Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

ОТЧЁТ

по лабораторной работе №1 (Week 1 Openedu)

Студенка Кузенкова Елизавета группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

Задача 1 «a+b»

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В данной задаче требуется вычислить сумму двух заданных чисел.

Формат входного файла

Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a, b \le 10^9$.

Формат выходного файла

В выходной файл выведите единственное целое число — результат сложения.

Примеры

input.txt	output.txt
23 11	34
-100 1	-99

Исходный код к задаче 1

```
public static void Main(string[] args)
{
     int a = 0, b = 0, c = 0;

     using (var file = new System.IO.StreamReader("input.txt"))
     {
         var input = file.ReadLine().Split(' ');
         a = int.Parse(input[0]);
         b = int.Parse(input[1]);
     }

     using (var file = new System.IO.StreamWriter("output.txt"))
     {
         c = a + b;
         file.WriteLine(c);
     }
}
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	10121216	25	13
1	OK	0.015	10076160	7	4
2	OK	0.031	10121216	8	5
3	OK	0.031	10063872	5	3
4	OK	0.015	10084352	5	3
5	OK	0.015	10059776	6	3
6	OK	0.015	10031104	9	6
7	OK	0.046	10076160	23	12
8	OK	0.015	10051584	25	13
9	OK	0.031	10022912	24	3
10	OK	0.015	10063872	24	3
11	OK	0.015	10080256	14	12
12	OK	0.015	10035200	23	12
13	OK	0.031	10047488	23	13
14	OK	0.015	10059776	20	11
15	OK	0.031	10059776	23	13
16	OK	0.031	10063872	20	11
17	OK	0.031	10088448	22	12
18	OK	0.031	10047488	23	13
19	OK	0.031	10088448	22	12
20	OK	0.031	10100736	22	12
21	OK	0.031	10076160	22	12

Задача 2 «a+b^2»

1.0 из 1.0 балла (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В данной задаче требуется вычислить значение выражения $a + b^2$.

Формат входного файла

Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a, \mathrm{b} \le 10^9$.

Формат выходного файла

В выходной файл выведите единственное целое число — результат вычисления выражения $a + b^2$.

Примеры

input.txt	output.txt
23 11	144
-100 1	-99

Исходный код к задаче 2

```
public static void Main(string[] args)
{
     long a = 0, b = 0, c = 0;

     using (var file = new System.IO.StreamReader("input.txt"))
     {
         var input = file.ReadLine().Split(' ');
         a = int.Parse(input[0]);
         b = int.Parse(input[1]);
     }

     using (var file = new System.IO.StreamWriter("output.txt"))
     {
         c = a + b * b;
         file.WriteLine(c);
     }
}
```

Бенчмарк к задаче 2

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	10117120	25	21
1	OK	0.031	10072064	7	5
2	OK	0.031	10072064	8	5
3	OK	0.031	10076160	5	3
4	OK	0.015	10113024	5	3
5	OK	0.031	10117120	6	3
6	OK	0.031	10059776	6	3
7	OK	0.031	10067968	23	21
8	OK	0.031	10059776	25	20
9	OK	0.015	10096640	24	20
10	OK	0.015	10084352	24	21
11	OK	0.031	10084352	23	20
12	OK	0.015	10067968	23	20
13	OK	0.015	10067968	20	17

14	OK	0.031	10100736	23	20
15	OK	0.015	10088448	20	20
16	OK	0.031	10084352	22	20
17	OK	0.015	10080256	23	20
18	OK	0.015	10076160	22	19
19	OK	0.031	10067968	22	19
20	OK	0.015	10100736	22	20

Задача 3 Сортировка вставками

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью сортировки вставками.

Сортировка вставками проходится по всем элементам массива от меньших индексов к большим («слева направо») для каждого элемента определяет его место в предшествующей ему отсортированной части массива и переносит его на это место (возможно, сдвигая некоторые элементы на один индекс вправо). Чтобы проконтролировать, что Вы используете именно сортировку вставками, мы попросим Вас для каждого элемента массива после того, как он будет обработан, выводить его новый индекс.

Формат входного файла

В первой строке входного файла содержится число $n(1 \leq n \leq 1000)$ — число элементов в массиве. Во второй строке находятся различных целых чисел, по модулю не превосходящих 10^9.

Формат выходного файла

В первой строке выходного файла выведите n чисел. При этом i-ое число равно индексу, на который, **в момент обработки его сортировкой вставками**, был перемещен i-ый элемент **исходного массива**. Индексы нумеруются, начиная с единицы. Между любыми двумя числами должен стоять ровно один пробел.

Во второй строке выходного файла выведите отсортированный массив. Между любыми двумя числами должен стоять ровно один пробел.

Пример

input.txt	output.txt
10 1 8 4 2 3 7 5 6 9 0	1 2 2 2 3 5 5 6 9 1 0 1 2 3 4 5 6 7 8 9

Комментарий к примеру

В примере сортировка вставками работает следующим образом:

1. Первый элемент остается на своем месте, поэтому первое число в ответе — единица. Отсортированная часть массива: [1]
2. Второй элемент больше первого, поэтому он тоже остается на своем месте, и второе число в ответе — двойка. [1 8]
3. Четверка меньше восьмерки, поэтому занимает второе место. [1 4 8]
4. Двойка занимает второе место. [1 2 4 8]
5. Тройка занимает третье место. [1 2 3 4 8]
6. Семерка занимает пятое место. [1 2 3 4 7 8]
7. Пятерка занимает пятое место. [1 2 3 4 5 7 8]
8. Шестерка занимает шестое место. [1 2 3 4 5 6 7 8]
9. Девятка занимает девятое место. [1 2 3 4 5 6 7 8 9]
10. Ноль занимает первое место. [0 1 2 3 4 5 6 7 8 9]

Исходный код к задаче 3

