计算机与信息工程学院实验报告

••••••••••••••••••••••••••••••••• 密 ••••••••••••••••••••••••••••••••• 封 ••••••••••••••••••••••••••••••••• 线 •••••••••••••••••••••••••••••••••

姓名：\_\_\_ 学号：\_20120200\_ 专业：\_计算机科学与技术\_ 年级：\_2020级\_

课程： 数据结构 主讲教师：\_\_杨\_\_ 辅导教师：\_杨

实验时间：\_2022\_年 \_3\_月 23\_日 \_下\_午\_0\_时至\_24\_时，实验地点\_计算机大楼\_\_

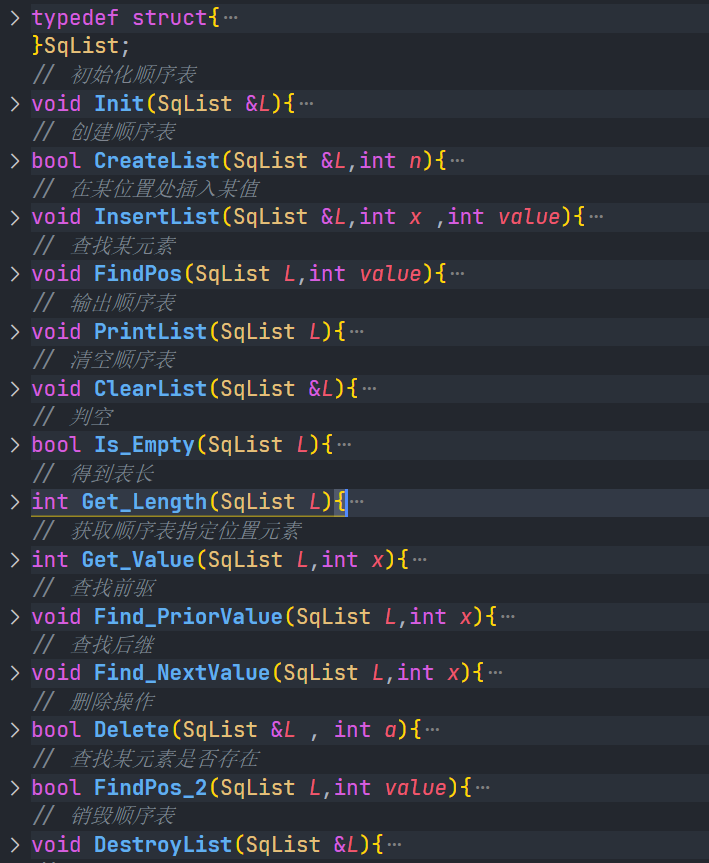
实验题目： 实验一 顺序表的操作（4学时）

实验目的：通过该实验，深入理解顺序表的逻辑结构、物理结构等概念，掌握顺序表基本操作的编程实现，注意顺序表插入、删除等操作过程中数据元素的移动现象，学生编写程序时，要考虑程序的健壮性，熟练掌握通过函数参数返回函数结果的办法。

实验环境（硬件和软件） Visual Studio Code/Win 10

实验内容： 用C++语言实现顺序表的各种操作：

1. 初始化顺序表
2. 创建顺序表
3. 在某位置处插入顺序表
4. 查找顺序表的某个元素
5. 输出顺序表的各个元素
6. 判断是否为空表
7. 得到顺序表的长度
8. 获取顺序表的指定位置的元素值大小
9. 查找前驱、后继
10. 删除顺序表的值的操作
11. 销毁顺序表



实验步骤：

1. 首先定义一个顺序表
2. *// 定义顺序表*
3. #define **MaxSize** 100
4. typedef struct{
5. int length ;
6. int \*data;
7. }SqList;

2. 对顺序表进行初始化

*// 初始化顺序表*

void **Init**(SqList &*L*){

*L*.data = new int[**MaxSize**];

*L*.length = 0 ;

}

3. 人机交互，创建顺序表，在向已创建的顺序表中插入时，先判断插入元素个数与表长之和是否大于顺序表的最大容量，也要保证个数大于0，即为合法个数。

*// 创建顺序表*

bool **CreateList**(SqList &*L*,int *n*){

    f = 0 ;

    if(*n*<0||*n*>**MaxSize**) return false ;

    cout **<<** "输入各个元素大小 :" ;

    for(int i = 0 ; i < *n* ; i ++){

        cin **>>** *L*.data[i] ;

*L*.length ++ ;

