#define \_CRT\_SECURE\_NO\_WARNINGS 1  
#include<stdio.h>  
#include<stdlib.h>  
#define InitSize 10  
  
typedef struct  
{  
int length;//当前顺序表长度  
int Maxsize;//顺序表最大长度  
int\* data;//定义顺序表中元素类型的数组指针  
}SqList;  
  
void InitList(SqList &L)//初始化顺序表  
{  
L.data = (int \*)malloc(InitSize\*sizeof(int));//使用malloc函数申请一片空间   
L.length = 0;//顺序表的当前长度为0   
L.Maxsize = InitSize;//顺序表的最大长度   
}  
  
void WriteList(SqList &L)//把元素放入顺序表  
{  
printf("请输入你要创建的顺序表的长度：");  
scanf("%d", &L.length);  
printf("请输入%d个你要放入顺序表里的元素：",L.length);  
for (int i = 0; i < L.length; i++)  
scanf("%d", &L.data[i]);  
}  
  
void IncreaseSize(SqList &L)//增加顺序表的长度  
{  
int len;  
int \*p = L.data;//\*p指向的地址和顺序表的首地址是一样的   
printf("请输入你要增加的顺序表的长度:");  
scanf("%d", &len);  
L.data = (int \*)malloc((L.Maxsize + len)\*sizeof(int));  
for (int i = 0; i < L.length; i++)  
L.data[i] = p[i];//把值一个个复制过去   
L.Maxsize = L.Maxsize + len;//顺序表最大长度增加Len   
free(p);//释放空间   
}  
/\*元素插入有一些要求：  
1.元素下标是否越界（有没有插队到奇怪的位置）  
2.顺序表存储空间是否满了（有没有位置让你插队）\*/   
bool ListInsret(SqList &L)//插入元素  
{  
int i, e;  
printf("请输入要插入顺序表的元素和元素位置：");  
scanf("%d %d", &e, &i);  
if (i<1 || i>L.length + 1)  
return false;  
if (L.length > L.Maxsize)  
return false;  
for (int j = L.length; j >= i; j--)  
{  
L.data[j] = L.data[j-1];  
}  
L.data[i-1] = e;  
L.length++;  
printf("插入的元素是%d，插入的位置是%d\n", e, i);  
return true;  
}  
/\*元素删除有一些要求：  
1.元素下标是否越界  
2.顺序表存储空间是否为空\*/   
bool ListDelete(SqList &L)//删除操作  
{  
int i, e;  
printf("请输入要删除的元素位置：");  
scanf("%d",&i);  
if (i<1 || i>L.length + 1)  
return false;  
if (!L.data)  
return false;  
e = L.data[i - 1];  
for (int j = i; j <= L.length; j++)  
{  
L.data[j-1] = L.data[j];  
}  
L.length--;  
printf("删除的元素是%d，这个元素的位置是%d\n", e, i);  
return true;  
}  
  
bool GetElem(SqList &L)//按位查找  
{  
int i;  
printf("你要找第几个元素：");  
scanf("%d", &i);  
if (i<1 || i>L.length + 1)//判断输入的i值是否合法   
{  
printf("查找失败\n");  
return false;//不合法返回一个false   
}  
printf("第%d个元素是%d\n", i, L.data[i - 1]);  
return true;//返回一个true   
}  
/\*false/true是bool型变量，C++独有，一般将非零值看做true，将零值看做false。\*/   
void LocateElem(SqList &L)//按值查找  
{  
int e;  
int k = 1;  
printf("输入你要查找的元素值：");  
scanf("%d", &e);  
for (int i = 0; i < L.length; i++)//判断输入的e值是否合法   
if (L.data[i] == e)  
{  
printf("找到了，是第%d个元素\n", i + 1);  
k = 0;  
break;  
}  
if (k)  
printf("找不到元素%d\n", e);  
}  
  
bool PrintList(SqList &L)//打印顺序表  
{  
if (!L.data)  
return false;  
printf("顺序表里的元素有：");  
for (int i = 0; i < L.length; i++)  
printf("%d ", L.data[i]);  
printf("\n");  
return true;  
}  
/\*顺序表初始化的时候是用malloc函数向系统申请的空间  
malloc函数申请的空间是在内存的堆区  
堆区的空间不会被系统自动回收，只把L.length改为0是不够的  
还需要用free函数释放空间  
与malloc一样，要引头文件#include<stdlib.h>\*/   
void DestroyList(SqList &L)//销毁顺序表  
{  
char a;  
getchar();  
printf("是否销毁顺序表(Y/N):");  
scanf("%c", &a);  
if (a == 'Y')  
{  
L.length = 0;  
L.Maxsize = 0;  
free(L.data);  
printf("顺序表已销毁\n");  
}  
}  
  
int Length(SqList &L)//求表长  
{  
if (L.length == 0)  
return 0;  
return L.length;  
}  
  
int main()  
{  
SqList L;  
InitList(L);  
WriteList(L);  
PrintList(L);  
IncreaseSize(L);  
ListInsret(L);  
PrintList(L);  
ListDelete(L);  
PrintList(L);  
GetElem(L);  
LocateElem(L);  
int len = Length(L);  
printf("顺序表的表长:%d\n", len);  
DestroyList(L);  
return 0;  
}