```
public class Lab1_3
{
    public int[] InsertionSort(int[] array, int[] indexesDiff)
    {
        int[] result = new int[array.Length];
        for (int i = 0; i < array.Length; i++)
        {
            int j = i;
            while (j > 0 && result[j - 1] > array[i])
            {
                result[j] = result[j - 1];
                j--;
            }
            indexesDiff[i] = j;
            result[j] = array[i];
        }
        return result;
    }

    private void DoWork(string[] args)
    {
        int[] array;
        int[] indexesDiff;
        using (var file = new System.IO.StreamReader("input.txt"))
        {
            var count = int.Parse(file.ReadLine());
            var input = file.ReadLine().Split(' ');
            array = new int[count];
            indexesDiff = new int[count];
            for (int i = 0; i < count; i++)
            {
                array[i] = int.Parse(input[i]);
            }
        }
```

```
        var result = this.InsertionSort(array, indexesDiff);

        using (var file = new System.IO.StreamWriter("output.txt"))
        {
            for (int i = 0; i < array.Length; i++)
            {
                file.Write($"{indexesDiff[i] + 1} ");
            }

            file.WriteLine();

            for (int i = 0; i < array.Length; i++)
            {
                file.Write($"{result[i]} ");
            }
        }
    }

        public static void Main(string[] args)
        {
            var app = new Lab1_3();
            app.DoWork(args);
        }
    }
```

Бенчмарк к задаче 3

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	10588160	10415	14298
1	OK	0.031	10264576	25	42
2	OK	0.015	10211328	7	7
3	OK	0.031	10149888	12	14
4	OK	0.031	10215424	8	10
5	OK	0.031	10203136	10	14
6	OK	0.015	10211328	29	33
7	OK	0.015	10215424	10	14
8	OK	0.031	10199040	10	14
9	OK	0.031	10145792	10	14
10	OK	0.015	10149888	10	14
11	OK	0.046	10182656	10	14
12	OK	0.031	10186752	57	65
13	OK	0.031	10264576	56	64
14	OK	0.031	10199040	57	65
15	OK	0.046	10182656	77	89
16	OK	0.031	10256384	76	88
17	OK	0.015	10285056	77	89
18	OK	0.031	10186752	112	129
19	OK	0.031	10219520	111	129
20	OK	0.031	10194944	110	127

21	OK	0.031	10203136	949	1192
22	OK	0.031	10256384	960	1221
23	OK	0.031	10227712	957	1136
24	OK	0.015	10252288	1490	1890
25	OK	0.031	10293248	1486	1946
26	OK	0.015	10235904	1481	1763
27	OK	0.031	10407936	3723	4890
28	OK	0.031	10309632	3729	5049
29	OK	0.031	10346496	3727	4439
30	OK	0.031	10498048	8456	11340
31	OK	0.031	10575872	8471	11611
32	OK	0.031	10563584	8415	10037
33	OK	0.031	10559488	10415	14037
34	OK	0.015	10588160	10410	14298
35	OK	0.031	10584064	10393	12388

Задача 4 Знакомство с жителями Сортлэнда

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Владелец графства Сортлэнд, граф Бабблсортер, решил познакомиться со своими подданными. Число жителей в графстве нечетно и составляет n, где n может быть достаточно велико, поэтому граф решил ограничиться знакомством с тремя представителями народонаселения: с самым бедным жителем, с жителем, обладающим средним достатком, и с самым богатым жителем.

Согласно традициям Сортлэнда, считается, что житель обладает средним достатком, если при сортировке жителей по сумме денежных сбережений он оказывается ровно посередине. Известно, что каждый житель графства имеет уникальный идентификационный номер, значение которого расположено в границах от единицы до n. Информация о размере денежных накоплений жителей хранится в массиве M таким образом, что сумма денежных накоплений жителя, обладающего идентификационным номером i, содержится в ячейке M[i]. Помогите секретарю графа мистеру Свопу вычислить идентификационные номера жителей, которые будут приглашены на встречу с графом.

Формат входного файла

Первая строка входного файла содержит число жителей n ($3 \leq n \leq 9999, n$ (, нечетно). Вторая строка содержит описание массива M, состоящее из положительных

вещественных чисел, разделенных пробелами. Гарантируется, что все элементы массива M различны, а их значения имеют точность не более двух знаков после запятой и не превышают 10^6

Формат выходного файла

В выходной файл выведите три целых положительных числа, разделенных пробелами — идентификационные номера беднейшего, среднего и самого богатого жителей Сортлэнда.

Пример

input.txt	output.txt
5 10.00 8.70 0.01 5.00 3.00	3 4 1

Комментарий к примеру

Если отсортировать жителей по их достатку, получится следующий массив:

[0.01, 3] [3.00, 5] [5.00, 4] [8.70, 2] [10.00, 1]

Здесь каждый житель указан в квадратных скобках, первое число — его достаток, второе число — его идентификационный номер. Таким образом, самый бедный житель имеет номер 3, самый богатый — номер 1, а средний — номер 4.

Исходный код к задаче 4

