# 四川师范大学 实验报告

学期： 2024 至 2025 第一学期 实验成绩：

课程名字：《程序设计基础——数据结构(C语言版)》 专业： 信息与计算科学

班级： 2023 级 9 班 实验编号： 01

实验项目： 实验一 指导老师： 冯山

姓名： 刘智恒 学号： 2023060522

**一、实验题目**

ADT的类C描述向C程序的转换实验

**二、实验目的及要求**

1.复习C语言编程基础(函数、指针、动态内存空间分配等)。

2.熟悉算法的类C描述及其C语言源程序实现的基本技巧。

3.加深对数据对象逻辑结构和物理结构关系的理解。

4.熟悉算法的时空间复杂度分析方法。

**三、实验内容**

1.输入一组数，存入数组中，寻找输入数据中的最大值和最小值。要求：数组中的元素个数由函数动态给出，结果通过函数参数形式返回。

2.对冒泡排序的以下算法类C描述，用C语言程序加以实现。要求：不改变排序算法的类C描述框架结构。

void bubble\_sort(int a[],int n)

{

for(i=n-1,change=TRUE;i>=1 && change;--i)

{

change=FALSE;

for (j=0;j<i;++j)

if (a[j]>a[j+1]) {a[j]<---->a[j+1];change=TRUE;}

}

} //bubble sort

3.对给定的一元多项式f(x)，当给定自变量x的值x0时，求f(x0)的值。试设计算法的类C描述，并用C语言程序实现之。

**四、实验准备**

1.计算机设备；

2.程序调试环境的准备，本实验采用**Microsoft Visual Stdio**环境；

3.实验源程序**Exp\_1(1,2)**和**Exp\_1(3)**准备；

4.实验测试用例：

问题1和2：

（1）**n = 10: 10个元素（1 2 3 4 5 6 7 8 9 0）；**

（2）**n = 12: 12个元素（577 21 909 -87 -11 78 12 0 98 43 1000 21）；**

（3）**n = 5: 5个元素（999999999 1 -2 2147483647 2147483648）；**

（4）**n = 6: 6个元素（1 0 -2147483648 -2147483649 -2147483647 -1）。**

问题3：

（1）**n = 5: 一共5项**；

**系数 指数**：**1 2** / **2.5 3** / **4 2** / **3 0 / 2 1**；

**x0 = 1.2。**

（2）**n = 4: 一共4项；**

**系数 指数：-2 3 / 0 2 / 3.4 1 / 0 0；**

**x0 = -2.5。**

**注：本实验中将问题1和2的求解进行了整合。**

**五、实验过程**

**（一）问题分析**

1.针对实验内容1，可以编写一个函数**findMaxAndMin**求解给定数组元素中的最大值和最小值,其中函数参数包括最大值和最小值，此处将其定义为**指针型参数**；

2.针对实验内容2，可以编写一个函数**BubbleSort**以使用“**冒泡法**”对数组元素进行排序，此处选择**升序排序**。注意到问题2的算法类**C**描述中，“**TRUE**”和“**FALSE**”应使用**宏**进行定义，其作用为：**若无元素进行交换，则标记为FALSE，即change = FALSE**;**若有元素进行交换，则标记为TRUE，即change = TRUE**;

3.针对实验内容3：

（1）若要实现一元多项式的求值运算，首先需要确定其数据结构。此处选择**单向链表**结构，类型定义如下：

**typedef struct PolyNode{**

**double coef; //系数**

**int exp; //指数**

**struct PolyNode \*next; //指向下一结点（下一项）的指针**

**}PolyNode, \*Polynomial; //Polynomial生成头节点，PolyNode生成中间结点**

（2）然后确定相关参数的**输入次序**：此处确定先输入一元多项式的项数**n**，然后输入每一项的系数**coef**和指数**exp**。其中系数**coef**和指数**exp**可以通过函数**创建一个新的多项式结点**来进行输入，然后将每个具有给定系数**coef**和指数**exp**的结点再通过另一个函数**插入到多项式链表**中。

（3）结点插入分析：本问题的结点插入**根据每一项的指数大小进行降序插入**，并针对不同情况作出不同处理。例如：如果**新节点的指数已经存在**，**则合并两项系数**；如果**合并后的系数为0，则删除该节点**。

（4）求值运算分析：首先应先初始化一个存储求和结果的变量**result=0.0**，然后可设定一个指针指向待求值的多项式首项。根据一元多项式的求值过程，代入求值点**x0**将每一项计算出来后再进行**累加**，同时伴随指针移动。最终将结果返回到**result**中。

**一元多项式求值的数学公式如下：**

**（二）算法描述**

1.根据以上分析，可将问题1和问题2进行整合。相关算法步骤如下：

【step 1】输入数组**arr**的大小**n**，同时对**n**进行检查；若**n**不合法，则**输出错误提示信息**；

【step 2】**动态分配数组arr的存储空间**，同时**检查是否分配成功**；然后依次输入**n**个数据元素；

【step 3】调用函数**findMaxAndMin**，代入相应参数，**通过参数返回最大值max和最小值min**并进行输出；

【step 4】调用函数**BubbleSort**，对输入的数据元素进行**升序排序**并输出排序结果；

【step 5】释放数组**arr**的存储空间。

2.根据以上分析，问题3的相关算法描述如下：

（1）**Status InitPoly(Polynomial\* poly);**

//本算法用于初始化一个空的多项式链表：将Polynomial\*类型的头指针poly置空，表示初始化后的多项式在结构上没有任何项

（2）**PolyNode\* createNode(double coef, int exp);**

//本算法用于创建结点:其中包含待给定double类型的系数coef，以及int类型的指数exp两个参数，并返回一个指向PolyNode结构体的指针

（3）**void InsertNode(Polynomial\* poly, double coef, int exp);**

//本算法用于将一个新的多项式项插入到多项式链表中，并按照项的指数大小进行降序排列插入：其中包含新项的double类型系数coef和int类型指数exp，以及Polynimial类型的指向多项式链表头指针的指针

