# Министерство науки и высшего образования Российской Федерации ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

# Факультет безопасности информационных технологий

### Дисциплина:

«Алгоритмы и структуры данных»

# ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«PSRS-сортировка»

	Выполнили:					
	Суханкулиев М.,					
студент группы N3246						
	(подпись)					
	Проверил:					
	Ерофеев С. А.					
	(отметка о выполнении)					
	(подпись)					

Санкт-Петербург 2025 г.

# СОДЕРЖАНИЕ

ние	4		
	1		

# **ВВЕДЕНИЕ**

**Цель работы** – Разработать программу сортировки методом PSRS (Parallel Sorting by Regular Sampling) для чисел из файла, которые считываются в статический или динамический массив по выбору пользователя.

Для достижения поставленной цели необходимо решить следующие задачи:

- Разработать блок-схему алгоритма;
- Составить спецификацию всех переменных;
- Реализовать программу на языке С;
- Провести тестирование программы.

#### 1 AЛГОРИТМ PSRS

#### 1.1 Описание алгоритма

Начало.

- **Шаг 1.** Исходный массив numbers из n элементов разделим поровну между р потоками. Каждый поток получит подмассив размера base\_size (с учетом возможного остатка remainder).
- **Шаг 2.** На каждом потоке запускаем быструю сортировку (qsort) для соответствующего подмассива (thread data[i].array).
- **Шаг 3.** Формируем вспомогательный массив (sample) из элементов каждого подмассива sub\_arrays, взятых под индексами  $0, \frac{n}{p^2}, \frac{2n}{p^2}, \dots, \frac{(p-1)n}{p^2}$ .
- **Шаг 4.** Сортируем вспомогательный массив sample с помощью быстрой сортировки (qsort).
- **Шаг 5.** Формируем массив разделителей (splitters) из элементов sample, взятых под индексами  $p+\left[\frac{p}{2}\right]-1,2p+\left[\frac{p}{2}\right]-1,...,(p-1)p+\left[\frac{p}{2}\right]-1.$
- **Шаг 6.** Делим данные в подмассивах sub\_arrays с помощью массива splitters. Пусть  $a_1, a_2, ..., a_j$  разделители. Тогда данные в каждом подмассиве sub\_arrays[i] разбиваются на группы элементов, попадающие в соответствующие полуинтервалы  $(-\infty, a_1], (a_1, a_2], ..., (a_j, +\infty)$ . Число элементов в каждой группе фиксируется в массиве counts.
- **Шаг 7.** Производим многопутевое слияние (p\_way\_merge) всех подмассивов sub\_arrays в массив result. Слияние выполняется поочередно: сначала объединяется первая группа со второй, затем результат с третьей и так далее. В итоге получаем отсортированный массив result.
- **Шаг 8.** Записываем отсортированные данные из result в выходной файл (write\_result\_to\_file).

Конец.

