实现顺序表的各种基本运算

姓名：凌强 班级：21计算机二班 学号：2021413579

实验目的：

1. 掌握线性表的基本运算。
2. 掌握顺序存储的概念，学会对顺序存储数据结构进行操作。
3. 加深对顺序存储数据结构的理解，逐步培养解决实际问题

的编程能力。

实验内容：

1. 初始化顺序表，接收输入的数据
2. 选择对应的功能以完成顺序表的插入、删除、查找、销毁等操作

# **需求分析**

1. 完成对顺序表的创建、销毁、插入、删除、定位等基本操作。
2. 演示程序中，规定 ElemType 为 int 型。
3. 演示程序以用户和计算机的对话方式执行，在运行框中出现提示信息之后，由用户在键盘上输入完成相关命令的执行；相结果显示在其后。
4. 程序执行的命令包括：
   1. 构造一个新的线性表。
   2. 求线性表中元素个数。
   3. 求线性表中第index个元素的值。
   4. 求L中第一item数据元素的位序。
   5. 在线性表的第l个位置插入数据 k。
   6. 删除线性表中第dl个数据。
   7. 销毁线性表。
   8. 退出程序。
5. 测试数据：六、测试结果。

# **概要设计**

顺序表的抽象数据定义：

*#define* MAX 100

typedef struct

{

    int a[MAX];

    int length;

} SqList;

基本操作：

*//定位元素*

void Locate(SqList \**sq*)

*//插入元素*

void Insert(SqList \**sq*)

*//删除元素*

void Delete(SqList \**sq*)

*//查找元素*

void Find(SqList \**sq*)

*//输出原始数据*

void OutData(SqList \**sq*)

*//输出操作之后的数据*

void OutList(SqList \**sq*)

*//打印主界面*

void welcome(SqList \**sq*)

程序包含两个模块：

* 1. 主函数模块：

用于与测试用户交换数据。

* 1. 操作函数模块：

用于指引测试用户下一步操作，操作顺序表。

# **详细设计**

## 顺序表的定义：

*#include* <iostream>

*#define* MAX 100

using namespace std;

typedef struct

{

    int a[MAX];

    int length;

} SqList;

## 操作函数定义：

void OutData(SqList \**sq*)

{

    int i;

*for* (i = 0; i < sq->length; i++)

    {

        cout << sq->a[i] << "  ";

    }

    cout << endl;

}

void OutList(SqList \**sq*)

{

    int i;

*for* (i = 0; i < sq->length; i++)

    {

        cout << sq->a[i] << " ";

    }

    cout << endl;

}

*//定位*

void Locate(SqList \**sq*)

{

    int index;

    cout << "请输入index:" << endl;

    cin >> index;

*if* (index < 1 || index > sq->length)

    {

        cout << "不存在第" << index << "个元素" << endl;

    }

*else*

    {

        cout << "第" << index << "个元素:" << sq->a[index - 1] << endl;

    }

}

*//查找元素*

void Find(SqList \**sq*)

{

    int i = 0, item;

    cout << "请输入要查找的元素:" << endl;

    cin >> item;

*while* (i < sq->length && sq->a[i] != item)

        i++;

*if* (i >= sq->length)

        cout << "待查找元素" << item << "不存在！" << endl;

*else*

        cout << "查找：元素" << item << "是第" << i + 1 << "个元素" << endl;

}

*//插入元素*

void Insert(SqList \**sq*)

{

    int j, l, k;

    cout << "请输入要插入的位置和元素:" << endl;

    cin >> l >> k;

*if* (l < 1 || l > sq->length)

    {

        cout << "插入位置非法！" << endl;

    }

*else*

    {

        l--;

*for* (j = sq->length; j > l; j--)

            sq->a[j] = sq->a[j - 1];

        sq->a[l] = k;

        sq->length++;

        cout << "插入后的顺序表为L: ";

        OutList(sq);

    }

}

*//删除元素*

void Delete(SqList \**sq*)

{

    int j, dl;

    cout << "请输入要删除的元素位置:" << endl;

    cin >> dl;

*if* (dl < 1 || dl > sq->length)

    {

        cout << "删除第" << dl << "个元素不存在！" << endl;

    }

*else*

    {

        dl--;

*for* (j = dl; j < sq->length - 1; j++)

            sq->a[j] = sq->a[j + 1];

        sq->length--;

        cout << "删除后的顺序表为L: ";

        OutList(sq);

    }

}

*//打印主界面*

void welcome(SqList \**sq*)

{

    system("cls");

    printf("〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓〓  ☆   顺 序 表 测 试 系 统     ☆  〓〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓★★★★★         ★★★★★★★         ★★★★★〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓★★★★★         ★★★★★★★         ★★★★★〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓★  ☆          1.定位元素        ☆  ★〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓★  ☆          2.插入元素        ☆  ★〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓★  ☆          3.删除元素        ☆  ★〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓★  ☆          4.查找元素        ☆  ★〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓★  ☆          5.输出元素        ☆  ★〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓★  ☆          0.退出系统        ☆  ★〓〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓★  ☆                              ☆  ★〓〓〓〓〓〓〓〓\n");

    printf("〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓〓\n");

    cout << "初始化链表成功！" << endl;

    int n = 0;

    int op;

    cout << "请输入元素个数:" << endl;

    cin >> sq->length;

    cout << "请输入" << sq->length << "个元素:" << endl;