（4）**double calculatePoly(Polynomial poly, double x);**

//本算法用于计算多项式的值：接受一个Polynomial类型的头节点poly

以及一个给定的double类型变量x

（5）**void freePoly(Polynomial poly);**

//本算法用于释放多项式链表的内存：接受一个Polynomial类型的头节点poly，逐个结点依次释放内存

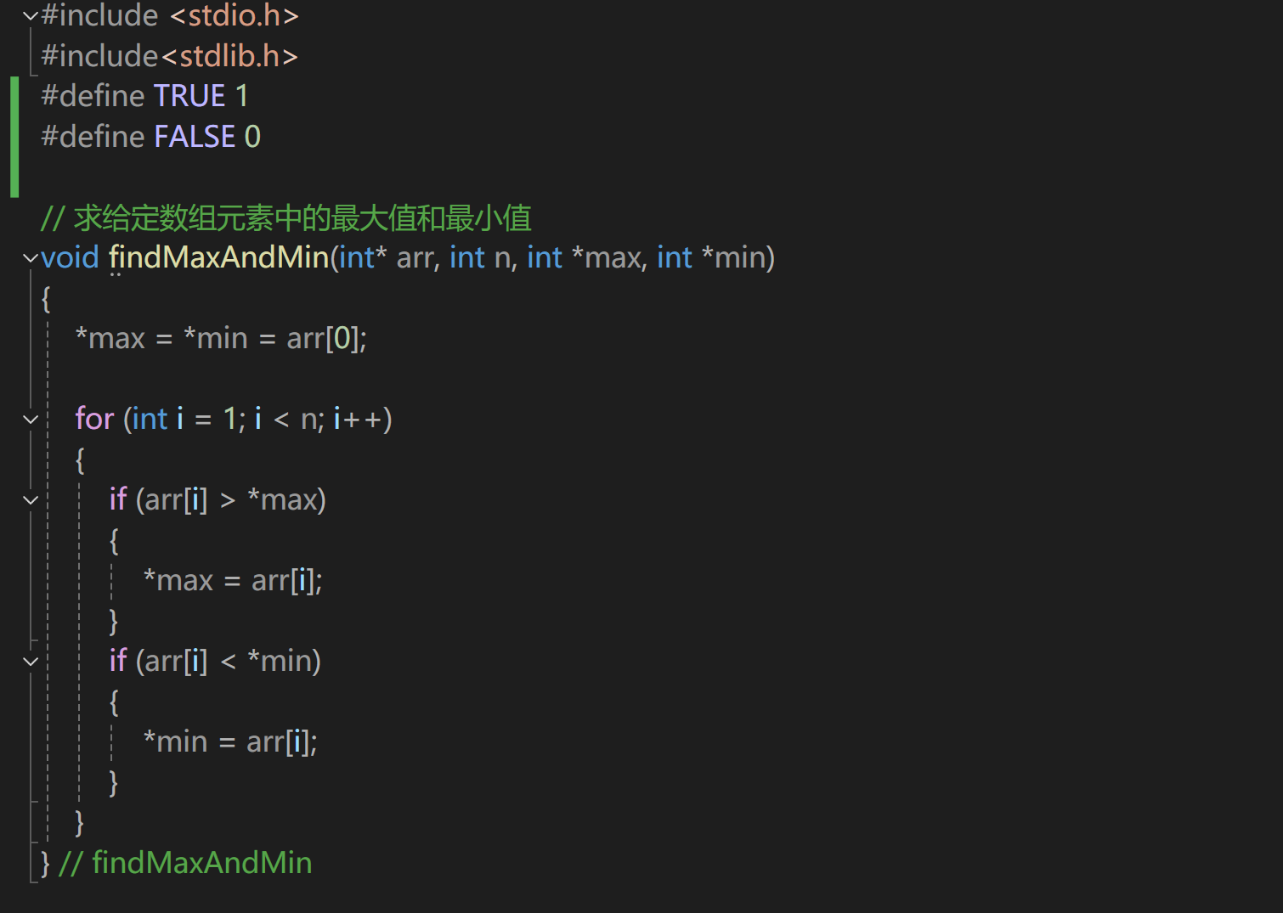
（6）**void printPoly(Polynomial poly);**

//本算法用于输出显示相应的多项式：接受一个Polynomial类型的头节点poly，同时考虑特殊情形下的简化表示

