Национальный исследовательский ядерный университет «МИФИ»
Институт интеллектуальных кибернетических систем
Кафедра №12 «Компьютерные системы и технологии»

ОТЧЕТ

О выполнении лабораторной работы №5

«Исследование методов сортировки массивов данных»

Студент: Гатченко А.С.

Группа: Б22-525

Преподаватель: Половнева Ю. А.

Москва – 2022

1. Формулировка индивидуального задания

Структура данных

Запись в журнале событий:

- идентификатор (целое число);
- уровень важности (целое число, обозначающее один из следующих уровней: debug, info, warn, error, fatal);
- текст (строка произвольной длины).

Алгоритмы сортировки

1. Пузырьковая сортировка (Bubble sort).
2. Двухсторонняя сортировка выбором (Double selection sort).

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовались встроенные типы данных int и char, предназначенные для работы с целыми числами, символами и строками, а также указатели на символы и на массивы символов.

3. Описание использованного алгоритма

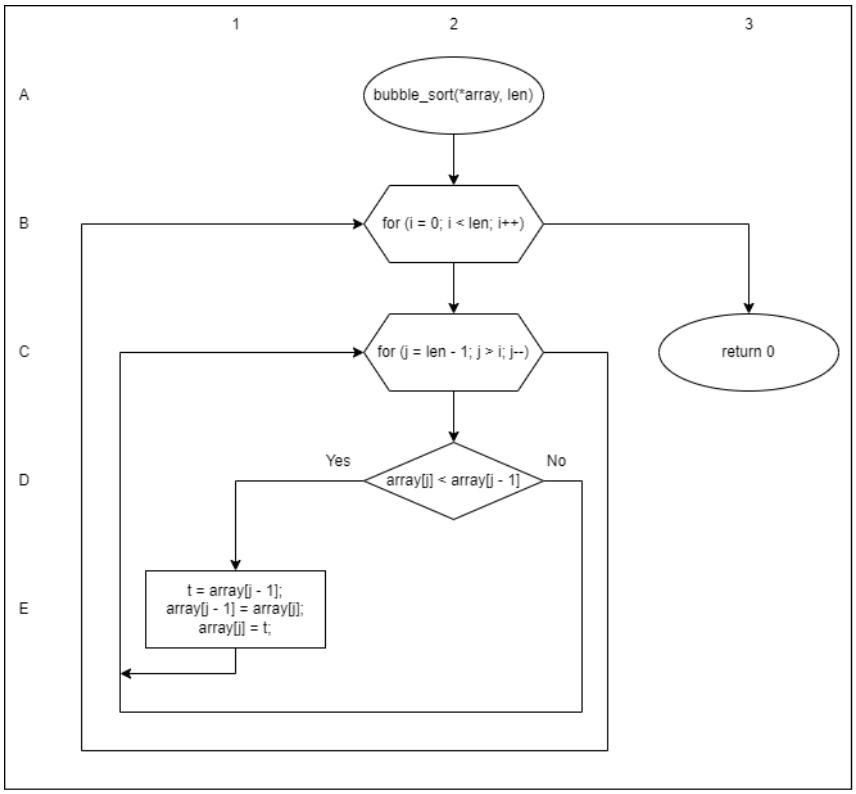


Рис. 1: Блок-схема алгоритма работы функции `bubble_sort()`

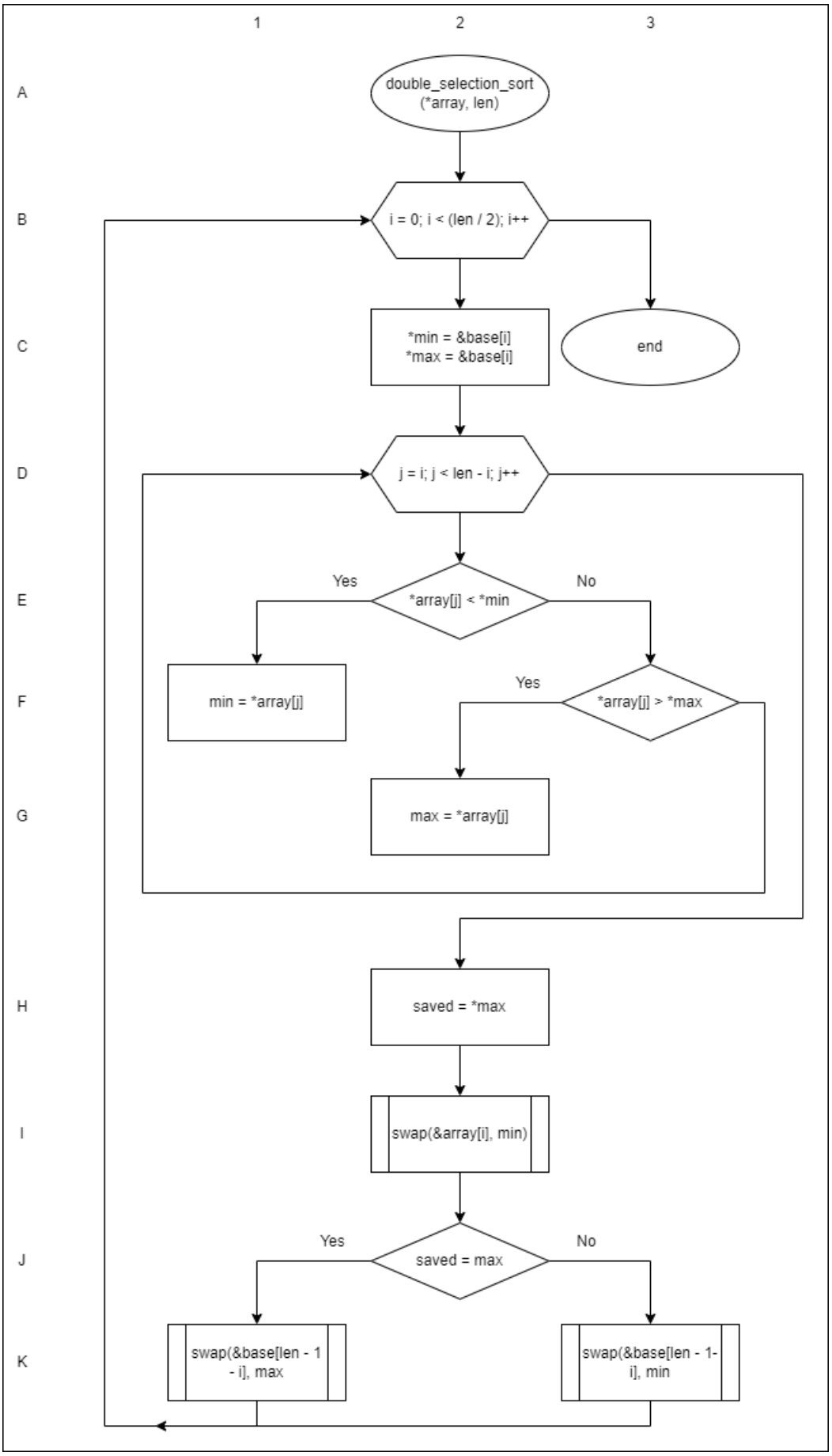


Рис. 2: Блок-схема алгоритма работы функции `double_selection_sort()`

4. Исходные коды разработанных программ

Листинг 1: Исходные коды программы lab5 (1)

1) Файл: prog.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "point.h"
#include "sort.h"

int read_from_file(FILE *file, int *num){
    int check = fscanf(file, "%d", num);
    if (check == -1){
        printf("Program has been stopped");
        return -1;
    }
    else if (check == 0){
        printf("Wrong data format!\n");
        return -1;
    }
    return 0;
}

int read_file(const char *path, int *len, Point **array, char ***str_arr){
    char buff[255];
    int x, y;
    int count = 0;
    Point new;

