如何撰写实验报告

汇报人：魏子铖

班级：软件1703
学号：201726010308
姓名：魏子铖
讨论周次：第六周
讨论课主题：如何撰写实验报告

目录

01 需求分析

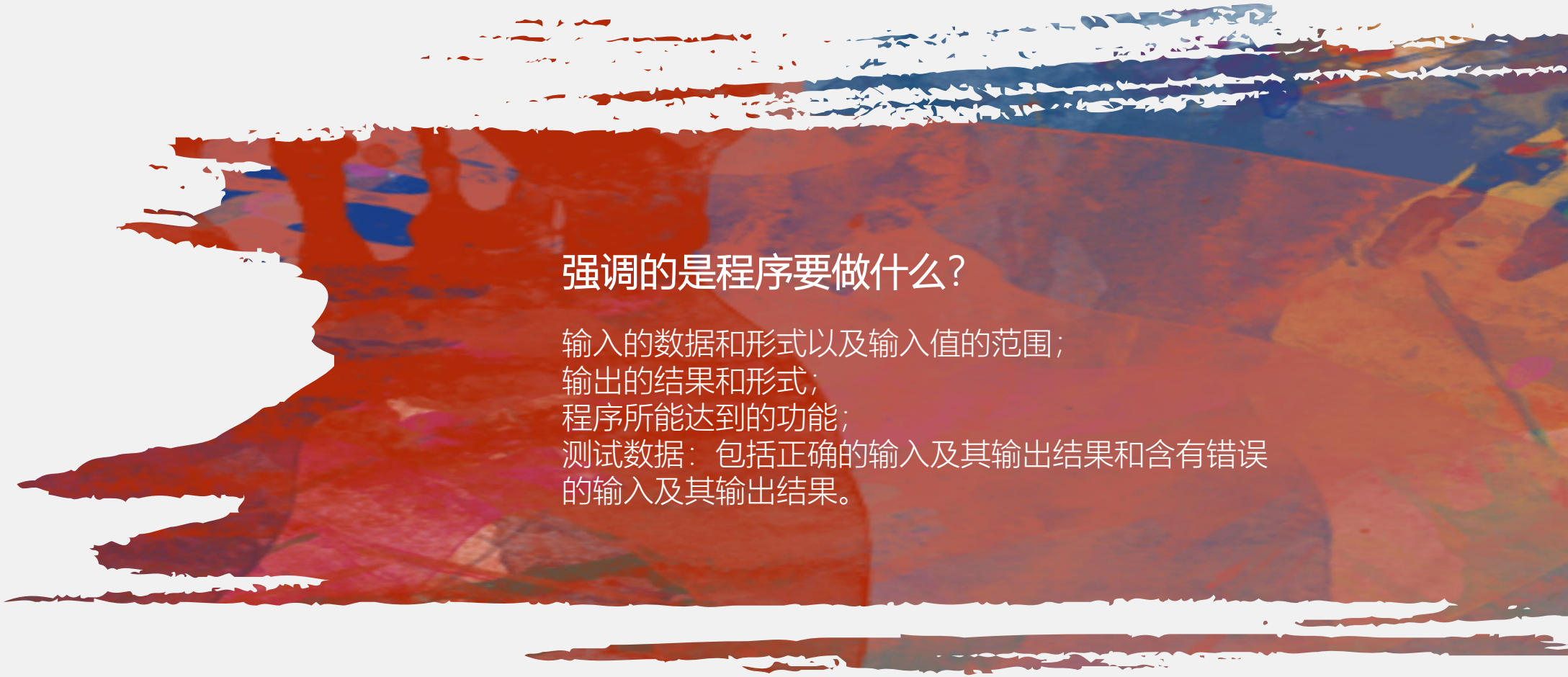
02 概要设计与详细设计

03 调试分析与测试结果

04 实验日志

01 需求分析

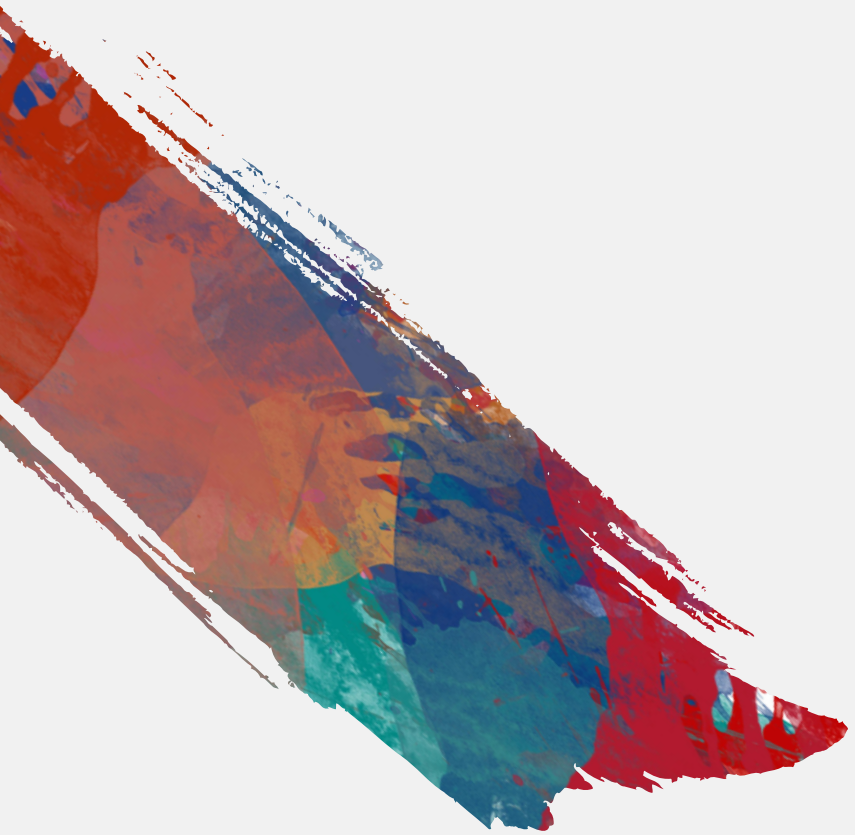
01 需求分析



强调的是程序要做什么？

输入的数据和形式以及输入值的范围；
输出的结果和形式；
程序所能达到的功能；
测试数据：包括正确的输入及其输出结果和含有错误的输入及其输出结果。

01 需求分析



0.问题描述

给定n个整数表示一个商店连续n天的销售量。如果某天之前销售量在增长，而后一天销售量减少，则称这一天为折点，反过来如果之前销售量减少而后一天销售量增长，也称这一天为折点。其他的天都不是折点。

