

**数据结构实验报告**

**实验报告题目：** **顺序表基本算法的实现**

**学生姓名**：刘凌杰

**专 业： 软件工程**

**学 号**： 201720180411

**指导教师： 邹 国 华**

**2018年 10 月 8 日**

**一. 实验内容：**

* **顺序表的初始化**
* **输入数据**
* **输出数据**
* **顺序表的插入操作**
* **顺序表的删除操作**
* **顺序表的查找操作**

**二. 实验算法实现：**

* **顺序表结构体**

typedef int DataType;

typedef struct SeqList

{

DataType \*qList;

int Length;

int Maxsize;

}SeqList;

1.顺序表的初始化———置空表。

void LnitList(SeqList \*list)

{

list->Length=0;

list->qList=(DataType\*)malloc(SIZE\*sizeof(DataType));

if(!list->qList)

{

printf("初始化失败！申请内存失败！");

return ;

}

list->Maxsize=SIZE;

}

2.判断线性表是否为空

//判断是否为空

bool ListEmpty(SeqList \*list)

{

return list->Length==0;

}

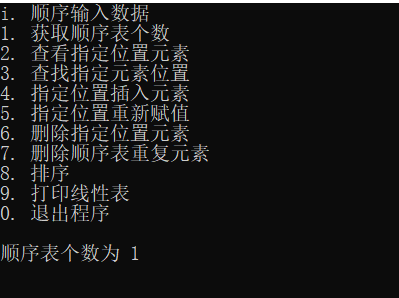
3.获取元素个数

int ListLength(SeqList \*list)

{

return list->Length;

}



4. 查找指定元素位置

//查看线性表L第i个元素

DataType GetElement(SeqList \*list,int i)

{

if(i<0||i>list->Length)

{

printf("位置超出范围 或 该链表为空！\n");

return ERROR;

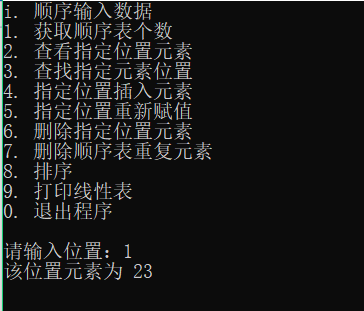
}

else

printf("该位置元素为 %d\n",list->qList[i-1]);

return OK;

}



5. 指定位置插入元素

//查找元素为e的位置

Status LocateElement(SeqList \*list,DataType e)

{

if(ListEmpty(list))

{

printf("空链表！");

return ERROR;

}

bool b=true;

int i;

for(i=0;i<list->Length;i++)

{

if(list->qList[i]==e)

{

printf("该元素位置为 %d\n",i+1);

b=false;

}

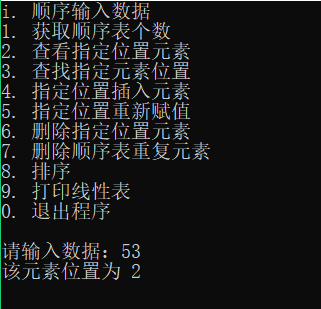
}

if(b)

printf("该元素不存在！\n");

return OK;

}



6. 指定位置插入元素

//在第i个位置插入元素e

Status InsertList(SeqList \*list,int i,DataType e)

{

if(i<=0||i>list->Length+1)

{

printf("位置超出范围或空链表！");

return ERROR;

}

if(list->Length>=list->Maxsize)

{

DataType \*p=(DataType\*)realloc(list->qList,(list->Maxsize+SIZE)\*sizeof(DataType));

if(!p)

{

printf("申请内存失败！\n");

return ERROR;

}

list->qList=p;

list->Maxsize+=SIZE;

}

for(int j=list->Length;j>=i;j--)

{

list->qList[j]=list->qList[j-1];

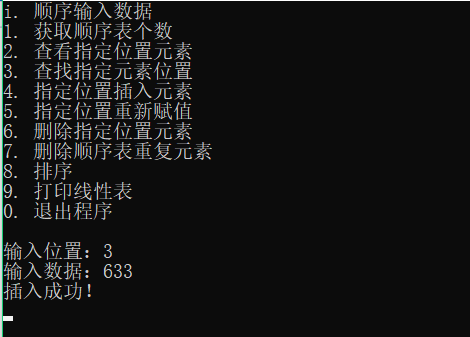
}

list->qList[i-1]=e;

list->Length++;

return OK;

}



7. 指定位置重新赋值

//在下标为i存入新值

Status ChangeList(SeqList \*list,int i,DataType e)