    }

    f = 1 ;

    return true ;

}

4. 插入函数时，先判断是否有足够空间放置新节点，及选择插入到位置是否合理，插入过程中，为了避免覆盖已存在节点，我们从最后一个节点开始移动。

*// 在某位置处插入某值*

void **InsertList**(SqList &*L*,int *x* ,int *value*){

    f = 0 ;

    if(*x*<1||*x*>*L*.length+1) {

        cout **<<** "插入位置错误" **<<** **endl** ;

    }

    if(*L*.length >= **MaxSize**){

        cout **<<** "当前空间已满" **<<endl** ;

    }

    for(int i = *L*.length - 1; i >= *x* - 1; i --){

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

    }

*L*.length ++ ;

*L*.data[*x*-1] = *value* ;

    f = 1 ;

}

5. 查找前驱与后继：

*// 查找前驱*

void **Find\_PriorValue**(SqList *L*,int *x*){

    int fff = 0 ;

    for (int i = 0; i < *L*.length;i ++){

        if(*L*.data[i] == *x*){

            if(i == 0)

                cout **<<** "第一个元素没有前驱" **<<** **endl**;

            else{

                cout **<<**"前驱为" **<<** *L*.data[i - 1]**<<endl** ;

                fff = 1;

                break;

            }

        }

    }

    if(!fff)

        cout **<<** "顺序表中无该元素" **<<** **endl**;

}

*// 查找后继*

void **Find\_NextValue**(SqList *L*,int *x*){

    int fff = 0 ;

    for (int i = 0; i < *L*.length;i ++){

        if(*L*.data[i] == *x*){

            if(i == *L*.length-1)

                cout **<<** "最一个元素没有后继" **<<** **endl**;

            else{

                cout **<<**"后继为" **<<** *L*.data[i + 1]**<<endl** ;

                fff = 1;

                break;

            }

        }

    }

    if(!fff)

        cout **<<** "顺序表中不存在该元素" **<<** **endl**;

}

6. 销毁顺序表，我们要主要将节点所占据的空间释放，用函数delete

*// 销毁顺序表*

void **DestroyList**(SqList &*L*){

    if(*L*.data)

        delete[] *L*.data;

*L*.length = 0;

*L*.data = **NULL**;

}

1. 顺序表实现集合的交集：利用双指针算法，定义两个指针从两个顺序表的起始位置开始，比较两个顺序表对应位置的大小，当找到其中较小的一个元素，将其插入新顺序表C中，同时移动指向较小元素的指针，继续下一轮的比较，同时，遍历完任一顺序表后，将另一顺序表的所有元素全部插入新顺序表，从而最终得到一个有序的顺序表，即为两个已知顺序表的交集。

void  **Function\_I** (SqList *A*,SqList *B*,SqList \**C*){

    int i = 0,k = 0,j = 0;

    while(i<=*A*.length&&j<=*B*.length){

        if(*A*.data[i]==*B*.data[j]){

*C*->data[k++]=*A*.data[i++];

            j++;

        }

        else if(*A*.data[i]<*B*.data[j]){

            i++;

        }

        else{

            j++;

        }

    }

*C*->length=k;

}

1. 顺序表实现集合的并集：利用顺序表实现集合的并集操作的思路与利用顺序表实现集合的交集操作相同，都是利用双指针算法，改变的是在while循环中，只有当查找到两顺序表中元素相同时，才向新顺序表中插入。核心代码如下：

void **Function\_T**(SqList *A*, SqList *B*, SqList \**C*)

{

    int i = 0, j = 0, k = 0;

    while (i <= *A*.length && j <= *B*.length)

    {

        if (*A*.data[i] <= *B*.data[j]){

*C*->data[k++] = *A*.data[i++];

        }

        else if(*A*.data[i] > *B*.data[j]){

*C*->data[k++] = *B*.data[j++];

        }

    }

    while (i <= *A*.length){

*C*->data[k++] = *A*.data[i++];

    }

    while (j <= *B*.length){

*C*->data[k++] = *B*.data[j++];

    }

*C*->length = k - 2;

}

1. 顺序表实现集合的差集: 利用顺序表实现集合的差集操作的思路与利用顺序表实现集合的交集、并集操作相同，都是利用双指针算法，改变的是在while循环中，当查找到两顺序表中元素相同时，只改变指针指向，而不向新顺序表中插入，当查找到两顺序表中元素不同时，将较小的插入新顺序表中，最后处理一下可能剩余的原顺序表中的元素，将其插入，得到有序的顺序表差集。核心代码如下：

void **Function\_S**(SqList *A*,SqList *B*,SqList \**C*){

    int l = 0, r = 0, k = 0;

    while(l <= *A*.length && r <= *B*.length){

        if(*A*.data[l] == *B*.data[r]) {

            l++, r++;