*while* (cin >> sq->a[n++])

    {

*if* (cin.get() == '\n')

*break*;

    }

    sq->length = n;

    cout << "创建成功！" << endl;

}

## 主函数完善：

int main()

{

    SqList \*sq;

    sq = (SqList \*)malloc(sizeof(SqList));

    int op;

*//打印主界面*

    welcome(sq);

*while* (true)

    {

*//指引用户操作*

        printf("\n请输入对应的功能键(数字): ");

        cin >> op;

*switch* (op)

        {

*case* 1:

            Locate(sq);

*break*;

*case* 2:

            Insert(sq);

*break*;

*case* 3:

            Delete(sq);

*break*;

*case* 4:

            Find(sq);

*break*;

*case* 5:

            cout<<"当前数据为:"<<endl;

            OutList(sq);

*break*;

*default*:

*break*;

        }

    }

    system("pause");

*return* 0;

}

# **调试分析**

1. 模块划分较合理。
2. 有头结点的算法相对简单，需处理的bug要少一些。
3. 顺序表示在删除，插入操作中的时间复杂度高于链式。O（n）
4. 定位、求前驱、求后继由于都要寻找e的位置，时间复杂度相当。
5. 遍历的时间复杂度均为O(n)。
6. 本次实验报告完成了对顺序表的各种基本运算表示，是一次良好的程序设计训练。

# **用户使用说明**

程序运行环境为vscode,执行文件为SqList.cpp,Seqist.exe.

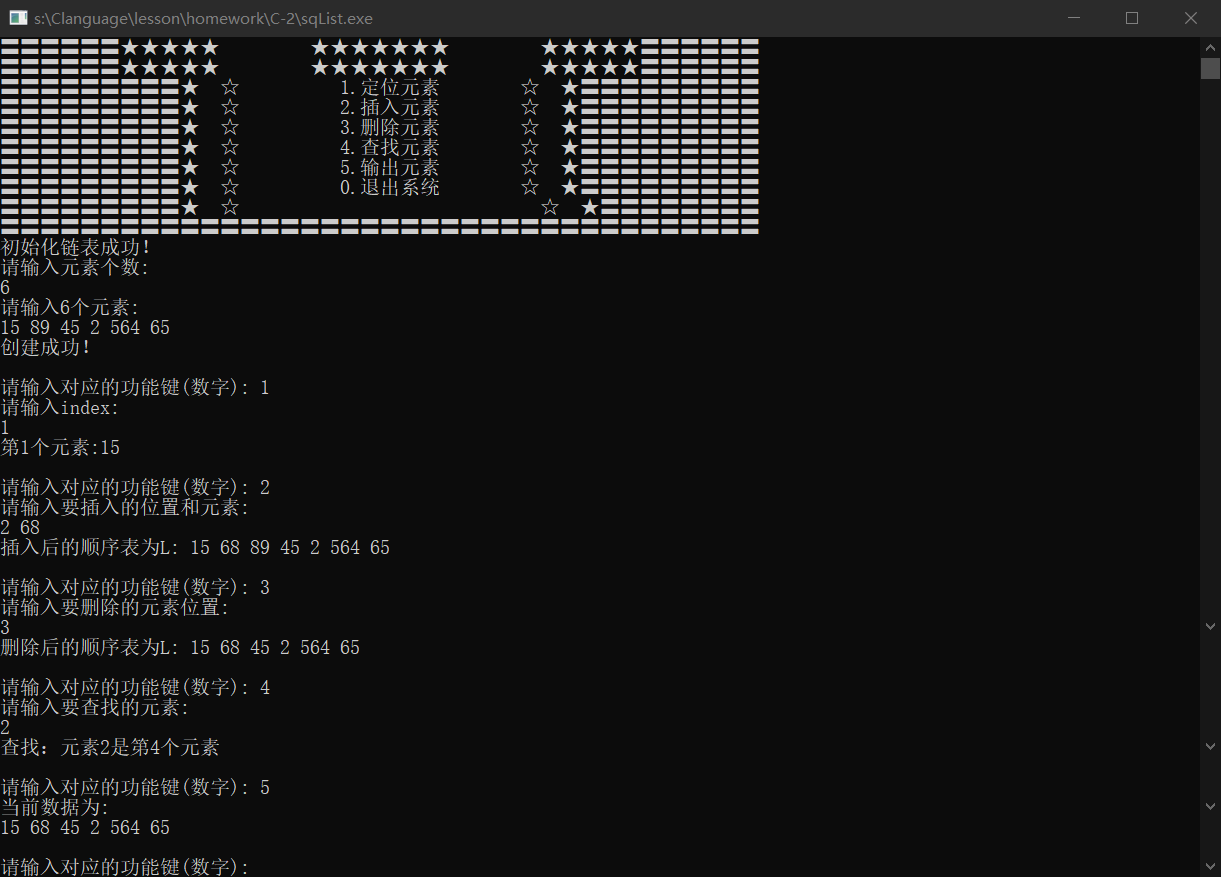
用户界面：



先输入元素个数，然后依次输入数据，最后创建成功，选择对应的功能。

# **测试结果**

## 顺序表



# **实验体会**

通过这次顺序表测试程序的编写，以及实验报告的完善，学习到了顺序表的存储特点和用途，同时感受到了c语言指针直接操作内存的魅力之处。

# **附录**

源程序：

SeqList.cpp