十二章实验报告

杨志航 12021242340

项目要求：

熟悉链表结构及其操作。

链表的操作：

1）创建链表（CreateLink）：从键盘输入n个学生的数据，每个学生信息都是链表中的一个结点，要求按照学号升序建立一个链表。此外，不可以输入学号字段相同的数据（每个学生的学号都是唯一的），若有重复则拒绝输入。

2）遍历链表（TraverseLink）：输出链表中每个结点的学生信息，每行显示一个学生的信息，要求要有表头（学号 姓名 成绩）。（可以自己设定显示的格式，尽量整齐、清晰）

3）查询学生信息（FindNode）：给定一个学生的学号，输出学生的全部信息。

4）插入学生信息（InsertNode）:从键盘输入一个学生的信息，将这个结点插入到链表中的正确位置。（同前，学号不能重复，否则拒绝插入）。

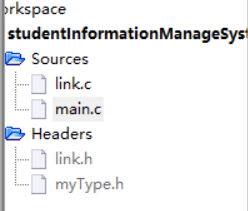
5）删除学生信息结点（DeleteNode）:从键盘输入学号，然后将该生的信息结点从链表中删除。

6）修改学生信息（UpdateNode）：从键盘输入学号，然后询问修改姓名还是成绩，确认后将链表中该生的结点信息进行修改。

7）生成一个链表的逆链表（ReversLink）:对符合要求1）的一个链表生成其逆序链表。

8\*）合并两个链表（BindLinks）：把按照要求1）生成的两个链表合并成一个符合要求1）的链表。如果有学号相同的两个结点，自己设定解决方案，在项目报告中加以说明。

Codeblocks中项目组成



代码及解读：

myType.h

#ifndef MYTYPE\_H\_INCLUDED

#define MYTYPE\_H\_INCLUDED

typedef struct stu

{ int num;

char name[20];

float score;

} Student;

typedef struct studentNode

{ Student data;

struct studentNode \*next;

} StuNode;

#endif // MYTYPE\_H\_INCLUDED

Link.h

#ifndef LINK\_H\_INCLUDED

#define LINK\_H\_INCLUDED

int CreateLink(StuNode \*\*head);

void TraverseLink(StuNode \*head,int n);

void FindNode(StuNode \*head,int n);

void InsertNode(StuNode \*head);

void DeleteNode(StuNode \*head);

void UpdateNode(StuNode \*head,int n);

void ReversLink(StuNode \*head,int n);

void BindLinks(StuNode \*head);

#endif // LINK\_H\_INCLUDED

Main.c

#include <stdio.h>

#include <stdlib.h>

#include "myType.h"

#include "link.h"

extern StuNode \*head;

extern int n;

int displayMenu()

{

int choose;

printf("\*\*欢迎使用学生成绩信息管理系统\*\*\n");

printf("\*\*输入数字选择功能 \*\*\n");

printf("\*\*1：创建链表 \*\*\n");

printf("\*\*2：遍历链表 \*\*\n");

printf("\*\*3：查询学生信息 \*\*\n");

printf("\*\*4：插入学生信息 \*\*\n");

printf("\*\*5：删除学生信息节点 \*\*\n");

printf("\*\*6：修改学生信息 \*\*\n");

printf("\*\*7：生成一个链表的逆链表 \*\*\n");

printf("\*\*8：合并两个链表 \*\*\n");

printf("\*\*0：退出程序 \*\*\n");

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

scanf("%d",&choose);

return choose;

}

int main()

{

int choose;

while(1)

{

choose=displayMenu();

switch(choose)

{

case 1: n=CreateLink(&head); break;

case 2: TraverseLink(head,n); break;

case 3: FindNode(head,n); break;

case 4: InsertNode(head); break;

case 5: DeleteNode(head); break;

case 6: UpdateNode(head,n); break;

case 7: ReversLink(head,n); break;

case 8: BindLinks(head); break;

case 0:exit(0);

default:printf("输入错误！");

}

}

return 0;

}

前几个难度不大，只对接下来的Link.c中的代码进行解读

Link.c

CreateLink：

int FindStudent(int num,Student a[],int n)

{

int i;

char s[20];

for(i=0;i<n;i++)

{

if(num==a[i].num)

{

printf("this student is already existed,please input another student:\n");

gets(s);

return 1;

}

}

return 0;

}

void MySort(Student a[],int n)

{

Student temp;

int i,j;

for(i=0;i<n-1;i++)

for(j=i+1;j<n;j++)

{

if(a[i].num>a[j].num)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

int CreateLink(StuNode \*\*head)

{

printf("录入多少学生？");

int n;

scanf("%d",&n);

