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 Жетесова Дана 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

```
#include <iostream>
#include <fstream>

int main()
{
	std::ifstream infile("input.txt");
	std::ofstream outfile("output.txt");
	int a, b;
	infile >> a >> b;
	infile.close();
	outfile << (a + b);
	outfile.close();
	return 0;
}
```

Бенчмарк к задаче 1

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	10121216	25	13
Max		0.031	2375680	25	11
1	OK	0.000	2363392	7	2
2	OK	0.000	2363392	8	3

3	OK	0.000	2363392	5	1
4	OK	0.000	2363392	5	1
5	OK	0.015	2363392	6	1
6	OK	0.015	2363392	9	4
7	OK	0.000	2363392	23	10
8	OK	0.015	2359296	25	11
9	OK	0.015	2363392	24	1
10	OK	0.015	2367488	24	1
11	OK	0.015	2359296	14	10
12	OK	0.000	2363392	23	10
13	OK	0.000	2359296	23	11
14	OK	0.015	2375680	20	9
15	OK	0.000	2359296	23	11
16	OK	0.015	2375680	20	9
17	OK	0.031	2359296	22	10
18	OK	0.000	2363392	23	11
19	OK	0.015	2363392	22	10
20	OK	0.000	2375680	22	10

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

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

```
#include <iostream>
#include <fstream>

int main()
{
	std::ifstream infile("input.txt");
	std::ofstream outfile("output.txt");
	long long a, b;
	infile >> a >> b;
	infile.close();
	outfile << (a + b * b);
	outfile.close();
	return 0;
}
```

Бенчмарк к задаче 2

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.015	2605056	25	19
1	OK	0.015	2588672	7	3
2	OK	0.000	2367488	8	3
3	OK	0.000	2367488	5	1
4	OK	0.000	2367488	5	1
5	OK	0.015	2367488	6	1
6	OK	0.015	2588672	6	1
7	OK	0.000	2592768	23	19
8	OK	0.000	2592768	25	18
9	OK	0.015	2600960	24	18
10	OK	0.015	2605056	24	19
11	OK	0.015	2592768	23	18
12	OK	0.000	2592768	23	18
13	OK	0.000	2363392	20	15
14	OK	0.000	2379776	23	18
15	OK	0.000	2367488	20	18
16	OK	0.000	2367488	22	18
17	OK	0.000	2379776	23	18
18	OK	0.000	2379776	22	17
19	OK	0.000	2367488	22	17
20	OK	0.000	2367488	22	18

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

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

int main()
{
	ifstream infile("input.txt");
	ofstream outfile("output.txt");
	int n, curValue;
	vector<int> arrayList;
	vector<int> indexList;
	infile >> n;
	for (int i = 0; i < n; i++) {
		infile >> curValue;
		arrayList.push_back(curValue); // добавляем в список новые элементы
	}
	infile.close();
	indexList.push_back(1);
	for (int i = 1; i < n; i++) { //сортируем методом вставок
		int j = i - 1;
		while (j >= 0 && arrayList[j] > arrayList[j + 1]) {
			curValue = arrayList[j];
			arrayList[j] = arrayList[j + 1];
			arrayList[j + 1] = curValue;
			j--;
		}
		indexList.push_back(j + 2);
	}
	for (int i = 0; i < n - 1; i++) {
		outfile << indexList[i] << " "; //вывод индексов
	}
	outfile << indexList[n - 1];
	outfile << endl;
	for (int i = 0; i < n - 1; i++) {
		outfile << arrayList[i] << " "; //вывод массива результа
	}
	outfile << arrayList[n - 1];
	outfile.close();
	return 0;
}
```

Бенчмарк к задаче 3

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	2588672	10415	14296
1	OK	0.000	2588672	25	40
2	OK	0.000	2375680	7	5
3	OK	0.015	2363392	12	12

4	OK	0.000	2584576	8	8
5	OK	0.015	2363392	10	12
6	OK	0.000	2379776	29	31
7	OK	0.000	2363392	10	12
8	OK	0.015	2363392	10	12
9	OK	0.000	2379776	10	12
10	OK	0.000	2363392	10	12
11	OK	0.015	2363392	10	12
12	OK	0.031	2375680	57	63
13	OK	0.015	2379776	56	62
14	OK	0.000	2359296	57	63
15	OK	0.000	2588672	77	87
16	OK	0.015	2367488	76	86
17	OK	0.015	2371584	77	87
18	OK	0.015	2367488	112	127
19	OK	0.000	2375680	111	127
20	OK	0.015	2367488	110	125
21	OK	0.015	2363392	949	1190
22	OK	0.000	2363392	960	1219
23	OK	0.000	2359296	957	1134
24	OK	0.000	2367488	1490	1888
25	OK	0.000	2371584	1486	1944
26	OK	0.015	2367488	1481	1761
27	OK	0.000	2375680	3723	4888
28	OK	0.000	2367488	3729	5047
29	OK	0.000	2375680	3727	4437
30	OK	0.015	2375680	8456	11338
31	OK	0.015	2371584	8471	11609
32	OK	0.015	2371584	8415	10035
33	OK	0.000	2375680	10415	14035
34	OK	0.000	2387968	10410	14296
35	OK	0.015	2371584	10393	12386

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

```
#include <iostream>
#include <fstream>
#include <vector>
//#include <iterator>
#include <map>
using namespace std;
```

```
int main()
{
	ifstream infile("input.txt");
	ofstream outfile("output.txt");
	int n;
	float curValue;
	vector<float> moneyList;
	map<float, int> moneyToIndex;
	infile >> n;
	for (int i = 0; i < n; i++) {
		infile >> curValue;
		moneyList.push_back(curValue); // добавляем в список новые элементы
		moneyToIndex[curValue] = i + 1; // добавляем элементы в словарь: (значение
элемента -> его индекс в исходном файле)
	}
	infile.close();