给定n个整数$a_1, a_2, \ldots, a_n$表示销售量，请计算出这些天总共有多少个折点。

为了减少歧义，我们给定的数据保证：在这n天中相邻两天的销售量总是不同的，即$a_{i-1} \neq a_i$。注意，如果两天不相邻，销售量可能相同。

1.问题分析

①存储一组销售量数据

②统计根据这组数据绘制的折线图上拐点的数量

2.输入数据

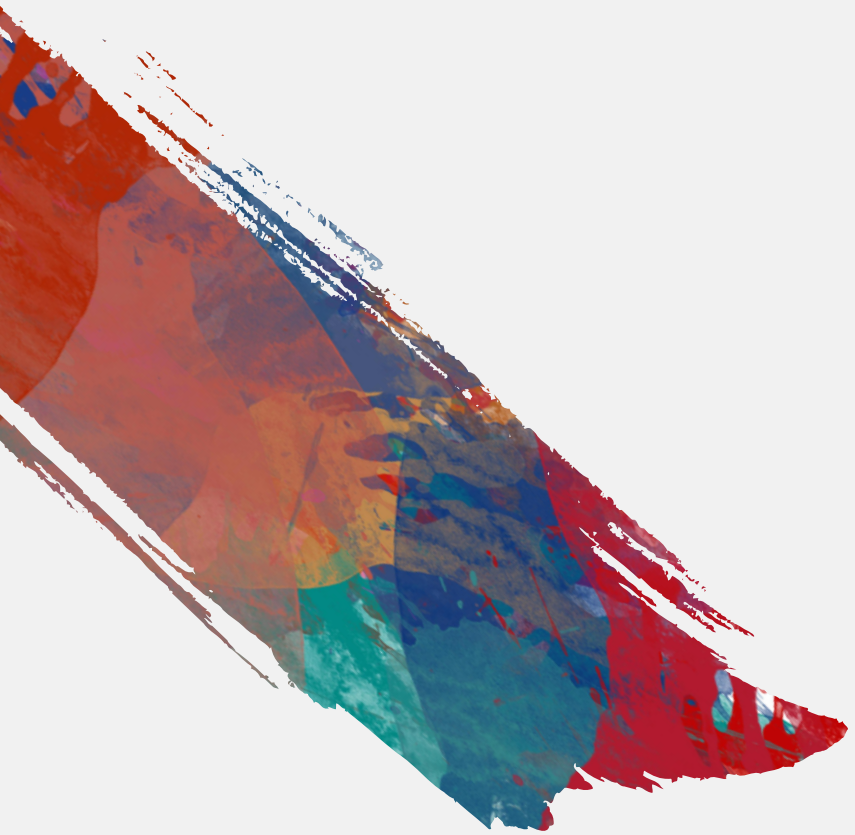
输入的第一行包含一个整数n。

第二行包含n个整数，用空格分隔，分别表示$a_1, a_2, \ldots, a_n$。

所有评测用例满足：$1 \leq n \leq 1000$，每天的销售量是不超过10000的非负整数。

3.输出数据

输出一个整数，表示折点出现的数量。



4.测试样例设计

样例一：

9

2 6 9 8 5 1 6 3 2

输出一：

3

样例二：

12

35 69 81 401 215 59 7 199 800 1593 12 1

输出二：

3

说明：样例一与样例二为一般情况，用于检验算法的正确性

样例三：

8

40 40 40 40 40 40 40 40

输出三：

0

样例四：

8

2 3 4 5 6 7 8 9

输出四：

0

样例五：

8

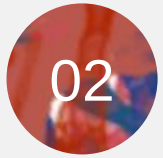
9 8 7 6 5 4 3 2

输出五：

0

说明：样例三、样例四与样例五中不存在折点，用于检验边界条件情况下的正确性

02 概要设计与详细设计



概要设计与详细设计

抽象数据类型

分析问题的数据和结构特征，设计合适的数据对象类型和抽象数据结构（ADT）。
要运用数据结构的知识阐述理由。
要运用ADT的设计和表示知识，设计题目的ADT。

算法的基本思想

简单扼要地阐述求解本问题的算法思路。（不能用计算机的语言）

程序的流程

设计并阐述程序的模块组成，简单描述每个模块的功能，整体描述各个模块之间的关系。

02

概要设计与详细设计

抽象数据类型

由于输入的数据元素均为非负整数，所以可以将输入存储在线性表中。
抽象数据类型设计：
数据对象：n个互相独立的整数
数据关系：从键盘读取的n个整数，按照输入的先后顺序存储，其形式满足线性特征，即$< a_i, a_{i+1} > (0 \le a < n)$
基本操作：开辟能够存储n个整型数据的内存空间
将输入的数据存储在线性表中
取出线性表中特定位置的数据
ADT：

```
template <typename E>
class Array_List : public List<E> {
private:
    int maxSize;//顺序表的最大长度
    int listSize;//顺序表的长度
    int curr;//当前位置
    E* listArray;//顺序表的物理结构

public:
    explicit Array_List(int size = defaultSize);//含默认参数的构造函数
    ~Array_List();//析构函数

    void append(const E&);//在顺序表后新增一个元素
    void moveToStart();//移动到顺序表的开头
    const E& getPrev() const;//获取当前位置的前驱的值
    const E& getNext() const;//获取当前位置的后继的值
    int length() const;//获取顺序表的长度
    const E& getValue() const;//获取当前位置的值
};
```

02 概要设计与详细设计

算法基本思想

折点计数问题中的数据是一个个整数，第一天与第n天的销售量即为“第一元素、最后元素”，并且第二天至第n-1天的数据是存在“唯一的前驱和后继的”，符合线性表的特点，由于输入是按照天数顺序输入的，所以可以采用顺序表来实现线性表，完成折点数量的统计。

