# 四川师范大学 实验报告

学期： 2025 至 2026 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： 2024 级 9 班 实验编号： 01

实验项目： 实验一 指导老师： 冯山

姓名： 杨晨 学号： 2024070646

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

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

3.实验测试用例：

问题1：

（1）**n = 10: （1 2 3 4 5 6 7 8 9 0）；**

（2）**n = 12:（-11 -9 -7 -5 -3 -1 1 3 5 7 9 11）；**

（3）**n = 5:（100000000 -100000000 -2 2147483647 2147483648）；**

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

问题2；

（1）**n = 10:（1 2 3 4 5 6 7 8 9 10）；**

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

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

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

问题3：

（1）**n = 5:**

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

**x0 = 2。**

（2）**n = 4:**

**系数 指数：-2 3 / 0 2 / 3.4 1 / 0 0；**

**x0 = -2.5。**

（3）**n = 2:**

**系数 指数：-2 -1 / 0 2 ;**

**x0 = -2.5。**

（4）**n = 2:**

**系数 指数：-2 3 / 0 2**

**x0 = -2.5。**

**五、实验过程**

**（一）问题分析**

1.考察实验内容1，

（1）设计数据结构vector：

Typedef struct {

Elemtype \*arr;

Int size;

Int capacity;

}

设计两个函数find\_array\_max和find\_array\_min分别求解给定数组元素中的最大值和最小值,并返回

2.考察实验内容2，

（1）设计一个函数**BubbleSort**以使用“**冒泡法**”对数组元素进行排序，此处选择**升序排序**。

（2）注意到问题2的算法类**C**描述中有布尔值，可以通过导入库<stlbool.h>实现，也可以用宏定义实现。前者导入的是真正的bool数据类型，是标准的实现方式，对非法输入的抵抗力较强。后者仅通过文本替换方式实现，是非标准的实现方式。

（3）设计一个函数实现两个元素的交换

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**顺序表**结构，如（a\_0,a\_1,…,a\_n）其中a\_i为x的第i项前面的系数。

（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最高次项的次数**n**，然后输入每一项的系数**coef**。缺项系数补0

（3）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**。根据一元多项式的求值过程，用递推公式result = a\_i + x0 \* result，迭代result值，最终将结果返回。

**（二）算法描述**

1．**问题1**相关算法步骤如下：

step 1：输入数组**arr**的大小**n**，同时对**n**进行检查是否为合法输入，若**n**不合法，则抛出错误；

step 2：**声明数组arr[n]**；然后依次输入**n**个数据元素，并检查是否为合法输入，若是则赋值给数组，若不合法则抛出错误；

step 3：调用函数find\_array\_max和find\_array\_min，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。

2.**问题2**相关算法步骤如下：

step 1：输入数组**arr**的大小**n**，同时对**n**进行检查是否为合法输入，若**n**不合法，则抛出错误；

step 2：**声明数组arr[n]**；然后依次输入**n**个数据元素，并检查是否为合法输入，若是则赋值给数组，若不合法则抛出错误；

step 3：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；

3**.问题3**的相关算法描述如下：

step 1：输入数组**arr**的大小**n**，同时对**n**进行检查是否为合法输入，若**n**不合法，则抛出错误；

step 2：**声明数组arr[n]，**并赋初值为0，然后输入次数exp和系数coef，检查是否为合法输入，不合法则抛出错误，将coef赋值给第exp个元素，约定exp为-1时为退出条件，

step 3：调用函数polynomial\_evaluation，对多项式进行求解

step 4:输出结果。

**（三）程序代码**

1.根据以上算法步骤，**问题1**的程序代码如下：

#include<stdio.h>

#include<limits.h>

int find\_array\_max(int arr[], int n) {

/\*

Function to find the maximum value in an array.