{

if(i<0||i>list->Length)

{

printf("位置超出范围或空链表！");

return ERROR;

}

else

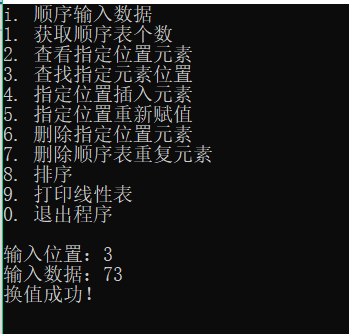
{

list->qList[i-1]=e;

return OK;

}

}



8. 删除指定位置元素

//删除位置为i的元素

Status DeleteList(SeqList \*list,int i)

{

if(i<0||i>list->Length)

{

printf("位置超出范围或空链表！");

return ERROR;

}

else

{

for(int j=i;j<=list->Length-1;j++)

{

list->qList[j-1]=list->qList[j];

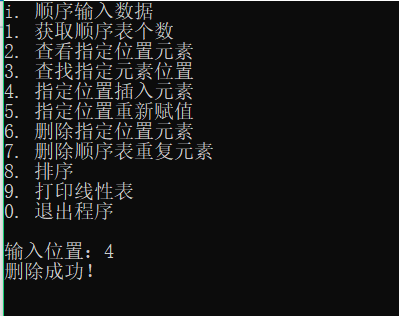
}

list->Length--;

return OK;

}

}



9. 删除顺序表重复元素

//删除重复的数据元素!

void DeleteRepetiion(SeqList \*list)

{

if(ListEmpty(list))

{

printf("空链表！");

return ;

}

int i=0,j;

while(i<list->Length)

{

j=i+1;

while(j<list->Length)

{

if(list->qList[i]==list->qList[j])

DeleteList(list,j+1);

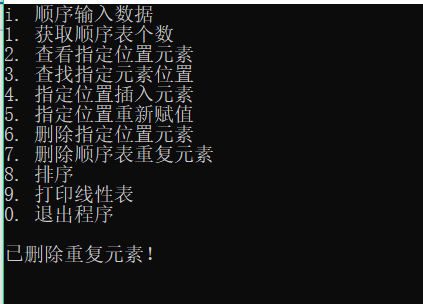
else

j++;

}

i++;

}



printf("已删除重复元素！");

}

10.输出

//显示所有元素

void LookList(SeqList \*list)

{

if(ListEmpty(list))

{

printf("线性表为空!\n");

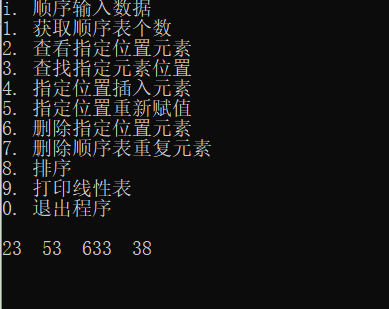
return ;

}

for(int i=0;i<list->Length;i++)

printf("%d ",list->qList[i]);

printf("\n");

}

11.排序

//排序

void Sort(SeqList \*list)

{

if(ListEmpty(list))

printf("空表！\n");

else

{

printf("升序 1.\n");

printf("降序 2.\n");

DataType t;

char i;

i=getch();

switch(i)

{

case '1':

for(int i=0;i<list->Length-1;i++) //冒泡排序

for(int j=0;j<list->Length-1-i;j++)

if(list->qList[j]>list->qList[j+1])

{

t=list->qList[j];

list->qList[j]=list->qList[j+1];

list->qList[j+1]=t;

}

printf("升序成功！\n");

break;

case '2':

for(int i=0;i<list->Length-1;i++) //选择排序

for(int j=i+1;j<list->Length;j++)

{

if(list->qList[i]<list->qList[j])

{

t=list->qList[i];

list->qList[i]=list->qList[j];

list->qList[j]=t;

}

}

printf("降序成功！\n");

break;

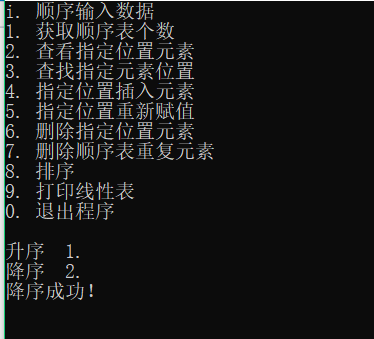
default:

printf("输入有误！");

}

}

}



12.合并顺序表

//合并线性表!