#### 1.2 Блок-схема алгоритма

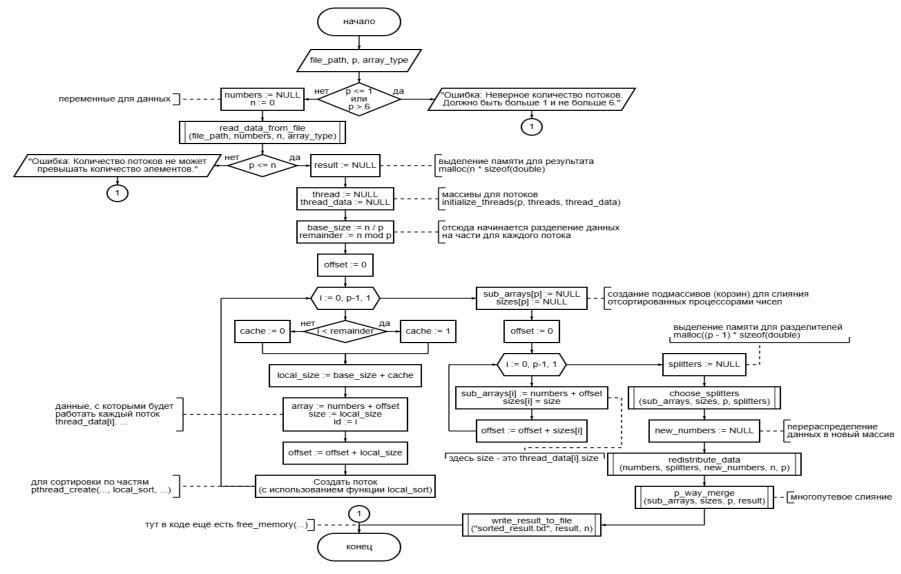


Рисунок 1 — Основная схема алгоритма

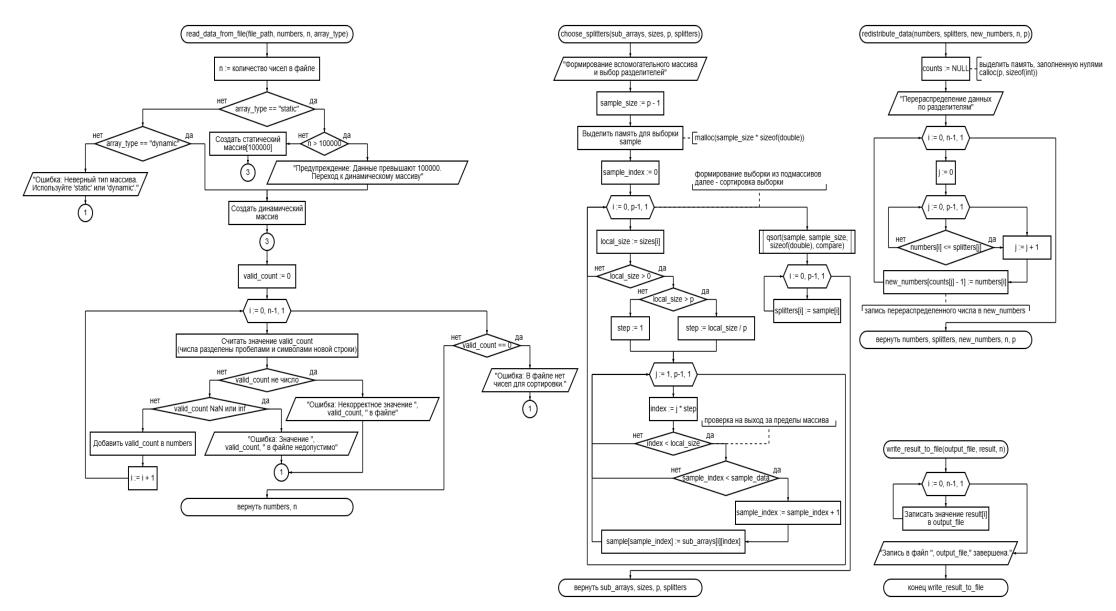


Рисунок 2 – Подпрограммы (I часть)

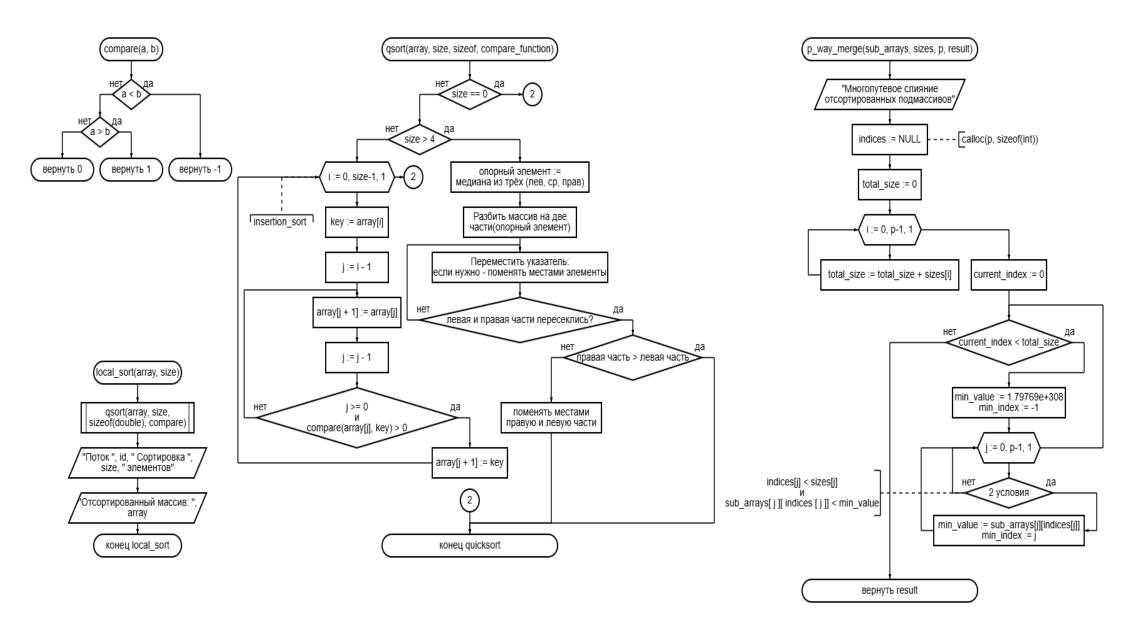


Рисунок 3 – Подпрограммы (II часть)

