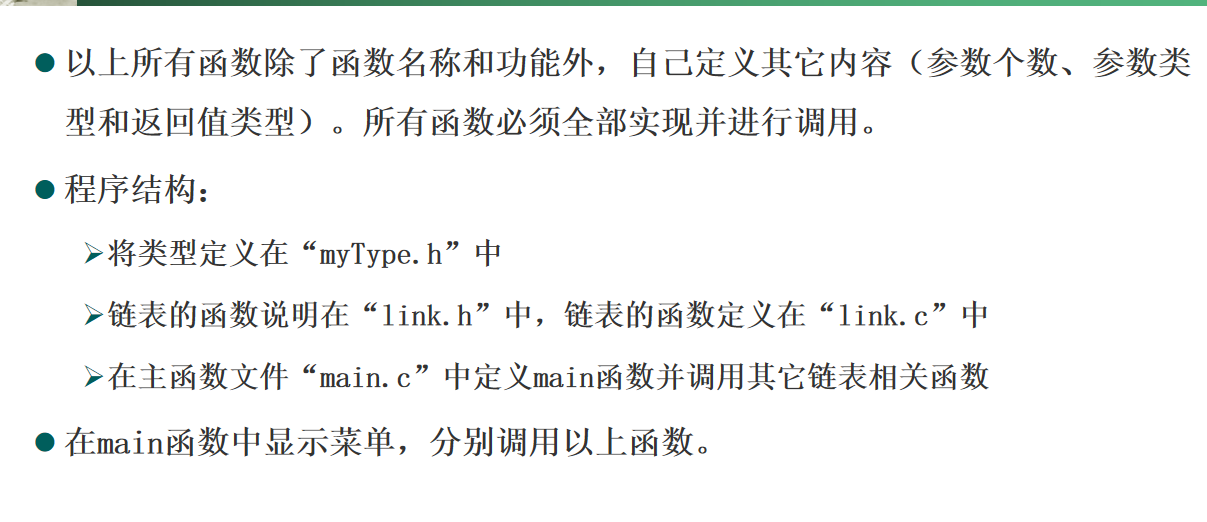
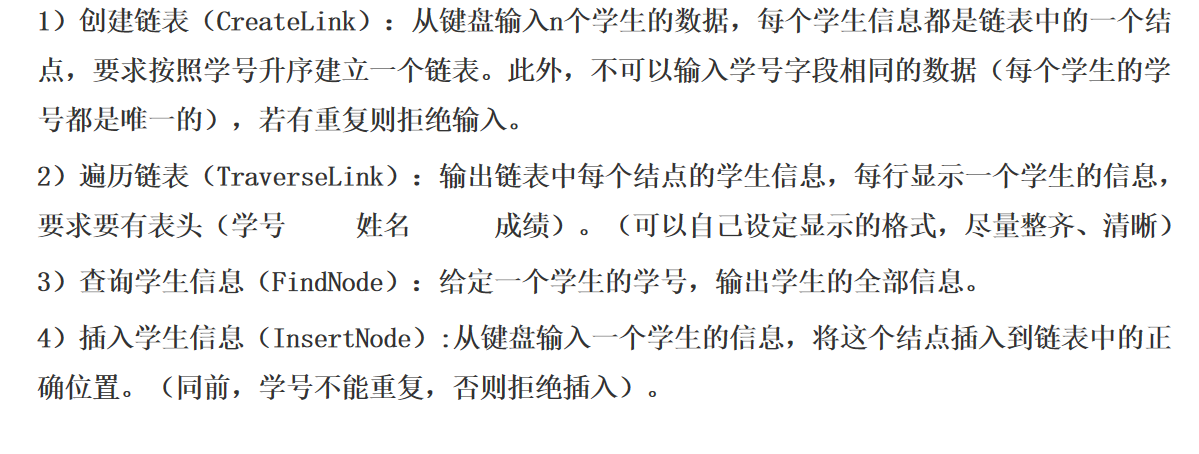
自顶而下，逐步求精。

