到这个成员！\n");

return ;

}

printf("\*\* 你想修改哪个地方？：\n");

printf("\*\* 输入1：修改姓名：\n");

printf("\*\* 输入2：修改性别：\n");

printf("\*\* 输入3：修改职业：\n");

printf("\*\* 输入4：修改学历：\n");

printf("\*\* 输入5：修改年龄：\n");

scanf(" %d", &choice);

getchar();

printf("请输入你想修改成的内容：\n");

switch (choice) {

case 1 :

gets(string -> name);

break;

case 2 :

gets(string -> sex);

break;

case 3 :

gets(string -> major);

break;

case 4 :

gets(string -> eduStatus);

break;

case 5:

scanf("%d", &string -> age);

break;

}

printf("修改成功！\n");

}

**（14）在链表中查找成员姓名相同的结构体模块**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能：在链表中查找成员姓名相同的结构体 \*

\*输入：1.链表的头；2.要查找的名字的指针 \*

\*返回：如果找到，返回对应结构体的指针 \*

\* 如果没找到，返回NULL； \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

data \*search(data \*head, char \*str) {

data \*p;

p = head -> next;

while (p != NULL) {

if (strcmp(p -> name, str) == 0)

break;

else

p = p -> next;

}

return p;

}

**四、调试分析**

1.因本程序没能使用直接在文件中任意位置插入数据的算法，所以在一定情况下会多次对磁盘文件进行读写，可能会造成计算机资源的浪费。

2.在程序开始的时候，会先读取一次本地的数据，并统计数据中的总人数，所以当数据量非常大的时候，程序的开启时间会变慢。（经过测试，当文件中的数据在750万条的时候，打开时间约为7秒。当文件中的数据在100万条左右时，并不需要多少等待的时间）。