# 2 СПЕЦИФИКАЦИЯ ПЕРЕМЕННЫХ

Переменная	Описание	Тип использования	Тип	Размер (байт)	Диапазон значений
file_path	Путь к файлу, содержащему данные для сортировки	Входная	const char	8	Путь к файлу в файловой системе (строка)
р	Количество потоков	Входная	char	8	[2, 6]
array_type	Тип массива (static/dynamic)	Входная	int	4	"static" или "dynamic"
numbers	Массив чисел для сортировки	Входная/Промежуточная	double	8	[-1.7977e + 308, 1.7977e + 308]
n	Количество чисел в массиве	Промежуточная	int	4	[0, 2 147 483 647]
threads	Массив потоков	Промежуточная	pthread_t	8	Системный указатель на поток (зависит от платформы)
thread_data	Массив структур с данными для каждого потока	Промежуточная	ThreadData	8	Указатель на структуры с информацией о каждом потоке
splitters	Массив разделителей для многопутевого слияния	Промежуточная	double	8	[-1.7977e + 308, 1.7977e + 308]
new_numbers	Массив для перераспределенных данных	Промежуточная	double	8	[-1.7977e + 308, 1.7977e + 308]
sub_arrays	Массив указателей на подмассивы для слияния	Промежуточная	double	8	Указатели на отсортированные подмассивы
sizes	Массив размеров подмассивов для каждого потока	Промежуточная	int	4	[0, 357 913 942]
indices	Массив индексов для слияния данных из подмассивов	Промежуточная	int	4	[0, 2 147 483 647]
sample	Вспомогательный массив для выборки разделителей	Промежуточная	double	8	[-1.7977e + 308, 1.7977e + 308]
line	Буфер для чтения строки из файла	Промежуточная	char[256]	256	Строка с ASCII-символами, длина до 255 символов + \0
token	Буфер для токенов строки при разборе данных из файла	Промежуточная	char	8	Указатель на текущий токен
temp	Временная переменная для хранения числовых значений	Промежуточная	double	8	[-1.7977e + 308, 1.7977e + 308]
valid_count	Количество корректно считанных чисел	Промежуточная	int	4	[0, 2 147 483 647]
file	Указатель на файл	Промежуточная	FILE	8	Указатель на открытый файл
sample_size	Размер выборки для разделителей	Промежуточная	int	4	[0, 5]
offset	Смещение для перераспределения данных	Промежуточная	int	4	[0, 2 147 483 647]
remainder	Остаток элементов при делении на количество потоков	Промежуточная	int	4	[0, 2 147 483 647]
step	Шаг для формирования выборки разделителей	Промежуточная	int	4	[1, 2 147 483 647]
min_value	Минимальное значение для слияния данных	Промежуточная	double	8	[-1.7977e + 308, 1.7977e + 308]
min_index	Индекс минимального значения при слиянии данных	Промежуточная	int	4	[0, 5]
result	Массив для хранения отсортированных данных	Выходная	double	8	[-1.7977e + 308, 1.7977e + 308]

Примечание: размер переменных в памяти указан для стандартных платформ х86-64.

#### 3 РЕАЛИЗАЦИЯ ПРОГРАММЫ

#### 3.1 Описание программы

Программа написана на языке С и состоит из трех основных файлов:

- 1) Заголовочный файл (psrs.h) содержит объявления функций и структур данных, которые используются в реализации программы.
- 2) Основной файл (main.c) включает функцию main(), которая управляет процессом выполнения программы.
- 3) Файл с реализациями функций (psrs.c) содержит определения всех функций, объявленных в заголовочном файле psrs.h.

#### 3.2 Код программы

#### 3.2.1 psrs.h

```
#ifndef PSRS H
        #define PSRS H
        #include <stdio.h>
        #include <stdlib.h>
        #include <pthread.h>
        #include <math.h>
        #include <string.h>
        #define MAX NUMBERS 100000
        typedef struct {
           double *array;
           int size;
            int id;
        } ThreadData;
        int read data from file(const char *file path, double **numbers, int *n, const char
*array type);
        int initialize_threads(int p, pthread_t **threads, ThreadData **thread data);
        void *local_sort(void *arg);
        void choose splitters(double **sub arrays, int *sizes, int p, double *splitters);
        void redistribute_data(double *numbers, double *splitters, double *new_numbers, int n, int
p);
        void p way merge(double **sub arrays, int *sizes, int p, double *result);
        void write_result_to_file(const char *output_file, double *result, int n);
        void free_memory(double *numbers, double *result, double *splitters, double *new_numbers,
pthread t *threads, ThreadData *thread data, const char *array type);
        #endif // PSRS H
```

