**实验2 顺序表的基本操作**

【实验目的】

1. 掌握顺序表的头部操作与尾部操作
2. 掌握顺序表的指定位置操作
3. 掌握顺序表的查找与删除的几种应用
4. 熟悉顺序表的几种排序方法

【实验要求】

1. 实验共5题，每题20分，满分100分；
2. 实验需在截至日期前提交实验平台，由实验平台自动判分，补交时间段内提交酌情扣分；
3. 学生对实验分数有异议，提交申请，带教老师修正得分；
4. 原则上学生需独立完成实验题，个别难题可互相讨论或查阅资料；
5. 若对试验题有疑问，且仔细阅读测试用例后仍然不理解，可咨询带教老师。

【实验内容】

1、顺序表的尾部操作

2、顺序表的头部操作

3、顺序表的指定位置操作

4、顺序表查找与删除的几种应用

5、顺序表的几种排序方法

**1.1顺序表（尾部操作）**

本题主要考查内容:

1. 顺序表初始化
2. 求顺序表的长度
3. 在顺序表的尾部插入值为data的元素，建立顺序表
4. 删除顺序表的尾部元素,其值存入指针x所指单元
5. 打印顺序表中所有元素

【程序片段】

#include <stdio.h>

#define MaxSize 100

typedef int ElemType;

typedef struct sqlist{

ElemType list[MaxSize];

int size;

}SqList;

void InitSqList(SqList \*L){//1初始化顺序表

L->size=0;

}

int ListLength(SqList L){//2求顺序表长度

return L.size;

}

//--------顺序表的尾部操作---------

int PushBack(SqList \*L,ElemType x){//3在顺序表的尾部插入值为data的元素

if(L->size>=MaxSize){

printf("list is full.\n");return 0;

}

【1】

【2】

return 1;

}

int PopBack(SqList \*L,ElemType \*x){//4删除顺序表的尾元素,其值存入指针x所指单元

if (L->size <= 0)

{

printf("There is no element in the sequence table that can be deleted.\n");

\*x = -12345;

return 0;

}

【3】 ;//3

【4】;//4

return 1;

}

int ShowSeqList(SqList \* L){//11打印顺序表中所有元素

int i;

if(L->size==0){

printf("There is no element in the sequence table.\n");return 0;

}

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

printf("%d ", L->list[i]);

printf("\n");

return 1;

}

int main(){

SqList my\_list ;

int x,n,i,num;

int switch\_num;

scanf("%d",&switch\_num);

switch(switch\_num){

case 1: InitSqList(&my\_list);

printf("%d\n",my\_list.size);

ShowSeqList(&my\_list);

break;

case 2: InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushBack(&my\_list,num);

}

ShowSeqList(&my\_list);

break;

case 3: InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushBack(&my\_list,num);

}

ShowSeqList(&my\_list);

for(i=1;i<=n-2;i++){

PopBack(&my\_list,&x);

printf("The %d PopBack delete num = %d\n",i,x);

}

ShowSeqList(&my\_list);

break;

case 4: InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushBack(&my\_list,num);

}

ShowSeqList(&my\_list);

for(i=1;i<=n;i++){

PopBack(&my\_list,&x);

printf("The %d PopBack delete num = %d\n",i,x);

}

ShowSeqList(&my\_list);

break;

case 5:InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushBack(&my\_list,num);

}

ShowSeqList(&my\_list);

for(i=1;i<=n+2;i++){

PopBack(&my\_list,&x);

printf("The %d PopBack delete num = %d\n",i,x);

}

ShowSeqList(&my\_list);

break;

}

return 0;

}

【测试用例】

本题主要测试顺序表的尾部操作，分以下情况：

1. 初始化顺序表后，打印顺序表的长度与元素
2. 初始化顺序表之后，依次在顺序表尾部插入若干元素，打印顺序表的长度与元素
3. 初始化顺序表并在顺序表尾部插入A个元素，依次删除尾部的B个元素（B<A），依次打印这B个元素，打印顺序表的长度与元素（顺序表的长度为|A-B|，顺序表中有前|A-B|元素）。
4. 初始化顺序表并在顺序表尾部插入A个元素，依次删除尾部的B个元素（B=A），依次打印这B个元素，打印顺序表的长度与元素（顺序表的长度为0，顺序表中没有元素）。
5. 初始化顺序表并做顺序表尾部插入操作A次，做顺序表尾部删除操作B次，其中B>A，打印顺序表的长度与元素（顺序表的长度为0，顺序表中没有元素）。

1/

输入：1，进行第1项测试

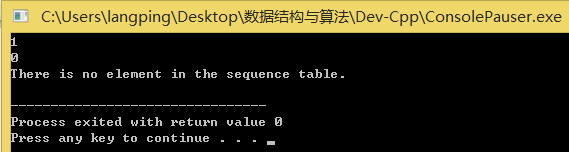
输出：第一行 0，表示顺序表长度为0

第二行There is no element in the sequence table.意思是顺序表里没有元素可打印

1

0

There is no element in the sequence table.



2/

输入：第一行2，进行第2项测试

第二行5，准备从键盘输入5个数，使用顺序表尾插入法插入顺序表

第三行2 4 6 3 5为从键盘输入的这5个数。

输出：第一行2 4 6 3 5意思是依次打印顺序表的结果。

2

5

2 4 6 3 5

2 4 6 3 5



3/

输入：第一行3，进行第三项测试

第二行6，准备输入6个数，用尾插法插入线性表

第三行，1 3 5 2 4 6为从键盘输入的这6个数

输出：第一行，1 3 5 2 4 6为插入完成后打印顺序表的结果

第二行The 1 PopBack delete num = 6，意思是第一次对顺序表进行尾删除操作的元素值是6

第二行The 2 PopBack delete num = 4，意思是第二次对顺序表进行尾删除操作的元素值是4

……

最后一行1 3表示，所有尾删除操作完成后，打印顺序表的结果

3

6

1 3 5 2 4 6

1 3 5 2 4 6

The 1 PopBack delete num = 6

The 2 PopBack delete num = 4

The 3 PopBack delete num = 2

The 4 PopBack delete num = 5

1 3



4/

输入：第一行4，进行第四项测试

第二行4，准备输入4个数，用尾插法插入线性表

第三行，1 2 3 4为从键盘输入的这4个数

输出：第一行，1 2 3 4为插入完成后打印顺序表的结果

第二行The 1 PopBack delete num = 4，意思是第一次对顺序表进行尾删除操作的元素值是4

第二行The 2 PopBack delete num = 3，意思是第二次对顺序表进行尾删除操作的元素值是3

……

最后一行There is no element in the sequence table.表示，所有尾删除操作完成后，打印顺序表的结果（顺序表已经没有元素）

4

4

1 2 3 4

1 2 3 4

The 1 PopBack delete num = 4

The 2 PopBack delete num = 3

The 3 PopBack delete num = 2

The 4 PopBack delete num = 1

There is no element in the sequence table.



