第5次实验

1. 线性表顺序存储——顺序表

线性表是由有限个同类型的数据元素组成的有序序列，一般记作（a1,a2,…,an）。除了a1和an之外，任意元素ai都有一个直接前驱ai-1和一个直接后继ai+1。a1无前驱，an无后继。

采用顺序存储结构的线性表称为顺序表，它的数据元素按照逻辑顺序依次存放在一组连续的存储单元中。下图用数组实现顺序表。

length=n maxsize

0 1 i-2 i-1 i n-1

a2

…

ai-1

ai

ai+1

a1

…

an

图 顺序表存储结构示意图

注意：顺序表的元素一定是连续存放的，中间没有空隙。

* **顺序表的主要算法**

判断顺序表是否为空只需判断length是否为零即可。顺序表的主要算法是在顺序表中插入元素、删除元素以及查找元素。

⑴ 在表中第 i 个位置**插入新元素** x

算法实现的主要步骤是：

第一步，判断插入位置的合理性以及表是否已满；

第二步，从最后一个元素开始依次向前，将每个元素向后移动一个位置，直到第i个元素为止。

第三步，向空出的第i个位置存入新元素 x

第四步，最后还要将线性表长度加1。

⑵ 在表中删除第i个元素

算法实现的主要步骤是：

第一步，判断删除位置的合理性

第二步，从第i+1个元素开始，依次向后直到最后一个元素为止，将每个元素向前移动一个位置。这时第i个元素已经被覆盖删除。

第三步，最后还要将线性表长度减1。

* 实验1——实现整形元素线性表

用数组实现在线性表中插入整数元素，在中间任何位置删除元素，查找一个元素。在main函数中验证

【算法】

const int maxsize=100; // 顺序表最大允许长度

// 定义顺序表

struct SeqList {

int data[maxsize]; // 顺序表存储数组的地址

int length; // 顺序表当前长度

};

void Insert( SeqList \*L, int i, int x ) // 插入

{

if ( 插入位置错误或表满 )

输出错误信息并返回;

else {

for( int j=L->length-1; j>=i-1; j-- ) {

第 j 个位置元素向后移动;

}

向第 i 个位置存入新元素 x;

表长度加一;

}

}

void Delete( SeqList \*L, int i ) // 删除

{

if ( 表中没有第i个元素或者大于表长 )

输出提示信息并返回;

else {

for ( int j=i; j<=L->length-1; j++ ) {

将第j个元素移植第j-1个元素处;

}

表长度减一;

}

}

int Find( SeqList \*L, int x )

{

for( int i = 0; i<L->length; i++ )

{

查找成功，返回元素位置

}

return 0; //查找失败, 返回 0

}

void Display( SeqList \*L ) //显示顺序表数据

{

for (int n = 0; n < L->length; ++n)

输出第n项元素信息;

}

// 在主函数定义顺序表，输入三个元素，显示，再删除一个元素、显示

int main()

{

SeqList L;

L.length=0;

int s;

cout<<"请输入三个整数：\n";

for(int i=0;i<3;i++)

{

cin>>s;

Insert( &L,i+1,s );

}

cout<<endl;

Display(&L); //显示顺序表数据

int ID;

cout<<"请输入将要删除的整数：\n";

cin>>ID;

Delete(&L, ID);

cout<<endl;

Display(&L); //显示顺序表数据

return 0;

}

* 实验2——学生信息管理线性表

将实验1中的元素用结构体代替，实现程序。

要求：

1. 插入学生记录时，学号按从小到大顺序插入
2. 删除学生信息时，按学号删除

|  |
| --- |
| 以下面例子说明，从小到大顺序插入信息的过程：  假设插入数据 5、7、2、3、9  第一步，顺序表空，插入5 // [5]  第二步，因为7>5，向下一个位置移动，因为到达尾部，所以插入7 // [5 7]  第三步，因为2<5，在当前位置插入2 // [2 5 7]  第四步，因为3>2，后移；因为3<5，在当前位置插入3 // [2 3 5 7]  第五步，因为9>2，后移；9>3，后移；9>5，后移；9>7，后移；  因为到达尾部，所以插入9 //[2 3 5 7 9] |

【算法】

const int maxsize=100; // 顺序表最大允许长度

// 定义数据元素

struct stu {

int id; //学号

char name[20]; //姓名

};

// 定义顺序表

struct SeqList {

stu data[maxsize]; // 顺序表存储数组的地址

int length; // 顺序表当前长度

};

// 插入元素

void Insert( SeqList \*L, stu x )

{

令j为表尾下标 ；

if( 表的长度大于0并且表的长度小于顺序表的容量)

{

while( 第j项的学号>插入学号并 且 序号j不在头部)

{

第j项元素向后移动1个位置;

序号j向前移动;

}

存入新元素x ；

表长度加1 ；

}

}

// 删除元素

void Delete( SeqList \*L, int ID )

{

定义j为元素下标

while(第j项的学号不等于ID 并且j小于表的长度) j++;

if(j大于等于表的长度)

输出"无此元素!"信息;

else {

for(int i=j; i<L->length-1; i++)

{

将第i+1个元素向前移动1个位置;

}

表长度减1；

}

}