        }else if(*A*.data[l] < *B*.data[r]){

*C*->data[k++] = *A*.data[l++];

*// C->data[k++] = B.data[r++];*

        }else if(*A*.data[l] > *B*.data[r]){

*// C->data[k++] = A.data[l++];*

*C*->data[k++] = *B*.data[r++];

        }

    }

    while(l <= *A*.length){

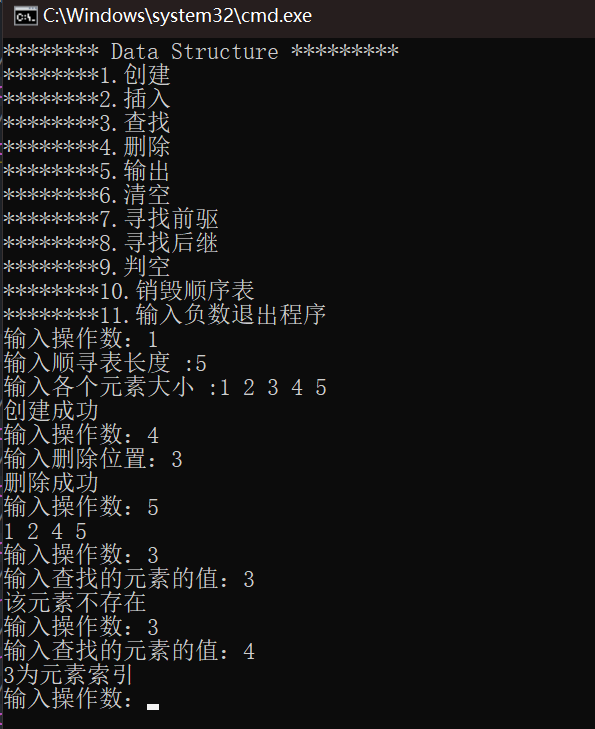
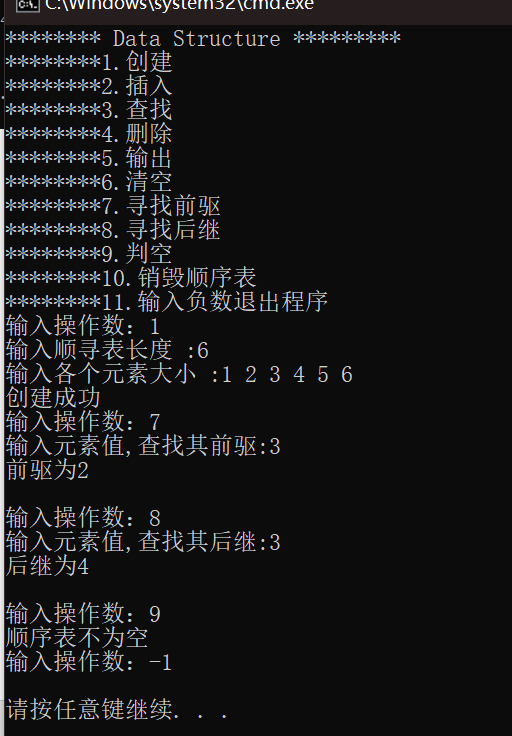
*C*->data[k++] = *A*.data[l++];

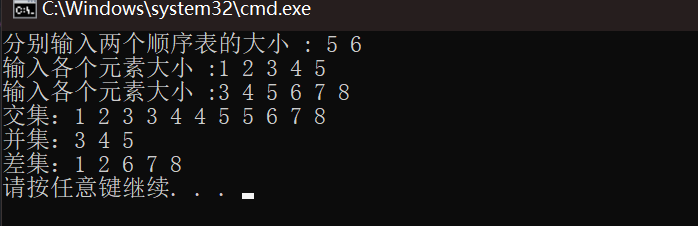
    }

*C*->length = k - 2;

}

实验数据记录：



问题讨论：

在用顺序表实现集合的交集，并集，差集时，在传参时，我发现传进的空顺序表并不能被改变，仿照普通参数传递改成Sqlist &C，并不能解决问题，查阅资料发现应该改为Sqlist \*C，同时主函数传递参数使用&C,解决问题。