实验题目：

**顺序表的实现**

实验环境：

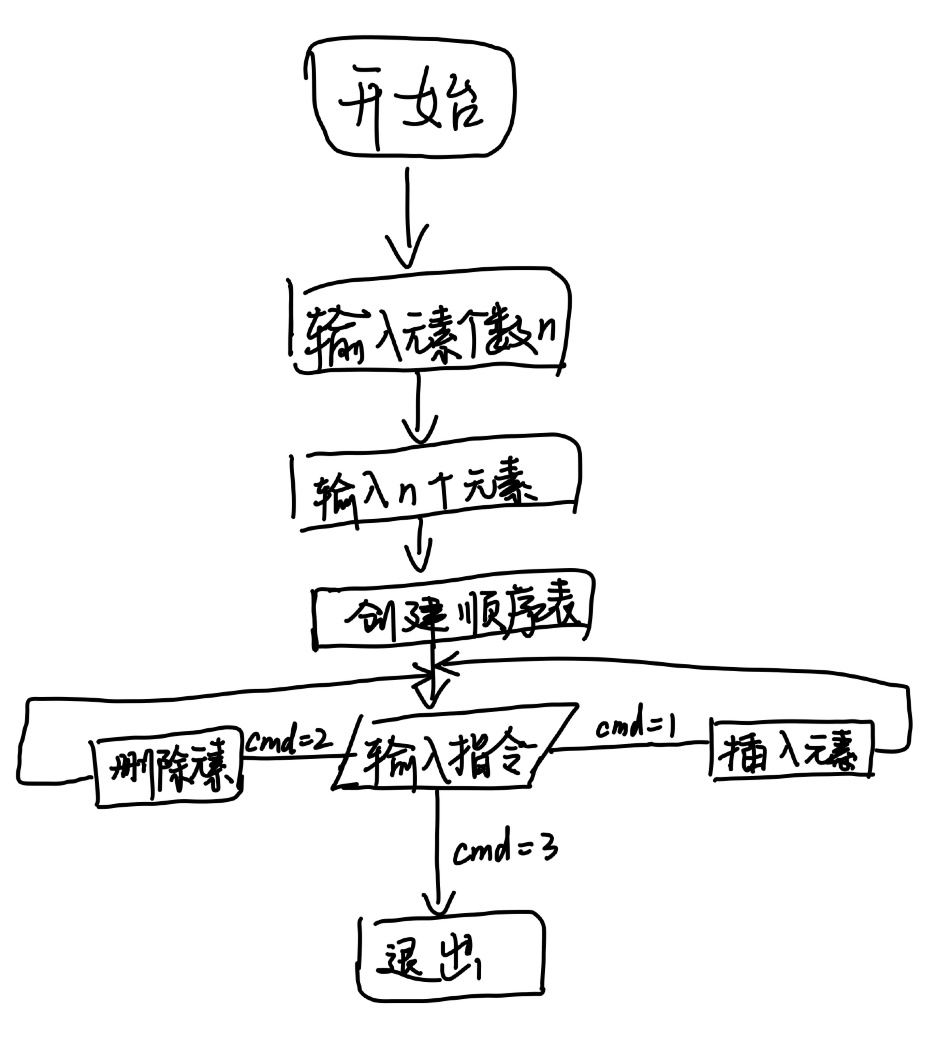
**Win10**

**DevCpp**

实验内容：

**实现顺序表的创建、插入、删除和自动扩容**

****实验流程：****



代码实现：

#include <stdio.h>

#include <stdlib.h>

#include <stdlib.h>

#define MORE 10 //每次扩大的内存

#define ListSize 50 //初始列表大小

typedef struct{

int \*data;//指向表里的数据

int length;

int size;

}List;

void CreateList(List &L,int n){//创建一个有n个元素的列表

if(n>ListSize){

L.data = (int\*)malloc((n+1)\*sizeof(int));

if(!L.data) exit(0);

L.size = n;

}

else{

L.data = (int\*)malloc((ListSize+1)\*sizeof(int));

if(!L.data) exit(0);

L.length = n;

L.size = ListSize;

}

printf("请输入链表中的%d个元素:\n",n);

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

scanf("%d",&L.data[i]);

}

}

void PrintList(List L){//打印列表

printf("这就是你的顺序表:\n");

for(int i =1;i<=L.length;i++){

printf("%d ",L.data[i]);

}

printf("\n");

}

void InsertList(List &L,int m,int k){//在第m个元素的位置插入k

if(L.length+1>L.size){//如果列表已满，扩大列表

int \*NewData;

NewData =(int\*)realloc(L.data,(MORE+L.size)\*sizeof(int));

L.data = NewData;

}

L.length++;

for(int i = L.length;i>m;i--){

L.data[i] = L.data[i-1];

}

L.data[m] = k;

PrintList(L);

}

void DeleteList(List &L,int m){//删除列表中第m个元素

L.length--;

for(int i=m;i<=L.length;i++){

L.data[i] = L.data[i+1];

}

PrintList(L);

}

int main(){

List L;

int n,cmd;

printf("请输入列表中的元素个数：");

scanf("%d",&n);

CreateList(L,n);

printf("\n指令：\n1.插入\n2.删除\n3.退出\n");

printf("你的操作是：\n") ;

scanf("%d",&cmd);

while(cmd!=3){

if(cmd==1) {

int m,k;

printf("您要插入的位置: ");

scanf("%d",&m);

printf("您要插入的元素是:");

scanf("%d",&k);

InsertList(L,m,k);

}

if(cmd==2){

int m;

printf("您要删除的位置是: ");

scanf("%d",&m);

DeleteList(L,m);

