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: Кругликова М. В.
Группа: Б22-504
Преподаватель: Комаров Т. И.

Москва – 2023

1. **Формулировка индивидуального задания**
   Вариант №67
   Структура данных Абонент:
   • ФИО (строка произвольной длины);
   • номер телефона (строка длиной до 16 символов, которая может включать в себя цифры, пробелы и знак «+» в начале);
   • время последнего звонка (целочисленное значение, соответствующее временной метке в формате UNIX time).
   Алгоритмы сортировки
   1. Шейкерная сортировка (Shaker sort).
   2. Сортировка вставками с бинарным поиском (Insertion sort with binary search)

2. **Описание использованных типов данных**
   При выполнении данной работы использовались такие типы данных, как:
   1. Встроенный тип данных int, предназначенный для работы с целыми числами
   2. Встроенный тип данных char, предназначенный для работы с символами
   3. Встроенный тип данных char *, предназначенный для работы с указателями на символы и массивы из них
   4. Встроенный тип данных FILE *, предназначенный для работы с файлами
   5. Собственный тип данных User, предназначенный для описания характеристик абонента
   6. Встроенный тип данных double, предназначенный для работы с вещественными числами
   7. Тип данных clock_t из библиотеки time.h, предназначенный для представления времени

3. **Описание использованного алгоритма**

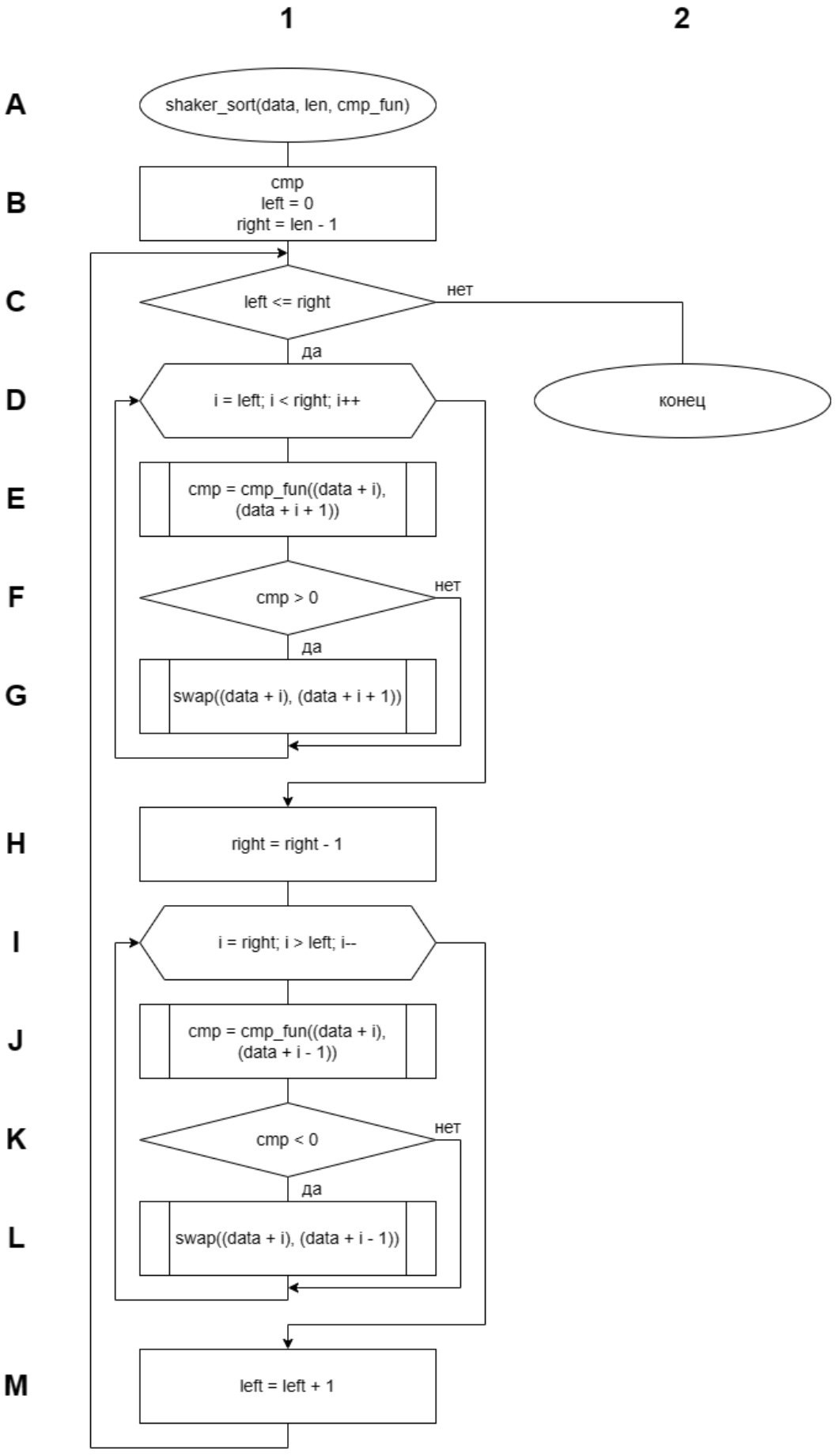

Рисунок 1. Блок-схема алгоритма работы функции shaker_sort (файл sorts.c)

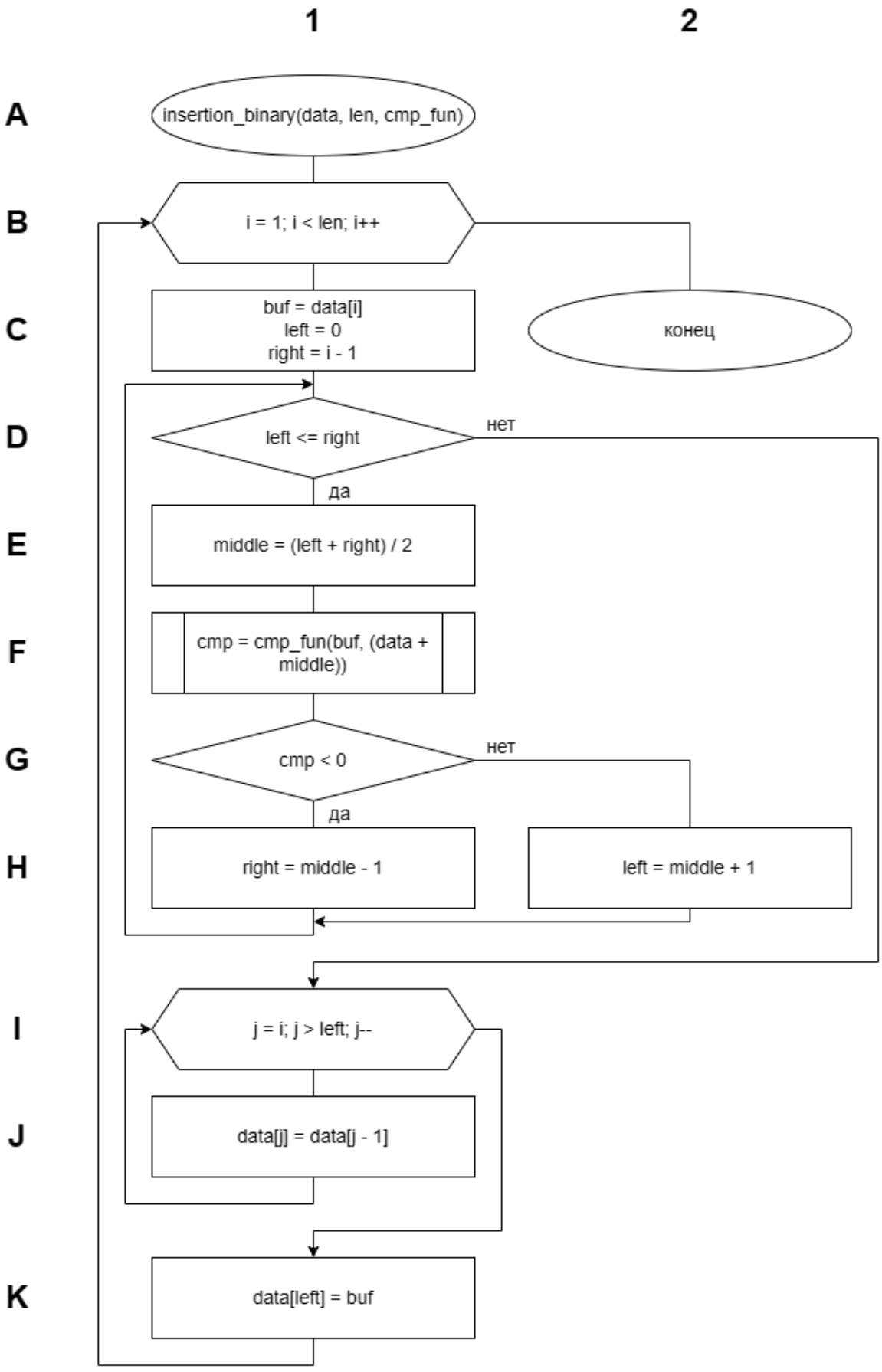

Рисунок 2. Блок-схема алгоритма работы функции insertion_binary (файл sorts.c)

