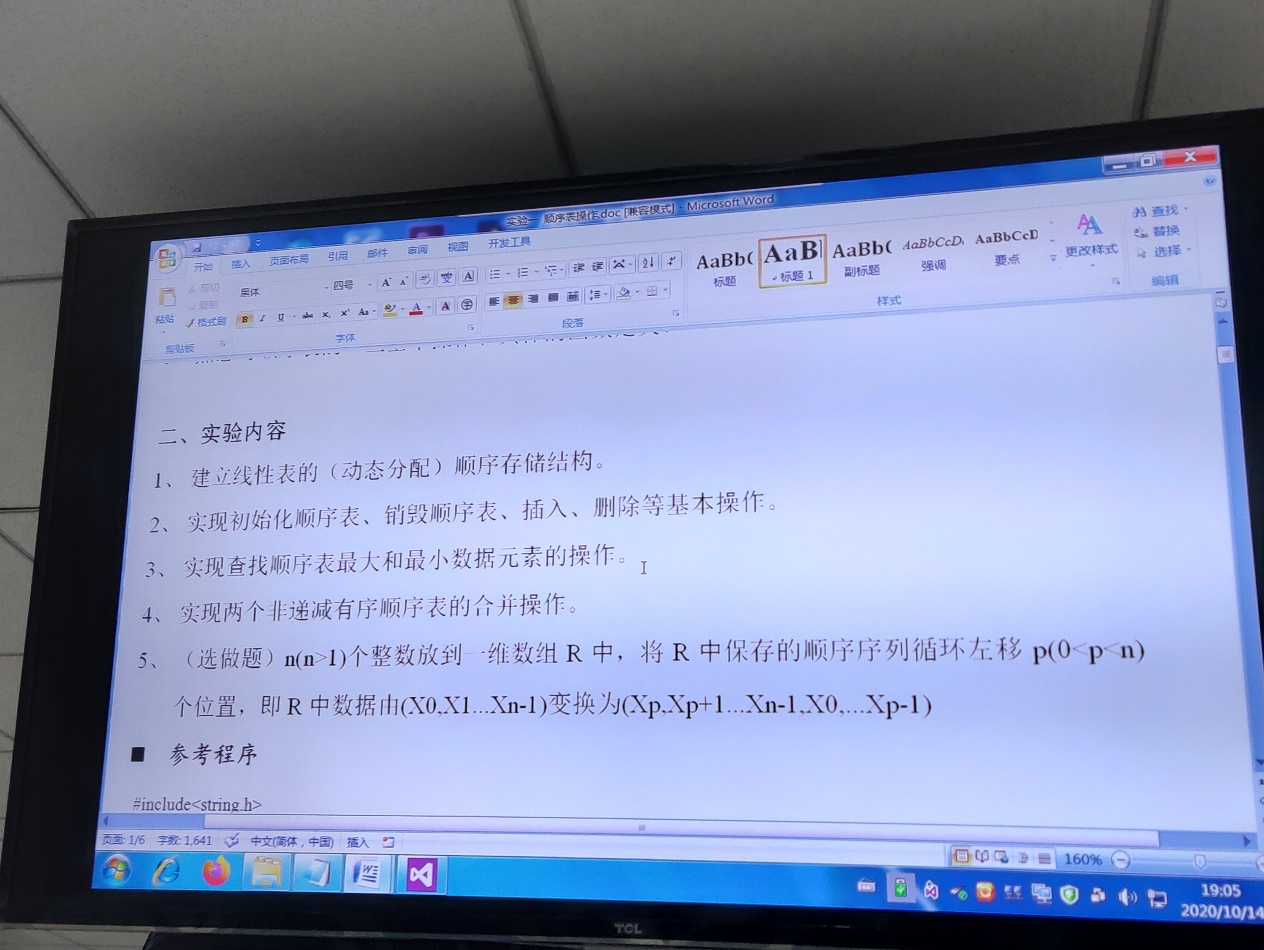
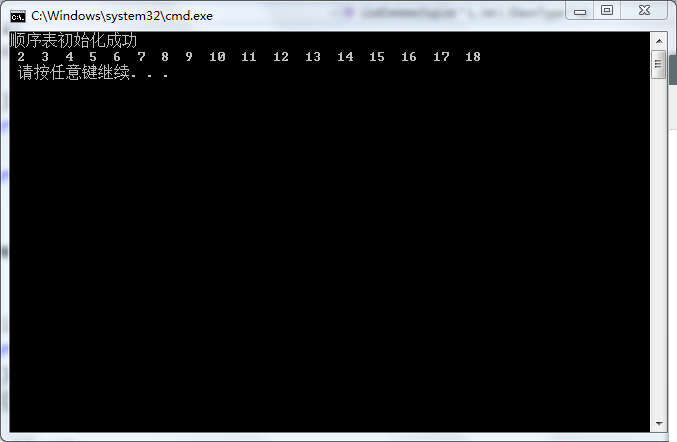
本次上机使用实验主要练习了线性链表的使用，

