# 《数据结构》实验报告一

|  |  |  |
| --- | --- | --- |
| 题目： 链表的基本操作 | | 班级： 通信1601 |
| 学号： 29 |  | 姓名： 郑祖闯 |
| 日期： |  | 程序名： |

### 一、上机实验的问题和要求：

单链表的查找、插入与删除。设计算法，实现线性结构上的单链表的产生以及元素的查找、插入与删除。具体实现要求：

1. 从键盘输入20个整数，产生不带表头的单链表，并输入结点值。
2. 从键盘输入1个整数，在单链表中查找该结点的位置。若找到，则显示“找到了”；否则，则显示“找不到”。
3. 从键盘输入2个整数，一个表示欲插入的位置i，另一个表示欲插入的数值x，将x插入在对应位置上，输出单链表所有结点值，观察输出结果。
4. 从键盘输入1个整数，表示欲删除结点的位置，输出单链表所有结点值，观察输出结果。
5. 将单链表中值重复的结点删除，使所得的结果表中个结点值均不相同，输出单链表所有结点值，观察输出结果。
6. 删除其中所有数据值为偶数的结点，输出单链表所有结点值，观察输出结果。
7. 把单链表变成带表头结点的循环链表，输出循环单链表所有结点值，观察输出结果。
8. （★）将单链表分解成两个单链表A和B，使A链表中含有原链表中序号为奇数的元素，而B链表中含有原链表中序号为偶数的元素，且保持原来的相对顺序，分别输出单链表A和单链表B的所有结点值，观察输出结果。

### 二、程序设计的基本思想，原理和算法描述：

（包括程序的结构，数据结构，输入/输出设计，符号名说明等）

这是一个带头结点的线性链表，数据域存放整形数据，由用户输入

头结点数据域存链表长度，所以程序中有个求链表长度的函数

int LengthList(LinkList L); //求链表长度

L是指向头结点的指针，将长度值存入语句为

L->data = LengthList(L);

为了实时观察链表情况，程序中有个输出链表数据的函数

void PrintList(LinkList L); //输出链表

程序可以实现8种不同的操作，这8种不同的操作由8个函数实现，分别是

void CreateList(LinkList &L); //创建链表

void Locate(LinkList L); //查询数值

void InsertList(LinkList &L); //插入数值

void DeleteList(LinkList &L); //选择删除

void Deleterepeat(LinkList &L); //删除重复结点

void DeleteEven(LinkList &L); //删除数值为偶数的结点

void Rotate(LinkList &L); //变为循环链表

void Divide(LinkList &L); //分解成两个链表

主程序中用 开关语句实现:

char operate;

printf("\n\n输入字符选择链表操作类型\nA:创建 B:查询 C:插入 D:选择删除 E:删除重复 F:删除偶数 \nG:变为循环链表 H:分解为两个链表\n");

### 三、源程序及注释：

#include<stdio.h>

#include<malloc.h>

typedef struct LNode //链表结点

{

int data;

struct LNode \*next;

}LNode,\*LinkList;

int over\_flag=0; //主函数结束标识符

void CreateList(LinkList &L); //创建链表

void Locate(LinkList L); //查询数值

void InsertList(LinkList &L); //插入数值

void DeleteList(LinkList &L); //选择删除

void Deleterepeat(LinkList &L); //删除重复结点

void DeleteEven(LinkList &L); //删除数值为偶数的结点

void Rotate(LinkList &L); //变为循环链表

void Divide(LinkList &L); //分解成两个链表

int LengthList(LinkList L); //求链表长度

void PrintList(LinkList L); //输出链表

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

主函数

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

void main(void)

{

char operate;

LinkList L;

int n;

for( n=0;n<40;n++)

{

printf("\n\n输入字符选择链表操作类型\nA:创建 B:查询 C:插入 D:选择删除 E:删除重复 F:删除偶数 \nG:变为循环链表 H:分解为两个链表\n");

label:scanf("%c",&operate);

switch (operate)

{

case 'a':

case 'A': CreateList(L);break;

case 'b':

case 'B': Locate(L);break;

case 'c':

case 'C': InsertList(L); break;

case 'd':

case 'D': DeleteList(L);break;

case 'e':

case 'E': Deleterepeat(L);break;

case 'f':

case 'F': DeleteEven(L);break;

case 'g':

case 'G': Rotate(L);break;

case 'h':

case 'H': Divide(L);break;

case '\n':goto label; //排除换行键的影响

default: printf("输入有误，请重新输入！");break;