    FILE *file = fopen(path, "r");

    if (file == NULL){
        printf("Wrong filepath!\n");
        return -1;
    }

    read_from_file(file, len);

    if (*len == 0){
        printf("File is empty!\n");
        return -1;
    }

    *array = calloc(*len, sizeof(Point));
    *str_arr = calloc(*len, sizeof(char *));

    for (int i = 0; i < *len; i++){
        read_from_file(file, &x);
        read_from_file(file, &y);
        fscanf(file,  "%255s", buff);
        printf("%d, %d, %s\n", x, y, buff);
        (*str_arr)[count] = strdup(buff);
        new = point_new(x, y, (*str_arr)[count]);
        (*array)[count] = new;
        count++;
    }

    point_array_print("Array from the file:\n", *array, *len);
    fclose(file);
    return 0;
}
```

```c
int write_to_file(const char *path, int len, Point *array){
    FILE *file = fopen(path, "w");

    if (file == NULL){
        printf("Wrong filepath!\n");
        return -1;
    }

    for (int i = 0; i < len; i++){
        fprintf(file, "%d %d %s\n", array[i].x, array[i].y, array[i].z);
    }

    fclose(file);
    return 0;
}

int str_input(char *str){
    int check = scanf("%s", str);
    if (check == -1){
        printf("Program has been stopped");
        return -1;
    }
    while (check == 0){
        scanf("%*[^\n]");
        printf("You can't str these symbols.\n");
        check = scanf("%s", str);
    }
    return 0;
}

int read_input(int *input, int num1, int num2){
    int check = scanf("%d", input);
    if (check == -1){
        printf("Program has been stopped");
        return -1;
    }
    while (check == 0 || (num1 > *input) || (num2 < *input)){
        scanf("%*[^\n]");
        printf("You can't input such symbols.\n");
        check = scanf("%d", input);
    }
    return 0;
}

int menu(char *input_file, char *output_file, int *res){
    int input = 0;

    printf("Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double
selection sort\n");
    if (read_input(&input, 1, 3) == -1){
        return -1;
    }
    res[0] = input;

    printf("Choose field: 1 - x, 2 - y, 3 - z\n");
    if (read_input(&input, 1, 3) == -1){
        return -1;
    }
    res[1] = input;

    printf("Choose direction: 1 - direct, 2 - inversed\n");
    if (read_input(&input, 1, 2) == -1){
        return -1;
```

```
    }
    res[2] = input;

    printf("Choose an input file:\n");
    if (str_input(input_file) == -1){
        return -1;
    }

    printf("Choose an output file:\n");
    if (str_input(output_file) == -1){
        return -1;
    }

    return 0;
}

int main(){
    char input_file[255];
    char output_file[255];
    int *res = calloc(3, sizeof(int));
    int len;
    Point *arr = NULL;
    char **str_arr = NULL;

    if (menu(input_file, output_file, res) == -1){
        return -1;
    }

    int check = read_file(input_file, &len, &arr, &str_arr);

    if (check >= 0){
        point_array_print("Before sort:\n", arr, len);
        void *comp_func = comp(res[1], res[2]);

        if (res[0] == 1){
            qsort(arr, len, sizeof(Point), comp_func);
        }
        else if (res[0] == 2){
            bubble_sort(arr, len, comp_func);
        }
        else if (res[0] == 3){
            double_selection_sort(arr, len, comp_func);
        }

        point_array_print("After sort:\n", arr, len);
        write_to_file(output_file, len, arr);
    }

    for (int i = 0; i < len; i++){
        free(str_arr[i]);
    }

    free(str_arr);
    free(arr);
    free(res);
    return 0;
}
```

2) Файл: point.c

```
#include <stdio.h>
#include <string.h>
#include "point.h"
```

```c
Point point_new(int x, int y, char *z){
    Point p = {x, y, z};
    return p;
}

void point_print(const Point *p) {
    printf("{%d, %d, %s}", p->x, p->y, p->z);
}

void point_array_print(const char *msg, const Point *arr, int len) {
    printf("%s", msg);

    for (int i = 0; i < len; ++i) {
        printf("a[%d] = ", i);
        point_print(&arr[i]);
        printf("\n");
    }
}

int point_cmp_x(const Point *p1, const Point *p2) {
    return p1->x - p2->x;
}

int point_cmp_x_inv(const Point *p1, const Point *p2) {
    return p2->x - p1->x;
}