#### 3.2.2 psrs.c

```
#include "psrs.h"
// Функция сортировки
int compare(const void *a, const void *b) {
```

```
if (*(double*)a < *(double*)b) return -1;
            if (*(double*)a > *(double*)b) return 1;
            return 0;
        // Локальная сортировка для каждого потока
        void *local_sort(void *arg) {
            ThreadData *data = (ThreadData*)arg;
qsort(data->array, data->size, sizeof(double), compare);
            printf("Поток %d: Сортировка %d элементов...\nОтсортированный подмассив: \n", data-
>id, data->size);
            for (int i = 0; i < data->size; ++i) {
                printf("%.15g ", data->array[i]);
            printf("\n");
            return NULL;
        // Чтение данных из файла и заполнение массива
        int read data from file(const char *file path, double **numbers, int *n, const char
*array_type) {
            FILE *file = fopen(file path, "r");
            if (!file) {
                printf("Ошибка: Не удалось открыть файл %s\n", file path);
            double temp;
            *n = 0;
            // Считываем количество чисел в файле
            while (fscanf(file, "%lf", &temp) == 1) {
                (*n)++;
            rewind(file);
            // Выбор типа массива
            if (strcmp(array_type, "static") == 0) {
                if (*n > MAX NUMBERS) {
                    printf("Предупреждение: Данные превышают MAX_NUMBERS. Переход к динамическому
массиву. \п");
                    *numbers = (double*)malloc(*n * sizeof(double)); // Динамическая память, если
превышен лимит
                } else {
                    static double static numbers[MAX NUMBERS]; // Статический массив
                    *numbers = static numbers; // Указываем на начало статического массива
            } else if (strcmp(array type, "dynamic") == 0) {
                *numbers = (double*)malloc(*n * sizeof(double)); // Динамическая память
            } else {
                printf("Ошибка: Неверный тип массива. Используйте 'static' или 'dynamic'.\n");
                fclose(file);
                return 0;
            if (!*numbers) {
                printf("Ошибка: Не удалось выделить память для чисел\n");
                fclose(file);
                return 0;
            // Считываем данные из файла с проверкой на ошибки
            int valid count = 0;
            char line[256]; // Буфер для строки
            while (fgets(line, sizeof(line), file)) {
                char *token = strtok(line, " \n\r\t"); // Разделяем строку на токены по пробелам
и символам новой строки
                while (token) {
                    // Проверка, что строка является корректным числом
                    char *endptr;
                    temp = strtod(token, &endptr);
                    // Если strtod не смог преобразовать строку в число
                    if (*endptr != '\0') {
```

```
printf("Ошибка: Некорректное значение \"%s\" в файле\n", token);
                        if (array type && strcmp(array type, "dynamic") == 0) { // Освобождаем
только динамически выделенную память
                            free(*numbers);
                        fclose(file);
                        return 0;
                    }
                    // Проверка на NaN и Infinity
                    if (isnan(temp) || isinf(temp)) {
                        printf("Ошибка: Значение \"%s\" в файле недопустимо\n", token);
                        if (array_type && strcmp(array_type, "dynamic") == 0) { // Освобождаем
только динамически выделенную память
                            free(*numbers);
                        fclose(file);
                        return 0;
                    if (valid count < *n) {
                        (*numbers)[valid count++] = temp;
                    token = strtok(NULL, " \n\r\t"); // Переход к следующему токену
                }
            if (valid count == 0) {
                printf("Ошибка: В файле нет чисел для сортировки.\n");
                if (array type && strcmp(array type, "dynamic") == 0) { // Освобождаем только
динамически выделенную память
                   free(*numbers);
                fclose(file);
                return 0;
            fclose(file);
            return 1;
        }
        // Функция для выделения памяти для потоков и данных потока
        int initialize_threads(int p, pthread_t **threads, ThreadData **thread_data) {
            *threads = (pthread_t*)malloc(p * sizeof(pthread_t));
            *thread data = (ThreadData*) malloc(p * sizeof(ThreadData));
            if (!*threads || !*thread data) {
                printf("Ошибка: Не удалось выделить память для потоков или данных потока\n");
                return 0;
            }
           return 1:
        // Выбор регулярных образцов для разделителей
        void choose splitters (double **sub arrays, int *sizes, int p, double *splitters) {
           printf("Формирование вспомогательного массива и выбор разделителей...\n");
            // Выделение памяти для выборки
            int sample size = p - 1;
            double *sample = (double*)malloc(sample size * sizeof(double));
            if (!sample) {
                printf("Ошибка: Не удалось выделить память для выборки\n");
                return;
           int sample index = 0;
            // Формирование выборки из подмассивов
            for (int i = 0; i < p; ++i) {
                int local_size = sizes[i];
                if (local size > 0) {
                    int \overline{\text{slep}} = (local size > p) ? (local size / p) : 1; // Защита от деления на 0
                    for (int j = 1; j < p; ++j) {
                        int index = j * step;
                        if (index < local_size) {</pre>
                            // Убедиться, что индекс не выходит за пределы массива
```

```
if (sample index < sample size) {</pre>
                                 sample[sample index++] = sub arrays[i][index];
                             } else {
                                 break;
                        }
                   }
                }
            // Сортировка выборки
            qsort(sample, sample size, sizeof(double), compare);
            // Выбор разделителей из отсортированной выборки
            for (int i = 0; i ; ++i) {
                splitters[i] = sample[i];
            free (sample);
        // Перераспределение данных по разделителям
        void redistribute data(double *numbers, double *splitters, double *new numbers, int n, int
p) {
            int *counts = (int*)calloc(p, sizeof(int));
            if (!counts) {
                printf("Ошибка: Не удалось выделить память для counts\n");
                return;
            memset(counts, 0, p * sizeof(int));
            printf("Перераспределение данных по разделителям...\n");
            for (int i = 0; i < n; ++i) {
                int j;
                for (j = 0; j  {
                    if (numbers[i] <= splitters[j]) {</pre>
                        break;
                    }
                }
                counts[j]++;
                new_numbers[counts[j] - 1] = numbers[i]; // Запись перераспределенного числа в
new numbers
            free(counts);
        // Многопутевое слияние отсортированных подмассивов
        void p_way_merge(double **sub_arrays, int *sizes, int p, double *result) {
            printf("Многопутевое слияние отсортированных подмассивов...\n");
            int *indices = (int*)calloc(p, sizeof(int));
            if (!indices) {
                printf("Ошибка: Не удалось выделить память для индексов\n");
                return;
            memset(indices, 0, p * sizeof(int));
            int total size = 0;
            for (int \bar{i} = 0; i < p; ++i) {
                total_size += sizes[i];
            int current_index = 0;
            while (current_index < total_size) {</pre>
                double min_value = __DBL_MAX__;
int min_index = -1;
                for (int j = 0; j < p; ++j) {
                    if (indices[j] < sizes[j] && sub_arrays[j][indices[j]] < min_value) {</pre>
                        min value = sub arrays[j][indices[j]];
                        min index = j;
                result[current_index] = min_value;
                indices[min index]++;
```

```
current index++;
            free (indices);
        }
        // Запись результата в файл
        void write_result_to_file(const char *output_file, double *result, int n) {
            FILE *file = fopen(output file, "w");
            if (!file) {
                printf("Ошибка: Не удалось открыть выходной файл %s\n", output file);
                return;
            for (int i = 0; i < n; i++) {
                fprintf(file, "%.15g ", result[i]);
            fclose(file);
            printf("Запись в файл \"%s\" завершена.\n", output file);
        // Освобождение памяти
        void free_memory(double *numbers, double *result, double *splitters, double *new_numbers,
pthread t *threadS, ThreadData *thread_data, const char *array_type) {
           if (array_type && strcmp(array_type, "dynamic") == 0) {
                free(numbers); // Освобождаем только если это динамический массив
            free (result);
            free(splitters);
            free(new numbers);
            free (threads);
            free (thread data);
```