5/

输入：第一行5，进行第五项测试

第二行3，准备输入3个数，用尾插法插入线性表

第三行，1 2 3 为从键盘输入的这3个数

输出：第一行，1 2 3 为插入完成后打印顺序表的结果

第二行The 1 PopBack delete num = 3，意思是第一次对顺序表进行尾删除操作的元素值是4

第二行The 2 PopBack delete num = 2，意思是第二次对顺序表进行尾删除操作的元素值是3

第三行The 2 PopBack delete num = 1，意思是第二次对顺序表进行尾删除操作的元素值是1

第四行There is no element in the sequence table that can be deleted.（顺序表无元素可删除）

第五行The 4 PopBack delete num = -12345（顺序表无元素可删除时x变量为一特殊值）

……

最后一行There is no element in the sequence table.表示，所有尾删除操作完成后，打印顺序表的结果（顺序表已经没有元素）

5

3

1 2 3

1 2 3

The 1 PopBack delete num = 3

The 2 PopBack delete num = 2

The 3 PopBack delete num = 1

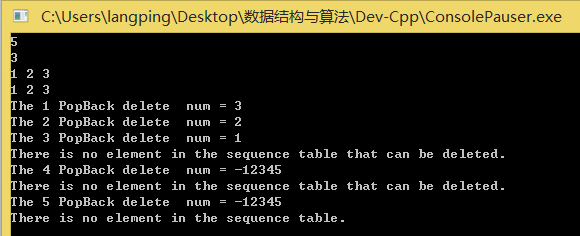
There is no element in the sequence table that can be deleted.

The 4 PopBack delete num = -12345

There is no element in the sequence table that can be deleted.

The 5 PopBack delete num = -12345

There is no element in the sequence table.



**1.2顺序表（头部操作）**

本题主要考查内容：

1. 在顺序表的头部插入值为data的元素，建立顺序表
2. 删除顺序表的头部元素,其值存入指针x所指单元
3. 打印顺序表中所有元素

【程序片段】

#include <stdio.h>

#define MaxSize 100

typedef int ElemType;

typedef struct sqlist{

ElemType list[MaxSize];

int size;

}SqList;

void InitSqList(SqList \*L){//1初始化顺序表

L->size=0;

}

int ListLength(SqList L){//2求顺序表长度

return L.size;

}

int PushFront(SqList \*L,ElemType x){//5在顺序表的头部插入值为data的元素

int j;

if(L->size>=MaxSize){

printf("list is full.\n");return 0;

}

for(j=L->size;j>=1;j--){

L->list[j]=L->list[j-1];

}

L->list[0]=【1】;//1

【2】;//2

return 1;

}

int PopFront(SqList \*L,ElemType \*x){//6删除顺序表的头部元素,其值存入指针x所指单元

int j;

if (L->size <= 0)

{

printf("There is no element in the sequence table that can be deleted.\n");

\*x = -12345;

return 0;

}

//3

【3】

return 1;

}

int ShowSeqList(SqList \* L){//11打印顺序表中所有元素

int i;

if(L->size==0){

printf("There is no element in the sequence table.\n");return 0;

}

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

printf("%d ", L->list[i]);

printf("\n");

return 1;

}

int main(){

SqList my\_list ;

int x,n,i,num;

int switch\_num;

scanf("%d",&switch\_num);

switch(switch\_num){

case 1: InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushFront(&my\_list,num);

}

ShowSeqList(&my\_list);

break;

case 2: InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushFront(&my\_list,num);

}

ShowSeqList(&my\_list);

for(i=1;i<=n-2;i++){

PopFront(&my\_list,&x);

printf("The %d PopBack delete num = %d\n",i,x);

}

ShowSeqList(&my\_list);

break;

case 3: InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushFront(&my\_list,num);

}

ShowSeqList(&my\_list);

for(i=1;i<=n;i++){

PopFront(&my\_list,&x);

printf("The %d PopBack delete num = %d\n",i,x);

}

ShowSeqList(&my\_list);

break;

case 4: InitSqList(&my\_list);

scanf("%d",&n);

for(i=1;i<=n;i++){

scanf("%d",&num);

PushFront(&my\_list,num);

}

ShowSeqList(&my\_list);

for(i=1;i<=n+2;i++){

PopFront(&my\_list,&x);

printf("The %d PopBack delete num = %d\n",i,x);

}

ShowSeqList(&my\_list);

break;

case 5: InitSqList(&my\_list);

for(i=1;i<=3;i++){

PopFront(&my\_list,&x);

printf("The %d PopBack delete num = %d\n",i,x);

}

ShowSeqList(&my\_list);

break;

}

return 0;

}

【测试用例】

本题主要测试顺序表的头部操作，分以下情况：

1. 初始化顺序表之后，依次在顺序表头部插入若干元素，打印顺序表的长度与元素
2. 初始化顺序表并在顺序表头部插入A个元素，依次删除头部的B个元素（B<A），依次打印这B个元素，打印顺序表的长度与元素（顺序表的长度为|A-B|，顺序表中有前|A-B|元素）。
3. 初始化顺序表并在顺序表头部插入A个元素，依次删除头部的B个元素（B=A），依次打印这B个元素，打印顺序表的长度与元素（顺序表的长度为0，顺序表中没有元素）。
4. 初始化顺序表并做顺序表头部插入操作A次，做顺序表头部删除操作B次，其中B>A，打印顺序表的长度与元素（顺序表的长度为0，顺序表中没有元素）。
5. 初始化顺序表并做顺序表头部删除操作B次，打印顺序表的长度与元素（顺序表的长度为0，顺序表中没有元素）。