}

if(over\_flag)return;

}

}

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

创建链表

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

void CreateList(LinkList &L)

{

int temp;

printf("创建链表：\n请输入创建链表所需的整数值(以-1结束):");

L = (LinkList)malloc(sizeof(LNode));

L->next = NULL;

LinkList q=L;

scanf("%d",&temp);

while(temp!=-1)

{

LinkList p;

p = (LinkList)malloc(sizeof(LNode));

p->data = temp;

p->next = NULL;

q->next = p;

q = q->next;

scanf("%d",&temp);

}

L->data = LengthList(L);

PrintList(L);

}

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

查询元素

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

void Locate(LinkList L)

{ if(!L)printf("错误：链表未创建！");

int element;

printf("查询数值：\n输入要查询的数值：");

scanf("%d",&element);

LinkList p=L->next;

int i =1;

while(p)

{

if(p->data==element)

{

printf("找到了，它是链表的第%d个元素

\n",i);

return ;

}

p=p->next;

i++;

}

printf("找不到

\n");

}

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

插入数值

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

void InsertList(LinkList &L)

{

int x,i;

printf("插入数值：\n输入要插入的数值和插入的位置：");

scanf("%d",&x);

scanf("%d",&i);

LinkList p = L;

int j = 0;

while (p && j < i-1)

{

p = p->next;

++j;

}

if (!p || j > i-1)

{

printf("输入位置错误!") ;

return;

}

LinkList s = (LinkList)malloc(sizeof(LNode));

s->data = x;

s->next = p->next;

p->next = s;

L->data = LengthList(L);

PrintList(L);

}

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

选择位置删除节点

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

void DeleteList(LinkList &L)

{

int i;

LinkList p = L;

printf("选择位置删除结点：\n输入要删除数值的位置：");

scanf("%d",&i);

int j = 0;

while (p->next && j < i-1)

{

p = p->next;

++j;

}

if (!(p->next) || j > i-1)

{

printf("输入位置错误!") ;

return;

}

LinkList q = p->next;

p->next = q->next;

free(q);

L->data = LengthList(L);

PrintList(L);

}

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

删除重复结点

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

void Deleterepeat(LinkList &L)

{

printf("删除重复结点后的链表为：\n");

int n=1;

int a[20];

LinkList q=L->next;

LinkList p=q->next;

a[0]=q->data;

while(p)

{

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

{

if(p->data==a[i])

{

LinkList r=p;

q->next=p->next;

p=p->next;

free(r);

break;

}

}

if(i==n)

{

a[n++]=p->data;

p=p->next;

q=q->next;

}

}

L->data = LengthList(L);

PrintList(L);

}

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

删除数值为偶数的结点

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

void DeleteEven(LinkList &L)

{

printf("删除偶数结点后的链表为：\n");

LinkList q=L;

LinkList p=L->next;

while(p)

{

if(p->data%2==0)

{

LinkList r=p;

q->next=p->next;

p=p->next;

free(r);

}

else

{

p=p->next;

q=q->next;

}

}

L->data = LengthList(L);

PrintList(L);

}

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

变为循环链表

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

void Rotate(LinkList &L)

{

printf("变为循环链表：\n");

LinkList p=L;

while(p->next)p=p->next;

p->next=L;

LinkList t=L->next;

printf("长度：%d\t",L->data);

printf("各个结点数值为：");

while(t!=L)

{

printf("%d\t",t->data);

t=t->next;

}

printf("\n");

printf("已经变为循环链表，其他操作将受影响,程序结束！\n");

over\_flag=1;

}

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

分解成两个链表

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

void Divide(LinkList &L)

{

printf("分解成两个链表：\n");

LinkList A=L;

LinkList B=(LinkList)malloc(sizeof(LNode));

B->next=NULL;

LinkList Lb=B;

int i=1;

LinkList La=L;

LinkList p=L->next;

while(p)

{

if(i++%2==0)

{

La->next=p->next;

p->next=NULL;

Lb->next=p;

Lb=Lb->next;

p=La->next;

}

else

{

p=p->next;

La=La->next;

}

}

A->data = LengthList(A);

printf("链表A:");

PrintList(A);

B->data = LengthList(B);

printf("链表B:");

PrintList(B);

printf("已经分解成两个链表，其他操作将受影响,程序结束！\n");

over\_flag=1;