Student \*a=NULL;

a=(Student\*)calloc(n,sizeof(Student));

int i;

for(i=0;i<n;i++)

{

int nowNum;

printf("please input studentID, name, grade\n");

scanf("%d",&nowNum);

if(!FindStudent(nowNum,a,i))

{

a[i].num=nowNum;

scanf("%s",a[i].name);

scanf("%f",&a[i].score);

}else

{

i--;

}

}

MySort(a,n);

StuNode \*p,\*last;

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

last=\*head;

for(i=0;i<n;i++)

{

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

p->data.num=a[i].num;

strcpy(p->data.name,a[i].name);

p->data.score=a[i].score;

last->next=p;

last=p;

}

last->next=NULL;

printf("创建成功，录入的学生有：\n");

TraverseLink(\*head,n);

return n;

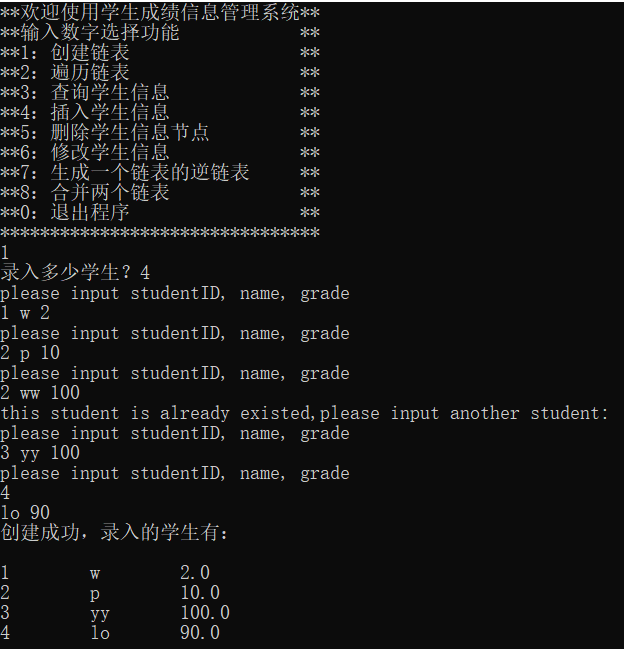
}

FindStudent函数通过输入学号返回不同的值（0或1）来检测该学生是否已经存在，进而实现对重复输入的鉴别。

为了降低排序的难度，先把学生的信息读入结构体数组中，通过MySort函数进行排序，再把排好序的学生信息由结构体数组复制进入链表之中，优点是降低了排序的难度，缺点也十分明显，加大了程序的时间和空间开销，以后直接对链表排序。

要求关键字：升序，重复则拒绝输入

代码效果：



TraverseLink：

void TraverseLink(StuNode \*head,int n)

{

int j=0;

StuNode \*p=head;

printf("\n");

while(p!=NULL&&j<n)

{

j++;

p=p->next;

printf("%-8d ",p->data.num);

printf("%-8s ",p->data.name);

printf("%-8.1f\n",p->data.score);

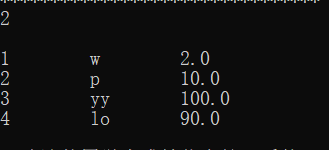
}

printf("\n");

}

难度不大，不解读。

代码效果：



FindNode：

void FindNode(StuNode \*head,int n)

{

printf("please input the student's number which you want to inquire:\n");

int num;

scanf("%d",&num);

int i=1;

StuNode \*p=head->next;

while(p->data.num!=num&&p!=NULL&&i<=n)

{

i++;

p=p->next;

}

if(i<=n)

{

printf("%-8d ",p->data.num);

printf("%-8s ",p->data.name);

printf("%-8.1f\n",p->data.score);

}else

{

printf("the student does not exit!\n");

}

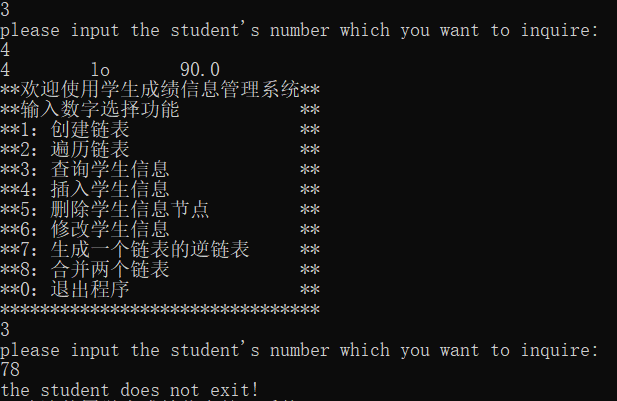
}

代码解读：代码难度不带，但是有个小细节。

while(p->data.num!=num&&p!=NULL&&i<=n)这句话在查询学生编号不存在的时候会访问到NULL的data区域，导致程序被强制停止，应该为

while(p!=NULL&&p->data.num!=num&&i<=n)判断会从左到右，这样在p!=NULL就会停止判断，而不会访问NULL的data区域。

代码效果：



InsertNode：

void InsertNode(StuNode \*head)

