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 Неделя №1

OpenEdu

Группа: P3217
Выполнил: Минин Александр

г. Санкт-Петербург
2018г.

Задача №1 a+b

Условие

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 $-10^9 \le a \le 10^9$, $-10^9 \le b \le 10^9$.

Формат выходного файла

В выходной файл выведите единственное целое число — результат сложения $a + b$.

Решение

```
public class Week1_1 {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new FileReader("input.txt"));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter("output.txt"));
        String[] str = bufferedReader.readLine().split(" ");

        int a = Integer.valueOf(str[0]);
        int b = Integer.valueOf(str[1]);
        int sum = a + b;

        bufferedWriter.write(sum + "\n");
        bufferedWriter.flush();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Результаты бенчмарка:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		171	20983808	25	12
1	OK	125	20914176	7	3
2	OK	125	20844544	8	4
3	OK	109	20926464	5	2
4	OK	109	20971520	5	2
5	OK	109	20869120	6	2
6	OK	125	20955136	9	5
7	OK	109	20983808	23	11
8	OK	125	20910080	25	12
9	OK	125	20910080	24	2
10	OK	109	20930560	24	2

11	OK	109	20922368	14	11
12	OK	125	20951040	23	11
13	OK	171	20934656	23	12
14	OK	109	20905984	20	10
15	OK	140	20897792	23	12
16	OK	125	20926464	20	10
17	OK	109	20877312	22	11
18	OK	140	20959232	23	12
19	OK	125	20955136	22	11
20	OK	125	20930560	22	11
21	OK	125	20869120	22	11

Задача №2 a+b^2

Условие

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 $-10^9 \le a \le 10^9$, $-10^9 \le b \le 10^9$.

Формат выходного файла

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

Решение

```
public class Week1_2 {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new FileReader("input.txt"));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter("output.txt"));
        String[] str = bufferedReader.readLine().split(" ");

        long a = Integer.valueOf(str[0]);
        long b = Integer.valueOf(str[1]);
        long result = a + b * b;

        bufferedWriter.write(result + "\n");
        bufferedWriter.flush();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		156	20975616	25	20

1	OK	109	20881408	7	4
2	OK	125	20975616	8	4
3	OK	125	20873216	5	2
4	OK	125	20856832	5	2
5	OK	109	20918272	6	2
6	OK	109	20918272	6	2
7	OK	109	20893696	23	20
8	OK	125	20975616	25	19
9	OK	109	20967424	24	19
10	OK	109	20922368	24	20
11	OK	109	20959232	23	19
12	OK	109	20865024	23	19
13	OK	109	20910080	20	16
14	OK	109	20971520	23	19
15	OK	125	20897792	20	19
16	OK	109	20959232	22	19
17	OK	125	20971520	23	19
18	OK	156	20938752	22	18
19	OK	109	20922368	22	18
20	OK	125	20967424	22	19

Задача №3 Сортировка вставками

Условие

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 n 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

Решение

```
package week1;

import java.io.*;

public class Week1_3 {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new FileReader("input.txt"));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter("output.txt"));

        int arraySize = Integer.valueOf(bufferedReader.readLine());
        int[] array = new int[arraySize];
        String[] arrayStr = bufferedReader.readLine().split(" ");
        for (int i = 0; i < arraySize; i++) {
            array[i] = Integer.valueOf(arrayStr[i]);
        }

        for (int i = 0; i < arraySize; i++) {
            int j = i;
            while (j > 0 && array[j] < array[j - 1]) {
                int tmp = array[j];
                array[j] = array[j - 1];
                array[j - 1] = tmp;
                j--;
```

```
            }
            bufferedWriter.write(j + 1 + " ");
        }
        bufferedWriter.write("\n");
        for (int i = 0; i < arraySize; i++) {
            bufferedWriter.write(Integer.toString(array[i]));
            if (i + 1 == arraySize) {
                bufferedWriter.write("\n");
            } else {
                bufferedWriter.write(" ");
            }
        }

        bufferedWriter.flush();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		171	2288435	10415	14297
1	OK	93	2091008	25	41
2	OK	125	2096742	7	6
3	OK	125	2091827	12	13
4	OK	109	2095104	8	9
5	OK	125	2087321	10	13
6	OK	125	2093465	29	32
7	OK	109	2095104	10	13
8	OK	125	2090598	10	13
9	OK	140	2091008	10	13
10	OK	109	2093465	10	13
11	OK	125	2091827	10	13
12	OK	125	2095104	57	64
13	OK	109	2095104	56	63
14	OK	125	2089369	57	64
15	OK	125	2092236	77	88
16	OK	171	2092236	76	87
17	OK	125	2092236	77	88
18	OK	156	2092646	112	128
19	OK	125	2097971	111	128
20	OK	125	2095104	110	126

21	OK	140	20983808	949	1191
22	OK	125	21004288	960	1220
23	OK	125	21032960	957	1135
24	OK	171	20967424	1490	1889
25	OK	156	21028864	1486	1945
26	OK	140	21016576	1481	1762
27	OK	125	21135360	3723	4889
28	OK	140	21164032	3729	5048
29	OK	125	21561344	3727	4438
30	OK	140	22216704	8456	11339
31	OK	156	22138880	8471	11610
32	OK	171	22515712	8415	10036
33	OK	140	22286336	10415	14036
34	OK	140	22294528	10410	14297
35	OK	171	22884352	10393	12387

Задача №4

Условие

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 ($3 \le n \le 9999$, n нечетно). Вторая строка содержит описание массива M, состоящее из n 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.

Решение