}

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

求链表长度

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

int LengthList(LinkList L)

{

int i=0;

LinkList p=L->next;

while(p)

{

p=p->next;

i++;

}

return i;

}

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

输出链表

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

void PrintList(LinkList L)

{

LinkList t=L->next;

printf("长度：%d\t",L->data);

printf("结点数值：");

while(t)

{

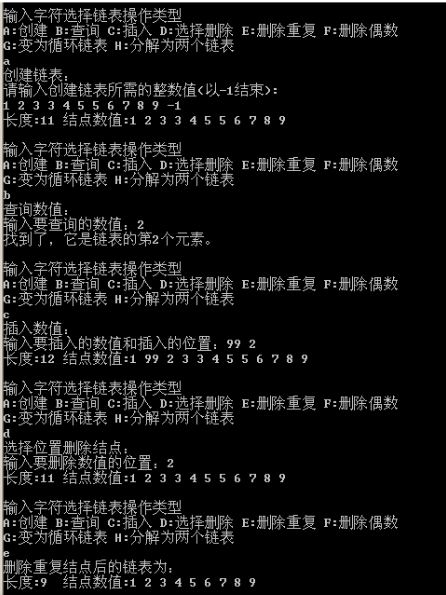
printf("%d\t",t->data);

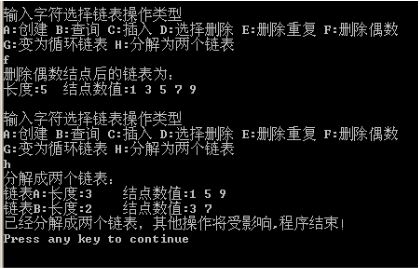
t=t->next;

}

printf("\n");

### 四、运行输出结果：





### 五、调试和运行程序过程中产生的问题及采取的措施：

1. 主程序中我用到 char operate;

scanf("%c",&operate);

Operate存放用户选择操作类型的字母A-H，但是当用户键入字母后，要按ENTER 键

表示输入完毕，所以以后执行scanf("%c",&operate);是会把ENTER输入到

operate中，从而影响了后面的操作，解决方案是加一个标记位

label :scanf("%c",&operate);

当程序发现输入为ENTER时，回到labeL处，这样解决了问题

请看下面

语句，注意 case '\n':goto label;

label : scanf("%c",&operate);

switch (operate)

{

case 'a':

case 'A': CreateList(L);break;

case 'b':

case 'B': Locate(L);break;

case 'c':

case 'C': InsertList(L); break;

case 'd':

caseD':DeleteList(L);break;

case 'e':

case 'E': Deleterepeat(L);break;

case 'f':

case 'F': DeleteEven(L);break;

case 'g':

case 'G': Rotate(L);break;

case 'h':

case 'H': Divide(L);break;

case '\n':goto label; //排除换行键的影响

default: printf("输入有误，请重新输入！");break;

}

2.最后两个操作

void Rotate(LinkList &L); //变为循环链表

void Divide(LinkList &L); //分解成两个链表

一旦执行，将破坏链表的结构，无法再进行其他操作，需要结束整个程序

而在 Rotate和Divide子程序中无法直接结束主函数，所以我设置了一个主函数结束标志位

int over\_flag=0; //主函数结束标识符

初值为0，当它变为1时 表示主函数需要结束

子函数中写

:void Rotate(LinkList &L)

{

..........

over\_flag=1;

}

void Divide(LinkList &L)

{

...........

over\_flag=1;

}

主函数中写

void main (void)

{

..........

if(over\_flag)return;

}

### 六、对算法的程序的讨论、分析，改进设想，其它经验教训：

原来我写的程序是8个基本操作按顺序自动执行，后来听取老师的意见，设置不同的字母对应这些基本操作，由用户自己选择执行，这样就比原来的程序要灵活很多，也便于用户使用和观察。在创建链表时，原来设计的是输入固定长度的链表，后来改进为由用户输入任意长度的数据，只需要用 -1 表示结束即可

### 七、对实验方式、组织、设备、题目的意见和建议：

这个试验题目还可以，基本涵盖了单链表所有的简单操作，只要能写完程序并顺利运行，也就掌握了单链表的相关知识内容，但是由于涵盖内容太多，课内无法完成，必须在课外花很多时间，所以我认为在上完这一章的内容后，老师就应该要求我们做这个程序，这样早点开始，做的时间长一些，效果更应该好一点。