int point_cmp_y(const Point *p1, const Point *p2) {
    return p1->y - p2->y;
}

int point_cmp_y_inv(const Point *p1, const Point *p2) {
    return p2->y - p1->y;
}

int point_cmp_z(const Point *p1, const Point *p2) {
    return strcmp(p1->z, p2->z);
}

int point_cmp_z_inv(const Point *p1, const Point *p2) {
    return strcmp(p2->z, p1->z);
}

void *comp(int num1, int num2){
    if (num1 == 1 && num2 == 1){
        return (int (*)(const void *, const void *)) point_cmp_x;
    }
    else if (num1 == 1 && num2 == 2){
        return (int (*)(const void *, const void *)) point_cmp_x_inv;
    }
    else if (num1 == 2 && num2 == 1){
        return (int (*)(const void *, const void *)) point_cmp_y;
    }
    else if (num1 == 2 && num2 == 2){
        return (int (*)(const void *, const void *)) point_cmp_y_inv;
    }
    else if (num1 == 3 && num2 == 1){
        return (int (*)(const void *, const void *)) point_cmp_z;
    }
    else if (num1 == 3 && num2 == 2){
        return (int (*)(const void *, const void *)) point_cmp_z_inv;
    }
```

```
    else {
        return NULL;
    }
}
```

3) Файл: sort.c

```
#include <string.h>
#include "point.h"

void swap(void* v1, void* v2, int size){
    char buffer[size];

    memcpy(buffer, v1, size);
    memcpy(v1, v2, size);
    memcpy(v2, buffer, size);
}

void bubble_sort(Point *base, int len, int(*compar)(const Point *, const
Point *)){
    for (int i = 0; i < len; i++){
        for (int j = len - 1; j > i; j--){
            if (compar(base + j, base + j - 1) < 0){
                swap(&base[j], &base[j - 1], sizeof(Point));
            }
        }
    }
}

void double_selection_sort(Point *base, int len, int(*compar)(const Point *,
const Point *)){
    Point saved;
    for (int i = 0; i < (len / 2); i++){
        Point *min = &base[i];
        Point *max = &base[i];

        for (int j = i; j < len - i; j++){
            if (compar(base + j, min) < 0){
                min = (base + j);
            }
            else if (compar(base + j, max) > 0){
                max = (base + j);
            }
        }

        saved = *max;
        swap(&base[i], min, sizeof(Point));

        if (max->x == saved.x && max->y == saved.y){
            swap(&base[len - 1 - i], max, sizeof(Point));
        }
        else {
            swap(&base[len - 1- i], min, sizeof(Point));
        }
    }
}
```

4) Файл: point.h

```
#ifndef POINT_H
#define POINT_H

typedef struct {
    int x, y;
    char *z;
} Point;

Point point_new(int x, int y, char *z);
void point_print(const Point *p);
void point_array_print(const char *msg, const Point *arr, int len);
void *comp(int num1, int num2);

#endif
```

5) Файл: sort.h

```
#ifndef SORT_H
#define SORT_H

void bubble_sort(Point *base, int len, int(*compar)(const Point *, const
Point *));
void double_selection_sort(Point *base, int len, int(*compar)(const Point *,
const Point *));