核心算法主要分为两步：

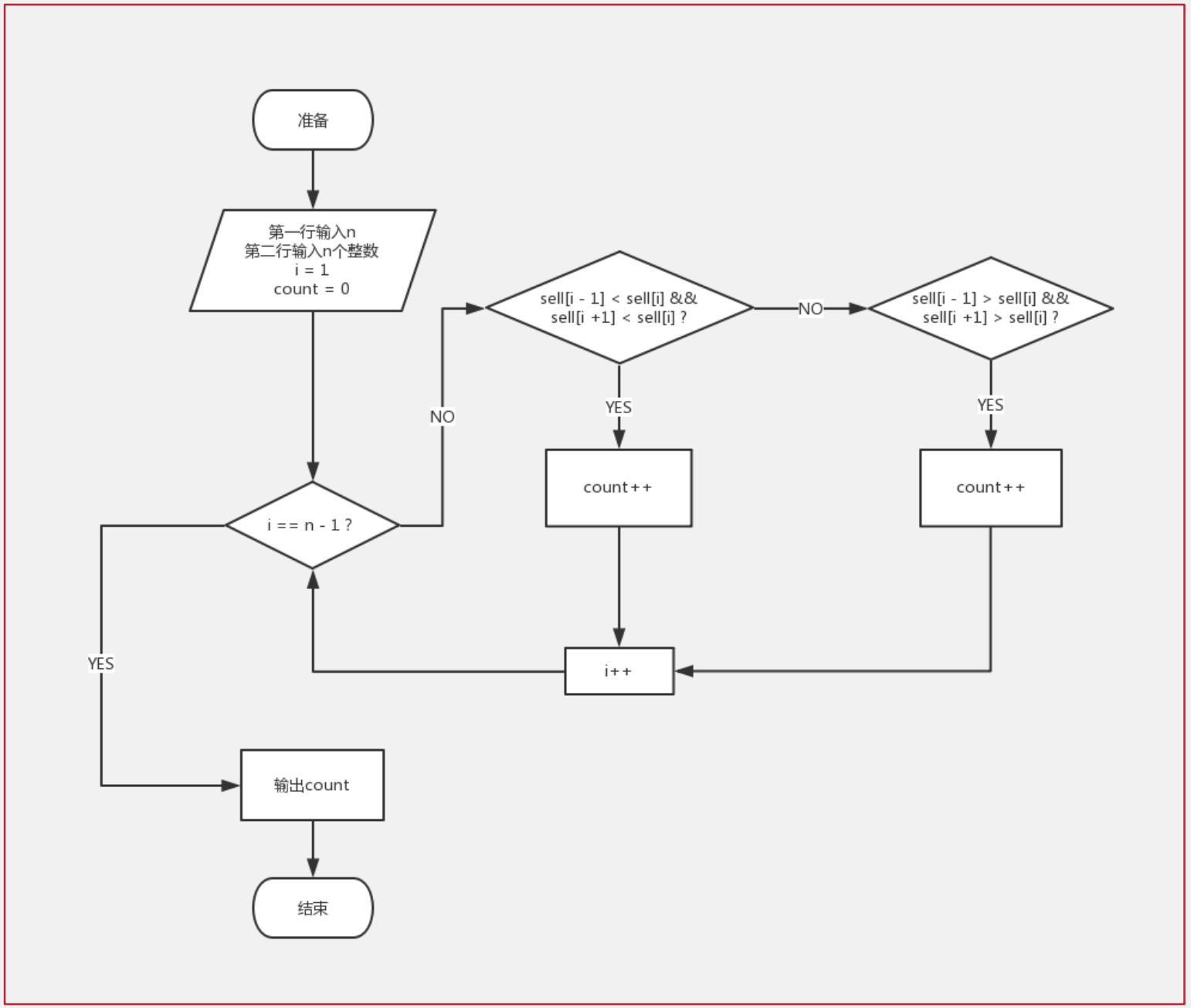
1.初始化一个计数器count = 0；

2.从第二天开始，一直到第n-1天，依次比较前后连续三天的销售量数据。

3.根据题目中折点的定义，第一天与第n天的销售量不存在出现折点的情况，所以我们从第二天开始查询。如果在查询的连续三天中，中间那天的销售量低于或高于前后两天的销售量，那么计数器的数值自增1，直到查询到第n-1天，循环结束后输出计数器的值即为所求结果。

