顺序表定义：用**顺序存储**的方式实现线性表
顺序存储定义：把**逻辑上相邻**的元素存储在**物理位置上也相邻**的存储单元中，元素之间的关系由存储单元的邻接关系来体现。
线性表定义：线性表具有**相同**特性**的数据元素**的一个有限序列。

静态顺序表创建

1、创建数据

2、初始化顺序表

```
typedef struct {                //创建数据
    ElemType data[MaxSize];   //ElemType数据类型
    int length;
}SqList;
```

```
void InitList(SqList &L) {          //初始化一个顺序表
    for (int i = 0;i < MaxSize;i++) {//MaxSize为最大容量
        L.data[i] = 0
        L.length = 0;
```

```
int main() {
    SqList L;//声明一个顺序表
    InitList(L);//初始化一个顺序表
    return 0;
}
```

动态顺序表创建

```
#define _CRT_SECURE_NO_WARNINGS
#define MAX 200
typedef struct SqList {
    int* data;             //指向动态数组的指针
    int MaxSize;          //最大容量
    int length;          //当前长度
}L;
```

初始化

```
SqList* InitList(SqList *L) {
    L->data = (SqList*)malloc(sizeof(int)*MAX);  //指针指向一块空间
    L->MaxSize = MAX;
    L->length = 0;            //初始长度为0
}
```

增加动态数组的长度

```
void IncreaseSize (SqList* W, int len){
    int* p = W->data;          //存储原来空间的数据
        W->data = (int*)malloc((W->MaxSize + len) * sizeof(int));
        for (int i = 0;i < W->length;i++) {
            W->data[i] = p[i];         //将数据复制到新区域
        }
        W->MaxSize = W->MaxSize + len;
        free(p);
}
```

顺序表插入操作

```
//在中间插入几个元素
bool ListInsert(L, int e, int i){//e为插入点 i为插入数值
    if (e<1 || i>L.length + 1)//判断e的范围是否正确
        return false
        if (L.length >= MaxSize)//当前存储空间已满，不能插入
            return false;
    for (int j = L.lenth;j >= e;j--) {// 从L.length 循环至插入点
        L.data[j] = L.data[j - 1];//前后换位
        L.data[j - 1] = i;//插入点换成插入值
        L.length++;//替换后总长度+1
        return true;
    }
}
```

顺序表删除操作

```
bool ListDelete(SqList* L, int i, int* e) {
    if (i<1 || i>L.length)
        return false;
    e = L.data[i - 1];
    if (int j = i;j < L.length;j++))
    L.data[j - 1] = L.data[j];
    L.length--;
    return true;
}
```