	// сортировка вставками
	for (int i = 1; i < n; i++) {
		int j = i - 1;
		while (j >= 0 && moneyList[j] > moneyList[j + 1]) {
			curValue = moneyList[j];
			moneyList[j] = moneyList[j + 1];
			moneyList[j + 1] = curValue;
			j--;
		}
	} // вывод индексов первого, среднего и последнего элементов отсортированного
массива через словарь
	outfile << moneyToIndex[moneyList[0]] << " " <<
		moneyToIndex[moneyList[(n - 1) / 2]] << " " <<
		moneyToIndex[moneyList[n - 1]];
	outfile.close();

	/*
	map <float, int> :: iterator it = moneyToIndex.begin();
	outfile << it->second << " ";
	advance(it, (n - 1) / 2);
	outfile << it->second << " ";
	advance(it, (n - 1) / 2);
	outfile << it->second;
	outfile.close();

	почему-то автоматическая сортировка map на большом количестве элементов (5861) не
работает, вероятно это немного оптимальнее
	а так это была бы очень хитрая оптимизация)
	*/

	return 0;
}
```

Бенчмарк к задаче 4

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.062	2666496	98892	14
1	OK	0.000	2396160	30	5
2	OK	0.000	2396160	33	5
3	OK	0.015	2396160	1065	8
4	OK	0.000	2416640	3732	10
5	OK	0.015	2510848	14975	13

6	OK	0.015	2461696	14998	11
7	OK	0.015	2445312	28749	14
8	OK	0.015	2441216	34791	12
9	OK	0.031	2482176	38037	13
10	OK	0.015	2478080	38074	14
11	OK	0.015	2433024	39288	13
12	OK	0.015	2555904	48638	13
13	OK	0.015	2580480	50722	12
14	OK	0.031	2560000	52757	14
15	OK	0.031	2633728	58008	13
16	OK	0.031	2498560	66504	14
17	OK	0.046	2535424	71786	14
18	OK	0.046	2482176	72346	14
19	OK	0.046	2547712	73304	13
20	OK	0.046	2555904	76139	14
21	OK	0.062	2584576	83944	14
22	OK	0.046	2592768	85179	13
23	OK	0.062	2588672	86522	12
24	OK	0.062	2592768	89202	13
25	OK	0.062	2666496	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 $n(1 \leq n \leq 5000)$ — 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 $(1 \leq X, Y \leq n)$. 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

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

int main()
{
	ifstream infile("input.txt");
```

```
	ofstream outfile("output.txt");
	int n, curValue;
	vector<int> arrayList;
	infile >> n;
	for (int i = 0; i < n; i++) {
		infile >> curValue;
		arrayList.push_back(curValue); // добавляем в список новые элементы
	}
	infile.close();
	// сортировка перестановками
	for (int i = 0; i < n - 1; ++i) {
		int min_index = i;
		for (int j = i + 1; j < n; ++j) {
			if (arrayList[min_index] > arrayList[j]) { // ищем минимальное число
в срезе массива от текущего индекса до конца
				min_index = j;
			}
		}
		if (i != min_index) { // минимальный элемент в срезе должен оказаться по
текущему индексу. он или уже там, или делаем перестановку
			curValue = arrayList[i];
			arrayList[i] = arrayList[min_index];
			arrayList[min_index] = curValue;
			outfile << "Swap elements at indices " << i + 1 << " and " <<
min_index + 1 << "." << endl; // ыводим перестановку
		}
	}
	outfile << "No more swaps needed.";
	outfile << endl;
	for (int i = 0; i < n - 1; i++) {
		outfile << arrayList[i] << " ";
	}
	outfile << arrayList[n - 1];
	outfile.close();
	return 0;
}
```

Бенчмарк к задаче 5

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.046	2379776	51993	255415
1	OK	0.000	2371584	14	137
2	OK	0.000	2359296	7	25
3	OK	0.000	2359296	12	30
4	OK	0.015	2359296	8	61
5	OK	0.015	2363392	10	28
6	OK	0.000	2355200	10	28
7	OK	0.015	2355200	29	47
8	OK	0.000	2359296	10	63
9	OK	0.000	2371584	10	63
10	OK	0.000	2355200	10	98
11	OK	0.000	2359296	10	63
12	OK	0.015	2355200	10	98
13	OK	0.000	2371584	50	138
14	OK	0.000	2359296	56	179
15	OK	0.000	2359296	57	75

16	OK	0.000	2371584	55	143
17	OK	0.000	2359296	75	303
18	OK	0.015	2371584	76	94
19	OK	0.031	2359296	78	201
20	OK	0.000	2355200	108	266
21	OK	0.000	2359296	107	124
22	OK	0.031	2367488	108	301
23	OK	0.000	2359296	948	4175
24	OK	0.000	2359296	947	964
25	OK	0.000	2359296	948	2621
26	OK	0.000	2359296	3720	17382
27	OK	0.000	2375680	3735	3751
28	OK	0.000	2363392	3722	10611
29	OK	0.015	2379776	8463	39802
30	OK	0.000	2371584	8441	8457
31	OK	0.015	2379776	8434	24176
32	OK	0.015	2379776	22822	111244
33	OK	0.015	2379776	22825	22840
34	OK	0.031	2379776	22877	66844
35	OK	0.046	2359296	51987	255415
36	OK	0.031	2363392	51940	51955
37	OK	0.046	2359296	51993	153401