4. Исходные коды разработанных программ

Листинг 1. Исходные коды программы lab5_1 (файлы getOptions.c, getOptions.h, main5_1.c, myFun.c, myFun.h, readData.c, readData.h, sorts.c, sorts.h, user.c, user.h, writeData.c, writeData.h)

Файл getOptions.c

```
#include <stdio.h>
#include <stdlib.h>

int get_option(int *opt1, int *opt23);
void printOpt1();
void printOpt2();
void printOpt3();
int readOpt12(int *n);
int readOpt3(int *n);

int get_option(int *opt1, int *opt23) {
    int opt2 = 0;
    int opt3 = 0;

    printOpt1();
    int ch = readOpt12(opt1);
    if (ch == 1) {
        return 1;
    }

    printOpt2();
    ch = readOpt12(&opt2);
    if (ch == 1) {
        return 1;
    }

    printOpt3();
    ch = readOpt3(&opt3);
    if (ch == 1) {
        return 1;
    }

    (*opt23) = opt2 * 10 + opt3;

    return 0;
}
```

```
void printOpt1() {
    printf("Выберите алгоритм сортировки:\n");
    printf("1 - qsort\n");
    printf("2 - Shaker sort\n");
    printf("3 - Insertion sort with binary
search\n");
}

void printOpt2() {
    printf("Выберите поле, по которому
осуществляется сортировка:\n");
    printf("1 - ФИО\n");
    printf("2 - номер телефона\n");
    printf("3 - время последнего звонка\n");
}

void printOpt3() {
    printf("Выберите направление сортировки:\n");
    printf("1 - по возрастанию\n");
    printf("2 - по убыванию\n");
}

int readOpt12(int *n) {
    int ch = 0;
    printf("Введите команду: ");
    ch = scanf("%d", n);
    if ((ch == EOF) || ((*n != 1) && (*n != 2) &&
(*n != 3))) {
        return 1;
    }

    return 0;
}

int readOpt3(int *n) {
    int ch = 0;
    printf("Введите команду: ");
    ch = scanf("%d", n);
    if ((ch == EOF) || ((*n != 1) && (*n != 2))) {
        return 1;
    }
```

```
    return 0;
}
```

Файл getOptions.h

```
#ifndef getOptions_H
#define getOptions_H

int get_option(int *opt1, int *opt23);
void printOpt1();
void printOpt2();
void printOpt3();
int readOpt12(int *n);
int readOpt3(int *n);

#endif
```

Файл main5_1.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <readline/readline.h>
#include "user.h"
#include "getOptions.h"
#include "readData.h"
#include "sorts.h"
#include "writeData.h"

int check_format();

int main() {
    int alg = 0;
    int field_way = 0;
    int check_opt = get_option(&alg, &field_way);
    if (check_opt == 1) {
        printf("Некорректный параметр\n");
        return 1;
    }

    char *name_input = readline("Введите имя
входного текстового файла: ");
    int check_f = check_format(name_input);
```

```
    if (check_f == -1) {
        printf("Некорректный формат файла\n");
        free(name_input);
        return -1;
    }
    FILE *file_input = fopen(name_input, "r");
    if (file_input == NULL) {
        printf("Файл не найден\n");
        free(name_input);
        return -1;
    }

    char *name_output = readline("Введите имя
выходного текстового файла: ");
    check_f = check_format(name_output);
    if (check_f == -1) {
        printf("Некорректный формат файла\n");
        free(name_input);
        free(name_output);
        return -1;
    }
    FILE *file_output = fopen(name_output, "w");
    if (file_output == NULL) {
        printf("Файл не найден\n");
        free(name_input);
        free(name_output);
        return -1;
    }

    int len_data = 0;
    User *data_user = read_data(file_input,
&len_data);
    if (len_data == 0) {
        printf("Массив данных пустой, так как в
файле отсутствуют корректные данные\n");
        free(name_input);
        fclose(file_input);
        user_free(data_user, len_data);
        free(data_user);
        return 1;
    }
```