具体题目如下所示：

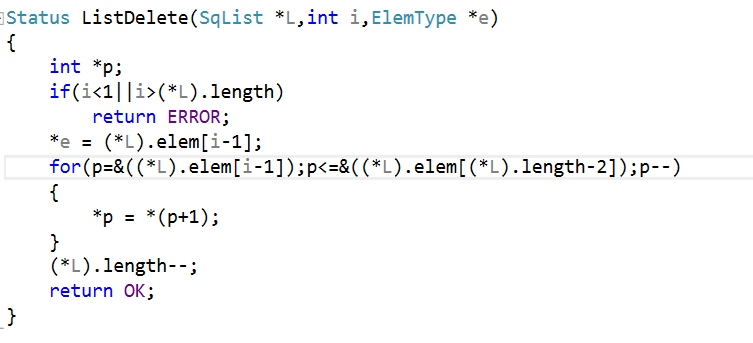


1. 直接使用老师给定的线性链表程序，不用修改。
2. 使用了初始化顺序表
3. 销毁顺序表
4. 删除顺序表

其中在实现将所删除的值输出时遇到无法输出的问题，即使改变printf的输出值为纯文本，依然无法输出。

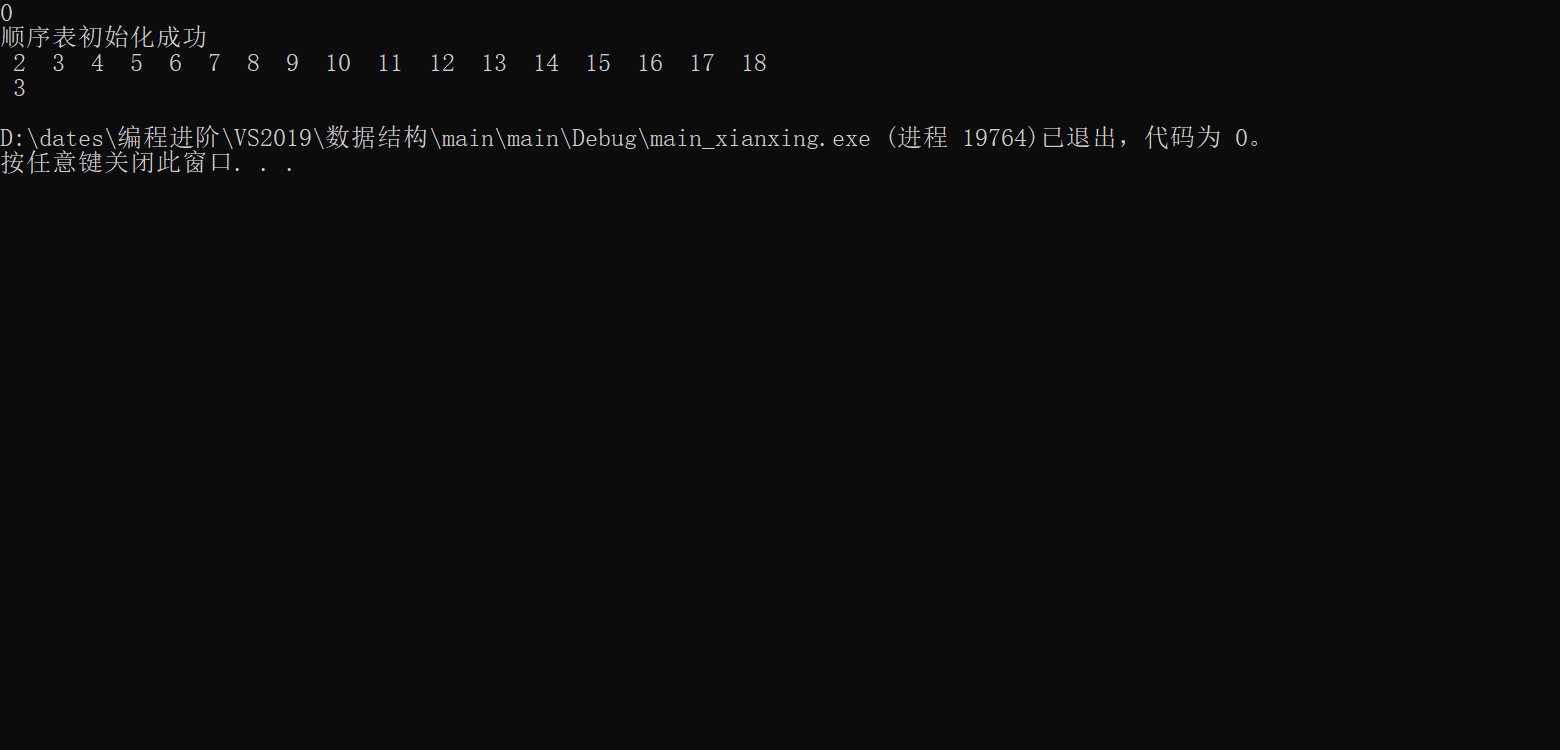


最终通过老师的帮忙调试，发现了问题所在，在**ListDelete**函数中，p--使用不当，应该改为P++；因为删除当前元素后，之后的所有元素指针都要后移一位。