}

printf("\n指令：\n1.插入\n2.删除\n3.退出");

printf("你的操作是：") ;

scanf("%d",&cmd);

system("cls");

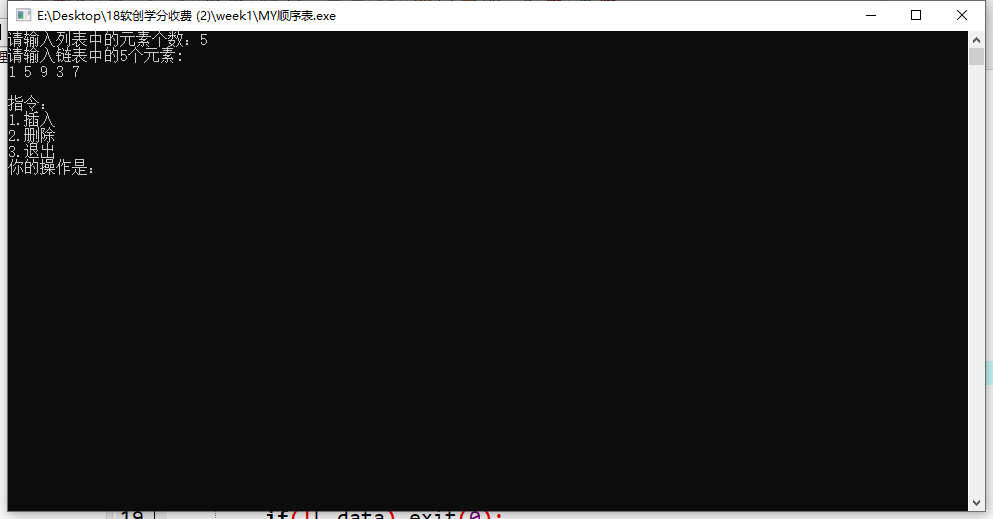
}

return 0;

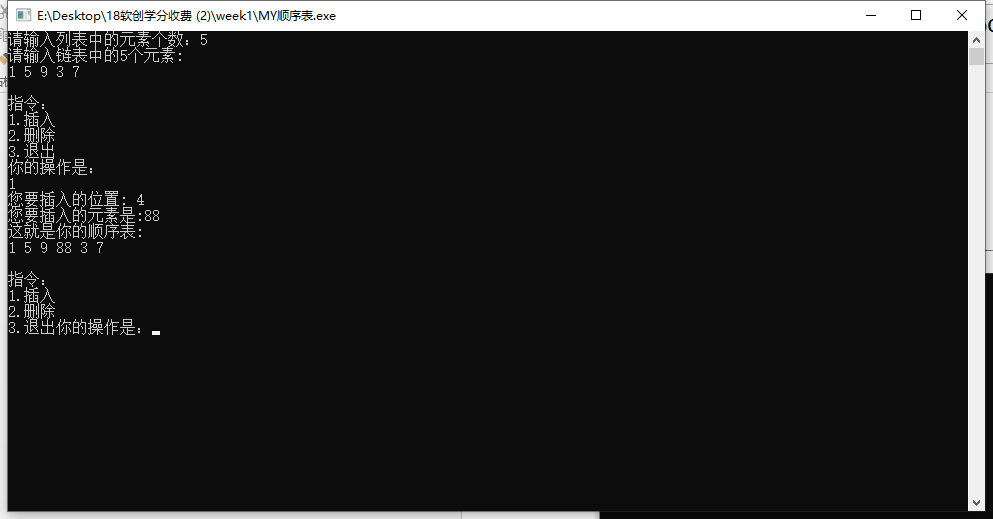
}

实验结果分析：

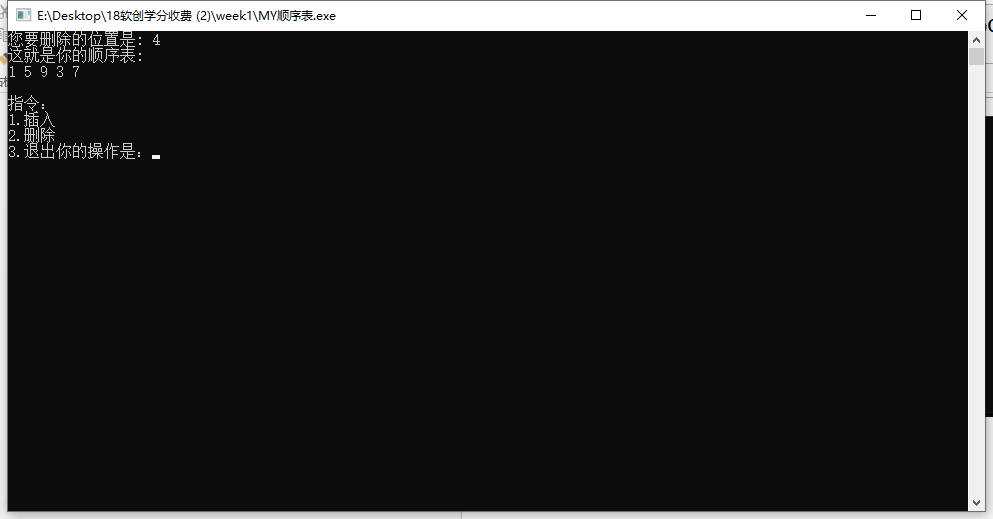
***顺序表的创建：***



顺序表的扩容插入：



***顺序表的删除：***



结果分析：

**程序可以自动根据插入元素数量创建顺序表，然后在插入过程中，如果顺序表已经满了，程序会自动对顺序表进行扩容**

**在删除操作中，程序可以根据输入的位置删除该位置的元素**