

数据结构实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 题目 | 实验一：线性表的基本操作及其应用 | | |
|  |  | | |
|  |  |  |  | |
| 学院 | 计算机学院 | 年级 | 网络24-1 | |
| 专业 | 网络 | 学号 | 111111111111111 | |
| 姓名 | 杨一一 | | |
| 指导老师 | 刘坤 | | |

2025 年04月01日**实验一 线性表**

**一、实验目的**

1、帮助读者复习C++语言程序设计中的知识。

2、熟悉线性表的逻辑结构。

3、熟悉线性表的基本运算在两种存储结构上的实现，其中以熟悉链表的操作为侧重点。

**二、实验内容**

本次实验提供4个题目，每个题目都标有难度系数，\*越多难度越大，题目一、二是必做题。题目三、题目四选作。

**三、实验准备知识**

1、请简述线性表的基本特性和线性表的几种基本操作的机制

①答：线性表的基本特性是：对线性表中某个元素ai来说，称其前面的元素ai-1为ai的直接前驱，称其后前面的元素ai+1为ai的直接后继。显然，线性表中每个元素最多有一个直接前驱和一个直接后继。

②答：线性表的几种基本操作的机制有六个：

（1）初始化线性表 initial\_List(L)——建立线性表的初始结构，即建空表。这也是各种结构都可能要用的运算。

（2）求表长度 List\_length(L)——即求表中的元素个数。

（3）按序号取元素 get\_element(L,i)——取出表中序号为i的元素 。

（4）按值查询 List\_locate(L,x)——取出指定值为x的元素，若存在该元素，则返回其地址；否则，返回一个能指示其不存在的地址值或标记。

（5）插入元素 List\_insert(L,i,x)——在表L的第i个位置上插入值为x的元素。显然，若表中的元素个数为n，则插入序号i应满足1<=i<=n+1。

（6）删除元素 List\_delete（L，i）——删除表L中序号为i的元素，显然，待删除元素的序号应满足1<=i<=n。

**四、实验内容**

**题目一：线性表的顺序表示**

[问题描述]

实现顺序表的建立、求长度，取元素、修改元素、插入、删除等基本操作。

[基本要求]

（1）依次从键盘读入数据，建立顺序表；

（2）输出顺序表中的数据元素；

（3）求顺序表的长度；

（4）根据指定条件能够取元素和修改元素；

（5）实现在指定位置插入和删除元素的功能。

[源代码]**（加注释）**

#include <iostream.h>

#include <malloc.h>

#define maxlen 100 //设元素个数最大为100

typedef struct //声明一个结构体来存放顺序表

{ int data[maxlen]; //定义存储表中元素的数组

int listlen; //定义表长度分量

}seqlist;

void get\_element(seqlist \*L,int I,int \*x); //按序号求元素运算的子函数

int List\_locate(seqlist L,int I,int x); //声明按值查询元素的子函数

bool List\_insert(seqlist \*L,int I,int x); //声明插入元素的子函数

bool List\_delete(seqlist \*L,int I); //声明删除元素的子函数

void main()

{ seqlist \*L; //表结构变量的定义

L=(seqlist \*)malloc(sizeof(seqlist)); //为表分配空间

int i; //定义整型变量

for(i=1;i<=20;i++) //赋值给表中的元素

{ L->data[i-1]=i; //填入删除内容

}

L->listlen=20; //定义表的长度

cout<<"删除前的顺序表:\n";

for(i=1;i<=L->listlen;i++) //输出表中元素

{ cout<<L->data[i-1]<<" ";

}

cout<<"\n输入要删除的纪录号:";

cin>>i; //用户输入删除的记录

if(List\_delete(L,i)) //用删除元素的子函数

{ cout<<"\n删除成功！删除后的顺序表为:\n";

for(i=1;i<=L->listlen;i++) //向前批量移动元素

{ cout<<L->data[i-1]<<" ";

}

}

}

bool List\_delete(seqlist \*L,int i) //删除元素的子函数

{ int j; // 定义一个整型变量

if(L->listlen<=0) //空表不能删除元素

{ cout<<"下溢出错！";

return false; //返回false

}

if(i>L->listlen||i<=0) //删除元素不存在

{ cout<<"删除位置错！";

return false; //返回false

}

else{ for(j=i;j<=L->listlen-1;j++) //向前批量移动元素

{ L->data[j-1]=L->data[j];;

}

L->listlen--; //表长度减1

return true; //返回true

}

}

void main()

{ seqlist \*L; //表结构变量的定义

L=(seqlist \*)malloc(sizeof(seqlist)); //为表分配空间

int i; //定义整型变量

for(i=1;i<=20;i++) //赋值给表中的元素

{ L->data[i-1]=i;

}

L->listlen=20; //定义表的长度

cout<<"取元素前的顺序表:\n";

for(i=1;i<=L->listlen;i++) //输出表中元素

{ cout<<L->data[i-1]<<" ";

}

cout<<"\n输入要取元素的序号:";

cin>>i; //用户输入要取元素的记录

if(get\_element( L, i)) //用取元素的子函数

{ cout<<"\n取元素成功！取元素后的顺序表为:\n";

for(i=1;i<=L->listlen;i++)

{ cout<<L->data[i-1]<<" ";