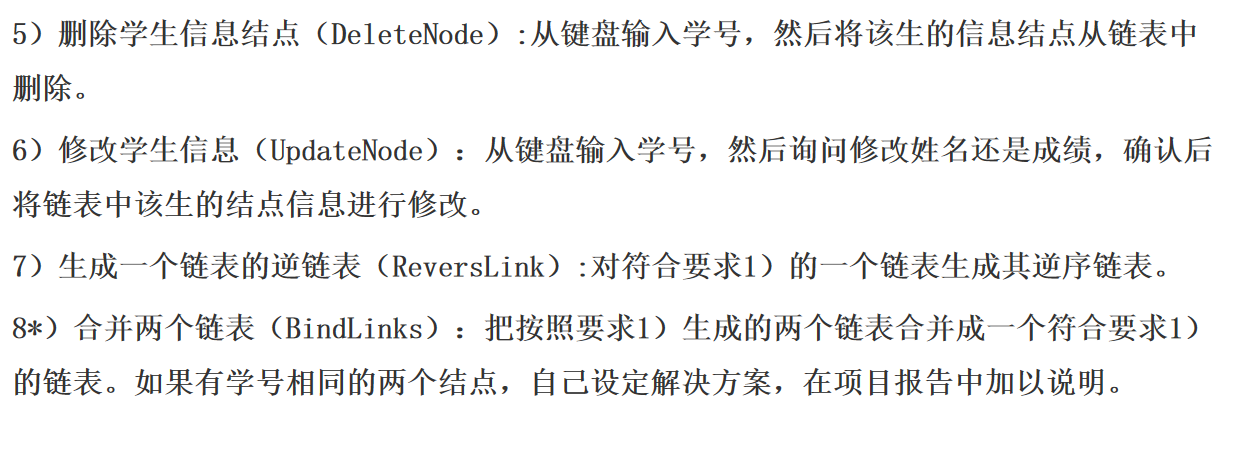
需求->变量的设置->搭建main框架->写出子函数

读需求：



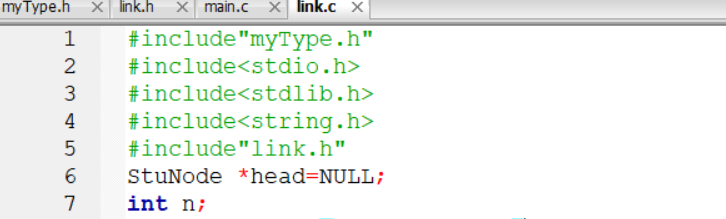
想想main设置？

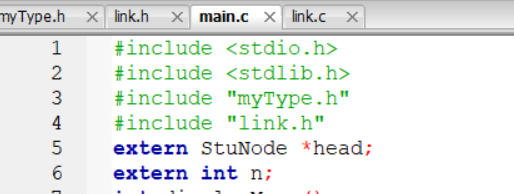




基于8个函数想想变量该如何设置。

设置变量





Mian：

讲解while部分

#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;

}

n=CreateLink(&head); 为什么返回值赋值给n?

子函数

再次分析需求设计子函数。

CreateLink：

需求？

代码：

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);//读走输入缓存区多余的输入

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;

}

}

}

//先存入结构体数组，排序再拷入链表，优点？缺点？

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);//为什么这样？

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

{

a[i].num=nowNum;

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

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

}else

{

i--;//为什么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=a[i];

// 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;

}

TraverseLink 略

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;//i代表了指针走到哪里了。

StuNode \*p=head->next;

while(p!=NULL&&p->data.num!=num&&i<=n)//小细节，为什么崩溃？

{

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");

}

}

插入：

需求？

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);//读走输入缓存区数据。

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)//尾插

{

StuNode \*last=head;

while(last->next!=NULL)

last=last->next;

last->next=p;

p->next=NULL;

}else//插中间，last名字有问题

{

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++;//很关键

TraverseLink(head,n);

}

删除：

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)//为啥《=？

{

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)//找到i-1个节点

{

k++;

pre=pre->next;

}

pre->next=p->next;

free(p);//内存泄露

n--;//关键

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

TraverseLink(head,n);

}

更新：（简单）

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);

}

}

逆链表：（交换节点的数据）

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;//重置

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

{

p1=p1->next;

j++;

}

Student temp;

temp=p->data;

p->data=p1->data;

p1->data=temp;

}

}

合并：

需求->算法

了解归并排序。

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);

}

项目中有许多地方可以改进，但是精力有限，也就做成这样了，有同学有时间，空间开销更小的方法，可以交流学习。

感谢聆听，谢谢！