Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЁТ**

по лабораторной работе № 2

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Оценка времени выполнения программ»

Выполнили ст. гр. 22ВВ2:

Рябов С. И.

Китаев Я. Е.

Аргаткин А. А.

Приняли:

Митрохин М.А.

Акифьев И.В

ПЕНЗА 2023

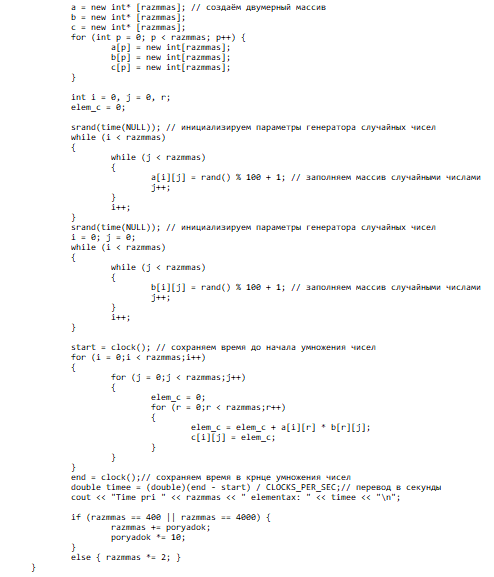
**Цель работы**

Научиться анализировать программы, определять их сложность. Научиться считать время работы при помощи библиотеки time.h.

**Лабораторное задание**

**Задание 1:**

1) Вычислить порядок сложности программы (О-символику).



O(n3)

O(n2)

O(n2)

O(n2)

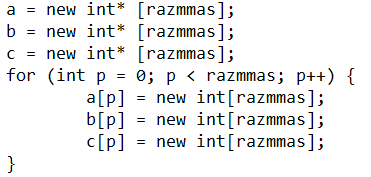
Сложность: O(n3)

2) Оценить время выполнения программы и кода, выполняющего перемножение матриц, используя функции библиотеки time.h для матриц размерами от 100, 200, 400, 1000, 2000, 4000, 10000.

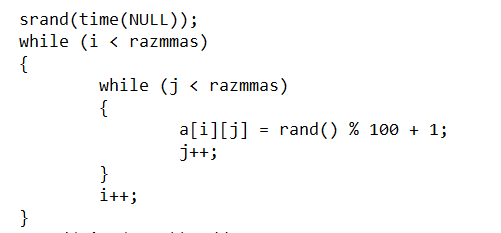
3) Построить график зависимости времени выполнения программы от размера матриц и сравнить полученный результат с теоретической оценкой.

**Описание программы**

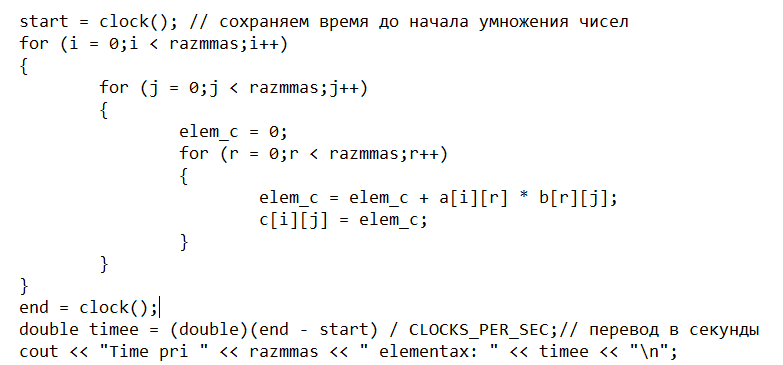
Сначала создаем двумерный массив, размер которого сможем изменять в ходе действия программы.



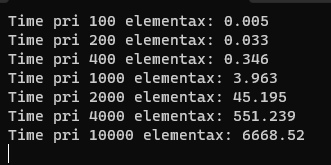
Заполняем массив случайными числами, а также делаем это со вторым.



Выполняем перемножение, заранее начав подсчёт времени, когда программа завершается, выводится время выполнения всех операций.



После этого изменяем размер массива и выполняем программу снова.



В результате видно, что при больших количествах элементов, операция перемножения выполняется очень долго.