02 概要设计与详细设计

程序的流程



02

概要设计与详细设计

1.物理数据类型

综合分析问题的数据和结构特征以及性能需求，设计合适的物理数据类型和物理数据结构。要运用数据结构的知识阐述理由。要用伪代码的形式给出概要设计中的ADT的具体实现。

2.输入和输出的格式

设计并阐述输入和输出的流程，格式，交互过程。

3.算法的具体步骤

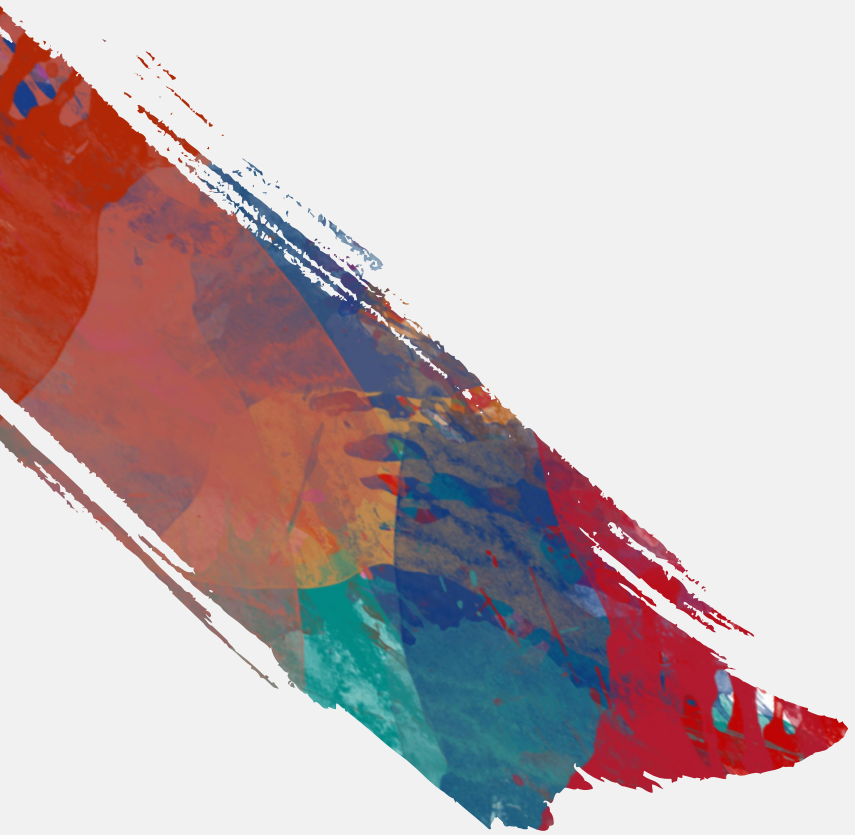
针对每一个模块，设计并阐述算法的具体步骤，要用文字的形式描述步骤，也可以用流程图的形式描述步骤，要给出伪代码。如果一个模块算法复杂，可以采用分为更小功能模块的方式来分别阐述。

4.算法的时空分析

运用算法分析的知识，分析并阐述求解本问题算法的各个模块的时空复杂度。

详细设计

02 概要设计与详细设计



1.物理数据类型

由于输入数据是整型数据，且所有评测用例满足：$1 \le n \le 1000$，每天的销售量是不超过10000的非负整数，为了节省内存空间，数据类型用short；由于输入按照日期先后顺序，满足线性特征，所以物理数据结构为线性结构。