**五、用户手册**

1.此程序为个人简历录入程序。

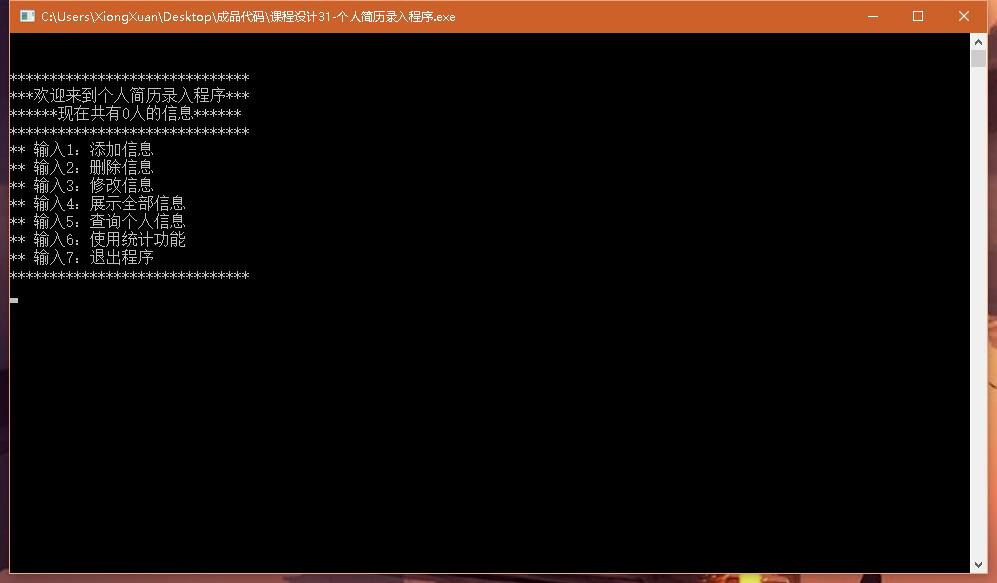
2.在启动时，程序会自动读取已保存的数据。若未保存过数据，则需要用户自行添加。

3.程序可在原有的成员中添加新的成员。

4.统计功能中，用户可以设置任意个筛选条件，程序会输出符合条件的所有人的数据，并统计总人数。

5.退出程序。

**六、测试数据**



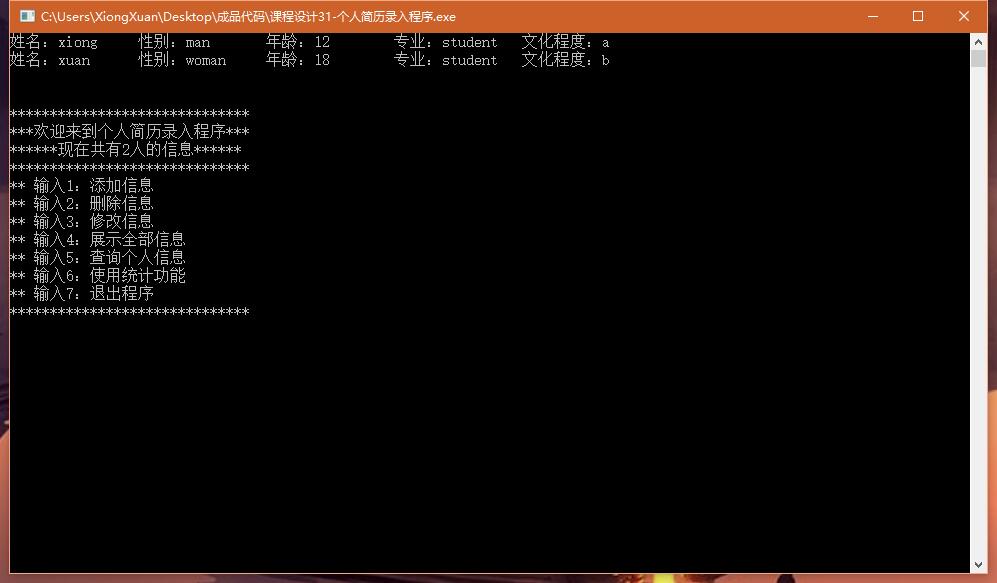
↑↑↑↑↑↑这是第一次启动的画面，此时显示共有0人的信息



↑↑↑↑↑↑开始添加信息，输入2，删除原有的成员并创建新的信息