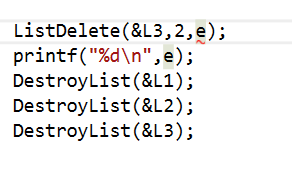


修改后运行，

结果如下：

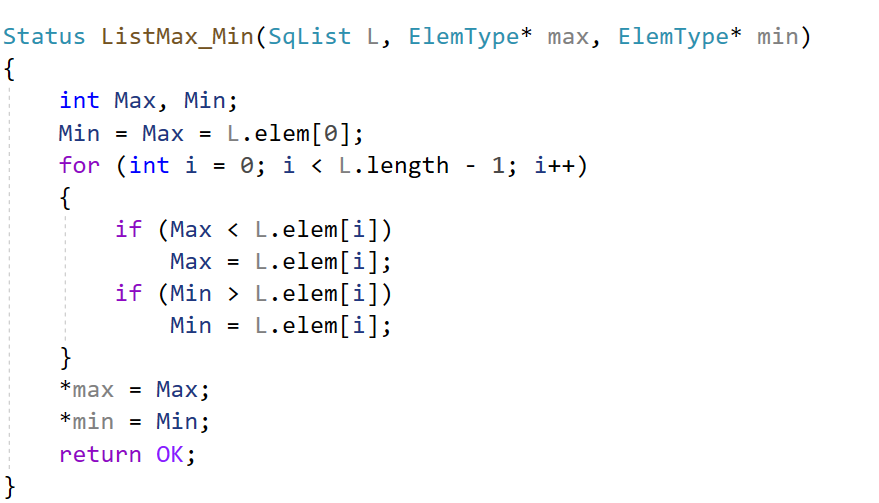


另外：需注意函数ListDelete(SqList\* L, int i, ElemType\* e)的参数，\*e只起储存值的作用。

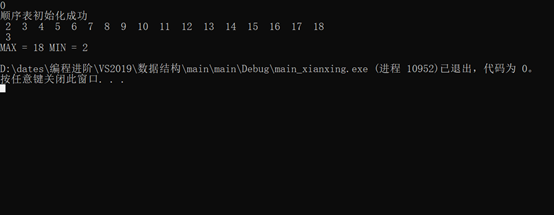


1. 寻找最大值和最小值



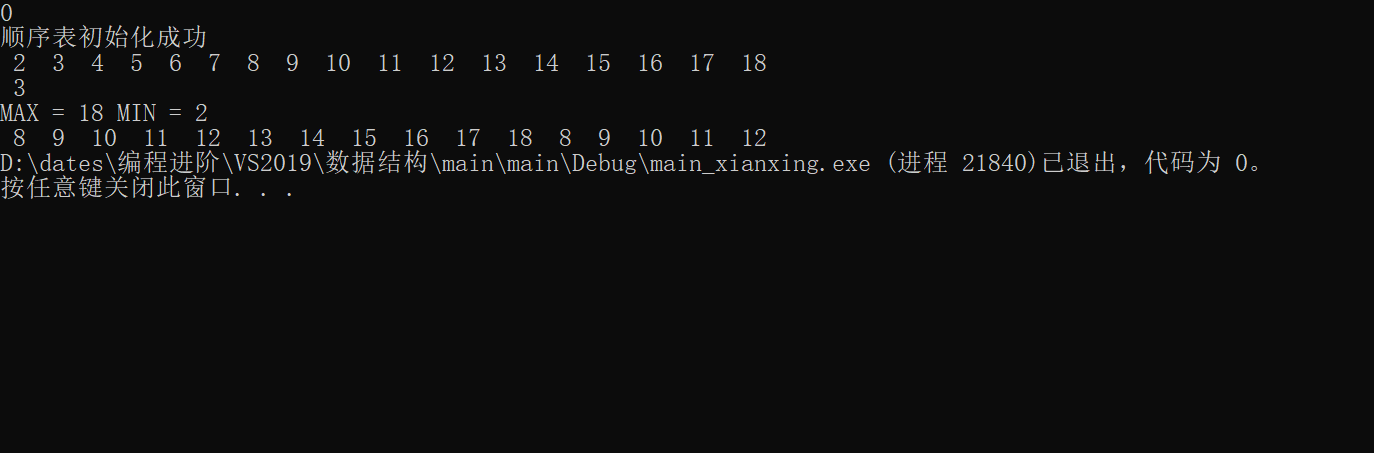


运行结果：

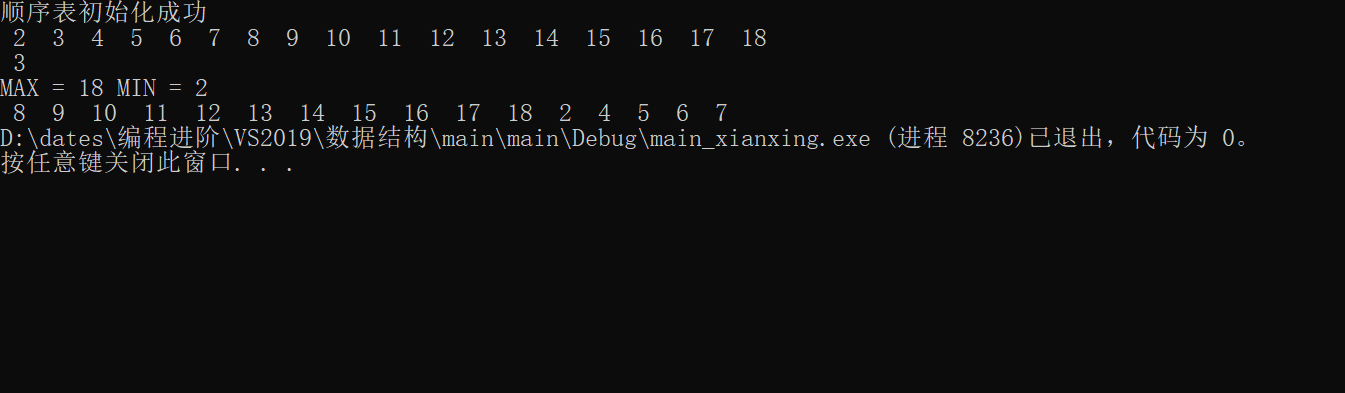


1. 合并两个线性表
2. 循环左移，思路是定义两个数组str[100]、str1[100]，将线性表内的值赋给str[100]，然后再将循环变换后的值赋到str1[100]，最后输出str1[100]；

问题：由于一开始只定义了一个数组，故在交换时，有的数值就被覆盖了。



修改后运行如下：



总结：

1. 本次上机事先准备工作没做好，导致上机时时间安排出问题。
2. 对线性表、数组、指针的理解有待进一步强化。
3. 写程序时要谨慎，防止出现低级错误，先将程序设计的逻辑问题考虑周全再动手敲代码。

代码如下：

#include<string.h>

#include<ctype.h>

#include<malloc.h> /\* malloc()等 \*/

#include<limits.h> /\* INT\_MAX等 \*/

#include<stdio.h> /\* EOF(=^Z或F6),NULL \*/

#include<stdlib.h> /\* atoi() \*/

#include<io.h> /\* eof() \*/

#include<math.h> /\* floor(),ceil(),abs() \*/

#include<process.h> /\* exit() \*/

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define LIST\_INIT\_SIZE 100

#define LISTINCREMRNT 10

typedef int ElemType;

 typedef int Status;

 typedef int Boolean;

typedef struct

{

  ElemType \* elem; //储存空间基地址

  int length; // 记录当前链表长度

  int listsize; //链表规模

} SqList;