void MergeList(SeqList \*list1,SeqList \*list2)

{

if(list1->Length+list2->Length>list1->Maxsize)

{

DataType \*p=(DataType\*)realloc(list1->qList,(list1->Length+list2->Length+SIZE)\*sizeof(DataType));

if(!p)

{

printf("申请内存失败！\n");

return ;

}

list1->qList=p;

list1->Maxsize=list1->Length+list2->Length+SIZE;

}

for(int i=0;i<list2->Length;i++)

list1->qList[i+list1->Length]=list2->qList[i];

list1->Length=list1->Length+list2->Length;

}

13.复制线性表

//复制线性表

SeqList\* Copy(SeqList \*list)

{

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

if(!p)

{

printf("SeqList申请内存失败！\n");

return NULL;

}

p->qList=(DataType\*)malloc(list->Maxsize\*sizeof(DataType));

if(!p->qList)

{

printf("SeqList->qList申请内存失败！\n");

return NULL;

}

p->Maxsize=list->Maxsize;

for(int i=0;i<list->Length;i++)

{

p->qList[i]=list->qList[i];

}

p->Length=list->Length;

return p;

}

14.输入

//顺序输入

Status Input(SeqList \*list,DataType data)

{

if(list->Length>=list->Maxsize)

{

DataType \*p=(DataType\*)realloc(list->qList,(list->Maxsize+SIZE)\*sizeof(DataType));

if(!p)

{

printf("申请内存失败！\n");

return ERROR;

}

list->qList=p;

list->Maxsize+=SIZE;

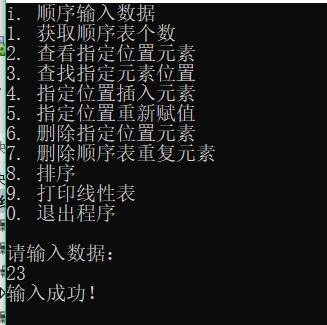
}

list->qList[list->Length]=data;

list->Length++;

printf("输入成功！\n");

}



15.主函数。

int main()

{

SeqList s;

LnitList(&s);

char ch;

int i;

DataType data;

while(1)

{

printf("i. 顺序输入数据\n");

printf("1. 获取顺序表个数\n");

printf("2. 查看指定位置元素\n");

printf("3. 查找指定元素位置\n");

printf("4. 指定位置插入元素\n");

printf("5. 指定位置重新赋值\n");

printf("6. 删除指定位置元素\n");

printf("7. 删除顺序表重复元素\n");

printf("8. 排序\n");

printf("9. 打印线性表\n");

printf("0. 退出程序\n\n");

ch=getch();

switch(ch)

{

case'i':

printf("请输入数据：\n");

scanf("%d",&data);

Input(&s,data);

getch();

system("cls");

break;

case'1':

printf("顺序表个数为 %d\n",ListLength(&s));

getch();

system("cls");

break;

case'2':

printf("请输入位置：");

scanf("%d",&i);

GetElement(&s,i);

getch();

system("cls");

break;

case'3':

printf("请输入数据：");

scanf("%d",&data);

LocateElement(&s,data);

getch();

system("cls");

break;

case'4':

printf("输入位置：");

scanf("%d",&i);

printf("输入数据：");

scanf("%d",&data);

if(InsertList(&s,i,data)==OK)

printf("插入成功！\n");

else

printf("插入失败！\n");

getch();

system("cls");

break;

case'5':

printf("输入位置：");

scanf("%d",&i);

printf("输入数据：");

scanf("%d",&data);

if(ChangeList(&s,i,data)==OK)

printf("换值成功！\n");

else

printf("换值失败！\n");

getch();

system("cls");

break;

case'6':

printf("输入位置：");

scanf("%d",&i);

if(DeleteList(&s,i)==OK)

printf("删除成功！\n");

else

printf("删除失败！\n");

getch();

system("cls");

break;

case'7':

DeleteRepetiion(&s);

getch();

system("cls");

break;

case'8':

Sort(&s);

getch();

system("cls");

break;

case'9':

LookList(&s);

getch();

system("cls");

break;

case'0':

exit(0);

default:

printf("输入序号无效！请重新输入！");

system("cls");

}

}

system("pause");

return 0;

}**三．实验总结：**

经过一段时间的学习，顺序表算法的实现基本掌握了，就是代码打得还不太熟练，还有待提高，有些功能不太熟要问同学或查资料，尤其在做插入、删除运算时顺序表的长度忘记改变。以后在这方面我会多花点时间，让自己更加熟练。