#endif
```

Листинг 2: Исходные коды программы lab4 (2)

1) Файл: prog.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "point.h"
#include "sort.h"

int read_input(int *input, int num1, int num2){
    int check = scanf("%d", input);
    if (check == -1){
        printf("Program has been stopped");
        return -1;
    }
    while (check == 0 || (num1 > *input) || (num2 < *input)){
        scanf("%*[^\n]");
        printf("You can't input such symbols.\n");
        check = scanf("%d", input);
    }
    return 0;
}

int menu(int *res, int *len, int *runs){
    int input = 0;

    printf("Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double
selection sort\n");
    if (read_input(&input, 1, 3) == -1){
        return -1;
    }
```

```
    res[0] = input;

    printf("Input a number of arrays:\n");
    if (read_input(&input, 1, 100000000) == -1){
        return -1;
    }
    *len = input;

    printf("Input a number of runs:\n");
    if (read_input(&input, 1, 100000000) == -1){
        return -1;
    }
    *runs = input;

    printf("Choose field: 1 - x, 2 - y, 3 - z\n");
    if (read_input(&input, 1, 3) == -1){
        return -1;
    }
    res[1] = input;

    printf("Choose direction: 1 - direct, 2 - inversed\n");
    if (read_input(&input, 1, 2) == -1){
        return -1;
    }
    res[2] = input;

    return 0;
}

int main(){
    int *res = calloc(3, sizeof(int));
    int len, runs, string_len;
    char *str = calloc(101, sizeof(char));
    char **str_arr = NULL;
    double ftime = 0;
    Point *arr = NULL;
    srand(time(NULL));
    if (menu(res, &len, &runs) == -1){
        return -1;
    }

    arr = calloc(len, sizeof(Point));
    str_arr = calloc(len, sizeof(char *));

    for (int i = 0; i < len; i++){
        string_len = rand() % 100 + 1;
        for (int j = 0; j < string_len; j++)
            str[j] = 'a' + rand() % 26;
        str[string_len] = '\0';
        str_arr[i] = strdup(str);
        arr[i] = point_new(rand(), rand() % 5, str_arr[i]);
        memset(str,0,101);
    }

    for (int i = 0; i < runs; i++){
        clock_t t0 = clock();
        if (res[0] == 1){
            qsort(arr, len, sizeof(Point), comp(res[1], res[2]));
        } else if (res[0] == 2){
            bubble_sort(arr, len, comp(res[1], res[2]));
        } else if (res[0] == 3){
            double_selection_sort(arr, len, comp(res[1], res[2]));
        }
```

```
        clock_t t1 = clock();
        ftime += (double)(t1 - t0) / CLOCKS_PER_SEC;
    }
    printf("arrays: %d\nsort: %d\nruns: %d\ntime: %lf\n", len, res[0], runs,
ftime / runs);

    for (int i = 0; i < len; i++){
        free(str_arr[i]);
    }

    free(str_arr);
    free(str);
    free(arr);
    free(res);
    return 0;
}
```

Файлы point.c, point.h, sort.c, sort.h идентичны соответствующим в lab5 (1).

5. Описание тестовых примеров

Таблица 1: Тестовые примеры lab5 (1)

Содержание текстового файла input.txt	Содержание текстового файла output.txt (первое поле, прямой порядок)	Содержание текстового файла output.txt (второе поле, обратный порядок)	Содержание текстового файла output.txt (третье поле, обратный порядок)
5 10 4 qwerty 7 0 abcdfglsigjjlitmobtsfilsrgkjugusilgrgilrsgirl 2 1 ui 68 3 qwer 11231324 2 uifo	2 1 ui 7 0 abcdfglsigjjlitmobtsfilsrgkjugusilgrgilrsgirl 10 4 qwerty 68 3 qwer 11231324 2 uifo	10 4 qwerty 68 3 qwer 11231324 2 uifo 2 1 ui 7 0 abcdfglsigjjlitmobtsfilsrgkjugusilgrgilrsgirl	11231324 2 uifo 2 1 ui 10 4 qwerty 68 3 qwer 7 0 abcdfglsigjjlitmobtsfilsrgkjugusilgrgilrsgirl

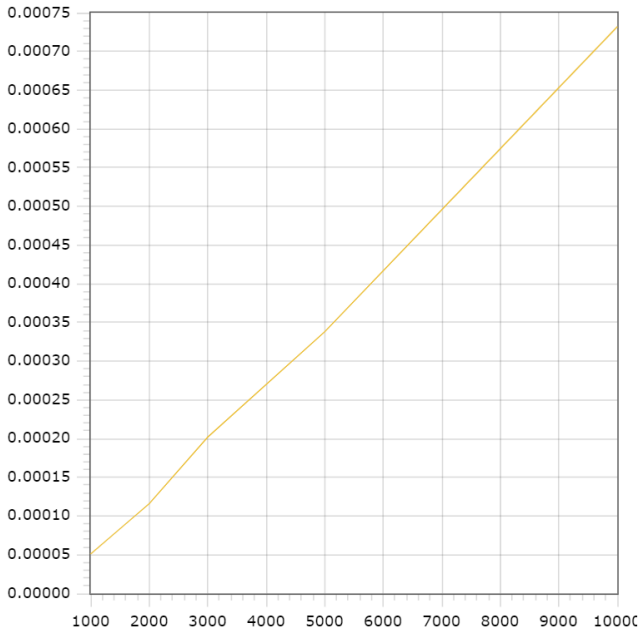
Таблица 2: Тестовые примеры lab5 (2)

