线性表

概念： 拥有n个**相同数据类型**的数据元素的**有限序列**，其中n为表长

L=(a1,a2,a3.....an)

其中an中"第n个"表示元素的**位序**（从0开始的是**下标**）

a1是表头元素，an是表尾元素

除了a1其他元素都有一个**直接前驱**，除了an其他元素都有**直接后继**

基本操作

InitList(&L)：初始化表

DestroyList(&L)：销毁表

ListInsert(&L,i,e)：插入操作

ListDelete(&L,i,&e)：删除操作

LocateElem(L,e)：按值查找

GetElem(L,i)：按位查找

Length(L)：求表长

PrintList(L)：打印表

Empty(L)：判空操作

顺序表

逻辑上相邻的元素其物理位置上也相邻

静态分配

```
#define MAXSIZE 10

typedef struct {

    int data[MAXSIZE];

}
```

提前定义指定长度的数组，但是这样的话不能确定实际上的数据元素个数。在初始化表的时候，数组内数据可以不用初始化，但是int length需要初始化为0，定义时会不会自动设为0主要看编译

器。

动态分配

```
typedef struct{
    int *data;
}
```

动态申请空间：

C：malloc free

C++：new delete

```
//加长表
addLength(SqList &L,int length){
    int *p = L.data;
    L.data = (int *)malloc(sizeof(int)x(L.MaxLength + length));
    for(int i = 0;i < L.length;i++){
        L.data[i] = p[i];
    }
    L.MaxLength+=length;
free p;
}
```

顺序表特点： 随机访问、存储密度高、拓展容量不方便、插入删除不方便

```
15  bool ListInsert(SqList &L,int i,int e){
16      if (L.length >= 10)
17          return false;
18      if (i<1 || i>L.length+1)
19          return false;
20      if (i>10)
21          return false;
22      for (int j = L.length; j >= i; j--)
23      {
24          L.data[j] = L.data[j-1];
25      }
26      L.data[i-1] = e;
27      L.length++;
28      return true;
29  }
```

插入方法

```
33  bool ListDelete(SqList &L,int i,int &e){
34      if (i<1 || i>L.length)
35          return false;
36      e = L.data[i-1];
37      for (int j = i; j < L.length; j++)
38      {
39          L.data[j-1] = L.data[j];
40      }
41      L.length--;
42      return true;
43  }
```

删除方法

```
46  int GetElem(SqList &L,int i){
47      if (i<1 || i>L.length)
48          return 0;
49      return L.data[i-1];
50  }
```

按位查找

```
53  int ElemLocal(SqList &L,int e){
54      for (int i = 0; i < L.length; i++)
55      {
56          if (L.data[i] == e)
57          {
58              return i+1;
59          }
60      }
61      return 0;
62  }
```

按值查找

结构体不能直接使用“==”来判断是否相等

单链表

typedef struct{

    int data;//数据域

    struct LNode *next;//指针域

}LNode,*LinkList;

不带头结点的单链表

L（头指针）->LNode（存放数据的）

写代码会更麻烦

带头结点的单链表

L（头指针）->LNode（头结点）->（存放数据的）

带头结点的话后续进行操作的时候会更加方便