```
    int (*compare)(const User*, const User*);
    choose_cmp(field_way, &compare);

    switch(alg) {
        case 1:
            qsort(data_user, len_data,
sizeof(User), (int (*)(const void *, const void *))
compare);
            break;
        case 2:
            shaker_sort(data_user, len_data,
compare);
            break;
        case 3:
            insertion_binary(data_user, len_data,
compare);
            break;
    }

    write_data(file_output, data_user, len_data);

    free(name_input);
    fclose(file_input);
    free(name_output);
    fclose(file_output);
    user_free(data_user, len_data);
    free(data_user);
    return 0;
}



int check_format(const char *name) {
    int len = strlen(name);
    if (len <= 4) return -1;
    if ((name[len - 4] != '.') || (name[len - 3] !=
't') || (name[len - 2] != 'x') || (name[len - 1] !=
't')) return -1;

    return 0;
}
```

Файл myFun.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int str_int(const char *str) {
    int sign = 1;
    int ind = 0;
    if (str[0] == '-') {
        sign = -1;
        ind = 1;
    }
    int res = 0;
    int i = ind;
    while (str[i] != '\0') {
        int d = str[i] - '0';
        res *= 10;
        res += d;
        i++;
    }
    res *= sign;
    return res;
}

double str_num(const char *str) {
    double res = 0;
    int ind = 0;
    while (str[ind] != '\0') {
        if ((str[ind] != ' ') && (str[ind] != '+'))
{
            int d = str[ind] - '0';
            res *= 10;
            res += d;
        }
        ind++;
    }
    return res;
}
```

Файл myFun.h

```
#ifndef myFun_H
```

```
#define myFun_H

int str_int();
double str_num();

#endif
```

Файл readData.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "user.h"
#include "myFun.h"
#include "readData.h"

#define str1
"aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyY
zZ -"
#define str2 "1234567890 "
#define str3 "1234567890"

User *read_data(FILE *file, int *len) {
    User *data = NULL;
    char *line = freadline(file);
    int count_str = 1;

    char *field1 = NULL;
    char field2[17];
    int field3 = 0;
    while (line) {
        int checkD = check_data(line, &field1,
field2, &field3);
        if (checkD == -1) {
            fprintf(stderr, "Некорректные данные в
%d строке файла\n", count_str);
        } else {
            (*len)++;
            data = realloc(data, (*len) *
sizeof(User));
            data[(*len) - 1] = user_new(field1,
field2, field3);
        }
```

```
            free(line);
            line = freadline(file);
            count_str++;
        }

        return data;
    }

    char *freadline(FILE *file) {
        char *ptr = (char *)malloc(1);
        char buf[81];
        int n, len = 0;
        *ptr = '\0';
        do {
            n = fscanf(file, "%80[^\n]", buf);
            if (n < 0) {
                free(ptr);
                ptr = NULL;
                continue;
            } else if (n == 0) {
                fscanf(file, "%*c");
            } else {
                len += strlen(buf);
                ptr = (char *)realloc(ptr, len + 1);
                strcat(ptr, buf);
            }
        } while (n > 0);
        return ptr;
    }

    int check_data(char *str, char **f1, char f2[17],
    int *f3) {
        int len_str = strlen(str);
        if (len_str == 0) {
            return -1;
        }
        int sep = count_sym(str, '|');
        if (sep != 2) {
            return -1;
        }
        if ((str[0] == '|') || (str[len_str - 1] ==
    '|')) {
```

```
        return -1;
    }
    if (strstr(str, "||") != NULL) {
        return -1;
    }

    (*f1) = strtok(str, "|");
    int len1 = strlen(*f1);
    int ch1 = strspn((*f1), str1);
    if ((ch1 < len1) || (count_sym((*f1), ' ') ==
len1)) {
        return -1;
    }

    char *f = strtok(NULL, "|");
    int len2 = strlen(f);
    if (len2 > 16) {
        return -1;
    }
    int ind = 0;
    while (f[ind] != '\0') {
        f2[ind] = f[ind];
        ind++;
    }
    f2[ind] = '\0';
    if (strspn(f, str3) == 0) {
        if ((*f) == '+') {
            f++;
            len2--;
        } else {
            return -1;
        }
    }
    int ch2 = strspn(f, str2);
    if ((ch2 < len2) || (count_sym(f, ' ') ==
len2)) {
        return -1;
    }

    f = strtok(NULL, "|");
    (*f3) = str_int(f);
    int len3 = strlen(f);
```

```
    if (strspn(f, str3) == 0) {
        if ((*f) == '-') {
            f++;
            len3--;
        } else {
            return -1;
        }
    }
    int ch3 = strspn(f, str3);
    if (ch3 < len3) {
        return -1;
    }

    return 0;
}

int count_sym(const char *s, char sym) {
    int len = strlen(s);
    int res = 0;
    for (int i = 0; i < len; i++) {
        if (s[i] == sym) {
            res++;
        }
    }
    return res;
}
```

Файл readData.h