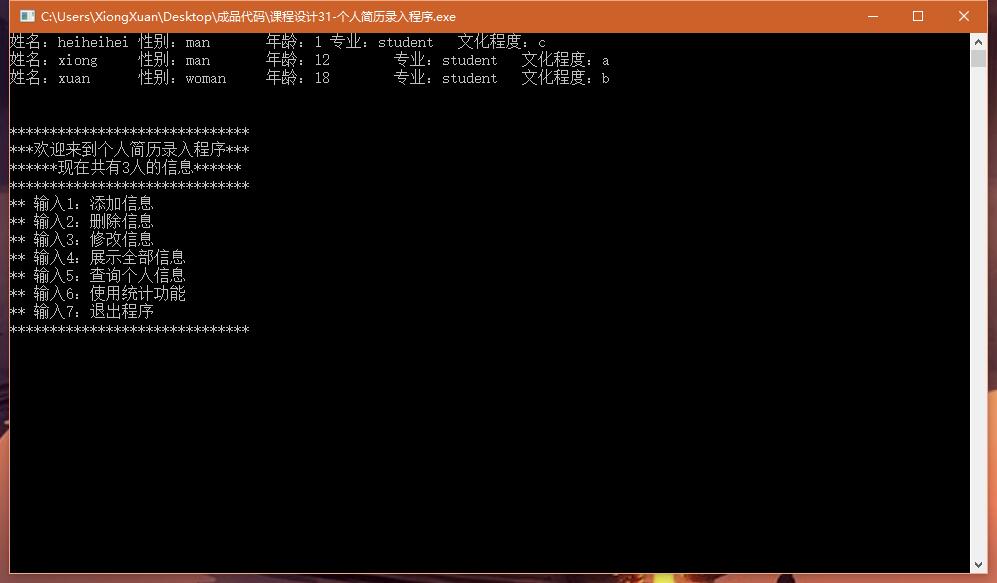
↑↑↑↑↑↑↑这里先输入两个人的信息，并按一个符号’@’结束输入



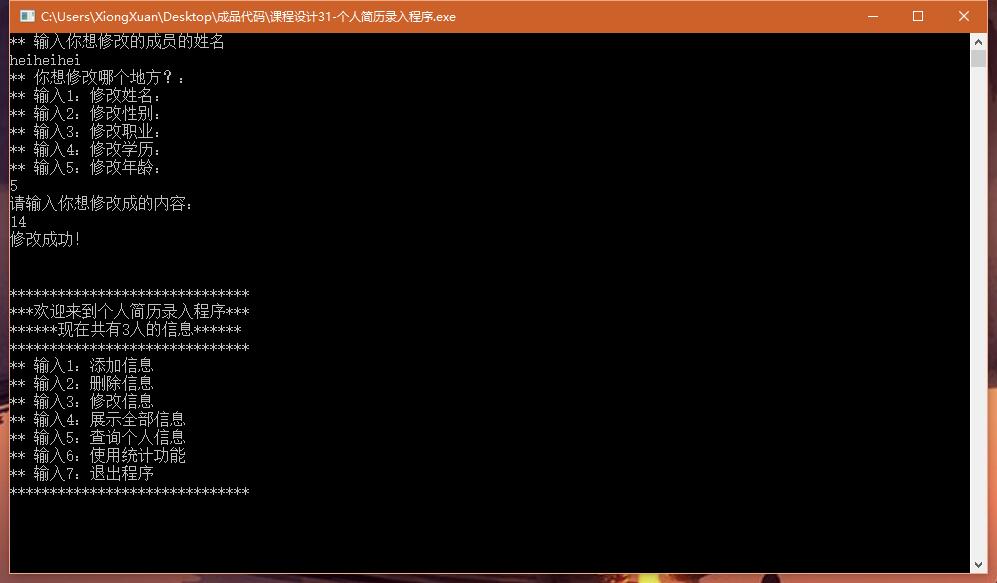
↑↑↑↑↑输入结束后，按4展示刚才输入的数据



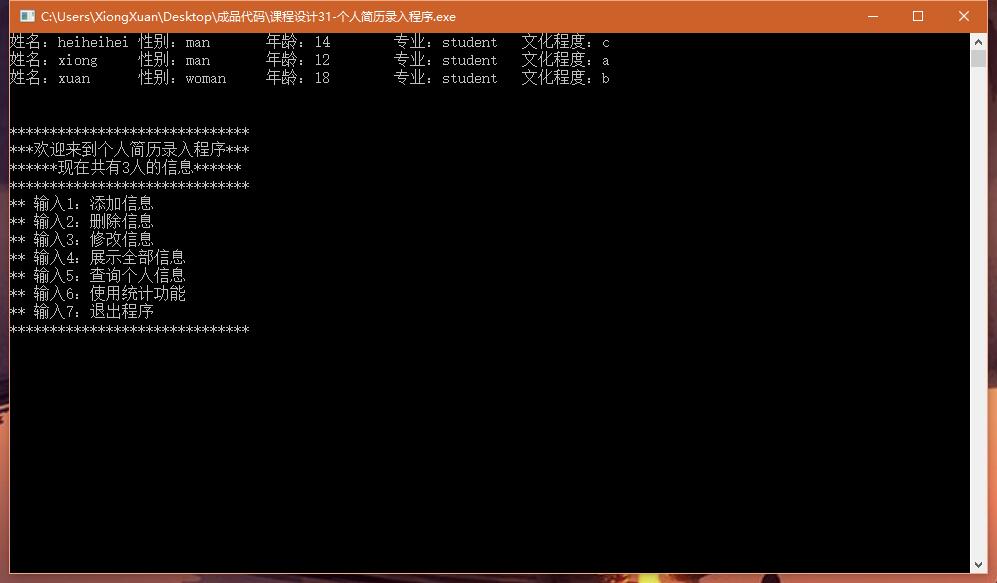
↑↑↑↑↑↑再使用1来再原有的成员里添加新的成员



↑↑↑↑↑↑按4展示所有数据，刚才的新数据插入成功！



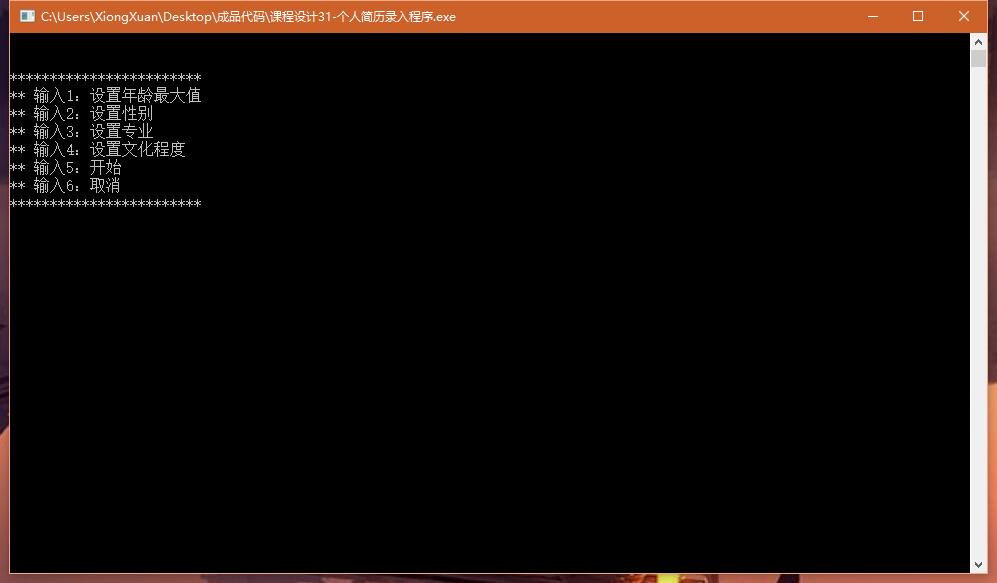
↑↑↑↑↑↑现在开始修改数据，修改的是姓名为heiheihei的成员的年龄，改为14