2.输入和输出的格式

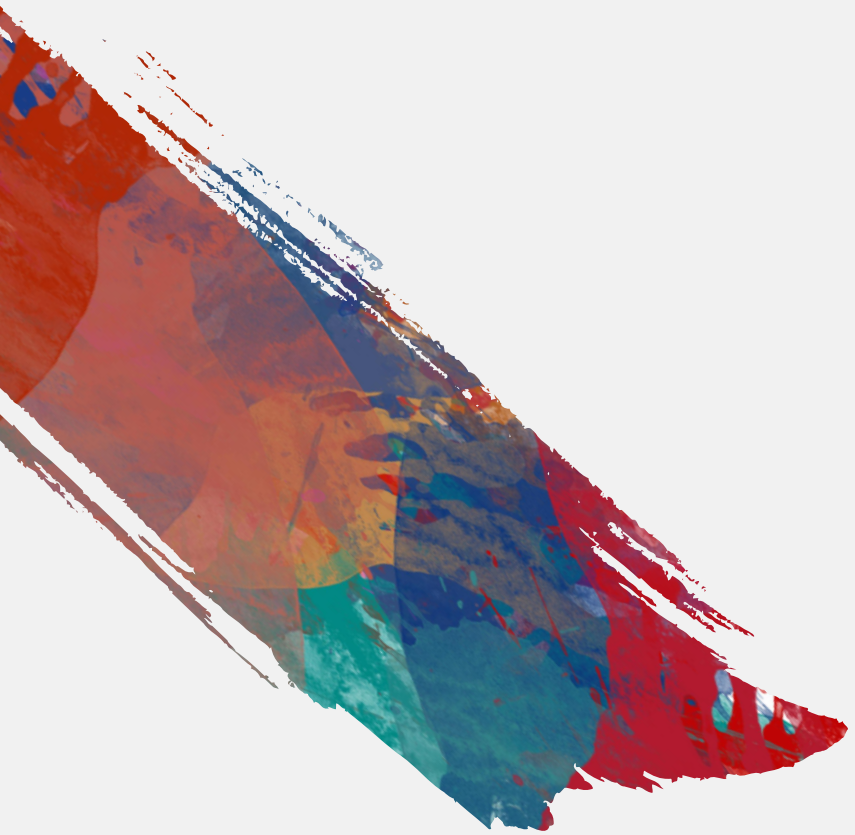
从键盘输入数据规模n和每天的销售量sell[i],程序将处理这些数据并将结果通过DOS输出。输出结果为折点的数量。

3.算法的具体步骤

1、设置数组，通过for循环将数据存入

2、已知输入总数n，依次根据题目中折点的定义，第一天与第n天的销售量不存在出现折点的情况，所以我们从顺序表的第二位开始查询。如果在查询的前后连续三个数据中，中间位置的数据同时小于或大于前后位置的数据，那么计数器的数值自增1，直到查询到第n-1个数据，循环结束后输出计数器的值即为所求结果。

02 概要设计与详细设计



ADT具体实现：

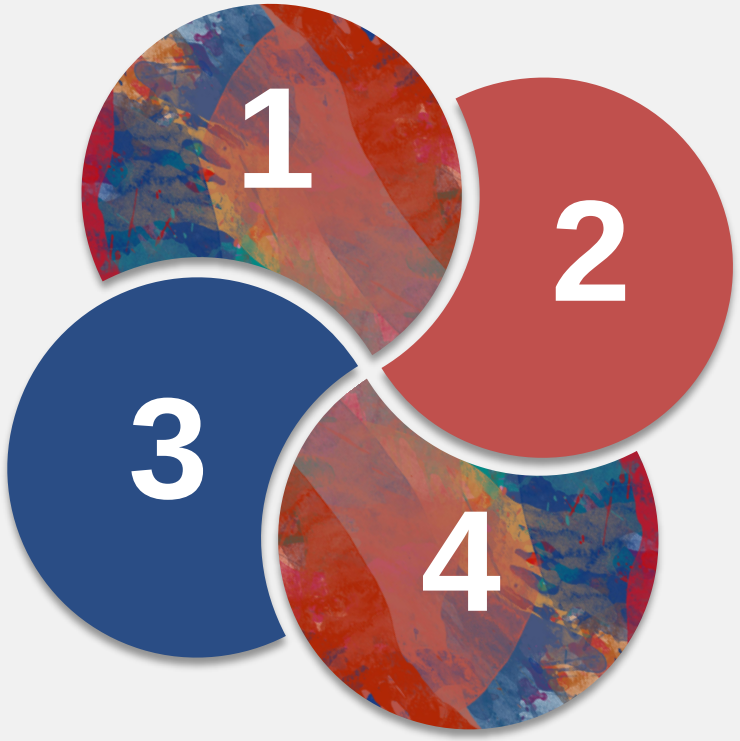
```
Array_List<int> sell(n);
for (int i = 0; i < n; i++) {
    int a = 0;
    cin >> a;
    sell.append(a);
}

int count = 0;
for (sell.moveToStart(), sell.next(); sell.currPosition() < sell.length()
- 1; sell.next()) {
    int day1 = sell.getPrev();
    int day2 = sell.getValue();
    int day3 = sell.getNext();
    if (day1 > day2 && day3 > day2)
        count++;
    if (day1 < day2 && day3 < day2)
        count++;
}
cout << count << endl;
```