#### 3.2.3 main.c

```
#include "psrs.h"
// Главная функция
int main(int argc, char *argv[]) {
    if (argc < 4) {
        printf("Использование: %s <путь к файлу> <количество потоков> <тип массива:
        static/dynamic>\n", argv[0]);
        return 1;
    const char *file path = argv[1];
    int p = atoi(argv[2]);
    if (p \le 1 \mid | p > 6) {
       printf("Ошибка: Неверное количество потоков. Должно быть больше 1 и не больше 6.\n");
        return 1;
    const char *array type = argv[3];
    // Переменные для данных
    double *numbers = NULL;
    int n = 0;
    if (!read_data_from_file(file_path, &numbers, &n, array_type)) {
       printf("Ошибка при чтении файла\n");
        return 1;
    if (p > n) {
        printf("Ошибка: Количество потоков не может превышать количество элементов.\n");
        free (numbers);
        return 1;
    // Резервирование памяти для результата
    double *result = (double*)malloc(n * sizeof(double));
    if (!result) {
        printf("Ошибка: Не удалось выделить память для результата\n");
        free (numbers);
```

```
return 1;
// Массивы для потоков
pthread t *threads;
ThreadData *thread data;
if (!initialize_threads(p, &threads, &thread_data)) {
   printf("Ошибка: Не удалось создать потоки threads\n");
    free(numbers);
    free (result);
   return 1;
// Разделение данных на части для каждого потока
int base size = n / p;
int remainder = n % p;
int offset = 0;
for (int i = 0; i < p; ++i) {
   int local size = base size + (i < remainder ? 1 : 0);</pre>
    thread_data[i].array = numbers + offset;
    thread_data[i].size = local_size;
    thread data[i].id = i;
    offset += local size;
   pthread create (&threads[i], NULL, local sort, &thread data[i]);
// Ожидание завершения всех потоков
for (int i = 0; i < p; ++i) {
    pthread_join(threads[i], NULL);
// Создание подмассивов для слияния
double *sub_arrays[p];
int sizes[p];
offset = 0;
for (int i = 0; i < p; ++i) {
    sub arrays[i] = numbers + offset;
    sizes[i] = thread data[i].size;
    offset += sizes[i];
// Выбор разделителей
double *splitters = (double*)malloc((p - 1) * sizeof(double));
if (!splitters) {
   printf("Ошибка: Не удалось выделить память для разделителей\n");
    free memory(numbers, result, NULL, NULL, threads, thread data, array type);
    return 1;
choose splitters (sub arrays, sizes, p, splitters);
// Перераспределение данных
double *new numbers = (double*)malloc(n * sizeof(double));
if (!new_numbers) {
   printf("Ошибка: He удалось выделить память для new_numbers\n");
   free memory(numbers, result, splitters, NULL, threads, thread data, array type);
redistribute data(numbers, splitters, new numbers, n, p);
// Многопутевое слияние
p_way_merge(sub_arrays, sizes, p, result);
// Запись результата в файл
write_result_to_file("sorted_result.txt", result, n);
// Освобождение памяти
free memory(numbers, result, splitters, new numbers, threads, thread data, array type);
return 0;
```

#### 4 ТЕСТИРОВАНИЕ ПРОГРАММЫ

Программа скомпилирована с использованием следующих флагов:

```
gcc -03 main.c psrs.c -o psrs -lpthread -lm -Wall -Wextra -Werror
```

Для проверки на наличие утечек памяти и других ошибок использовалась утилита Valgrind.

```
valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes ./psrs...
```

В процессе тестирования не было обнаружено утечек памяти или других проблем (примеры ниже).