1/

输入：第一行1，进行第一项测试

第二行5，准备输入5个数，用头插法插入线性表

第三行，1 2 3 4 5 为从键盘输入的这5个数

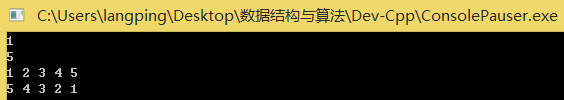
输出：第一行，5 4 3 2 1 为插入完成后打印顺序表的结果

1

5

1 2 3 4 5

5 4 3 2 1



2/

输入：第一行2，进行第二项测试

第二行4，准备输入4个数，用头插法插入线性表

第三行，2 4 6 8 为从键盘输入的这4个数

输出：第一行，8 6 4 2 为插入完成后打印顺序表的结果

第二行The 1 PopBack delete num = 8，意思是第一次对顺序表进行头删除操作的元素值是8

第三行The 2 PopBack delete num = 6，意思是第二次对顺序表进行头删除操作的元素值是6

最后一行4 2表示，所有头删除操作完成后，打印顺序表的结果

2

4

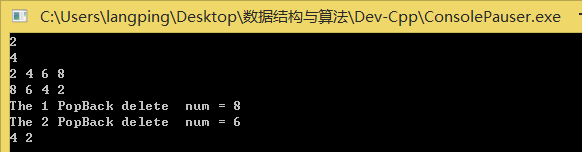
2 4 6 8

8 6 4 2

The 1 PopBack delete num = 8

The 2 PopBack delete num = 6

4 2



3/

输入：第一行3，进行第三项测试

第二行5，准备输入5个数，用头插法插入线性表

第三行，1 3 5 7 9为从键盘输入的这45个数

输出：第一行，9 7 5 3 1为插入完成后打印顺序表的结果

第二行The 1 PopBack delete num = 9，意思是第一次对顺序表进行头删除操作的元素值是9

第二行The 2 PopBack delete num = 7，意思是第二次对顺序表进行头删除操作的元素值是7

……

最后一行There is no element in the sequence table.表示，所有头删除操作完成后，打印顺序表的结果（顺序表已经没有元素）

3

5

1 3 5 7 9

9 7 5 3 1

The 1 PopBack delete num = 9

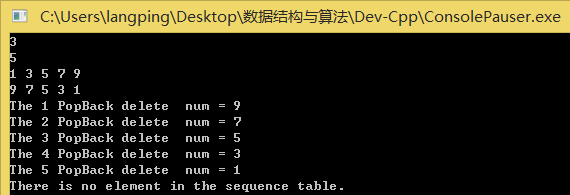
The 2 PopBack delete num = 7

The 3 PopBack delete num = 5

The 4 PopBack delete num = 3

The 5 PopBack delete num = 1

There is no element in the sequence table.



4/

输入：第一行4，进行第四项测试

第二行5，准备输入5个数，用头插法插入线性表

第三行，1 2 3 6 7 为从键盘输入的这5个数

输出：第一行，7 6 2 1为插入完成后打印顺序表的结果

第二行The 1 PopBack delete num = 7，意思是第一次对顺序表进行头删除操作的元素值是7

第二行The 2 PopBack delete num = 6，意思是第二次对顺序表进行头删除操作的元素值是6

……

第七行There is no element in the sequence table that can be deleted.（顺序表无元素可删除）

第八行The 4 PopBack delete num = -12345（顺序表无元素可删除时x变量为一特殊值）

……

最后一行There is no element in the sequence table.表示，所有头删除操作完成后，打印顺序表的结果（顺序表已经没有元素）

4

5

1 2 3 6 7

7 6 3 2 1

The 1 PopBack delete num = 7

The 2 PopBack delete num = 6

The 3 PopBack delete num = 3

The 4 PopBack delete num = 2

The 5 PopBack delete num = 1

There is no element in the sequence table that can be deleted.

The 6 PopBack delete num = -12345

There is no element in the sequence table that can be deleted.

The 7 PopBack delete num = -12345

There is no element in the sequence table.



5/

输入：第一行5，进行第五项测试

输出：第一行，There is no element in the sequence table that can be deleted.（顺序表无元素可删除，因为没有进行插入操作）

第二行The 4 PopBack delete num = -12345（顺序表无元素可删除时x变量为一特殊值）

……

最后一行There is no element in the sequence table.表示，所有头删除操作完成后，打印顺序表的结果（顺序表已经没有元素）

5

There is no element in the sequence table that can be deleted.

The 1 PopBack delete num = -12345

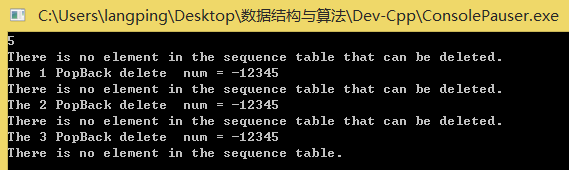
There is no element in the sequence table that can be deleted.

The 2 PopBack delete num = -12345

There is no element in the sequence table that can be deleted.

The 3 PopBack delete num = -12345

There is no element in the sequence table.



**1.3顺序表（指定位置操作）**

本题主要考查内容：

1. 在顺序表的位置i插入值为x的元素
2. 删除顺序表的位置i元素，其值存入指针x所指单元
3. 修改顺序表的位置i元素之值为x
4. 获取顺序表的i位置元素之值,其值存入指针x所指单元

【程序片段】

#include <stdio.h>

#define MaxSize 100

typedef int ElemType;

typedef struct sqlist{

ElemType list[MaxSize];

int size;

}SqList;

void InitSqList(SqList \*L){//1初始化顺序表

L->size=0;

}

int ListLength(SqList L){//2求顺序表长度

return L.size;

}

//--------顺序表的指定位置操作---------

int ListInsert(SqList \*L,int i,ElemType x){//7在顺序表的位置i插入值为x的元素

int j;

if(L->size>=MaxSize){

printf("list is full.\n");return 0;

}

if(i<0||i>L->size){

printf("Parameter illegal\n");return 0;

}

//1

for(【1.1】){

【1.2】;

}

//2

L->list[i]=x;

【2】;

return 1;

}

int ListDelete(SqList \*L,int i,ElemType \*x){ //8删除顺序表的位置i元素，其值存入指针x所指单元

int j;

if(L->size<=0){

printf("There is no element in the sequence table that can be deleted.\n");