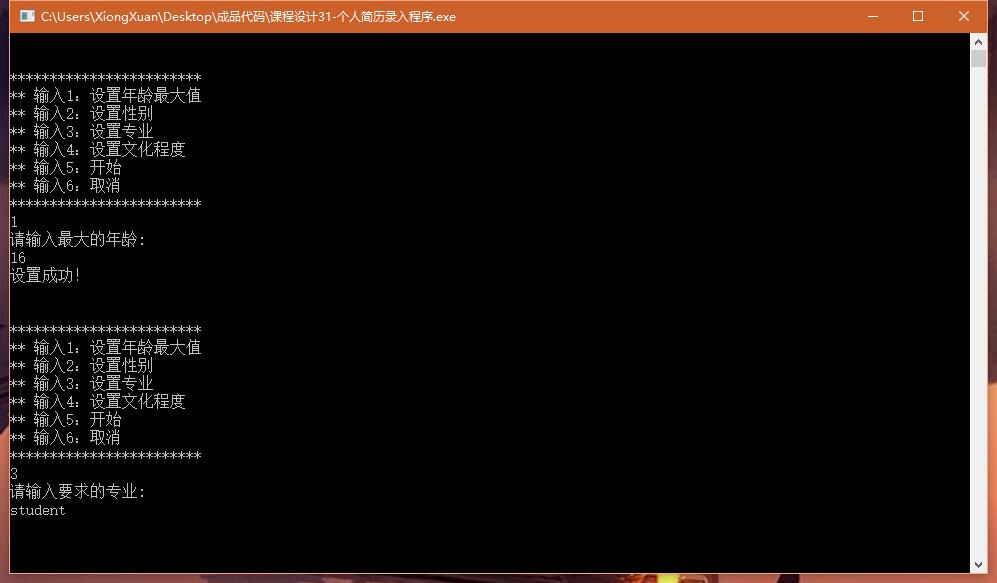
↑↑↑↑↑↑展示修改后的成员信息，此处修改成功



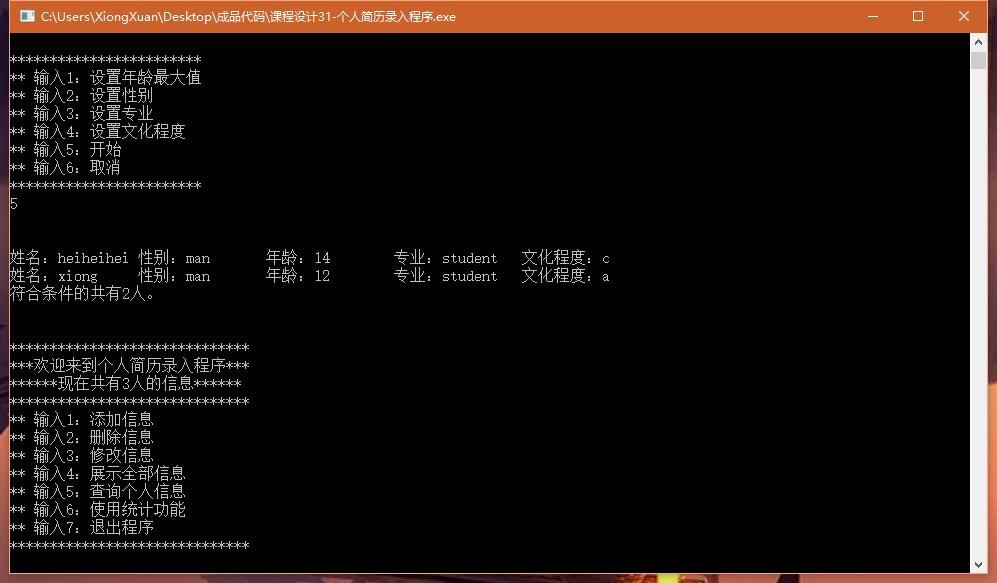
↑↑↑↑↑↑这是查询功能，输入成员的名字即可查询到他的信息



↑↑↑↑↑↑这是统计功能的选项面板



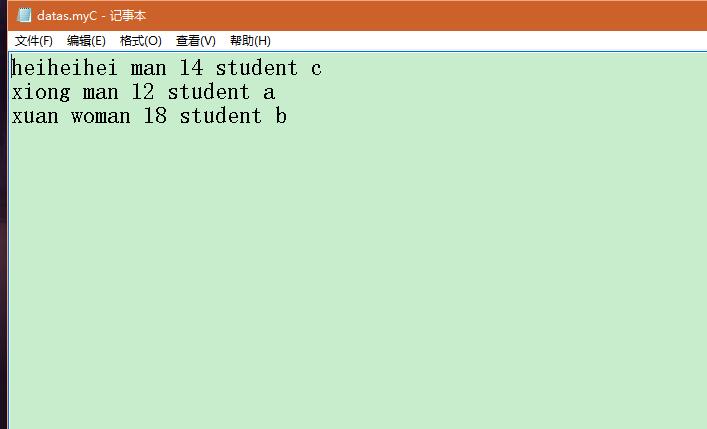
↑↑↑↑↑↑设置了两个要求，在这里可以设置任意个条件



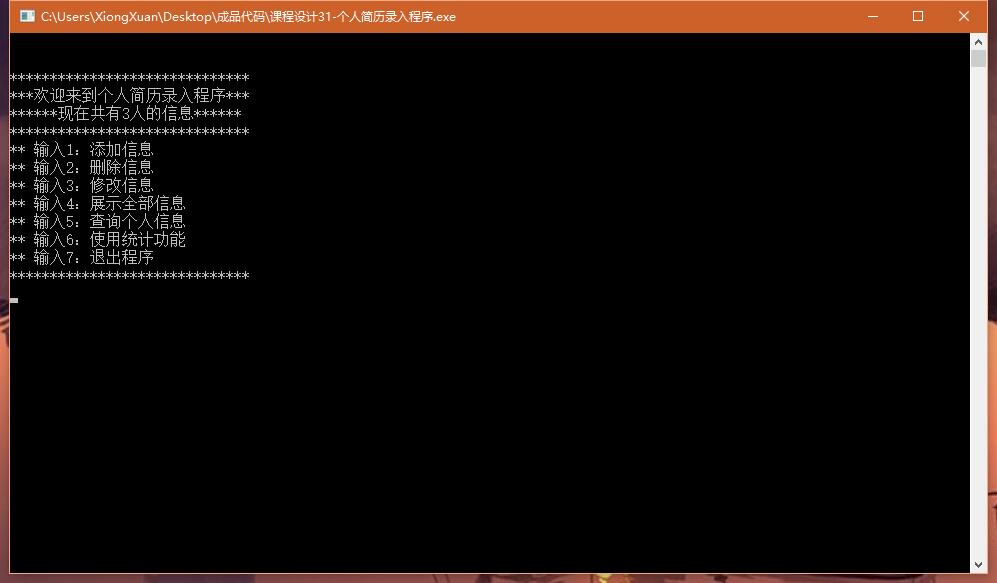
↑↑↑↑↑↑开始查询，并输出查询后的结果，符合要求的一共两人



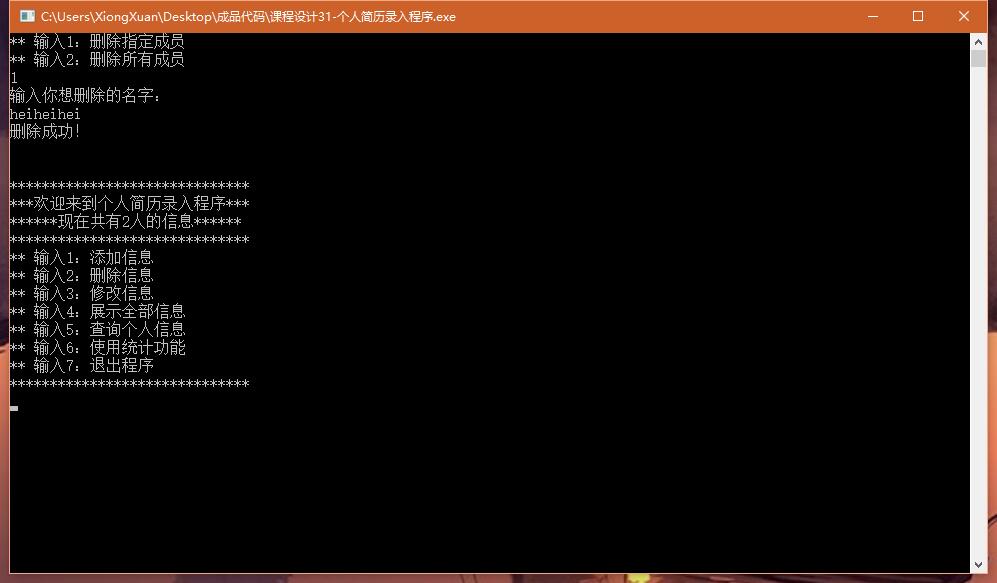
↑↑↑↑↑↑在主面板选择退出程序后，程序会自动将数据写入文件



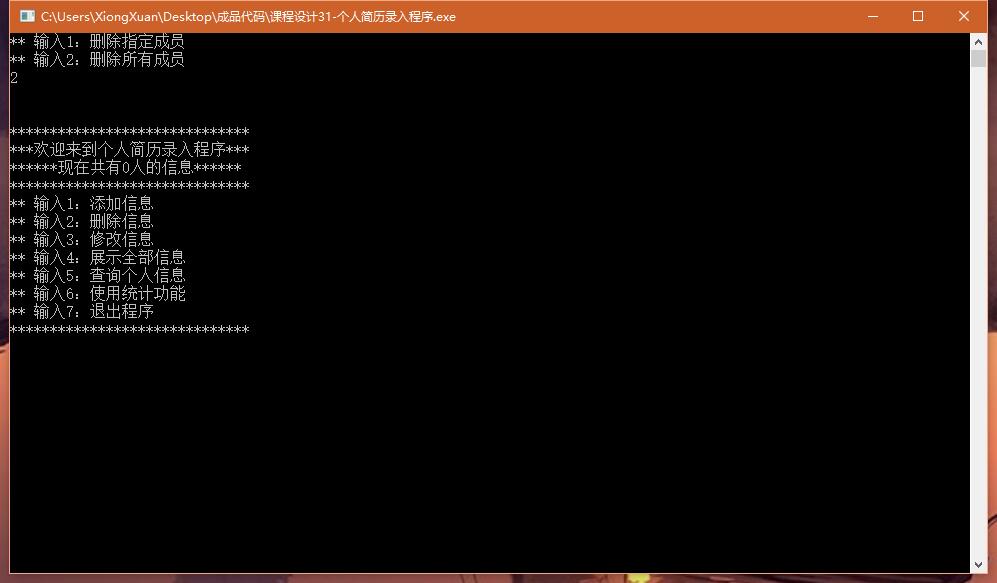
↑↑↑↑↑↑这是程序自己写的文件



↑↑↑↑↑↑重新打开程序，程序会自动读取上次保存的数据



↑↑↑↑↑↑现在是删除功能，可以删除指定的成员，也可以删除所有的成员



↑↑↑↑↑↑删除所有的成员后，显示共有0人的信息

**至此，所有的演示结束！**

**（七）附录**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<string.h>**

**#include<unistd.h>**

**struct DATA {**

**char name[20], sex[10], major[20], eduStatus[20];**

**int age;**

**struct DATA \*next;**

**};**

**typedef struct DATA data;**

**data \* readFile(data \*); //读取文件中的信息**

**void displayItem(int); //展示选项界面**

**data \*count(data \*); //统计功能**

**data \*displayAll(data \*); //展示链表成员**

**int HowManyPeople(data \*); //判断链表中有多少人**

**data \*creatNew(data \*); //添加信息功能**

**data \*creatLink(data \*); //创建链表**

**data \*addPeople(data \*); //在原链表里添加成员**

**data \*deleteAll(data \*); //删除所有信息**

**void writeAddFile(data \*); //重新写入新的数据**

**data \*delAStruct(data \*, data \*);//删除链表里的一个节点**

**void revise(data \*); //修改链表里的一个节点**

**data \*search(data \*, char \*); //在链表中查找名字相同的结构体**

**void delFunction(data \*); //删除信息功能的函数**

**void writeFileAll(data \*); //将链表里的所有数据写入文件**

**int main() {**

**int choice;**

**data \*head, \*p;**

**char hisName[20];**

**head = (data \*)malloc(sizeof(data));**

**head -> next = NULL;**

**head = readFile(head); //这里是先读取保存的文件**

**displayItem(HowManyPeople(head)); //展示选项界面**

**while (scanf("%d", &choice)) { //当scanf成功接收到一个数字的时候会返回一个正值，满足while里面的条件。若接收到非数字则返回0，就不满足条件，直接退出**

**system("cls"); //清屏函数，保持界面整洁**

**switch (choice) { //判断用户的输入的数字**

**case 1:**

**getchar();**

**head = creatNew(head); //调用creatNew函数，创建新链表 或 添加一个节点**

**break;**

**case 2:**

**delFunction(head); //调用delFunction函数，删除整个链表 或 删除链表中指定节点**

**break;**

**case 3:**

**getchar();**

**revise(head); //调用revise函数，修改某个成员**

**break;**

**case 4:**

**head = displayAll(head);//调用displayAll函数，展示链表中所有数据**

**break;**

**case 5:**