Для сравнения методов сортировки использовалась прямая сортировка по первому полю, время усреднялось на основании 20 запусков сортировки для наборов структур со случайными данными в них (соответствующими заданным типам данных в полях структуры). Слева указано количество структур в массиве. Время указано в секундах.

	qsort	**bubble**	**double selection**
1000	0.000051	0.003253	0.002643
2000	0.000116	0.013104	0.009829
3000	0.000202	0.029338	0.022666
5000	0.000338	0.081961	0.065466

10000	0.000732	0.325592	0.257637

6. Скриншоты



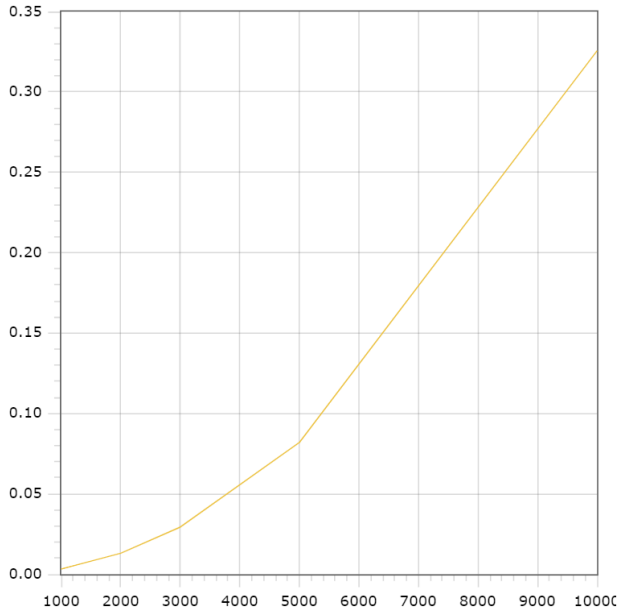


Рис. 3: График зависимости времени сортировки от количества элементов алгоритма qsort