{

StuNode \*p;

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

printf("input student's information:\n ");

scanf("%d",&(p->data.num));

int nowNum=p->data.num;

scanf("%s",p->data.name);

scanf("%f",&(p->data.score));

int i;

StuNode \*pp=head->next;

for(i=1;i<=n;i++)

{

if(nowNum==pp->data.num)

{

printf("this student is already existed!");

return;

}

if(nowNum<pp->data.num)

break;

pp=pp->next;

}

if(i>n)

{

StuNode \*last=head;

while(last->next!=NULL)

last=last->next;

last->next=p;

p->next=NULL;

}else

{

StuNode \*last=head;

int k=0;

while(last!=NULL&&k<i-1)

{

k++;

last=last->next;

}

last->next=p;

p->next=pp;

}

printf("do successfully! students now:\n");

n++;

TraverseLink(head,n);

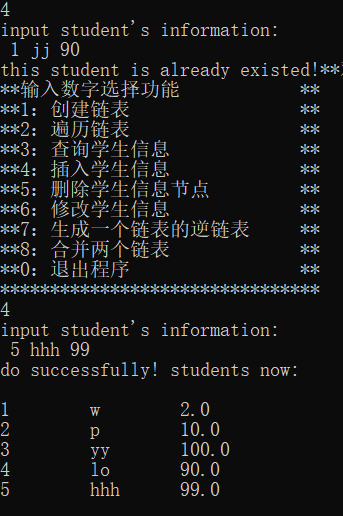
}

代码解读：遍历链表，如果nowNum==pp->data.num，说明学生存在，拒绝插入。如果nowNum<pp->data.num，就停下，并插入。还有一种情况是在尾部插入。

插入后一定要n++；

要求关键字：正确位置，不能重复

代码效果：



DeleteNode：

void DeleteNode(StuNode \*head)

{

printf("input student's number to delete student:");

int nowNum;

scanf("%d",&nowNum);

StuNode \*p=head;

int i=0;

while(p!=NULL&&i<=n)

{

p=p->next;

i++;

if(p->data.num==nowNum)

break;

}

if(i>n)

{

printf("no such student!\n");

return;

}

StuNode \*pre=head;

int k=0;

while(pre!=NULL&&k<i-1)

{

k++;

pre=pre->next;

}

pre->next=p->next;

free(p);

n--;

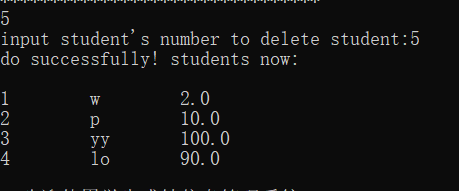
printf("do successfully! students now:\n");

TraverseLink(head,n);

}

难度不大。

代码效果：



UpdateNode：

void UpdateNode(StuNode \*head,int n)

{

printf("input student's number to update student's information:");

int nowNum;

scanf("%d",&nowNum);

StuNode \*p=head;

int i=0;

while(p!=NULL&&i<=n)

{

p=p->next;

i++;

if(p->data.num==nowNum)

break;

}

if(i>n)

{

printf("no such student!\n");

return;

}

printf("update name(0) or grade(1)?\n");

int choose,grade;

char s[10];

scanf("%d",&choose);

if(choose)

{

printf("input:");

scanf("%d",&grade);

p->data.score=grade;

}else

{

printf("input:");

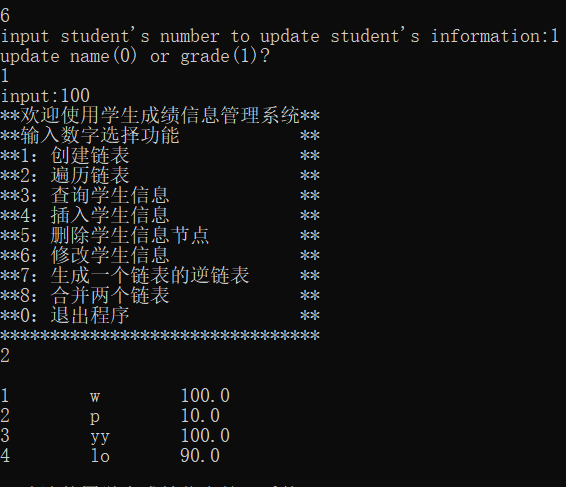
scanf("%s",s);

strcpy(p->data.name,s);