03 调试分析与测试结果

03 调试分析与测试结果

设计调试方案

调试目的，样例，调式计划和设置。



描述调试过程

调试中的问题

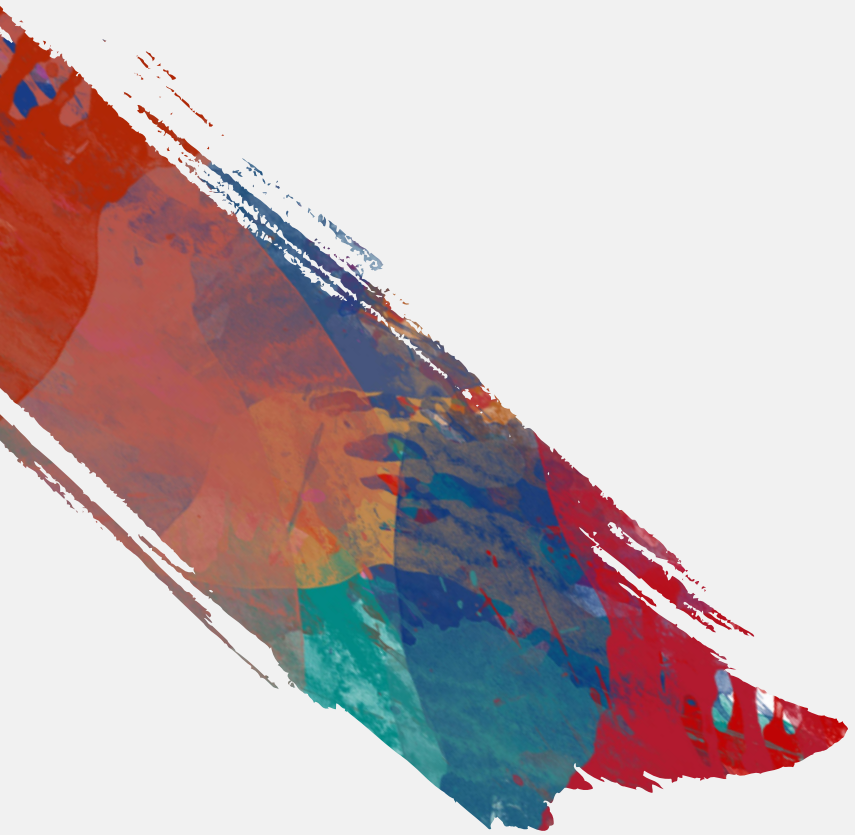
如果发现问题，简单阐述如何解决问题。

测试结果

贴出每个测试样例的运行结果，以及给出分析结论

03

调试分析与测试结果



1.调试方案设计

调试目的：

发现思维逻辑与代码实现上的区别，改进代码结构，排除语法、逻辑上的错误。

样例：

9

2 6 9 8 5 1 6 3 2

调试计划：

单步调试，在调试过程中注意i与count的值的改变，发现问题。

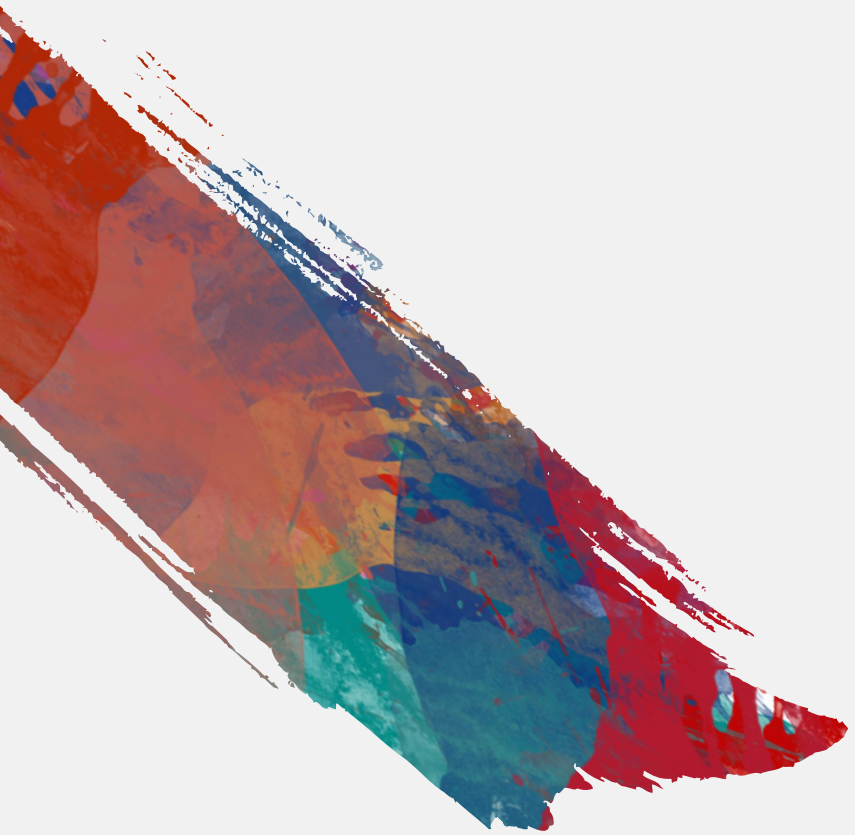
设置：

在循环开始处设置断点，单步调试。

2.调试过程和结果，及分析

一开始由于对折点定义的不充分理解，导致从顺序表的第一位开始查询，但第一位的数据没有前驱，于是发现了一个越界的问题。修改逻辑上的问题后，样例通过调试，提交到评测机上，该份代码得分100分。

03 调试分析与测试结果



```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
9
2 6 9 8 5 1 6 3 2
9
2 6 9 8 5 1 6 3 2
3

Process finished with exit code 0
```

样例一
折点数量为3，结果正确

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
12
35 69 81 401 215 59 7 199 800 1593 12 1