**printf("输入你想查询的成员名字：\n");**

**getchar(); //在输入1~7选项的时候，还按了一个回车键，这个回车键会让下面的gets获取字符串出错，所以要用getchar()函数把回车键吃掉**

**gets(hisName);**

**p = search(head, hisName);//p返回的是查询到的那个人所在的节点的指针，如果返回NULL说明没找到**

**if (p == NULL)**

**printf("没有这个人！");**

**else**

**printf("姓名：%s\t性别：%s\t年龄：%d\t专业：%s\t文化程度：%s\n",**

**p -> name, p -> sex, p -> age, p -> major, p -> eduStatus);**

**break;**

**case 6:**

**head = count(head); //调用count函数，统计符合要求的成员**

**break;**

**case 7:**

**printf("正在更新数据，请稍等。");**

**writeFileAll(head); //调用writeFileAll函数，重新将链表中的数据写入文件**

**system("cls"); //清屏**

**printf("谢谢使用，再见！");**

**sleep(3); //延时函数，在这里是延时三秒**

**return 0;**

**}**

**displayItem(HowManyPeople(head));//展示功能面板**

**}**

**return 0;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：展示现有项目的列表和现有人数 \***

**\*输入：现有人数 \***

**\*返回：空 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**void displayItem(int sum) {**

**printf("\n\n");**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf("\*\*\*欢迎来到个人简历录入程序\*\*\*\n");**

**printf("\*\*\*\*\*\*现在共有%d人的信息\*\*\*\*\*\*\n", sum);**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf("\*\* 输入1：添加信息\n");**

**printf("\*\* 输入2：删除信息\n");**

**printf("\*\* 输入3：修改信息\n");**

**printf("\*\* 输入4：展示全部信息\n");**

**printf("\*\* 输入5：查询个人信息\n");**

**printf("\*\* 输入6：使用统计功能\n");**

**printf("\*\* 输入7：退出程序\n");**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：列表中添加信息功能 \***

**\*输入：要写入的链表头指针 \***

**\*返回：链表的头指针 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*creatNew(data \*head){**

**int choice;**

**printf("\*\* 输入1：在原有成员里添加新成员\n");**

**printf("\*\* 输入2：删除原有成员并创建新信息\n");**

**scanf("%d", &choice);**

**getchar(); //把输入的数字后面的回车键吃掉**

**switch(choice) { //通过用户输入的数字来调用不同的函数**

**case 1:**

**head = addPeople(head); //调用addPeople函数在原有的基础上添加成员**

**break;**

**case 2:**

**head = deleteAll(head); //删除整个链表**

**head = creatLink(head); //调用creatLink函数来创建新的链表**

**break;**

**}**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：列表中删除信息功能 \***

**\*输入：要写入的链表头指针 \***

**\*返回：空 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**void delFunction(data \*head) {**

**printf("\*\* 输入1：删除指定成员\n");**

**printf("\*\* 输入2：删除所有成员\n");**

**int choice;**

**char Fname[20];**

**data \*del;**

**scanf("%d", &choice);**

**getchar(); //吃掉回车**

**if (choice == 2)**

**head = deleteAll(head); //删除整个链表**

**else {**

**printf("输入你想删除的名字：\n");**

**gets(Fname);**

**del = search(head, Fname); //先找到要删除成员的位置，后删除**

**while (del == NULL) { //如果del指针指向了一个空的位置，则代表链表中没有要删除的这个成员**

**printf("没有找到这个人！