}

}

代码解读：和删除是差不多的，都是先找到这个节点。



ReversLink：

void ReversLink(StuNode \*head,int n)

{

StuNode \*p=head,\*p1;

int i,mid=n/2,j;

for(i=0;i<mid;)

{

i++;

p=p->next;

j=0;

p1=head；

while(p1!=NULL&&j<n+1-i)

{

p1=p1->next;

j++;

}

Student temp;

temp=p->data;

p->data=p1->data;

p1->data=temp;

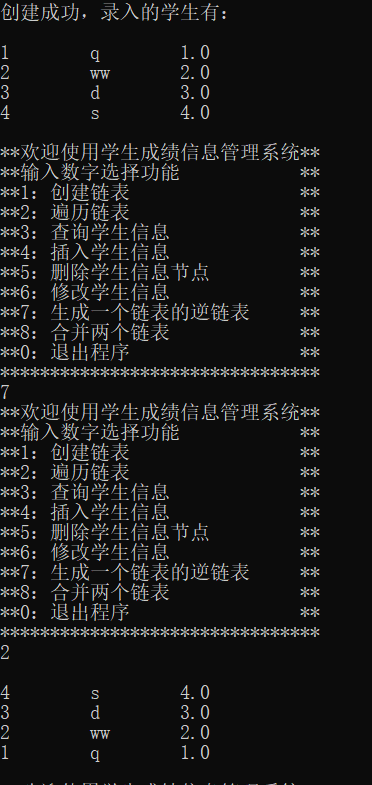
}

}

代码解读：折半遍历，每次交换(1,n),(2,n-1),(3,n-2)……

代码效果：

程序出Bug了，修了一下，所以数据跟之前不一样了。



BindLinks：

void BindLinks(StuNode \*head)

{

StuNode \*head2;

int m;

m=CreateLink(&head2);

int i=1,j=1;

StuNode \*p1=head->next,\*p2=head2->next;

StuNode \*head3,\*p,\*last;

head3=(StuNode\*)malloc(sizeof(StuNode));

last=head3;

int count=0;

while(i<=n&&j<=m)

{

if(p1->data.num<p2->data.num)

{

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

p->data=p1->data;

last->next=p;

last=p;

i++;

p1=p1->next;

}else if(p1->data.num>p2->data.num)

{

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

p->data=p2->data;

last->next=p;

last=p;

j++;

p2=p2->next;

}else

{

count++;

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

p->data=p2->data;

last->next=p;

last=p;

j++;

p2=p2->next;

i++;

p1=p1->next;

}

}

while(i<=n)

{

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

p->data=p1->data;

last->next=p;

last=p;

i++;

p1=p1->next;

}

while(j<=m)

{

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

p->data=p2->data;

last->next=p;

last=p;

j++;

p2=p2->next;

}

last->next=NULL;

TraverseLink(head3,m+n-count);

}

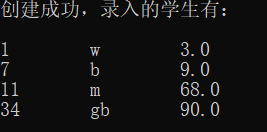
代码解读：用双指针(i,j)遍历,若\*i<\*j,\*i填入第三张链表，i++.反之同理。为了达到去重的效果，当\*i==\*j时，填入\*i或者\*j,i++,j++;

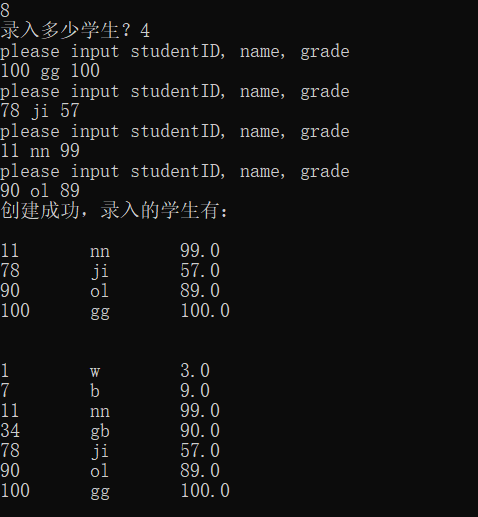
当一个指针先遍历完了，就直接把另一张链表的数据填入第三张链表即可。

类似于归并排序中的归并步骤。

要求关键字：去重，升序。

代码效果：





升序且去掉了重复的11号。

总结：项目难度中等偏上，比较考察总体规划能力，细节处理能力，通过这次作业，我的代码能力得到了很大的提升，基本实现了项目入门。