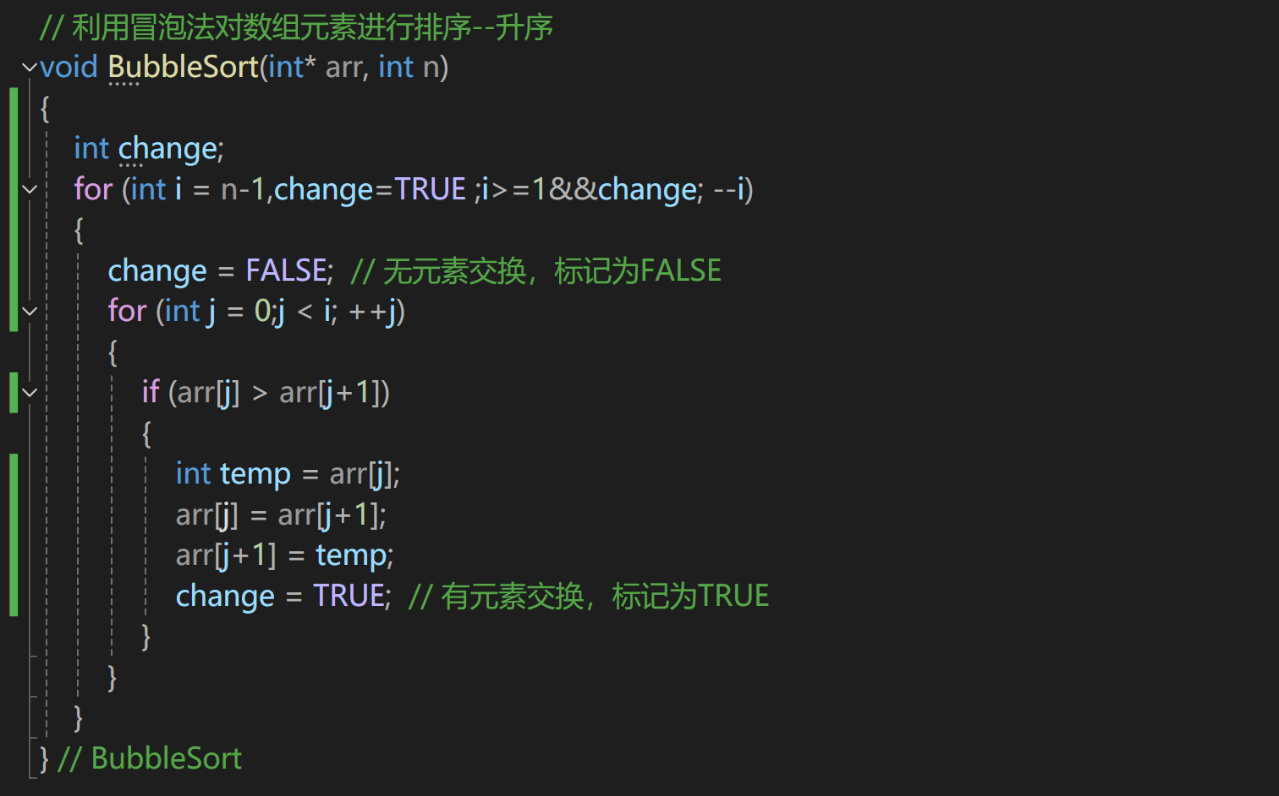
**（三）程序代码**

1.根据以上算法步骤，**问题1和2**的程序代码如下：

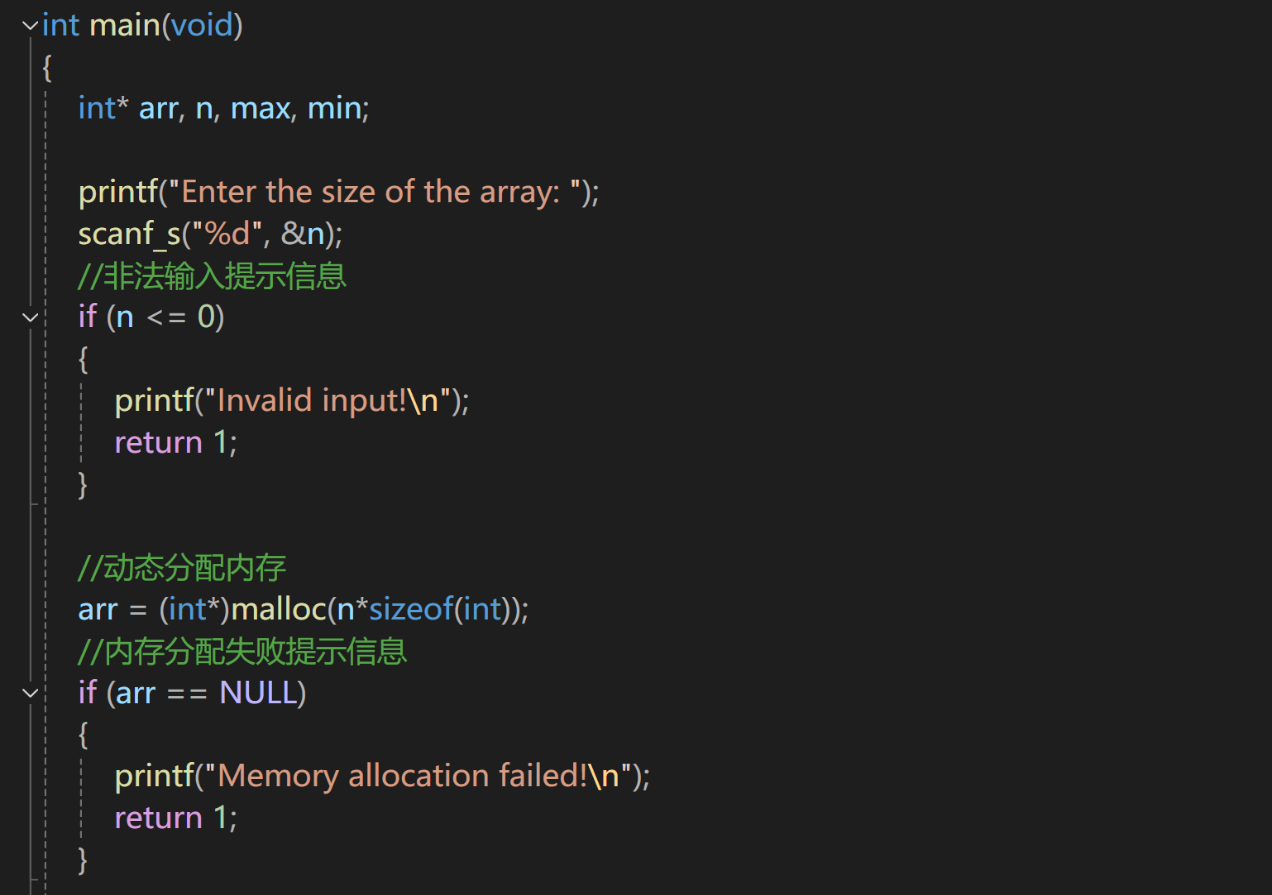




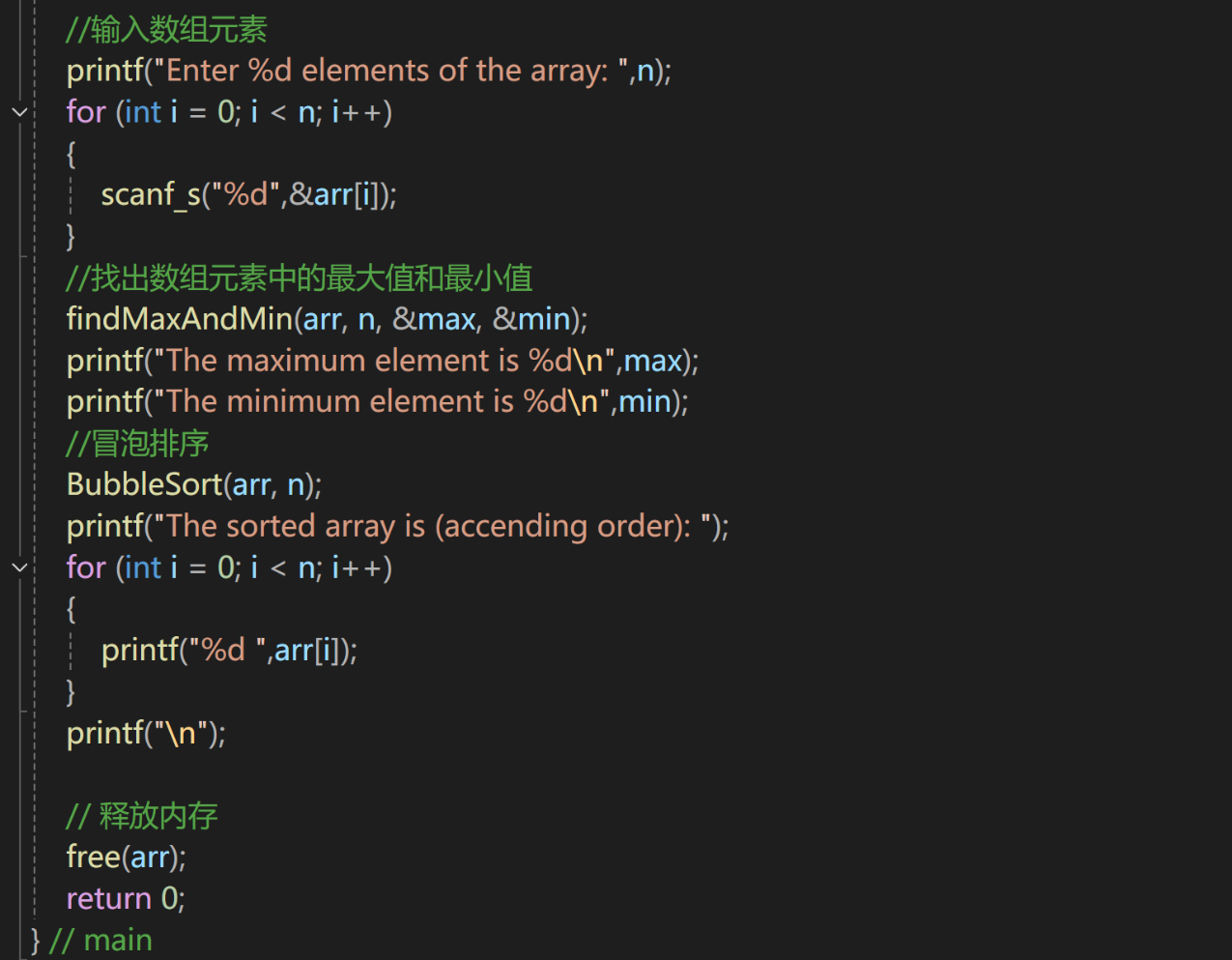