}

}

}

bool get\_element(seqlist \*L,int i) //取元素的子函数

{ int x;

if(i<1||i>L->listlen) //要取的元素不存在

{ cout<<"下溢出错！"; //空表无元素

return false;}

else

{ x=L->data[i-1]; //取出相应元素

cout<<"\n所取的元素\n";

cout<<x<<" "; //输出对应元素

return true;}

}

void main()

{ seqlist \*L; //表结构变量的定义

L=(seqlist \*)malloc(sizeof(seqlist)); //为表分配空间

int i,x,a; //定义三个整型变量

for(i=1;i<=20;i++) //赋值给表中的元素

{ L->data[i-1]=i;

}

L->listlen=20; //定义表的长度

cout<<"查找元素前的顺序表:\n";

for(i=1;i<=L->listlen;i++) //输出表中元素

{ cout<<L->data[i-1]<<" ";

}

cout<<"\n输入要查元素:";

cin>>x; //用户输入查找的记录

a=List\_locate(L,x);

if(a!=0)

{ cout<<"\n查元素成功！查元素后的顺序表为:\n";

cout<<a<<" ";

cout<<"\n取元素后的顺序表为:\n"; //取元素后的顺序

for(i=1;i<=L->listlen;i++) //输出表中元素

{ cout<<L->data[i-1]<<" ";

}

}

else cout<<"\n查无此元素成功！:\n";

}

int List\_locate(seqlist \*L,int x) //按值查询元素的子函数

{ int i; //定义一个整型变量

for(i=0;i<L->listlen;i++) //依次比较各个元素

{ if(L->data[i]==x) return(i+1); //找到元素X的位置返回其序号

}

return(0); //未找到则返回0值

}

void main()

{ seqlist \*L; //表结构变量的定义

L=(seqlist \*)malloc(sizeof(seqlist)); //为表分配空间

int i,x,a; //定义三个整型变量

for(i=1;i<=20;i++) //赋值给表中的元素

{ L->data[i-1]=i; //填入插入内容

}

L->listlen=20; //定义表的长度

cout<<"插入元素前的顺序表:\n";

for(i=1;i<=L->listlen;i++) //输出表中元素

{ cout<<L->data[i-1]<<" ";

}

cout<<"\n输入要插的元素:";

cin>>x; //用户输入插入的记录

cout<<"\n输入要插的位置:";

cin>>i; //用户输入插入的位置

a=List\_insert(L,i,x);

if(a==1)

{ cout<<"\n插入元素成功！:\n";

cout<<"\n插入元素后的顺序表为:\n";

for(i=1;i<=L->listlen;i++) //输出表中元素

{ cout<<L->data[i-1]<<" ";

}

}

else if(a==0)

cout<<"溢出,不能插入！"; //溢出,不能插入

else cout<<"插入位置错！";

}

int List\_insert(seqlist \*L,int i,int x) //插入元素的子函数

{ int j; //定义一个整型变量

if(L->listlen==maxlen)

{ cout<<"溢出,不能插入！"; //溢出，不能插入

return 0; //返回

}

else if(i<1||i>L->listlen+1)

{ cout<<"插入位置错！"; //插入范围错

return -1 ;} //返回并结束

else

{for(j=L->listlen-1;j>=i-1;j--) //往后移动元素

L->data[j+1]= L->data[j];

L->data[i-1]=x; //填入插入的元素

L->listlen++; //修改表长度

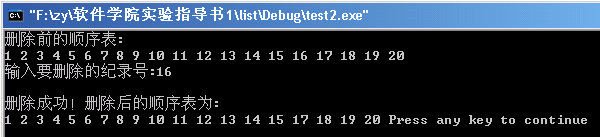
return 1; //返回

}

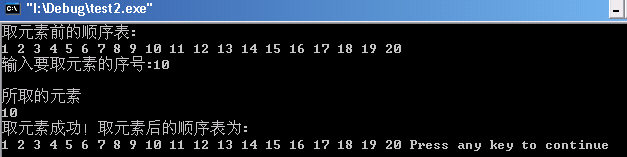
}

**运行结果如下图：**

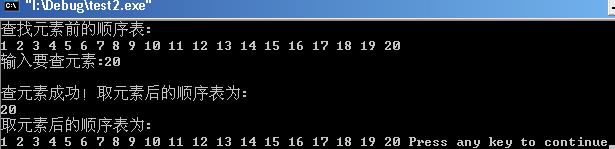
1）实现删除功能



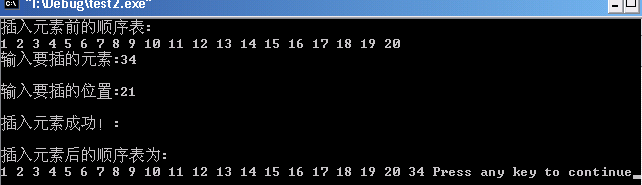
2）实现取元素功能



3）实现查找功能



4）实现插入功能



**题目二：线性表的链式表示**

[问题描述]

实现带头结点的单链表的建立、求长度，取元素、修改元素、插入、删除等单链表的基本操作。

[基本要求]

（1）依次从键盘读入数据，建立带头结点的单链表；

（2）输出单链表中的数据元素

（3）求单链表的长度；

（4）根据指定条件能够取元素和修改元素；

（5）实现在指定位置插入和删除元素的功能。

[源代码]

……

**五、实验完成时间、地点**

2015.3.19（星期一）软件学院405机房

**六、实验总结**

（在设计调试程序过程中出现错误、问题及解决方案）