\*x=-12345;

return 0;

}

if(i<0||i>L->size-1){

printf("Parameter illegal.\n");\*x=-12345;return 0;

}

\*x=【3】;//3

//4

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

【4】;

}

【5】; //5

return 1;

}

int ListUpdate(SqList \*L,int i,ElemType x){//9修改顺序表的位置i元素之值为x

if(i<0||i>L->size-1){

printf("Parameter illegal.\n");return 0;

}

【6】;// 6

return 1;

}

int ListGet(SqList L,int i,ElemType \*x){////10获取顺序表的i位置元素之值,其值存入指针x所指单元

if(i<0||i>L.size-1){

printf("Parameter illegal.\n");\*x=-12345;return 0;

}

【7】;//

return 1;

}

int ShowSeqList(SqList \* L){//11打印顺序表中所有元素

int i;

if(L->size==0){

printf("There is no element in the sequence table.\n");return 0;

}

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

printf("%d ", L->list[i]);

printf("\n");

return 1;

}

int main(){

SqList my\_list ;

int x,n,i,num;

int switch\_num;

scanf("%d",&switch\_num);

switch(switch\_num){

case 1: InitSqList(&my\_list);

scanf("%d",&x);

ListInsert(&my\_list,0,x);//第一次头插入

scanf("%d",&x);

ListInsert(&my\_list,0,x);//第二次头插入

scanf("%d",&x);

ListInsert(&my\_list,0,x);//第三次头插入

scanf("%d",&x);

ShowSeqList(&my\_list);

ListInsert(&my\_list,3,x);//第一次尾插入 ，第二个参数为什么是3？

scanf("%d",&x);

ListInsert(&my\_list,4,x);//第二次尾插入 ，第二个参数为什么是4？

ShowSeqList(&my\_list);

scanf("%d",&x);

ListInsert(&my\_list,2,x);// 在位置2插入

ShowSeqList(&my\_list);

scanf("%d",&x);

ListInsert(&my\_list,3,x);// 在位置3插入

ShowSeqList(&my\_list);

scanf("%d",&x);

ListInsert(&my\_list,5,x);// 在位置5插入

ShowSeqList(&my\_list);

break;

case 2: InitSqList(&my\_list);

for(i=1;i<=10;i++)

ListInsert(&my\_list,0,i);

ShowSeqList(&my\_list);

ListInsert(&my\_list,-1,100);//第一个非法位置，小于0位置

ListInsert(&my\_list,my\_list.size+1,100);//第二个非法位置，大于尾位置 ,为什么最小是

//ShowSeqList(&my\_list); // my\_list.size+1？

break;

case 3: InitSqList(&my\_list);

for(i=1;i<=10;i++)

ListInsert(&my\_list,0,i);

ShowSeqList(&my\_list);

ListDelete(&my\_list,0,&x);

printf("The 1st PopBack delete num = %d . \n",x);

ListDelete(&my\_list,0,&x);

printf("The 2nd PopBack delete num = %d . \n",x);

ListDelete(&my\_list,0,&x);

printf("The 3rd PopBack delete num = %d . \n",x);

ShowSeqList(&my\_list);//3次头删除后的结果

ListDelete(&my\_list,6,&x);

printf("The 1st PopBack delete num = %d . \n",x);

ListDelete(&my\_list,5,&x);

printf("The 2rd PopBack delete num = %d . \n",x);

ShowSeqList(&my\_list);//2次尾删除后的结果

ListDelete(&my\_list,2,&x);

printf("The 1st PopBack delete num = %d . \n",x);

ListDelete(&my\_list,3,&x);

printf("The 2rd PopBack delete num = %d . \n",x);

ShowSeqList(&my\_list);//2次指定位置删除后的结果

ListDelete(&my\_list,-1,&x);//第一个非法位置，小于0位置

ListDelete(&my\_list,my\_list.size,&x);//第二个非法位置，大于尾位置

ShowSeqList(&my\_list);

break;

case 4: InitSqList(&my\_list);

for(i=1;i<=10;i++)

ListInsert(&my\_list,0,i);

ShowSeqList(&my\_list);

ListUpdate(&my\_list,0,100);

ListUpdate(&my\_list,3,1000);

ListUpdate(&my\_list,my\_list.size-1,10000);

ShowSeqList(&my\_list);

ListUpdate(&my\_list,-1,100000);

ListUpdate(&my\_list,my\_list.size,100000);

break;

case 5: InitSqList(&my\_list);

for(i=1;i<=10;i++)

ListInsert(&my\_list,0,i);

ShowSeqList(&my\_list);

ListGet(my\_list,0,&x);

printf("The 0 num = %d . \n",x);

ListGet(my\_list,3,&x);

printf("The 3 num = %d . \n",x);

ListGet(my\_list,my\_list.size-1,&x);

printf("The last num = %d . \n",x);

ListGet(my\_list,-1,&x);

ListGet(my\_list,my\_list.size,&x);

break;

}

return 0;

}

【测试用例】

本题主要测试顺序表的指定位置操作，分以下情况：

1）插入操作一：初始化顺序表后，在指定位置i插入值为data的元素，首先连续在顺序表的0位置插入3个元素（相当于头插入法）；然后连续在顺序表的最后一个元素之后插入2个元素（相当于尾插入）；最后在顺序表的2，3，5插入元素（指定位置）。

2）插入操作二：初始化顺序表后，首先连续在顺序表的0位置插入5个元素建立顺序表（相当于头插入法）；然后测试插入位置非法的情况（-1位置，以及大于顺序表长度的位置）。

3）删除操作：初始化顺序表后，首先连续在顺序表的0位置插入10个元素建立顺序表（相当于头插入法）；然后连续2次删除0位置的元素（相当于头删除）；连续3次删除尾位置的元素（相当于尾删除）删除指定位置的元素（2，3）；最后测试删除位置非法的情况（-1位置，以及大于顺序表长度的位置）。

4）修改操作：初始化顺序表后，首先连续在顺序表的0位置插入10个元素建立顺序表（相当于头插入法）；依次修改0，3，最后一个元素的值；最后测试修改位置非法的情况（-1位置，以及大于顺序表长度的位置）。