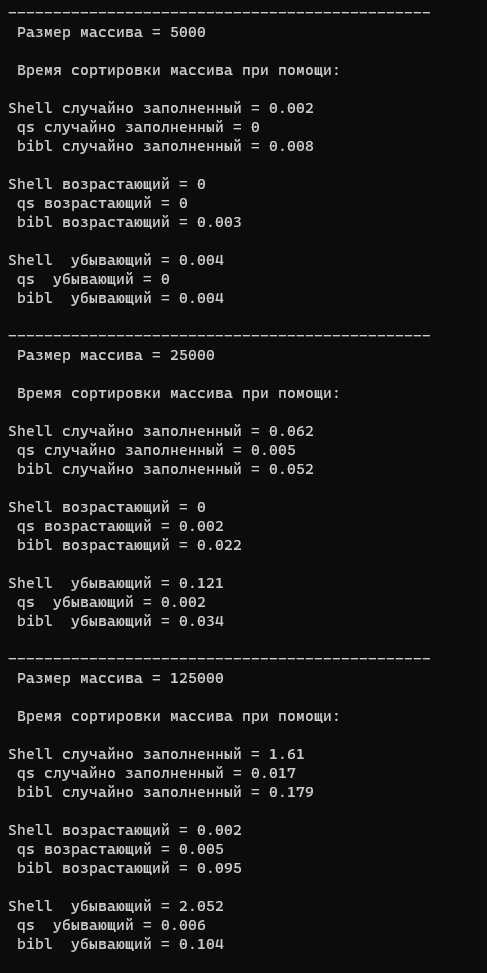


График изменяется по параболической зависимости, что соответствует его сложности (n2).

**Задание 2:**

1. Оценить время работы каждого из реализованных алгоритмов на случайном наборе значений массива.
2. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой возрастающую последовательность чисел.
3. Оценить время работы каждого из реализованных алгоритмов на массиве, представляющем собой убывающую последовательность чисел.
4. Оценить время работы каждого из реализованных алгоритмов на массиве, одна половина которого представляет собой возрастающую последовательность чисел, а вторая, – убывающую.
5. Оценить время работы стандартной функции qsort, реализующей алгоритм быстрой сортировки на выше указанных наборах данных.

Разработали программу выполняющую сортировку массива разными способами: быстрая сортировка, шелл и библиотечная сортировка. Выполнили проверку на массивах разных размеров, а также на массивах, где числа возрастают и где числа убывают.



С сортировкой массива размером 5 тысяч все алгоритмы справились почти моментально, лучше всех справился алгоритм quick sort, время его работы меньше миллисекунды во всех вариациях массива.

Для размера 25000, хуже всех справился алгоритм shel, за исключением варианта, где элементы не надо сортировать, тут он показал себя лучше всех. quick sort справляется также хорошо, выполняя алгоритм почти моментально.

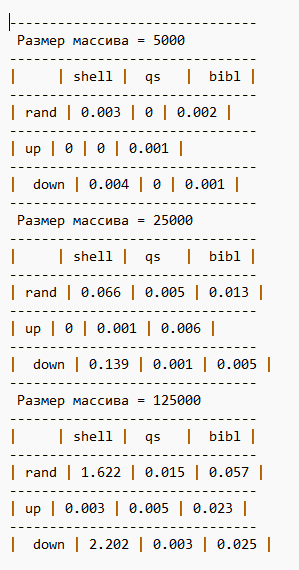
В ситуации с 125000 элементы ситуация такая же.

Вывод: Алгоритм быстрой сортировки справляется лучше всех, он выполняет операции значительно быстрее остальных.

Алгоритм Shell самый медленный, он справляется гораздо хуже других, особенно это отражается на больших количествах элементов, но в случаях, когда массив отсортирован или почти все элементы отсортировано – он лучший.

Библиотечный алгоритм сортировки основан на быстрой сортировке, но справляется с задачей дольше. Он прост в реализации, но теряется эффективность.

На занятии добавили вывод результат в таблицу. Вывод осуществляется при помощи библиотеки fstream.



**Вывод:** Научились оценивать сложность программ, а также измерять время ее работы.