**图5.1**



**图5.2**



**图5.3**



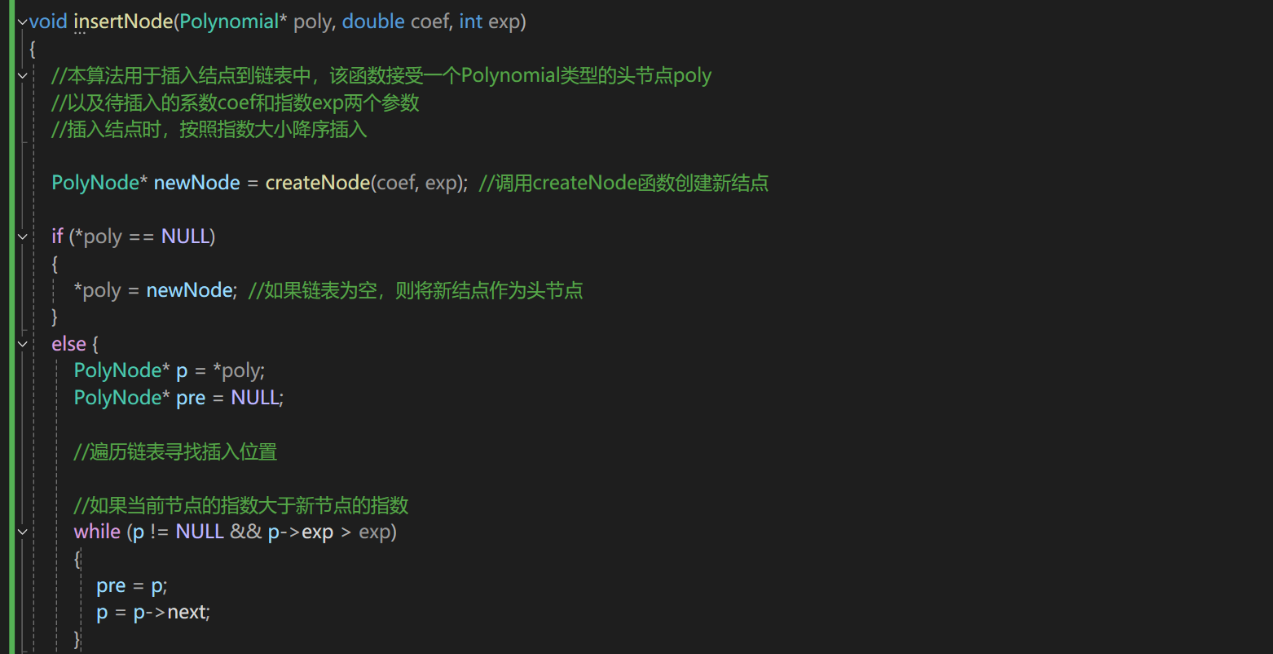
**图5.4**

2.根据以上算法描述，**问题3**的程序代码如下：

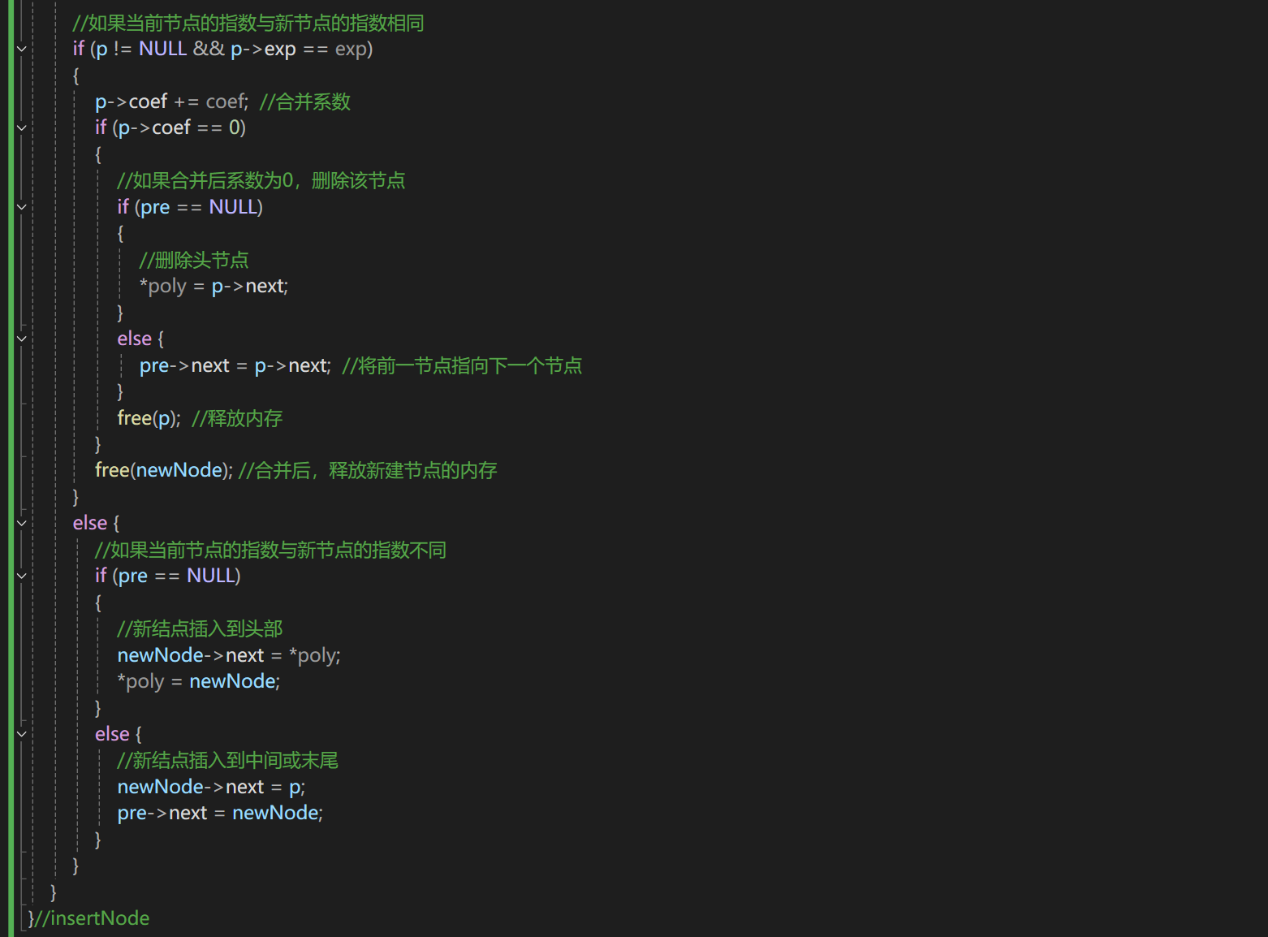