5）获取操作：初始化顺序表后，首先连续在顺序表的0位置插入10个元素建立顺序表（相当于头插入法）；依次获取0，3，最后一个元素的值；最后测试获取位置非法的情况（-1位置，以及大于顺序表长度的位置）。

1/

输入：第一行1，进行第一项测试

第二行1 2 3 4 5 6 7 8，输入8个数，准备指定位置插入顺序表

输出：第一行，3 2 1 为依次在头部插入1 2 3后打印顺序表的结果

第二行，3 2 1 4 5为上述操作基础上，在顺序表尾部依次插入4 5后打印顺序表的结果

第三行，3 2 6 1 4 5为上述操作基础上，在顺序表2位置插入6后打印顺序表的结果

第四行，3 2 6 7 1 4 5为上述操作基础上，在顺序表3位置插入7后打印顺序表的结果

第五行，3 2 6 1 8 4 5为上述操作基础上，在顺序表5位置插入8后打印顺序表的结果

1

1 2 3 4 5 6 7 8

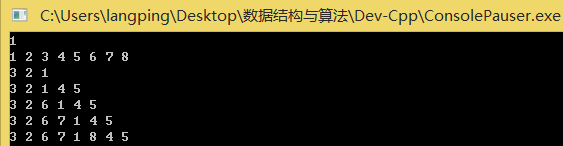
3 2 1

3 2 1 4 5

3 2 6 1 4 5

3 2 6 7 1 4 5

3 2 6 7 1 8 4 5



2/

输入：第一行2，进行第二项测试

输出：第一行，10 9 8 7 6 5 4 3 2 1为依次在头部插入1 2 3 4 5 6 7 8 9 10后打印顺序表的结果

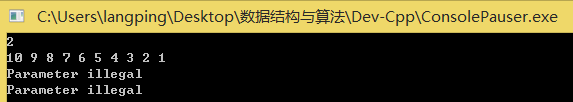
第二、三行，插入非法位置元素出错提示（-1、尾元素+1）

2

10 9 8 7 6 5 4 3 2 1

Parameter illegal

Parameter illegal



3/

输入：第一行3，进行第三项测试

输出：第一行，10 9 8 7 6 5 4 3 2 1为依次在头部插入1 2 34 5 6 7 8 9 10后打印顺序表的结果

第二行至第四行，连续对顺序表做3次0位置的删除操作，删除的值分别为10，9，8

第五行，7 6 5 4 3 2 1为上述操作基础后，打印顺序表的结果

第六，七行，上述操作基础上，连续对顺序表做2次尾位置的删除操作，删除的值分别为1，2

第八行，7 6 5 4 3为上述操作基础后，打印顺序表的结果

第九，十行，上述操作基础上，连续对顺序表做2，3位置的删除操作，删除的值分别为5，3

第十一行，7 6 4为上述操作基础后，打印顺序表的结果

第十二/十三行，连续2次对顺序表做非法位置的删除操作（-1、尾元素+1）

第十四行，7 6 4为上述操作基础后，打印顺序表的结果

3

10 9 8 7 6 5 4 3 2 1

The 1st PopBack delete num = 10 .

The 2nd PopBack delete num = 9 .

The 3rd PopBack delete num = 8 .

7 6 5 4 3 2 1

The 1st PopBack delete num = 1 .

The 2rd PopBack delete num = 2 .

7 6 5 4 3

The 1st PopBack delete num = 5 .

The 2rd PopBack delete num = 3 .

7 6 4

Parameter illegal.

Parameter illegal.

7 6 4



4/

输入：第一行4，进行第四项测试

输出：第一行，10 9 8 7 6 5 4 3 2 1为依次在头部插入1 2 34 5 6 7 8 9 10后打印顺序表的结果

第二行至第四行，修改顺序表做0，3，尾位置元素的值为100，1000，10000

第三、四行，修改非法位置元素出错提示（-1、尾元素+1）

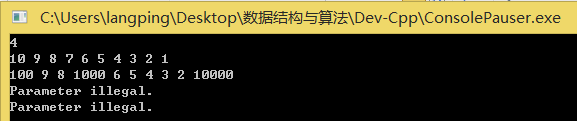
4

10 9 8 7 6 5 4 3 2 1

100 9 8 1000 6 5 4 3 2 10000

Parameter illegal.

Parameter illegal.



5/

输入：第一行5，进行第五项测试

输出：第一行，10 9 8 7 6 5 4 3 2 1为依次在头部插入1 2 34 5 6 7 8 9 10后打印顺序表的结果

第二行至第四行，获取顺序表做0，3，尾位置元素的值为10，7，1

第五，六行，获取非法位置出错提示（-1、尾元素+1）

5

10 9 8 7 6 5 4 3 2 1

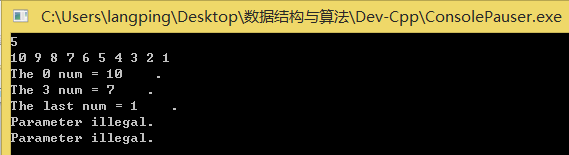
The 0 num = 10 .

The 3 num = 7 .

The last num = 1 .

Parameter illegal.

Parameter illegal.



**1.4顺序表（查找与删除的几种应用）**

本题主要考查内容：

1. 查找顺序表中值为x的的第一个元素，找到则返回该元素的位置，否则返回-1
2. 删除顺序表中第一个值为data的元素
3. 删除顺序表中所有值为data的元素
4. 使用二分查找降序排列顺序表中的元素

【程序片段】

#include <stdio.h>

#define MaxSize 100

typedef int ElemType;

typedef struct sqlist{

ElemType list[MaxSize];

int size;

}SqList;

void InitSqList(SqList \*L){//1初始化顺序表

L->size=0;

}

int ListLength(SqList L){//2求顺序表长度

return L.size;

}

int PushFront(SqList \*L,ElemType x){//5在顺序表的头部插入值为data的元素

int j;

if(L->size>=MaxSize){

printf("list is full.\n");return 0;

}

for(j=L->size;j>=1;j--){

【1】;

}

L->list[0]=【2】;

L->size++;

return 1;

}

int ListUpdate(SqList \*L,int i,ElemType x){//9修改顺序表的位置i元素之值为x

if(i<0||i>L->size-1){

printf("Parameter illegal.\n");return 0;