请重新输入：\n");**

**gets(Fname);**

**del = search(head, Fname); //调用查找函数**

**}**

**head = delAStruct(head, del); //调用delAStruct函数来删除链表中的一个节点**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：删除链表中所有数据 \***

**\*输入：链表的头指针 \***

**\*返回：链表的头指针 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*deleteAll(data \*head) {**

**data \*p, \*q;**

**p = head -> next;**

**while (p != NULL) {**

**q = p -> next; //指针q用来暂时存放p -> next（也就是p的下一个节点），因为free(p)会把p释放掉，释放以后p -> next就找不到了**

**free(p);**

**p = q;**

**}**

**remove("datas.myC"); //remove函数的作用是删除电脑中的某个文件，在stdio.h这个头文件中，传入的参数是文件的路径**

**head -> next = NULL;**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：创建一个新的链表 \***

**\*输入：链表的头指针 \***

**\*返回：链表的头指针 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*creatLink(data \*head) {**

**data \*p, \*q;**

**q = head;**

**p = (data \*) malloc(sizeof(data));**

**printf("请输入姓名：(输入非英文结束)\n");**

**gets(p -> name);**

**while ((\*p -> name >= 'a' && \*p -> name <= 'z') //\*p -> name中，p -> name是输入的字符串的指针，在前面加了一个\*代表取值，取的是字符数组第一个成员的值**

**|| (\*p -> name >= 'A' && \*p -> name <= 'A')) {**

**if (search(head, p -> name)){ //调用search函数查询链表中姓名相同的节点，如果没有找到，search函数会返回NULL，这时候if不会成立。**

**printf("已有相同的姓名，请重新输入姓名：(输入非英文结束)\n");**

**gets(p -> name);**

**continue;**

**}**

**printf("请输入他/她的性别：\n"); //这里是在输入要填写的内容**

**gets(p -> sex);**

**printf("请输入他/她的年龄：\n");**

**scanf("%d", &p -> age);**

**getchar();**

**printf("请输入他/她的专业：\n");**

**gets(p -> major);**

**printf("请输入他/她的文化程度：\n");**

**gets(p -> eduStatus);**

**q -> next = p; //脸上链表**

**q = p;**

**p -> next = NULL;**

**p = (data \*) malloc(sizeof(data)); //申请一块新的内存**

**printf("请输入姓名：(输入非英文结束)\n");**

**gets(p -> name);**

**}**

**free(p); //如果用户输入了非英文，代表输入结束，则上面申请的新空间不需要录入新信息。这里就把这块不需要用的空间释放掉了**

**q -> next = NULL; //把链表尾设为NULL**

**writeFileAll(head); //调用writeFileAll函数，把链表里的所有内容写入文件**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：在现有链表中添加一个成员 \***