12
35 69 81 401 215 59 7 199 800 1593 12 1
3

Process finished with exit code 0
```

样例二
折点数量为3，结果正确

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
8
40 40 40 40 40 40 40 40
8
40 40 40 40 40 40 40 40
0

Process finished with exit code 0
```

样例三
折点数量为0，结果正确

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
8
2 3 4 5 6 7 8 9
8
2 3 4 5 6 7 8 9
0

Process finished with exit code 0
```

样例四
折点数量为0，结果正确

```
C:\Users\istil\Desktop\istillmessup\DataStructures\List\cmake-build-debug\List.exe
8
9 8 7 6 5 4 3 2
8
9 8 7 6 5 4 3 2
0

Process finished with exit code 0
```

样例五
折点数量为0，结果正确

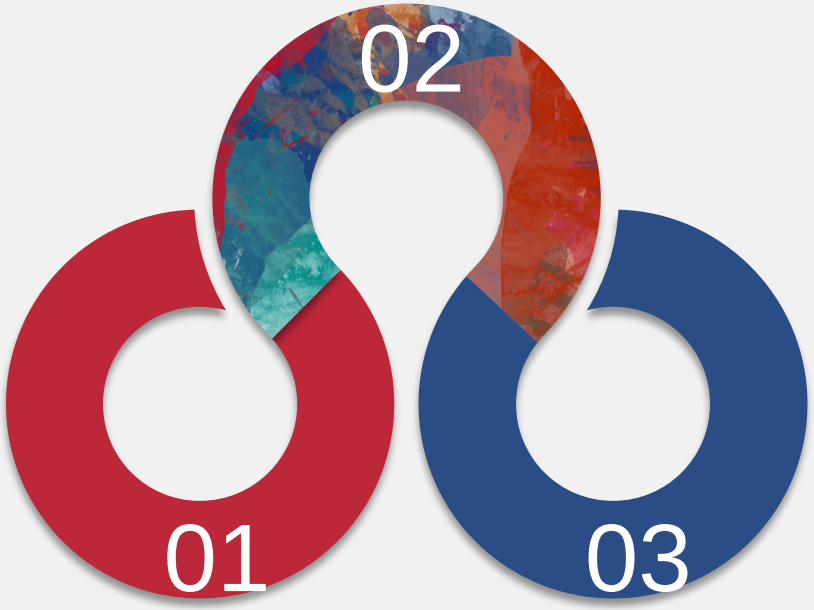
04 实验日志

04 实验日志

01 完整记录本次实验内容

02 遇到的问题及解决方法

03 心得体会




THANKS!