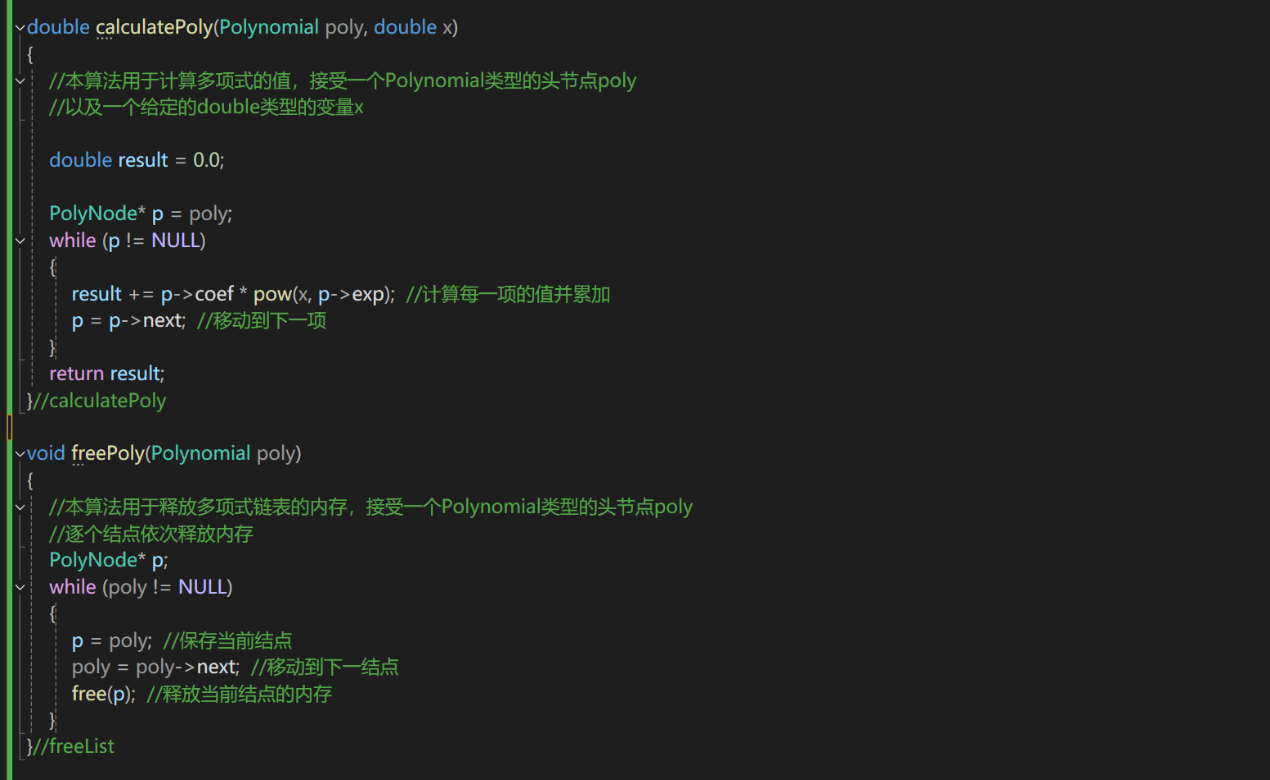
**图5.5**



**图5.6**



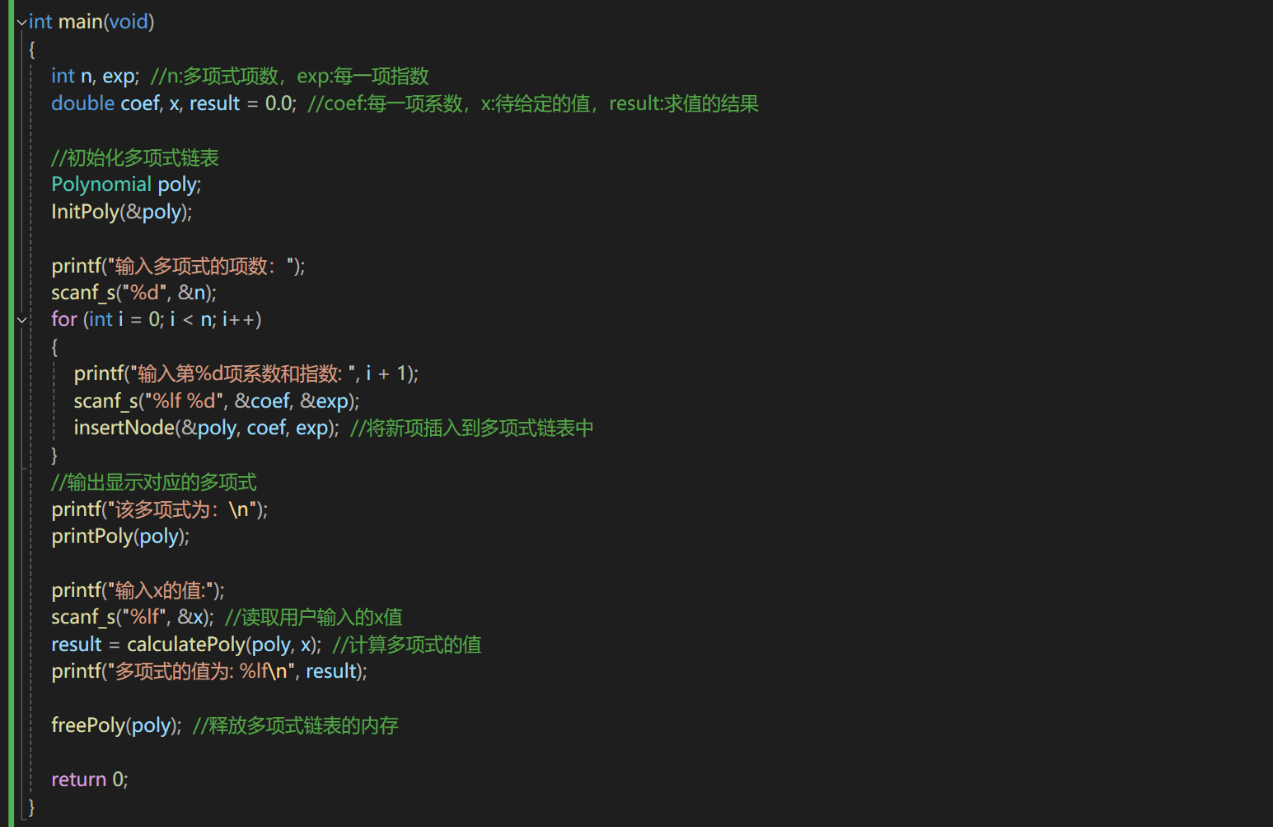
**图5.7**



**图5.8**



**图5.9**

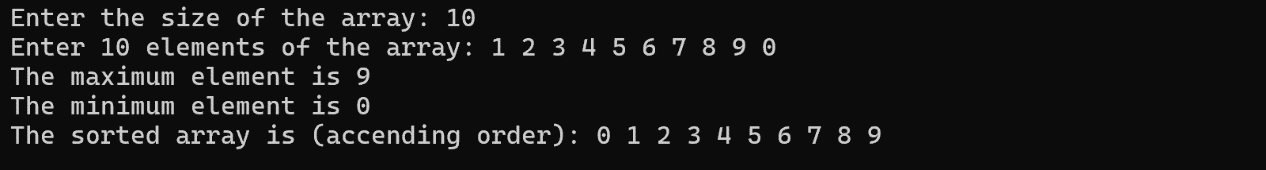