}

【3】x;

return 1;

}

int ShowSeqList(SqList \* L){//11打印顺序表中所有元素

int i;

if(L->size==0){

printf("There is no element in the sequence table.\n");return 0;

}

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

printf("%d ", L->list[i]);

printf("\n");

return 1;

}

int ListSearch(SqList L,ElemType x){//12查找顺序表中值为x的的第一个元素，找到则返回该元素的位置，否则返回-1

int j;

for(j=0;j<L.size;j++){

if(L.list[j]==x){

return j;

}

}

【4】;

}

int ListDelete(SqList \*L,int i,ElemType \*x){ //8删除顺序表的位置i元素，其值存入指针x所指单元

int j;

if(L->size<=0){

printf("There is no element in the sequence table that can be deleted.\n");

return 0;

}

if(i<0||i>L->size){

printf("Parameter illegal.\n");return 0;

}

\*x=L->list[i] ;

for(【5】){

L->list[j-1]=L->list[j];

}

L->size--;

return 1;

}

int Remove(SqList \*L, ElemType data){//13删除顺序表中第一个值为data的元素,方法一

int i,x;

i=ListSearch(\*L,data);

if(i==-1){

printf("The deleted element was not found.\n");return -1;

}

ListDelete(【6】);

}

int Remove1(SqList \*L, ElemType data){//13删除顺序表中第一个值为data的元素，方法二

int i,j,x;

for(j=0;j<L->size;j++){

if(L->list[j]==data){

break;

}

}

i=j;

if(i>=L->size){

printf("The deleted element was not found.\n");

return -1;

}

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

【7】=L->list[j];

}

L->size--;

return 1;

}

int Remove\_all(SqList \*L, ElemType data){//14删除顺序表中所有值为data的元素

int i,x;

int k;

for(k=0;k<L->size; k++) {

i=ListSearch(【8】);

if(i!=-1){

ListDelete(L,i,&x);

k--;

}

}

}

int BinarySearch(SqList \*L, ElemType data){//15使用二分查找降序排列中的元素

int left = 0, right=L->size-1,mid;

while(left<=right){

mid = (left + right) / 2;

if (【9】)

left = mid + 1;

else if (【10】)

right = mid - 1;

else if (data==L->list[mid])

return mid;

}

return -1;

}

int main(){

SqList my\_list ;

int x,n,i,num;

int switch\_num;

scanf("%d",&switch\_num);

InitSqList(&my\_list);

for(i=1;i<=18;i++)

PushFront(&my\_list,i);

ListUpdate(&my\_list,17,-9); ListUpdate(&my\_list,15,10);

ListUpdate(&my\_list,13,-1); ListUpdate(&my\_list,11,10);

ListUpdate(&my\_list,9,-2); ListUpdate(&my\_list,7,10);

ListUpdate(&my\_list,5,-10); ListUpdate(&my\_list,3,21);

ListUpdate(&my\_list,1,100); ListUpdate(&my\_list,0,30);

switch(switch\_num){

case 1: ShowSeqList(&my\_list);

i=ListSearch(my\_list,10);

if(i>=0)

printf("The first one is equal to 10 at %d\n",i);

else

printf("The element 10 was not found.\n");

i=ListSearch(my\_list,30);

if(i>=0)

printf("The first one is equal to 30 at %d\n",i);

else

printf("The element 30 was not found.\n");

i=ListSearch(my\_list,-9);

if(i>=0)

printf("The first one is equal to -9 at %d\n",i);

else

printf("The element -9 was not found.\n");

i=ListSearch(my\_list,-100);

if(i>=0)

printf("The first one is equal to -100 at %d\n",i);

else

printf("The element -100 was not found.\n");

i=ListSearch(my\_list,200);

if(i>=0)

printf("The first one is equal to 200 at %d\n",i);

else

printf("The element 200 was not found.\n");

break;

case 2: ShowSeqList(&my\_list);

Remove(&my\_list,30 );ShowSeqList(&my\_list);

Remove(&my\_list,12 );ShowSeqList(&my\_list);

Remove(&my\_list,10 );ShowSeqList(&my\_list);

Remove(&my\_list,-30);ShowSeqList(&my\_list);

break;

case 3: ShowSeqList(&my\_list);

Remove1(&my\_list,30 );ShowSeqList(&my\_list);

Remove1(&my\_list,12 );ShowSeqList(&my\_list);

Remove1(&my\_list,10 );ShowSeqList(&my\_list);

Remove1(&my\_list,-30);ShowSeqList(&my\_list);

break;

case 4: ShowSeqList(&my\_list);

Remove\_all(&my\_list,30);ShowSeqList(&my\_list);

Remove\_all(&my\_list,10 );ShowSeqList(&my\_list);

//Remove\_all(&my\_list,-30);ShowSeqList(&my\_list);

break;

case 5: InitSqList(&my\_list);

for(i=1;i<=18;i++)

PushFront(&my\_list,i);

ShowSeqList(&my\_list);

i=BinarySearch(&my\_list,10);

if(i>=0)

printf("The first one is equal to 10 at %d\n",i);

else

printf("The element 10 was not found.\n");

i=BinarySearch(&my\_list,18);

if(i>=0)

printf("The first one is equal to 18 at %d\n",i);

else

printf("The element 18 was not found.\n");

i=BinarySearch(&my\_list,1);

if(i>=0)

printf("The first one is equal to 1 at %d\n",i);

else

printf("The element 1 was not found.\n");

i=BinarySearch(&my\_list,-100);

if(i>=0)

printf("The first one is equal to -100 at %d\n",i);

else

printf("The element -100 was not found.\n");

i=BinarySearch(&my\_list,200);

if(i>=0)

printf("The first one is equal to 200 at %d\n",i);

else

printf("The element 200 was not found.\n");

break;

}

return 0;

}

【测试用例】

本题主要测试顺序表的头部操作，分以下情况：

1. 查找顺序表（未排序）中值为x的的第一个元素，找到则返回该元素的位置，否则返回-1
2. 删除顺序表中第一个值为data的元素,方法一
3. 删除顺序表中第一个值为data的元素,方法二
4. 删除顺序表中所有值为data的元素
5. 使用二分查找降序排列中的元素

1/

输入：第一行1，进行第一项测试

输出：第一行30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9为顺序表各元素

第二行The first one is equal to 10 at 7指顺序表中第一个值为10的位置是7

第三行The first one is equal to 30 at 0指顺序表中第一个值为30的位置是0

第四行The first one is equal to -9 at 17指顺序表中第一个值为-9的位置是17

第五行The element -100 was not found.指顺序表中没找到元素-100

第六行The element 200 was not found.指顺序表中没找到元素200

1

30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9

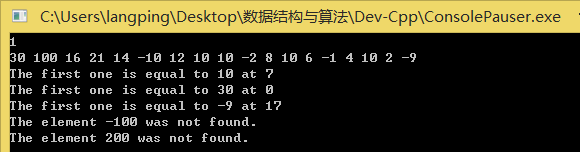
The first one is equal to 10 at 7

The first one is equal to 30 at 0

The first one is equal to -9 at 17

The element -100 was not found.