```
package week1;

import java.io.*;

public class Week1_4 {

    private static <T> void swap(T[] array, int i, int j) {
        T tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
    }

    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new FileReader("input.txt"));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter("output.txt"));

        int n = Integer.parseInt(bufferedReader.readLine());
        String[] moneyStrArray = bufferedReader.readLine().split(" ");
        Integer[] aIdentifiers = new Integer[n];
        Double[] aMoney = new Double[n];
        for (int i = 0; i < n; i++) {
            aIdentifiers[i] = i + 1;
            aMoney[i] = Double.parseDouble(moneyStrArray[i]);
        }

        for (int i = 0; i < n; i++) {
            int j = i;
            while (j > 0 && aMoney[j] < aMoney[j - 1]) {
                swap(aMoney, j, j - 1);
                swap(aIdentifiers, j, j - 1);
                j--;
            }
        }

        bufferedWriter.write(aIdentifiers[0] + " ");
        bufferedWriter.write(aIdentifiers[n / 2] + " ");
        bufferedWriter.write(aIdentifiers[n - 1].toString());

        bufferedWriter.flush();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		421	27086848	98892	14
1	OK	125	21065728	30	5
2	OK	125	20983808	33	5
3	OK	156	21114880	1065	8

4	OK	140	22638592	3732	10
5	OK	140	23379968	14975	13
6	OK	187	23416832	14998	11
7	OK	234	24686592	28749	14
8	OK	218	24911872	34791	12
9	OK	234	25063424	38037	13
10	OK	250	25182208	38074	14
11	OK	218	24936448	39288	13
12	OK	234	25509888	48638	13
13	OK	281	25542656	50722	12
14	OK	250	25600000	52757	14
15	OK	265	25653248	58008	13
16	OK	265	26284032	66504	14
17	OK	296	26292224	71786	14
18	OK	281	26394624	72346	14
19	OK	281	26378240	73304	13
20	OK	281	26390528	76139	14
21	OK	312	26603520	83944	14
22	OK	296	26705920	85179	13
23	OK	312	26636288	86522	12
24	OK	375	26742784	89202	13
25	OK	421	27086848	98892	14

Задача №5

Условие

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 n 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

Решение

```
package week1;

import java.io.*;

public class Week1_5 {


    private static BufferedReader bufferedReader;
    private static BufferedWriter bufferedWriter;

    private static void swap(int[] array, int i, int j) throws IOException {
        int tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
        bufferedWriter.write("Swap elements at indices " + (i+1) + " and " + (j+1) + ".\n");
    }

    private static void quickSort(int[] array, int startIndex, int endIndex) throws IOException {
        if (endIndex < startIndex) {
            return;
        }

        int pivot = startIndex + (endIndex - startIndex) / 2; // startIndex <= pivot < endIndex
```

```
        int l = startIndex;
        int r = endIndex;
        while (l < r) {
            while (l < pivot) {
                if (array[l] > array[pivot]) {
                    swap(array, l, pivot);
                    pivot = l;
                } else {
                    l++;
                }
            }

            while (r > pivot) {
                if (array[r] < array[pivot]) {
                    swap(array, pivot, r);
                    pivot = r;
                } else {
                    r--;
                }
            }
        }

        quickSort(array, startIndex, pivot - 1);
        quickSort(array, pivot + 1, endIndex);
    }

    public static void main(String[] args) throws IOException {
        bufferedReader = new BufferedReader(new FileReader("input.txt"));
        bufferedWriter = new BufferedWriter(new FileWriter("output.txt"));

        int arraySize = Integer.valueOf(bufferedReader.readLine());
        int[] array = new int[arraySize];
        String[] arrayStr = bufferedReader.readLine().split(" ");

        for (int i = 0; i < arraySize; i++) {
            array[i] = Integer.valueOf(arrayStr[i]);
        }

        quickSort(array, 0, array.length - 1);

        bufferedWriter.write("No more swaps needed.\n");

        for (int i = 0; i < arraySize; i++) {
            bufferedWriter.write(array[i] + " ");
        }

        bufferedWriter.flush();
        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Результаты бенчмарка

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		421	27086848	98892	14
1	OK	125	21065728	30	5

2	OK	125	20983808	33	5
3	OK	156	21114880	1065	8
4	OK	140	22638592	3732	10
5	OK	140	23379968	14975	13
6	OK	187	23416832	14998	11
7	OK	234	24686592	28749	14
8	OK	218	24911872	34791	12
9	OK	234	25063424	38037	13
10	OK	250	25182208	38074	14
11	OK	218	24936448	39288	13
12	OK	234	25509888	48638	13
13	OK	281	25542656	50722	12
14	OK	250	25600000	52757	14
15	OK	265	25653248	58008	13
16	OK	265	26284032	66504	14
17	OK	296	26292224	71786	14
18	OK	281	26394624	72346	14
19	OK	281	26378240	73304	13
20	OK	281	26390528	76139	14
21	OK	312	26603520	83944	14
22	OK	296	26705920	85179	13
23	OK	312	26636288	86522	12
24	OK	375	26742784	89202	13
25	OK	421	27086848	98892	14