#### 4.1 Скриншоты тестирования

```
× sorted_result.txt
   1.00000000000001 1.0000000000000 1.2 5.3 42 3.14 -0.5 1e300
   54 45 1.2 5.3 42 3.14 0 6.0 605 450.5454654
   1e-300 1e+300 1e-100 1e100 3.14 2.71828 42 -42
   42 42 42 42 42 42 42 42 42 42
   1.0 -1.0 3.14 -3.14 0 2.71 -2.71 42 -42 1e-3
   1.0000000000000001 1.000000000
   2.000000000000001 2.000000000
   1.23 0 5 42 1.234 0.567 -0.12 File Actions Edit View Help
   1.1 2.2 3.3 4.4 5.5 6.6 7.7 8
   17.17 18.18 19.19 20.20 21.21 [kali⊕ kali]-[~/Desktop/algos1] 1e-300 1e-200 1e-100 1e0 1e10 $./psrs file.txt 3 dynamic
   1e3 1e4 1e5 1e6 1e7 1e8 1e9 Ошибка: Значение "1e500" в файле недопустимо
                                     Ошибка при чтении файла
3 42 0 0 0 0 0 0 0 0 1e500
   42 -42 3 1.2 7.5 0.4 1e3 100
1e+10 1.23e45 3.14159E-5
                                         -(kali®kali)-[~/Desktop/algos1]
   +42 -3.14 +1.6 1.5 +2 -5
```

Рисунок 4 – Неподдерживаемые значения

```
(kali⊗kali)-[~/Desktop/algos1]
$ ./psrs file.txt 8 dynamic
Ошибка: Неверное количество потоков. Должно быть больше 1 и не больше 6.

[(kali⊗kali)-[~/Desktop/algos1]
$ ./psrs file.txt 6 dynam
Ошибка: Неверный тип массива. Используйте 'static' или 'dynamic'.
Ошибка при чтении файла

[(kali⊗kali)-[~/Desktop/algos1]
$ ./psrs file(nofile).txt 6 dynamic
Ошибка: Не удалось открыть файл file(nofile).txt
Ошибка при чтении файла

[(kali⊗kali)-[~/Desktop/algos1]
$ ./psrs file
Использование: ./psrs <nyть к файлу> <количество потоков> <тип массива: static/dynamic>

[(kali⊗kali)-[~/Desktop/algos1]
$ ./psrs file.txt 3 dynamic
Поток 0: Сортировка 48 элементов ...
Отсортированный подмассив:
```