The element 200 was not found.



2/

输入：第一行2，进行第二项测试

输出：第一行30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9为执行删除操作前的顺序表

第二行100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9为删除顺序表中第一个值为30的元素后

第三行100 16 21 14 -10 10 10 -2 8 10 6 -1 4 10 2 -9为执行上述操作后又删除顺序表中第一个值为12的元素

第四行100 16 21 14 -10 10 -2 8 10 6 -1 4 10 2 -9为执行上述操作后又删除顺序表中第一个值为10的元素

第五行The deleted element was not found. 为执行上述操作后又删除顺序表中第一个值为-30的元素，因为顺序表不存在-30的元素给出的错误提示。

第六行为最后顺序表里的各元素。

2

30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9

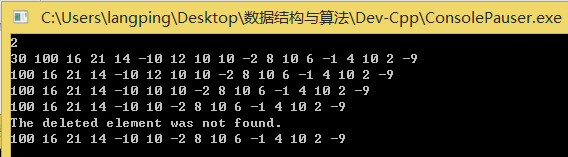
100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9

100 16 21 14 -10 10 10 -2 8 10 6 -1 4 10 2 -9

100 16 21 14 -10 10 -2 8 10 6 -1 4 10 2 -9

The deleted element was not found.

100 16 21 14 -10 10 -2 8 10 6 -1 4 10 2 -9



3/

输入：第一行3，进行第三项测试

输出：第一行30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9为执行删除操作前的顺序表

第二行100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9为删除顺序表中第一个值为30的元素后

第三行100 16 21 14 -10 10 10 -2 8 10 6 -1 4 10 2 -9为执行上述操作后又删除顺序表中第一个值为12的元素

第四行100 16 21 14 -10 10 -2 8 10 6 -1 4 10 2 -9为执行上述操作后又删除顺序表中第一个值为10的元素

第五行The deleted element was not found. 为执行上述操作后又删除顺序表中第一个值为-30的元素，因为顺序表不存在-30的元素给出的错误提示。

第六行为最后顺序表里的各元素。

3

30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9

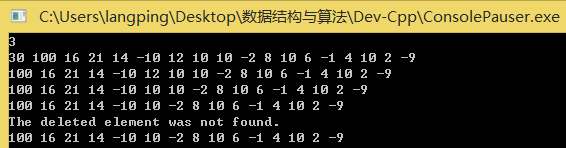
100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9

100 16 21 14 -10 10 10 -2 8 10 6 -1 4 10 2 -9

100 16 21 14 -10 10 -2 8 10 6 -1 4 10 2 -9

The deleted element was not found.

100 16 21 14 -10 10 -2 8 10 6 -1 4 10 2 -9



4/

输入：第一行4，进行第四测试

输出：第一行30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9为执行删除操作前的顺序表

第二行100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9为删除顺序表中所有值为30的元素后

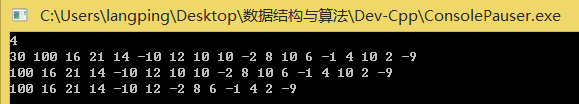
第三行00 16 21 14 -10 12 -2 8 6 -1 4 2 -9为删除顺序表中所有值为30的元素后

4

30 100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9

100 16 21 14 -10 12 10 10 -2 8 10 6 -1 4 10 2 -9

100 16 21 14 -10 12 -2 8 6 -1 4 2 -9



5/

输入：第一行5，进行第五项测试

输出：第一行18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1为顺序表各元素

第二行The first one is equal to 10 at 8指顺序表中第一个值为10的位置是8

第三行The first one is equal to 18 at 0指顺序表中第一个值为18的位置是0

第四行The first one is equal to 1 at 17指顺序表中第一个值为1的位置是17

第五行The element -100 was not found.指顺序表中没找到元素-100

第六行The element 200 was not found.指顺序表中没找到元素200

5

18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

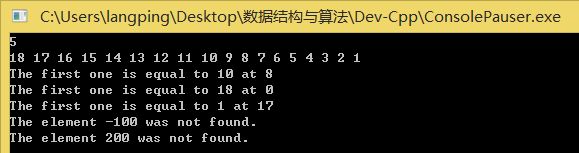
The first one is equal to 10 at 8

The first one is equal to 18 at 0

The first one is equal to 1 at 17

The element -100 was not found.

The element 200 was not found.



【程序片段】

**1.5顺序表（几种排序方法）**

本题主要考查内容：

1. 使用冒泡排序给顺序表中的元素排序
2. 使用选择排序给顺序表中的元素排序
3. 使用插入排序给顺序表中的元素排序（直接插入）
4. 使用插入排序给顺序表中的元素排序（二分插入）

【程序片段】

#include <stdio.h>

//#include <stdlib.h>

#define MaxSize 100

typedef int ElemType;

typedef struct sqlist{

ElemType list[MaxSize];

int size;

}SqList;

//1初始化顺序表

void InitSqList(SqList \*L){

L->size=0;

}

//2求顺序表长度

int ListLength(SqList L){

return L.size;

}

int PushFront(SqList \*L,ElemType x){//5在顺序表的头部插入值为data的元素

int j;

if(L->size>=MaxSize){

printf("list is full.\n");return 0;

}

for(j=L->size;j>=1;j--){

L->list[j]=L->list[j-1];

}

L->list[0]=x;

L->size++;

return 1;

}

int ListUpdate(SqList \*L,int i,ElemType x){//9修改顺序表的位置i元素之值为x

if(i<0||i>L->size-1){

printf("Parameter illegal.\n");return 0;