```
public class Lab1_4
{
    public double[] InsertionSort(double[] array, int[] ids)
    {
        double[] result = new double[array.Length];
        for (int i = 0; i < array.Length; i++)
        {
            int j = i;
            while (j > 0 && result[j - 1] > array[i])
            {
                var t = ids[j];
                ids[j] = ids[j - 1];
                ids[j - 1] = t;
                result[j] = result[j - 1];
                j--;
            }
            result[j] = array[i];
        }
        return result;
    }

    private void DoWork(string[] args)
    {
        double[] array;
        int[] ids;
        using (var file = new System.IO.StreamReader("input.txt"))
        {
            var count = int.Parse(file.ReadLine());
            var input = file.ReadLine().Split(' ');
            array = new double[count];
            ids = new int[count];
```

```
            for (int i = 0; i < ids.Length; i++)
            {
                ids[i] = i + 1;
            }
            for (int i = 0; i < count; i++)
            {
                array[i] = double.Parse(input[i]);
            }
        }

        var result = this.InsertionSort(array, ids);
        using (var file = new System.IO.StreamWriter("output.txt"))
        {
            file.Write($"{ids[0]} {ids[ids.Length / 2]} {ids[ids.Length - 1]}");
        }
    }

    public static void Main(string[] args)
    {
        var app = new Lab1_4();
        app.DoWork(args);
    }
}
```

Бенчмарк к задаче 4

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.109	11464704	98892	14
1	OK	0.031	10526720	30	5
2	OK	0.031	10493952	33	5
3	OK	0.015	10289152	1065	8
4	OK	0.031	10334208	3732	10
5	OK	0.031	10485760	14975	13
6	OK	0.031	10584064	14998	11
7	OK	0.031	10661888	28749	14
8	OK	0.031	10694656	34791	12
9	OK	0.046	10706944	38037	13
10	OK	0.046	10731520	38074	14
11	OK	0.031	10928128	39288	13
12	OK	0.046	10854400	48638	13
13	OK	0.046	10911744	50722	12
14	OK	0.109	10891264	52757	14
15	OK	0.062	10973184	58008	13
16	OK	0.078	11157504	66504	14
17	OK	0.078	11149312	71786	14
18	OK	0.062	11153408	72346	14
19	OK	0.078	11210752	73304	13
20	OK	0.062	11288576	76139	14
21	OK	0.078	11321344	83944	14

22	OK	0.078	11333632	85179	13
23	OK	0.093	11317248	86522	12
24	OK	0.093	11329536	89202	13
25	OK	0.093	11464704	98892	14

Задача 5 Секретарь Своп

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Уже знакомый нам из предыдущей задачи граф Бабблсортер поручил своему секретарю, мистеру Свопу, оформлять приглашения беднейшему, богатейшему и среднему по достатку жителю своих владений. Однако кто же, в отсутствие мистера Свопа, будет заниматься самым важным делом — сортировкой массивов чисел? Видимо, это придется сделать Вам!

Дан массив, состоящий из

целых чисел. Вам необходимо его отсортировать по неубыванию. Но делать это нужно так же, как это делает мистер Своп — то есть, каждое действие должно быть взаимной перестановкой пары элементов. Вам также придется записать все, что Вы делали, в файл, чтобы мистер Своп смог проверить Вашу работу.

Формат входного файла

В первой строке входного файла содержится число $n(1 \le n \le 5000)$ — число элементов в массиве. Во второй строке находятся n целых чисел, по модулю не превосходящих 10^9. Числа могут совпадать друг с другом.

Формат выходного файла

В первых нескольких строках выведите осуществленные Вами операции перестановки элементов. Каждая строка должна иметь следующий формат:

```
Swap elements at indices X and Y.
```

где X и Y — различные индексы массива, элементы на которых нужно переставить $(1 \le X, Y \le n)$. Мистер Своп любит порядок, поэтому сделайте так, чтобы X < Y.

После того, как все нужные перестановки выведены, выведите следующую фразу:

```
No more swaps needed.
```

Во последней строке выходного файла выведите отсортированный массив, чтобы мистер Своп не переделывал работу за Вас. Между любыми двумя числами должен стоять ровно один пробел.

Пример

input.txt	output.txt
5 3 1 4 2 2	Swap elements at indices 1 and 2. Swap elements at indices 2 and 4. Swap elements at indices 3 and 5. No more swaps needed. 1 2 2 3 4

Послесловие и предостережение

Семья секретаря Свопа занималась сортировками массивов, и именно с помощью перестановок пар элементов, как минимум с XII века, поэтому все Свопы владеют этим искусством в совершенстве. Мы не просим Вас произвести минимальную последовательность перестановок, приводящую к правильному ответу. Однако учтите, что для вывода слишком длинной последовательности у Вашего алгоритма может не хватить времени (или памяти — если выводимые строки хранятся в памяти перед выводом). Подумайте, что с этим можно сделать. Решение существует!

Исходный код к задаче 5

```
public class Lab1_5
{
    public void Quicksort(double[] elements, int left, int right, StreamWriter file)
    {
        int i = left, j = right;
        var pivot = elements[(left + right) / 2];
        while (i <= j)
        {
            while (elements[i].CompareTo(pivot) < 0)
            {
                i++;
            }

            while (elements[j].CompareTo(pivot) > 0)
            {
                j--;
            }

            if (i <= j)
            {
                // Swap
                var tmp = elements[i];
                elements[i] = elements[j];
                elements[j] = tmp;

                if (i != j)
                    file.WriteLine($"Swap elements at indices {i + 1} and {j + 1}.");

                i++;
                j--;
            }
        }
```

```
        // Recursive calls
        if (left < j)
        {
            this.Quicksort(elements, left, j, file);
        }

        if (i < right)
        {
            this.Quicksort(elements, i, right, file);
        }
    }

    private void DoWork(string[] args)
    {
        double[] array;
        using (var file = new System.IO.StreamReader("input.txt"))
        {
            var count = int.Parse(file.ReadLine());
            var input = file.ReadLine().Split(' ');
            array = new double[count];
            for (int i = 0; i < count; i++)
            {
                array[i] = double.Parse(input[i]);
            }
        }

        using (var file = new System.IO.StreamWriter("output.txt"))
        {
            this.Quicksort(array, 0, array.Length - 1, file);
            file.WriteLine("No more swaps needed.");
            file.WriteLine(string.Join(" ", array));
        }
    }

    public static void Main(string[] args)
    {
        var app = new Lab1_5();
        app.DoWork(args);
    }
}
```

Бенчмарк к задаче 5

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	12546048	51993	635372
1	OK	0.031	10616832	14	139
2	OK	0.031	10674176	7	27
3	OK	0.031	10711040	12	32
4	OK	0.015	10665984	8	63
5	OK	0.015	10678272	10	65
6	OK	0.031	10620928	10	30
7	OK	0.031	10657792	29	49
8	OK	0.015	10657792	10	65
9	OK	0.031	10702848	10	65
10	OK	0.031	10657792	10	100
11	OK	0.031	10633216	10	65

12	OK	0.031	10657792	10	100
13	OK	0.031	10715136	50	210
14	OK	0.031	10715136	56	181
15	OK	0.015	10620928	57	77
16	OK	0.031	10649600	55	145
17	OK	0.031	10633216	75	305
18	OK	0.031	10625024	76	96
19	OK	0.031	10764288	78	203
20	OK	0.031	10678272	108	268
21	OK	0.046	10633216	107	126
22	OK	0.031	10653696	108	303
23	OK	0.031	10743808	948	5932
24	OK	0.031	10665984	947	966
25	OK	0.031	10682368	948	2623
26	OK	0.015	10989568	3720	31952
27	OK	0.031	10752000	3735	3753
28	OK	0.015	10903552	3722	10613
29	OK	0.031	11452416	8463	78938
30	OK	0.031	10899456	8441	8459
31	OK	0.031	11030528	8434	24178
32	OK	0.031	11919360	22822	252665
33	OK	0.031	11165696	22825	22842
34	OK	0.031	11530240	22877	66846
35	OK	0.046	12546048	51987	635372
36	OK	0.031	11755520	51940	51957
37	OK	0.031	12169216	51993	153403