Рис. 4: График зависимости времени сортировки от количества элементов алгоритма

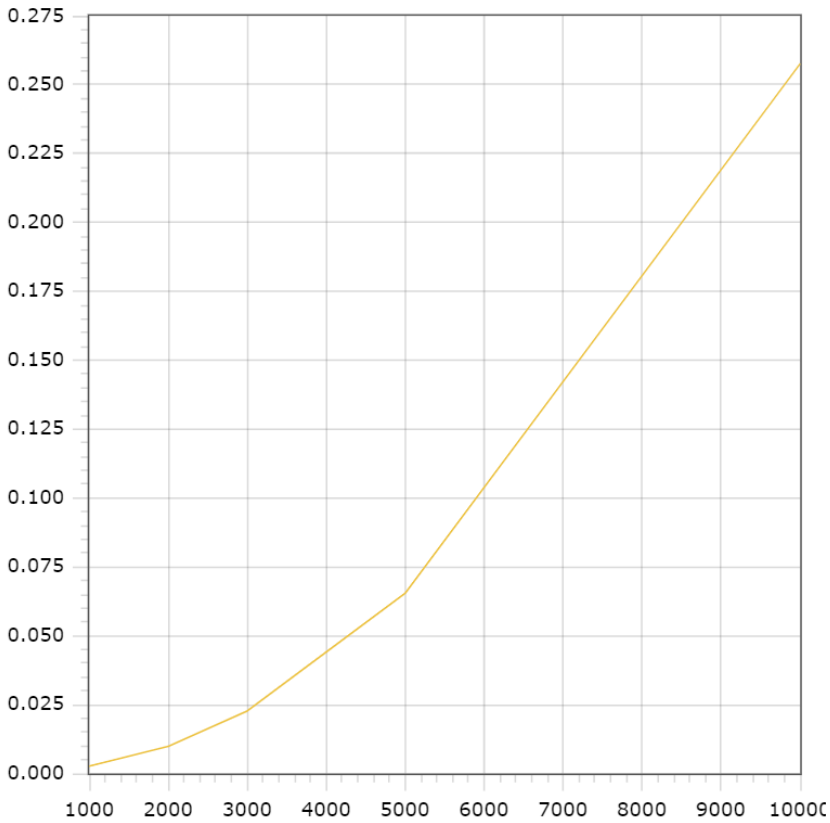


Рис. 5: График зависимости времени сортировки от количества элементов алгоритма double_selection_sort

```
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
1
Input a number of elements in arrays:
2
Input a number of arrays:
2000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 2000
sort: 1
runs: 20
time: 0.000116
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
1
Input a number of elements in arrays:
2
Input a number of arrays:
3000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 3000
sort: 1
runs: 20
time: 0.000202
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
1
Input a number of elements in arrays:
2
Input a number of arrays:
5000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 5000
sort: 1
runs: 20
time: 0.000338
```

```
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
2
Input a number of elements in arrays:
2
Input a number of arrays:
2000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 2000
sort: 2
runs: 20
time: 0.013104
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
2
Input a number of elements in arrays:
2
Input a number of arrays:
3000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 3000
sort: 2
runs: 20
time: 0.029338
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
2
Input a number of elements in arrays:
2
Input a number of arrays:
5000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 5000
sort: 2
runs: 20
time: 0.081961
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
2
Input a number of elements in arrays:
2
Input a number of arrays:
10000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 10000
sort: 2
runs: 20
time: 0.325592
[gatchenko.as@unix:~/Gatchenko/lab5/2]$
```

```
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
3
Input a number of elements in arrays:
2
Input a number of arrays:
3000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 3000
sort: 3
runs: 20
time: 0.022666
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
3
Input a number of elements in arrays:
2
Input a number of arrays:
5000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 5000
sort: 3
runs: 20
time: 0.065466
[gatchenko.as@unix:~/Gatchenko/lab5/2]$ ./run
Choose sorting algorithm: 1 - qsort, 2 - bubble sort, 3 - double selection sort
3
Input a number of elements in arrays:
2
Input a number of arrays:
10000
Choose field: 1 - x, 2 - y
1
Choose direction: 1 - direct, 2 - inversed
1
arrays: 10000
sort: 3
runs: 20
time: 0.257637
[gatchenko.as@unix:~/Gatchenko/lab5/2]$
```

Рис. 6: Сортировка методом quick_sort

Рис. 7: Сортировка методом bubble_sort

Рис. 8: Сортировка методом double_selection_sort

7. Выводы

В ходе выполнения данной работы на примере программы, выполняющей сортировку массивов данных, были рассмотрены базовые принципы работы построения программ на языке С и обработки строк:

1. Организация ввода/вывода, а также проверка корректности ввода.
2. Разработка функций.
3. Объявление и использование переменных.
4. Выполнение простейших арифметических операций над целочисленными и дробными операндами.
5. Использование циклов и условий.
6. Использование указателей (параметров) на символы, а также на массивы символов.
7. Разбиение программы на несколько файлов.
8. Работа с памятью.
9. Реализация различных методов сортировки.

Теоретическая скорость методов сортировки:

1. Qsort – $O(n\ log(n))$

2. Bubble_sort - $O(n^2)$

3. Double_selection_sort - $O(n^2)$

По графикам зависимости количества сортируемых элементов от времени видно, что для bubble_sort и double_selection_sort теоретическая и практическая скорости совпадают. График зависимости для quick_sort похож на линейную скорость, однако если взять значения времени для 1000 (0.000051 с) и для 10000 элементов (0.000732 с), то видно, что зависимость не линейная, а совпадает с теоретической, т.к. время выполнения возросло не в 10, а в большее количество раз.