Parameters:

arr - array of integers

n - number of elements in the array

Returns:

Maximum integer value in the array

time complexity: O(n)

space complexity: O(1)

\*/

int max = arr[0];

for(int i = 1; i < n; i++) {

if(arr[i] > max) {

max = arr[i];

}

}

return max;

}

int find\_array\_min(int arr[], int n) {

/\*

Function to find the minimum value in an array.

Parameters:

arr - array of integers

n - number of elements in the array

Returns:

Minimum integer value in the array

time complexity: O(n)

space complexity: O(1)

\*/

int min = arr[0];

for(int i = 1; i < n; i++) {

if(arr[i] < min) {

min = arr[i];

}

}

return min;

}

int main() {

int n;

printf("Enter the number of elements in the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d integers(between %d and %d):\n", n, INT\_MIN, INT\_MAX);

for(int i = 0; i < n; i++) {

long long temp;

scanf("%lld", &temp);

if (temp > INT\_MAX || temp < INT\_MIN) {

printf("Error: Input value %lld is out of int range.\n", temp);

return -1;

}

arr[i] = (int)temp;

}

int max\_value = find\_array\_max(arr, n);

int min\_value = find\_array\_min(arr, n);

printf("The maximum value is: %d\n", max\_value);

printf("The minimum value is: %d\n", min\_value);

return 0;

}

2. 根据以上算法步骤，**问题2**的程序代码如下：

#include<stdio.h>

#include<limits.h>

#include<stdbool.h>

void swap(int \*a, int \*b) {

/\*

Function to swap two integers.

Parameters:

a - pointer to the first integer

b - pointer to the second integer

time complexity: O(1)

space complexity: O(1)

\*/

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void bulltle\_sort(int arr[], int n) {

/\*

Function to sort an array using bubble sort algorithm.

Parameters:

arr - array of integers

n - number of elements in the array

time complexity: O(n^2)

space complexity: O(1)

\*/

bool change = true;

for(int i = n-1; i >=1 && change; i--) {

change = false;

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

if(arr[j] > arr[j+1]) {

swap(&arr[j], &arr[j+1]);

change = true;

}

}

}

}

int main() {

int n;

printf("Enter the number of elements in the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d integers(between %d and %d):\n", n, INT\_MIN, INT\_MAX);

for(int i = 0; i < n; i++) {

long long temp;

scanf("%lld", &temp);

if (temp > INT\_MAX || temp < INT\_MIN) {

printf("Error: Input value %lld is out of int range.\n", temp);

return -1;

}

arr[i] = (int)temp;

}

bulltle\_sort(arr, n);

printf("Sorted array: \n");

for(int i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

3.**问题3**的程序代码如下：

#include <stdio.h>

#include <limits.h>

#include <float.h>

int main(void){

int n;

printf("input the degree of polynomial:");

scanf("%d",&n);

double arr[n+1];

// Initialize array elements to 0

for(int i=0;i<=n;i++){

arr[i]=0;

}

// Input degree and coefficients

printf("input degree and the coefficients of term corresponding to degree (input -1 to end):\n");

for(int i=0;i<=n;i++){

long degree;

double coefficient;

scanf("%ld %lf",&degree,&coefficient);

if(degree==-1 && coefficient==-1){

break;

}

if(degree<0 || degree>n){

printf("Error: Input degree %ld is out of range.\n",degree);

return -1;

}

if(coefficient>DBL\_MAX || coefficient< -DBL\_MAX){

printf("Error: Input coefficient %lf is out of double range.\n",coefficient);

return -1;

}

arr[degree]=coefficient;

}

// Input x0

printf("input x0(between %d and %d):", INT\_MIN, INT\_MAX);

long long x0;

scanf("%lld",&x0);

if(x0>INT\_MAX || x0<INT\_MIN){

printf("Error: Input value %lld is out of int range.\n",x0);

return -1;

}

// Calculate polynomial value using Horner's method

double result=0;

for(int i=n;i>=0;i--){

result = result \* x0 + arr[i];

}

printf("P(%lld) = %lf\n", x0, result);