**\*输入：链表的头指针 \***

**\*返回：链表的头指针 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*addPeople(data \* head) {**

**data \*p, \*q;**

**q = head -> next;**

**p = (data \*) malloc(sizeof(data));**

**printf("请输入姓名：(输入非英文结束)\n"); //输入的这一段和上面的creatLink()函数代码一样，有不懂的可以参照上面的注释**

**gets(p -> name);**

**while ((\*p -> name >= 'a' && \*p -> name <= 'z')**

**|| (\*p -> name >= 'A' && \*p -> name <= 'Z')) {**

**if (search(head, p -> name)){**

**printf("已有相同的姓名，请重新输入姓名：(输入非英文结束)\n");**

**gets(p -> name);**

**continue;**

**}**

**printf("请输入他/她的性别：\n");**

**gets(p -> sex);**

**printf("请输入他/她的年龄：\n");**

**scanf("%d", &p -> age);**

**getchar();**

**printf("请输入他/她的专业：\n");**

**gets(p -> major);**

**printf("请输入他/她的文化程度：\n");**

**gets(p -> eduStatus);**

**head -> next = p;**

**p -> next = q; //从链表的头插入，而不是从它的尾插入。在这里从头插入避免了检索整个链表，会更快一些**

**writeAddFile(p); //这里是把新的数据写入文件的末尾，防止数据丢失**

**q = p;**

**p = (data \*) malloc(sizeof(data));**

**printf("请输入姓名：(输入非英文结束)\n");**

**gets(p -> name);**

**}**

**free(p);**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：统计链表中的人数 \***

**\*输入：链表的头指针 \***

**\*返回：链表中的人数 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**int HowManyPeople(data \*head) {**

**data \*p;**

**int sum = 0;**

**for (p = head -> next; p != NULL; p = p -> next) //每有一个链表，总数+1，每个链表代表一个人的信息**

**sum++;**

**return sum;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：输出链表中所有成员的信息 \***

**\*输入：链表的头指针 \***

**\*返回：链表的头指针 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*displayAll(data \*head) {**

**data \*p;**

**p = head -> next;**

**while (p != NULL) {**

**printf("姓名：%s\t性别：%s\t年龄：%d\t专业：%s\t文化程度：%s\n",**

**p -> name, p -> sex, p -> age, p -> major, p -> eduStatus);**

**p = p -> next;**

**}**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：统计功能 \***

**\*输入：链表的头指针 \***

**\*返回：链表的头指针 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*count(data \*head) {**

**int choice, Fage, a[5] = {0}; //a[5]代表4个选项，1为需要筛选，0为不需要筛选**

**char Fsex[20], Fmajor[50], FeduStatus[50];**

**/\*设置筛选项目\*/**

**while (1) {**

**printf("\n\n");**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**printf("\*\* 输入1：设置年龄最大值\n");**

**printf("\*\* 输入2：设置性别\n");**

**printf("\*\* 输入3：设置专业\n");**

**printf("\*\* 输入4：设置文化程度\n");**

**printf("\*\* 输入5：开始\n");**

**printf("\*\* 输入6：取消\n");**

**printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");**

**scanf("%d", &choice);**

**switch(choice) { //判断用户的选择**

**case 1:**

**printf("请输入最大的年龄:\n");**

**scanf("%d", &Fage);**

**a[1] = 1;**

**printf("设置成功！\n");**

**break;**

**case 2:**

**printf("请输入要求的性别:\n");**

**getchar(); //这里的getchar()是为了吃掉输入数字时的回车键**

**gets(Fsex);**

**a[2] = 1;**

**printf("设置成功！\n");**

**break;**

**case 3:**

**printf("请输入要求的专业:\n");**

**getchar();**

**gets(Fmajor);**

**a[3] = 1;**

**printf("设置成功！\n");**

**break;**

**case 4:**

**printf("请输入要求的文化程度:\n");**

**getchar();**

**gets(FeduStatus);**

**a[4] = 1;**

**printf("设置成功！\n");**

**break;**

**case 5:**

**break;**

**case 6:**

**return head;**

**}**

**if(choice == 5)**

**break;**

**}**

**/\*开始筛选\*/**

**printf("\n\n");**

**int i, key, sum = 0; //key的初始值为1，代表满足筛选的条件，如果有一个条件不符，则被修改成0.**

**data \*p;**

**p = head -> next;**

**while (p != NULL) { //筛选的原理是，遍历整个链表，在每个成员处依次判断a[1~5]（数组a代表是否需要筛选），如需要筛选则判断是否与输入的数据相符，不需要筛选则跳过**

**key = 1;**

**for (i = 1; i <= 4 && key == 1; i++){**

**if (a[i] != 1)**

**continue;**

**else {**

**switch (i) { //i 会从 1加到 4，这里会循环 4次，来判断是否需要筛选**

**case 1:**

**if (p -> age >= Fage)**

**key = 0;**

**break;**

**case 2:**

**if (strcmp(p -> sex, Fsex)) //strcmp中的两个指针指向的字符串如果相同，则返回 0.则不执行if中的语句**

**key = 0;**

**break;**

**case 3:**

**if (strcmp(p -> major, Fmajor))**

**key = 0;**

**break;**

**case 4:**

**if (strcmp(p -> eduStatus, FeduStatus))**

**key = 0;**

**break;**

**}**

**}**

**}**

**if (key == 1){**

**printf("姓名：%s\t性别：%s\t年龄：%d\t专业：%s\t文化程度：%s\n",**

**p -> name, p -> sex, p -> age, p -> major, p -> eduStatus);**

**sum++; //sum用来统计满足条件的人数**

**}**

**else**

**key = 1;**

**p = p -> next;**

**}**

**printf("符合条件的共有%d人。