Рисунок 5 – Обработка вводимых аргументов

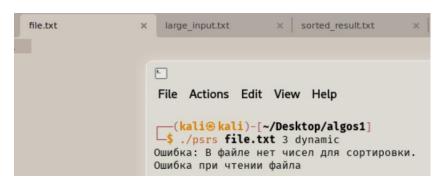


Рисунок 6 – Пустой файл или файл без чисел

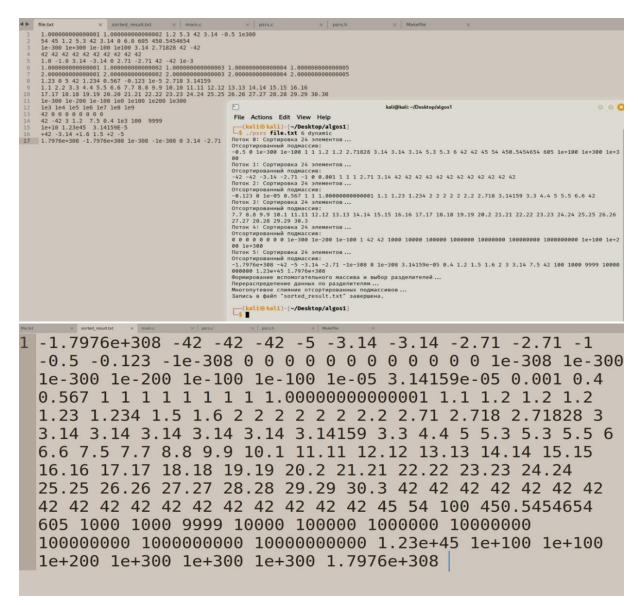


Рисунок 7 – Все поддерживаемые значения и вывод

```
C- C.C. 85 - 0.C. 85 - 780 - 85 - 780 - 85 - 780 - 85 - 87 - 85 - 685 - 685 - 186 - 85 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 865 - 8
 4988 -38.4769 -38.4667 -38.4571 -38.449 -38.4422 -38.4312 -38.4218 -38.407 -38.3999 -38.3664 -38.3578 -38.349 -38.3411 -38.
 2519 -38.235 -38.2247 -38.2149 -38.2048 -38.2042 -38.203 -38.2012 -38.1948 -38.1939 -38.1811 -38.1701 -38.1446 -38.1417 -38
 0984 -38.0955 -38.0763 -38.0703 -38.0554 -38.0532 -38.0388 -38.0353 -38.0169 -38.0106 -37.9991 -37.9948 -37.9817 -37.9581
 9076 -37.8989 -37.8821 -37.8779 -37.8737 -37.8693 -37.8676 -37.8539 -37.8255 -37.8016 -37.7969 -37.7963 -37.7705 -37.7692
 .7135 -37.7088 -37.707 -37.7067 -37.7019 -37.6873 -37.6855 -37.6839 -37.6781 -37.6648 -37.6576 -37.6491 -37.6414 -37.6317 -3
 .5848 -37.5788
                                                                                                                                                                                                                                                                                                               173 -37.
                                                                                                                                                                                                                                                                                                0 2 41 -37.
                                                                                                                                              kali@kali: ~/Desktop/algos1
397 -37.3812 -
532 -37.1439 -
                                                                                                                                                                                                                                                                                                                   823 -36
                                      File Actions Edit View Help
.9051 -36.8845
                                                                                                                                                                                                                                                                                                                    .7502
                                           -(kali@kali)-[~/Desktop/algos1]
 7069 -36.6796
                                                                                                                                                                                                                                                                                                                     -36.55
                                      _$ ./psrs large_input.txt 2 static
                                                                                                                                                                                                                                                                                                                    447 -36
 4927 - 36, 4892
                                      Предупреждение: Данные превышают MAX_NUMBERS. Переход к динамическому массиву.
.2459 -36.2426 Поток 1: Сортировка 100000 элементов...
                                                                                                                                                                                                                                                                                                                    31 - 36
 .0722 -36.0552 Отсортированный подмассив:
                                                                                                                                                                                                                                                                                                                     9478
.8736 -35.8731 -999.996 -999.984 -999.982 -999.976 -999.953 -999.943 -999.943 -999.94 -999.927 -999.917 -999.909 -999.904 -999.
                                                                                                                                                                                                                                                                                                                     .766 -
 .6897 -35.6847 893 -999.866 -999.865 -999.821 -999.792 -999.769 -999.747 -999.645 -999.635 -999.621 -999.564 -999.551 -999.525
 .5327 -35.5251 -999.497 -999.466 -999.429 -999.41 -999.391 -999.367 -999.318 -999.292 -999.287 -999.286 -999.278 -999.277 -999.
                                                                                                                                                                                                                                                                                                                    4113 -3
. 3664 - 35, 3355 249 - 999, 241 - 999, 229 - 999, 228 - 999, 191 - 999, 188 - 999, 143 - 999, 126 - 999, 109 - 999, 066 - 999, 064 - 999, 028 - 999, 017 - 3664 - 398, 345 - 998, 347 - 998, 348 - 998, 347 - 998, 348 - 998, 347 - 998, 348 - 998, 347 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 998, 348 - 988, 348 - 988, 348 - 998, 348 - 998, 348 - 98
                                                                                                                                                                                                                                                                                                                    1239 -
                                                                                                                                                                                                                                                                                                                     8827
 8354 -34.8045 681 -998.67 -998.654 -998.654 -998.613 -998.601 -998.573 -998.568 -998.551 -998.525 -998.524 -998.523 -998.517 -9
.6327 -34.6138 98.5 -998.475 -998.436 -998.42 -998.413 -998.384 -998.355 -998.327 -998.291 -998.263 -998.254 -998.244 -998.229
 362 -34.3608 -998.205 -998.195 -998.179 -998.129 -998.118 -998.118 -998.102 -998.096 -998.094 -998.092 -998.089 -998.089 -998
                                                                                                                                                                                                                                                                                                                    858 -34
                                      .081 \; -998.064 \; -998.037 \; -998.011 \; -997.999 \; -997.977 \; -997.957 \; -997.937 \; -997.911 \; -997.907 \; -997.869 \; -997.865 \; -997.865 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -997.869 \; -9
 .0216 -34.0105 -997.828 -997.805 -997.781 -997.744 -997.742 -997.726 -997.714 -997.705 -997.67 -997.626 -997.615 -997.581 -997
                                                                                                                                                                                                                                                                                                                    443 -33
 7688 -33,7559 .578 -997.532 -997.532 -997.524 -997.488 -997.419 -997.418 -997.34 -997.34 -997.307 -997.288 -997.287 -997.188 -
 4548 -33,4353 997.152 -997.15 -997.126 -997.101 -997.073 -997.061 -997.053 -997.051 -997.055 -997.027 -997.027 -997.027
                                                                                                                                                                                                                                                                                                                    2658 -3
                                      22 -997.012 -997.01 -996.989 -996.988 -996.97 -996.93 -996.917 -996.895 -996.89 -996.862 -996.862 -996.858 -996.
 .2282 -33.222
                                                                                                                                                                                                                                                                                                                    0829 -3
                                      802 -996.776 -996.768 -996.727 -996.719 -996.716 -996.703 -996.674 -996.674 -996.673 -996.663 -996.659 -996.653
                                      -996.651 -996.62 -996.522 -996.522 -996.515 -996.511 -996.511 -996.507 -996.488 -996.482 -996.48 -996.474 -996.4
 8806 -32.8716 69 -996.436 -996.425 -996.407 -996.386 -996.379 -996.334 -996.325 -996.315 -996.315 -996.31 -996.30 -996.27 -99
 6802 -32.6604 6.257 -996.226 -996.17 -996.169 -996.154 -996.132 -996.123 -996.123 -996.101 -996.097 -996.082 -996.06 -996.031
 .4926 -32.485 -995.983 -995.966 -995.956 -995.94 -995.85 -995.88 -995.829 -995.821 -995.82 -995.797 -995.794 -995.78 -995.762
                                                                                                                                                                                                                                                                                                                    3786 -
.3491 -32.3482 -995.759 -995.753 -995.736 -995.732 -995.728 -995.724 -995.709 -995.706 -995.673 -995.672 -995.631 -995.606 -99
.1897 -32.1834 5.59 -995.562 -995.531 -995.524 -995.521 -995.5 -995.481 -995.459 -995.391 -995.375 -995.317 -995.28 -995.276 -9
                                                                                                                                                                                                                                                                                                                    291 -32
 9822 -31, 9747 95.267 -995.265 -995.263 -995.259 -995.258 -995.217 -995.212 -995.194 -995.151 -995.147 -995.14 -995.152 -995.
                                                                                                                                                                                                                                                                                                                     .8919
 8384 -31.8335 -995.066 -995.061 -995.021 -994.992 -994.985 -994.978 -994.917 -994.899 -994.87 -994.827 -994.837 -994.83 -994.8
                                                                                                                                                                                                                                                                                                                      -31.69
                                      2 -994.801 -994.795 -994.794 -994.767 -994.764 -994.751 -994.741 -994.738 -994.734 -994.721 -994.683 -994.637 -9
 6242 -31.6241
                                                                                                                                                                                                                                                                                                                    4425 -3
                                      94.632 -994.61 -994.603 -994.548 -994.534 -994.513 -994.508 -994.502 -994.496 -994.479 -994.472 -994.471 -994.37
                                                                                                                                                                                                                                                                                                                    2652
                                      7 -994.366 -994.364 -994.35 -994.343 -994.248 -994.219 -994.157 -994.153 -994.148 -994.135 -994.121 -994.114 -99
                                                                                                                                                                                                                                                                                                                    0859
 2141 -31.2033
.0095 -31.0004 -30.9925 -30.9698 -30.9563 -30.955 -30.9388 -30.9314 -30.9274 -30.8877 -30.8827 -30.8819 -30.8708 -30.8598 -
```