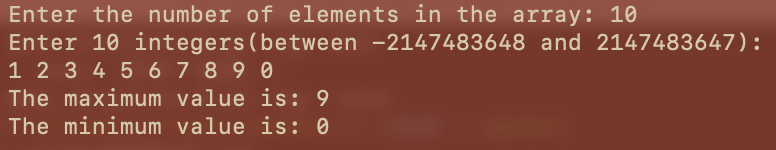
return 0;

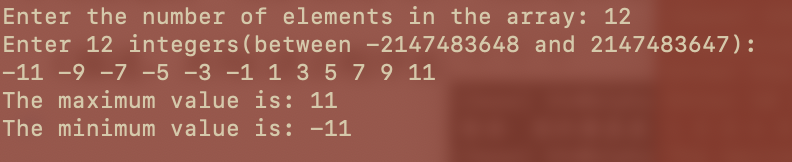
}

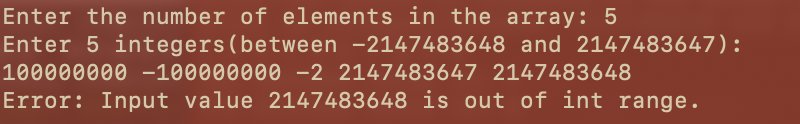
**六、实验结果**

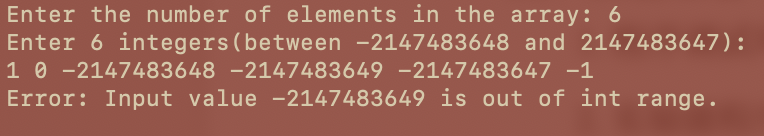
**（一）结果呈现：**

1.以下是问题一根据**测试用例**进行测试后的结果:

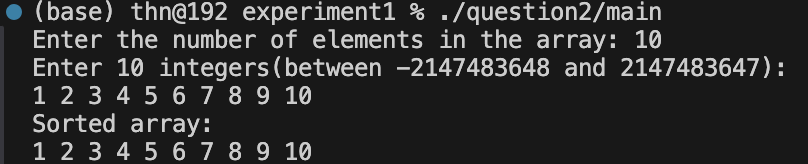


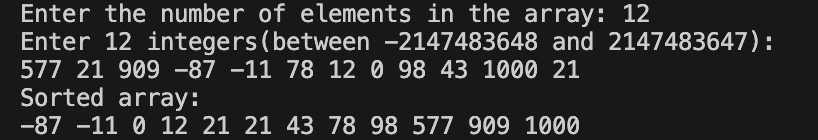


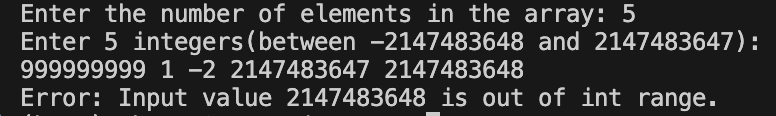


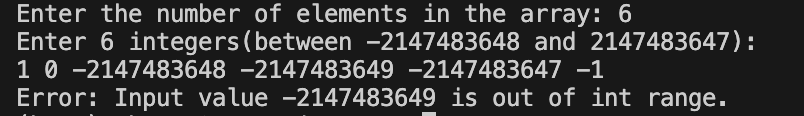


2.以下是问题二根据**测试用例**进行测试后的结果:

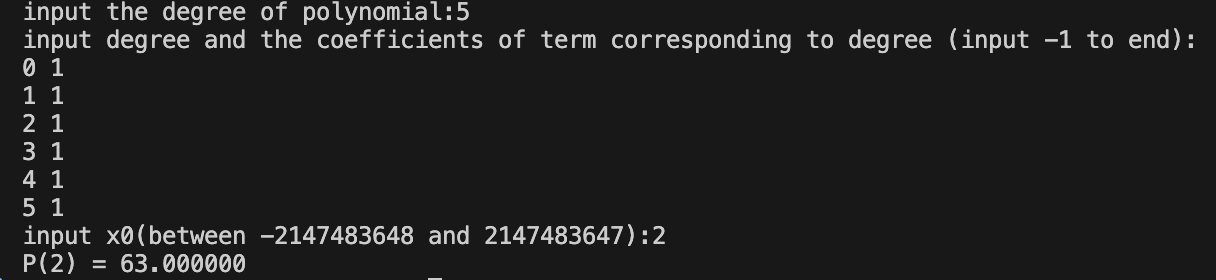


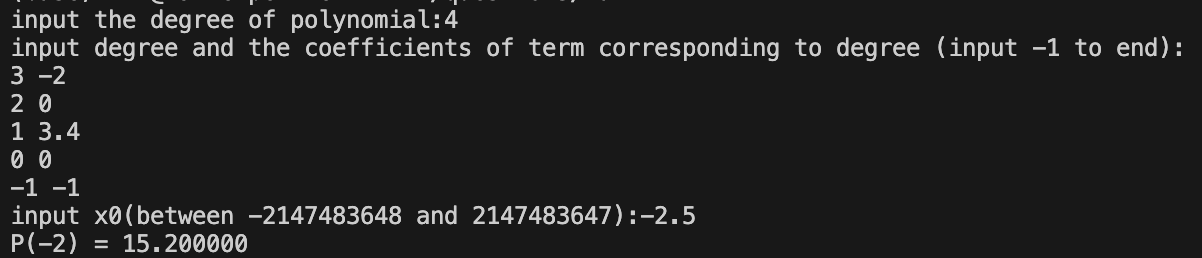


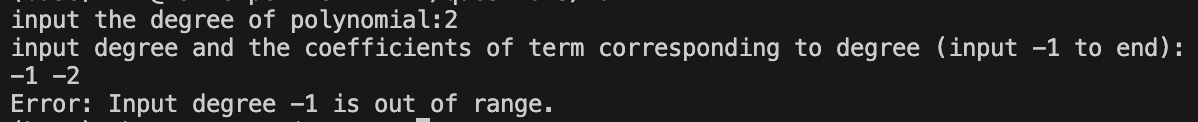


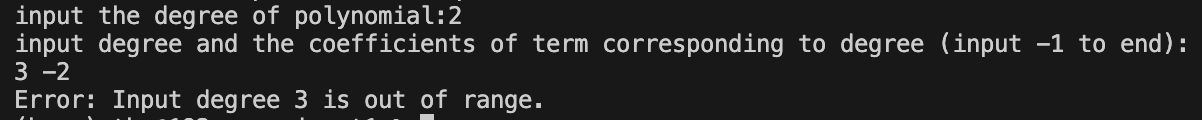


**2.**以下是根据**问题3的测试用例**进行测试后的结果：

****

****

****

****

**（二）结果分析**：

**1.基本分析：**

**（1）问题1**

测试结果**符合预期**，非法输入错误抛出正常

**（2）问题2**

测试结果**符合预期**，非法输入错误抛出正常

**（3）问题3**

测试结果**经验算后符合预期，**非法输入错误抛出正常

**2.时间复杂度分析**：

**（1）问题1**

【find\_array\_max和find\_array\_min**】—— 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。

**（2）问题2**

【**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：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**。

**（3）问题3**

**用递推公式求解多项式的值，只需要遍历一遍系数数组就行**

**故该程序时间复杂度为O(n)**

**3.空间复杂度分析**：

**（1）问题1和2**

**【数组】**

声明了一个大小为**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。

**（2）问题3**

**【系数矩阵**】

声明了一个大小为**n**的double型数组**arr**。因此，这部分的空间复杂度为**O(n)**，因为它直接与多项式的最高次项次数成正比。

**综上所述：整体的空间复杂度为：O(n)，其中n是多项式的最高次项的次数。**

**七、感悟体会**

写抛出异常好麻烦，不如python的try好用。