Status InitList(SqList \*L)

{

    (\*L).elem = (ElemType\*)malloc(LIST\_INIT\_SIZE\*sizeof(ElemType));

    if(!(\*L).elem)

        exit(OVERFLOW);

    (\*L).length = 0;

    (\*L).listsize = LIST\_INIT\_SIZE;

    return OK;

}

Status DestroyList(SqList \*L)

{ /\* 操作结果：三元组T被销毁 \*/

    free((\*L).elem);

   (\*L).elem=NULL;

   (\*L).length =0;

   return OK;

 }

void ClearList(SqList \*L) {

    (\*L).length = 0;

}

Status ListEmpty(SqList L)  //值拷贝

{

return L.length ==0 ?1:0;

}

Status ListLength(SqList L)

{

    return L.length ;

 }

Status GetEle(SqList L,int i,int \*e)

{

    if(i<1||i>L.length)

     return ERROR;

    \*e = L.elem[i-1];

    return OK;

}

Status ListInsert(SqList \*L ,int i, int e)

{

    int \*newbase;

    int \*p, \*q;

    if(i<1||i>(\*L).length+1)

        return ERROR;

    if((\*L).length>(\*L).listsize)

    {

        newbase = (ElemType\*)realloc((\*L).elem, ((\*L).listsize + LISTINCREMRNT) \* sizeof(ElemType));

        if(!newbase)

            exit(OVERFLOW);

        (\*L).elem = newbase;

        (\*L).listsize += LISTINCREMRNT;

    }

    q = &((\*L).elem[i-1]);

    for(p=&(\*L).elem[(\*L).length - 1]; p>=q; --p)

    {

        \*(p+1) = \*p;

    }

    \*q = e;

    ++(\*L).length;

    return OK;

}

Status LocateElem(SqList \*L,int e)

{

    int i = 1;

    while (i<(\*L).length && (\*L).elem[i-1])

        i++;

    if(i<(\*L).length)

        return i;

    else

        return ERROR;

}

Status ListDelete(SqList \*L,int i,ElemType \*e)

{

    int \*p;

    if(i<1||i>(\*L).length)

        return ERROR;

    \*e = (\*L).elem[i-1];

    for(p=&((\*L).elem[i-1]);p<=&((\*L).elem[(\*L).length-2]);p++)

    {

        \*p = \*(p+1);

    }

    (\*L).length--;

    return OK;

}

Status ListTraverse(SqList L)

{

    for(int i=0;i<L.length-1;i++)

    {

        printf(" %d ",L.elem [i]);

    }

    return OK;

}

void MergeList(SqList La, SqList Lb,SqList \*Lc)

{

    InitList(Lc);

    int i=1, j=1, k=0;

    int La\_length = ListLength(La);

    int Lb\_length = ListLength(Lb);

    int ai,bj;

    while ((i<=La\_length) && (j<=Lb\_length))

    {

        GetEle(La, i, &ai);

        GetEle(Lb, j, &bj);

        if(ai<=bj)

        {

            ListInsert(Lc, ++k, ai);

            ++i;

        }

        else

        {

            ListInsert(Lc, ++k, bj);

            ++j;

        }

    }

    while(i<=La\_length)

    {

        GetEle(La, i++, &ai);

        ListInsert(Lc, ++k, ai);

    }

    while(j<=Lb\_length)

    {

        GetEle(Lb, j++, &bj);

        ListInsert(Lc, ++k, bj);

    }

}

void main()

{

    int max,min;

    SqList L1,L2,L3;

    ElemType  e;

    int i;

    if(InitList(&L1)==1 && InitList(&L2)==1)

        printf("顺序表初始化成功\n");

    for(i=1;i<10;i++)

        ListInsert(&L1,i, 2\*i);

    for(i=1;i<10;i++)

        ListInsert(&L2,i, (2\*i+1));

    MergeList(L1, L2, &L3);

    ListTraverse(L3);

    printf("\n ");

    ListDelete(&L3,2,&e);

    printf("%d\n",e);

    DestroyList(&L1);

    DestroyList(&L2);

    DestroyList(&L3);

}