}

L->list[i]=x;

return 1;

}

int ShowSeqList(SqList \* L){//11打印顺序表中所有元素

int i;

if(L->size==0){

printf("There is no element in the sequence table.\n");return 0;

}

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

printf("%d ", L->list[i]);

printf("\n");

return 1;

}

void BubbleSort(SqList\* L){//16使用冒泡排序给顺序表中的元素排序

int i , j , temp ; int flag = 1;

for (i =1; i < L->size && flag==1; i++){

flag = 0;

for (j = 0; 【1】; j++){

if (L->list[j] > L->list[j+1]){

temp = L->list[j];

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

L->list[j+1] = temp;

【2】;

}

}

}

}

void SelectSort(SqList\* L){//17使用选择排序给顺序表中的元素排序

int i , j , min , temp ;

for(i=0;i<L->size-1;i++){

min = i ;

For【3】)

if(【4】)

min = j;

if(min!=i){

temp = L->list[i];

L->list[i]=L->list[min];

L->list[min]=temp;

}

}

}

void InsertSort(SqList\* L){//18直接插入排序：将第一个数据看做一个顺序表，将后面的数据一次插入表中

int i,j,x;

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

if(L->list[i] < L->list[i-1]){//若第i个元素大于i-1元素，直接插入。小于的话，移动有序表后插入

j= i-1; //表中最后一个数据

x = L->list[i]; //复制为哨兵，即存储待排序元素

L->list[i] = L->list[i-1]; //先后移一个元素 (因为a[i]就是X，所以不怕丢失)

while(j>=0 && x < L->list[j]){ //查找在有序表的插入位置 (遍历表)

L->list[j+1] =【5】;

【6】; //元素后移

}

L->list[j+1] = x; //插入到正确位置

}

}

}

void BInsertSort(SqList\* L) { //折半插入

int i,j,low,high,x,m;

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

low=0,high=i;

if(L->list[i] < L->list[i-1]){//若第i个元素大于i-1元素，直接插入。小于的话，移动有序表后插入

x = L->list[i]; //复制为哨兵，即存储待排序元素

L->list[i] = L->list[i-1]; //先后移一个元素 (因为a[i]就是X，所以不怕丢失)

while(low<=high){ //查找在有序表的插入位置 (遍历表)

m=【7】;

if(x<L->list[m]) high=m-1;

else low=m+1;

}

for(j=i-1;j>=high+1;j--)

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

L->list[【8】] = x; //插入到正确位置

}

}

}

void DecendSort(SqList\* L){//16使用冒泡排序给顺序表中的元素排序

【9】

}

//--------主函数--------

int main(){

SqList my\_list ;

int x,n,i,num;

int switch\_num;

scanf("%d",&switch\_num);

InitSqList(&my\_list);

for(i=1;i<=18;i++)

PushFront(&my\_list,i);

ListUpdate(&my\_list,17,-9);

ListUpdate(&my\_list,15,23);

ListUpdate(&my\_list,13,-1);

ListUpdate(&my\_list,11,-16);

ListUpdate(&my\_list,9,-2);

ListUpdate(&my\_list,7,10);

ListUpdate(&my\_list,5,-10);

ListUpdate(&my\_list,3,21);

ListUpdate(&my\_list,1,100);

ListUpdate(&my\_list,0,30);

ShowSeqList(&my\_list);

switch(switch\_num){

case 1: BubbleSort(&my\_list);ShowSeqList(&my\_list);break;

case 2: SelectSort(&my\_list);ShowSeqList(&my\_list);break;

case 3: InsertSort(&my\_list);ShowSeqList(&my\_list);break;

case 4: BInsertSort(&my\_list);ShowSeqList(&my\_list);break;

case 5: DecendSort(&my\_list);ShowSeqList(&my\_list);break;

}

return 0;

}

【测试用例】

本题主要测试顺序表的排序操作，分以下情况：

1）冒泡排序，将顺序表按升序排列。

2）选择排序，将顺序表按升序排列。

3）直接插入排序，将顺序表按升序排列。

4）折半插入排序，将顺序表按升序排列。

5）任选一种排序方法，将顺序表按降序排列。（函数原型为void DecendSort(SqList\* L)）

1/

输入：第一行1，进行第一项测试

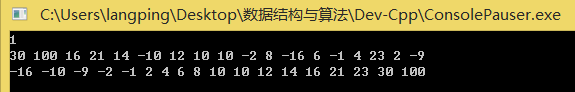
输出：第一行，30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9为排序前顺序表元素。

第二行，-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100为冒泡排序后顺序表元素。

1

30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9

-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100



2/

输入：第一行2，进行第二项测试

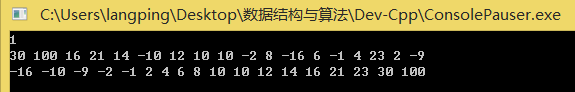
输出：第一行，30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9为排序前顺序表元素。

第二行，-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100为选择排序后顺序表元素。

2

30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9

-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100



3/

输入：第一行3，进行第三项测试

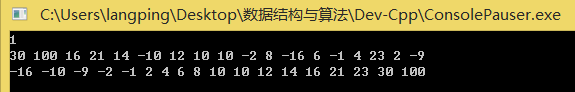
输出：第一行，30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9为排序前顺序表元素。

第二行，-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100为直接插入排序后顺序表元素。

3

30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9

-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100



4/

输入：第一行4，进行第四项测试

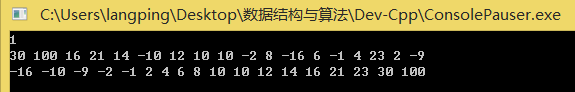
输出：第一行，30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9为排序前顺序表元素。

第二行，-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100为折半插入排序后顺序表元素。

4

30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9

-16 -10 -9 -2 -1 2 4 6 8 10 10 12 14 16 21 23 30 100



5/

输入：第一行5，进行第五项测试

输出：第一行，30 100 16 21 14 -10 12 10 10 -2 8 -16 6 -1 4 23 2 -9为排序前顺序表元素。

第二行，100 30 23 21 16 14 12 10 10 8 6 4 2 -1 -2 -9 -10 -16为按照降序排序后顺序表元素。