Рисунок 8 – Очень большое количество чисел

Рисунок 9 – Некорректные знаки

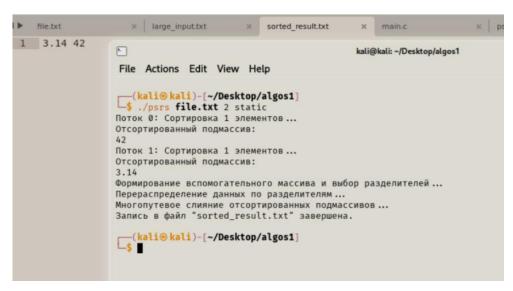


Рисунок 10 – Минимальное количество чисел

```
x large_input.txt
                              × sorted result.txt
                                              × main.c
file.txt
1.000000000000001 1.0000000000000 1.2 5.3 42 3.14 -0.5 1e300
54 45 1.2 5.3 42 3.14 0 6.0 605 450.5454654
1e-300 1e+300 1e-100 1e100 3.14 2.71828 42 -42
42 42 42 42 42 42 42 42 42 42
1.0 -1.0 3.14 -3.14 0 2.71 -2.71 42 -42 1e-3
1.23 0 5 42 1.234 0.567 -0.123 1e-5 2.718 3.14159
1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9 10.10 11.11 12.12 13.13 14.14 15.
17.17 18.18 19.19 20.20 2
1e-300 1e-200 1e-100 1e0
1e3 1e4 1e5 1e6 1e7 1e8 1e File Actions Edit View Help
42 0 0 0 0 0 0 0 0 asd
42 -42 3 1.2 7.5 0.4 le3
                        __(kali⊛ kali)-[~/Desktop/algos1]
                        _$ ./psrs file.txt 3 dynamic
1e+10 1.23e45 3.14159E-5
                        Ошибка: Некорректное значение "asd" в файле
+42 -3.14 +1.6 1.5 +2 -5
                        Ошибка при чтении файла
1.7976e+308 -1.7976e+308
```

Рисунок 11 – Строка в файле

#### ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был разработан и реализован алгоритм параллельной сортировки методом PSRS. Для проверки работоспособности программы проведено тестирование с различными входными данными, включая крайние случаи и большие массивы чисел. Анализ результатов показал корректность работы алгоритма и отсутствие утечек памяти.

Выполненная работа позволила закрепить навыки многопоточного программирования на языке С, работы с динамическими структурами данных и анализа алгоритмов сортировки.

#### СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1. Сортировки Викиконспекты 2022.
- 2. PSRS-сортировка Викиконспекты 2022.
- 3. Быстрая сортировка Викиконспекты 2022.
- 4. H. Shi and J. Schaeffer. "Parallel Sorting by Regular Sampling. Journal of Parallel and Distributed Computing," 14(4):361--372, 1992. URL: <a href="mailto:psrs1.pdf">psrs1.pdf</a> Yandex Documents
- 5. Parallel and Distributed Computing Department of Computer Science and Engineering (DEI) Instituto Superior Tencico 2012. URL : <u>parallelsort.pdf Yandex Documents</u>
- 6. Sathish Vadhiyar Parallel Sorting 2012. URL : <u>ParallelSorting.pdf Yandex Documents</u>
- 7. .Stack Overflow 2022. URL : Which parallel sorting algorithm has the best average case performance? Stack Overflow