## Федеральное государственное автономное образовательное учреждение высшего образования

## **Санкт-Петербургский национальный исследовательский университет**

# информационных технологий, механики и оптики

# Кафедра ИПМ

# Лабораторная работа №1

**Сортировки за O(n2)**

# Выполнил

Студент 2 курса

# Группы P3218

Гхази Даниэль

# Принял:

К.т.н.

Старший преподаватель

Симоненко Зинаида

Григорьевна

Санкт-Петербург

2016

**Цель работы:**

Для выполнения лабораторной работы необходимо сгенерировать тестовые файлы (используя генераторы случайных чисел), содержащие целые числа, в количестве от 26 до 220 , реализовать алгоритмы используя один из следующих языков программирования: C++, C#, C, Python, для каждого тестового файла из набора выполнить сортировку данных, вычислить среднее время сортировки по одному файлу, построить график зависимости времени сортировки от количества элементов в файле, выполнить сравнение алгоритмов

**Генератор исходных данных:**

intmain()

{

srand(time(NULL));

FILE \*file;

if ((file = fopen("mas.txt", "wt")) == NULL)

return 1;

for (inti = 0; i<ARR\_SIZE; i++)

fprintf(file, "%i", rand() % ARR\_SIZE);

fclose(file);

return 0;

}

**Сортировки:**

**Пузырьковая**

#defineARR\_SIZE 100000

int temp, mas1[ARR\_SIZE];

for (inti = 0; i<ARR\_SIZE - 1; i++)

{

for (int j = 0; j <ARR\_SIZE - 1; j++)

{

if (mas1[j] > mas1[j+1])

{

temp = mas1[j];

mas1[j] = mas1[j+1];

mas1[j+1] = temp;

}

}

}

**Методомвыбора:**

#defineARR\_SIZE 100000

int temp, mas1[ARR\_SIZE];

for (inti = 0; i<ARR\_SIZE - 1; i++)

{

int min = i;

for (int j = i + 1; j <ARR\_SIZE; j++)

{

if (mas1[j] < mas1[min])

min = j;

}

temp = mas1[i];

mas1[i] = mas1[min];

mas1[min] = temp;

}

**Простымивставками:**

#defineARR\_SIZE 100000

int temp, j, curElemVal,mas1[ARR\_SIZE];

for (inti = 0; i<ARR\_SIZE; i++)

{

curElemVal = mas1[i];

for (j = i - 1; (j >= 0) && (mas1[j] >curElemVal); j--)

mas1[j + 1] = mas1[j];

mas1[j + 1] = curElemVal;

}

**Результаты:**

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Пузырьковая, с | Выбором, с | Простыми вставками, с |
| 16588 | 3,201 | 1,799 | 2,193 |
| 32993 | 13,643 | 7,234 | 8,901 |
| 69544 | 53,426 | 27,127 | 34,623 |
| 141023 | 234,221 | 128,126 | 150,506 |

**Выводы:**

В результате сравнения алгоритмов на большом количестве значений удалось выяснить, что самой эффективной из представленных сортировок является сортировка выбором, которая отсортировывает наибольшее количество элементов за минимальное время.