**图5.10**

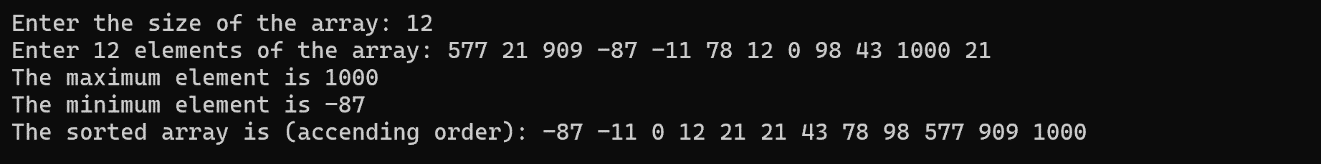
**六、实验结果**

**（一）结果呈现：**

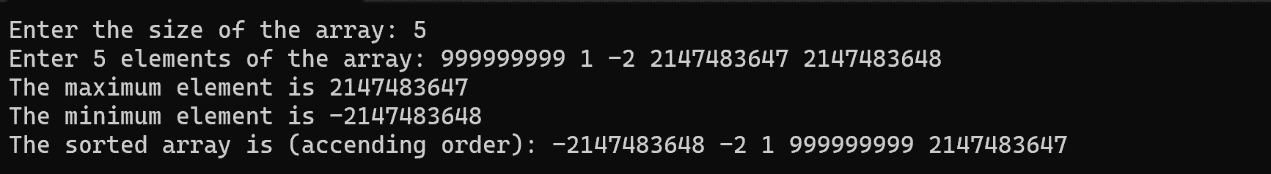
1.根据以上**问题1和2**程序代码运行后进行多次测试，以下是根据**问题1和2的测试用例**进行测试后的结果示意图：