\n", sum);**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：读取文件中的信息 \***

**\*输入：链表的头指针 \***

**\*返回：链表的头指针 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \* readFile(data \*head) {**

**FILE \* fp;**

**data \*p, \*last;**

**last = head;**

**if ((fp = fopen("datas.myC", "r")) == NULL) { //r 是以只读的方式打开**

**head -> next = NULL;**

**return head;**

**}**

**else {**

**while (!feof(fp)) { //判断fp指针是否指向了文件的末尾**

**p = (data \*) malloc(sizeof(data));**

**last->next = p;**

**fscanf(fp, "%s%s%d%s%s\n", p->name, p->sex, &p ->age, p->major, p->eduStatus);**

**last = p;**

**}**

**last->next = NULL;**

**}**

**if (fclose(fp) != 0) {**

**printf("Cannot close ""datas.myC!""");**

**exit(0);**

**}**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：在文件的末尾写入新数据 \***

**\*输入：要写入的数据链表指针 \***

**\*返回：空 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**void writeAddFile(data \*p) {**

**FILE \*fp;**

**if ((fp = fopen("datas.myC", "a+")) == NULL) { //a+ 是读和写的方式打开，不清空原有的文件**

**printf("写入失败！");**

**exit(0);**

**}**

**fseek(fp, 0, SEEK\_END); //将指针指向文件的末尾**

**fprintf(fp,"%s %s %d %s %s\n", p -> name, p -> sex, p ->age, p -> major, p -> eduStatus);**

**if (fclose(fp) != 0) {**

**printf("Cannot close the file !");**

**exit(0);**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：将链表里的所有数据写入文件 \***

**\*输入：要写入的数据链表指针 \***

**\*返回：空 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**void writeFileAll(data \*head) {**

**FILE \*fp;**

**data \*p;**

**p = head -> next;**

**if (head -> next == NULL)**

**return ;**

**if ((fp = fopen("datas.myC", "w")) == NULL) { //w 是以只读的方式打开，并且清空原来的文件**

**printf("写入失败");**

**exit(0);**

**}**

**while (p != NULL) {**

**fprintf(fp, "%s %s %d %s %s\n", p->name, p->sex, p -> age, p->major, p->eduStatus);**

**p = p -> next;**

**}**

**if (fclose(fp) != 0) {**

**printf("Cannot close the file !");**

**exit(0);**

**}**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：删除链表里的一个节点 \***

**\*输入：1.链表的头；2.要删除位置的指针 \***

**\*返回：链表的头 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*delAStruct(data \*head, data \*del ) {**

**data \*p, \*last;**

**last = head;**

**p = head -> next;**

**for (; p != del; p = p -> next) //找到del指针在链表中的位置，并用last指针记录del指针上一个节点的位置**

**last = p;**

**last -> next = p -> next; //删除 p节点，并释放它**

**free(p);**

**printf("删除成功！\n");**

**return head;**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：列表中修改信息功能 \***

**\*输入：要写入的链表头指针 \***

**\*返回：空 \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**void revise(data \*head) {**

**int choice;**

**char hisname[20];**

**data \*string;**

**printf("\*\* 输入你想修改的成员的姓名\n");**

**gets(hisname);**

**string = search(head, hisname); //调用search函数查找节点的位置，并用指针string接受这个位置**

**if (string == NULL) { //如果指针string为空的话，说明没有这个节点**

**printf("在数据中没有找到这个成员！\n");**

**return ;**

**}**

**printf("\*\* 你想修改哪个地方？：\n");**

**printf("\*\* 输入1：修改姓名：\n");**

**printf("\*\* 输入2：修改性别：\n");**

**printf("\*\* 输入3：修改职业：\n");**

**printf("\*\* 输入4：修改学历：\n");**

**printf("\*\* 输入5：修改年龄：\n");**

**scanf(" %d", &choice);**

**getchar();**

**printf("请输入你想修改成的内容：\n");**

**switch (choice) {**

**case 1 :**

**gets(string -> name);**

**break;**

**case 2 :**

**gets(string -> sex);**

**break;**

**case 3 :**

**gets(string -> major);**

**break;**

**case 4 :**

**gets(string -> eduStatus);**

**break;**

**case 5:**

**scanf("%d", &string -> age);**

**break;**

**}**

**printf("修改成功！\n");**

**}**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*功能：在链表中查找成员姓名相同的结构体 \***

**\*输入：1.链表的头；2.要查找的名字的指针 \***

**\*返回：如果找到，返回对应结构体的指针 \***

**\* 如果没找到，返回NULL； \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**data \*search(data \*head, char \*str) {**

**data \*p;**

**p = head -> next;**

**while (p != NULL) {**

**if (strcmp(p -> name, str) == 0)**

**break;**

**else**

**p = p -> next;**

**}**

**return p;**

**}**