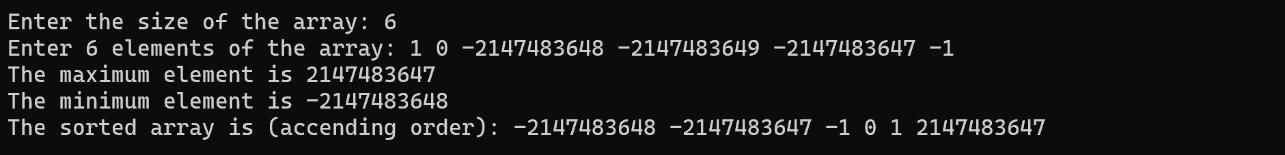
**图6.1（n = 10）**



**图6.2（n = 12）**

****

**图6.3（n = 5）**

****

**图6.4（n = 6）**

**2.**根据以上**问题3**程序代码运行后进行多次测试，以下是根据**问题3的测试用例**进行测试后的结果示意图：

****

**图4.5（n = 5）**

****

**图4.6（n = 4）**

**（二）结果分析**：

**1.基本分析：**

**（1）问题1和2**

测试结果**基本符合预期**，但仍存在一些问题：

在本问题的测试环境下，**int**数据类型的字节数为**4**，一共**32位**长度，取值范围为：

**-2147483648 ~ 2147483647**

当输入数据**超过规定数据类型取值边界**时，即：第三个测试用例中的数据**2147483648**，以及第四个测试用例中的数据**-2147483649**，此时产生了**溢出**，由此在最终排序结果中，相应的**超界数据被视为了负数或正数**。

**（2）问题3**

测试结果**经验算后符合预期**，对于**系数为0**或**系数不为0**等特殊的情况作出了调整。

例如：在第二个测试用例中，包含**系数为0**的项。针对这种情形，本问题中用**‘0’**字符进行代替该项。

**2.时间复杂度分析**：

**（1）问题1和2**

【**findMaxAndMin】—— 寻找最大值和最小值**

在该函数中通过一个循环遍历数组中的每个元素，循环从第一个元素开始，比较每个元素与当前的最大值和最小值。

时间复杂度：**O(n)**，其中**n**是数组的元素数量。因为需要遍历整个数组一次，以找到最大值和最小值。

【**BubbleSort】—— 冒泡排序**

冒泡排序通过两层循环实现。外层循环控制排序的轮数，内层循环通过相邻元素的交换，把当前未排序部分的最大元素“冒泡”到数组的末尾。在最坏的情况下，即数组元素是按降序排列输入时，每次内层循环的比较次数会逐渐减少，最终需要进行**n-1**次比较和交换。

时间复杂度：**O(n2)**，因为最坏情况下需要进行**n\*(n-1)/2**次比较和交换，大约是**O(n2)**。

**综上所述：该程序的总体时间复杂度由较高的冒泡排序决定，总时间复杂度为：O(n2)，其中n是数组元素数量**。

**（2）问题3**

**【InitPoly】—— 初始化多项式链表**

此操作只是简单地设置指针，时间复杂度为**O(1)**，。

**【createNode】—— 创建结点**

为每个插入操作分配一个新的结点，时间复杂度为**O(1)**。

**【insertNode】—— 插入结点**

在最坏情况下，可能需要遍历整个链表以找到插入位置。假设链表的规模为**n**，那么插入操作的时间复杂度为**O(n)**。插入操作总共会被执行n次，所以插入所有项的总时间复杂度为**O(n2)。**

**【calculatePoly】—— 计算多项式的值**

需要遍历整个多项式链表来累加每一项的值，该操作的时间复杂度为**O(n)**。

**【freePoly】—— 释放链表内存**

同样需要遍历整个链表，时间复杂度为**O(n)**。

**综上所述：主要的时间复杂度来源是插入操作，因此整体的时间复杂度为：O(n2)，其中n是多项式的项数。**

**3.空间复杂度分析**：

**（1）问题1和2**

**【动态数组】**

代码中使用**malloc**函数动态分配了一个大小为**n**的整型数组**arr**。因此，这部分的空间复杂度为**O(n)**，因为它直接与输入的数组大小成正比。

**【其余局部变量】**

**n, max, min, change, i, j**等都是常量时间使用的变量，它们的空间占用是常量级的，即**O(1)**。

**综上所述：整体的空间复杂度为O(n)，其中n是数组元素数量。**

**（2）问题3**

**【链表节点的空间**】

每插入一个节点，需要额外的空间来存储该节点，空间复杂度为**O(n)**（每个结点存储系数和指数，以及指向下一个节点的指针）。

**【其余变量和指针】**

虽然一些额外的变量用于存储指针和结果，但它们都是常量级别，因此不影响整体的空间复杂度。

**综上所述：整体的空间复杂度为：O(n)，其中n是多项式的项数。**

**七、感悟体会**

关于**参数传递**，需要注意以下问题;

1.问题1和2中的参数传递主要以**传递地址**为主；其中，函数**findMaxAndMin**的参数采用了**传递指针**和**传递数组名**两种方式。

2.传递指针时，指针变量作为参数，**形参变化影响实参**。例如问题1中数组元素最大值和最小值的求解要求通过函数参数返回，故需要注意在传递指针变量参数**max**和**min**时，传递的是其地址，函数**findMaxAndMin**内部的参数应为**\*max**和**\*min**。而在调用函数**findMaxAndMin**时，由于传递的是地址，故参数应加上‘**&**’符，即**&max**和**&min**。

3.传递数组名时，由于**数组名本身即为该数组的地址**（或首地址），故问题1中的函数**findMaxAndMin**相应参数为**arr[]**或**\*arr**，在调用函数**findMaxAndMin**时，相应参数直接为**arr**。

4.问题3中插入节点时，插入过程十分关键：在插入时，要确保链表按照指数的大小**降序排列**。链表为空时，此时**使新节点成为头节点**。同时使用两个指针**p**和**pre**寻找合适的插入位置：如果当前节点的指数大于新节点的指数，继续移动指针**p**。对于**指数相同**的情形也作出了合适的处理：如果新节点的指数已经存在，则**合并系数**；如果合并后的系数为0，则**删除**该节点。