```
#include "user.h"
#include <stdio.h>

#ifndef readData_H
#define readData_H

User *read_data(FILE *file, int *len);
char *freadline(FILE *file);
int check_data(char *line, char **f1, char f2[17],
int *f3);
int count_sym(const char *s, char sym);

#endif
```

Файл sorts.c

```
#include <stdio.h>
#include <stdlib.h>
#include "user.h"

void choose_cmp(int opt, int (**fun)(const User*,
const User*)) {
    switch(opt) {
        case 11:
            (*fun) = user_cmp_name_incr;
            break;
        case 12:
            (*fun) = user_cmp_name_decr;
            break;
        case 21:
            (*fun) = user_cmp_number_incr;
            break;
        case 22:
            (*fun) = user_cmp_number_decr;
            break;
        case 31:
            (*fun) = user_cmp_time_incr;
            break;
        case 32:
            (*fun) = user_cmp_time_decr;
            break;
    }
}

void swap(User *u1, User *u2) {
    User tmp = *u1;
    *u1 = *u2;
    *u2 = tmp;
}

void shaker_sort(User *data, int len, int
(*cmp_fun)(const User*, const User*)) {
    int cmp;
    int left = 0;
    int right = len - 1;
    while (left <= right) {
```

```
        for (int i = left; i < right; i++) {
            cmp = cmp_fun((data + i), (data + i +
1));
            if (cmp > 0) swap((data + i), (data +
i + 1));
        }
        right--;
        for (int i = right; i > left; i--) {
            cmp = cmp_fun((data + i), (data + i -
1));
            if (cmp < 0) swap((data + i), (data +
i - 1));
        }
        left++;
    }
}

void insertion_binary(User *data, int len, int
(*cmp_fun)(const User*, const User*)) {
    int cmp;
    for (int i = 1; i < len; i++) {
        User buf = data[i];
        int left = 0;
        int right = i - 1;
        while (left <= right) {
            int middle = (left + right) / 2;
            cmp = cmp_fun(&buf, (data + middle));
            if (cmp < 0) {
                right = middle - 1;
            } else {
                left = middle + 1;
            }
        }
        for (int j = i; j > left; j--) {
            data[j] = data[j - 1];
        }
        data[left] = buf;
    }
}
```

Файл sorts.h

```
#ifndef sorts_H
```

```
#define sorts_H

void choose_cmp();
void swap();
void shaker_sort();
void insertion_binary();

#endif
```

Файл user.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "user.h"
#include "myFun.h"

User user_new(char *name, char number[17], int
time) {
    User u;
    u.name = strdup(name);
    for (int i = 0; i < 17; i++) {
        u.number[i] = number[i];
    }
    u.time = time;
    return u;
}

void user_print(const User *u) {
    printf("{%s, %s, %d}", u->name, u->number, u-
>time);
}

void user_array_print(const User *arr, int len) {
    for (int i = 0; i < len; i++) {
        printf("a[%d] = ", i);
        user_print(arr + i);
        printf("\n");
    }
}

void user_free(User *arr, int len) {
    for (int i = 0; i < len; i++) {
```

```
            free(arr[i].name);
        }
    }


int user_cmp_name_incr(const User *u1, const User
*u2) {
    return strcmp(u1->name, u2->name);
}
int user_cmp_name_decr(const User *u1, const User
*u2) {
    return strcmp(u2->name, u1->name);
}

int user_cmp_number_incr(const User *u1, const User
*u2) {
    double num1 = str_num(u1->number);
    double num2 = str_num(u2->number);
    if (num1 > num2) return 1;
    else if (num1 == num2) return 0;
    else return -1;
}
int user_cmp_number_decr(const User *u1, const User
*u2) {
    double num1 = str_num(u1->number);
    double num2 = str_num(u2->number);
    if (num1 > num2) return -1;
    else if (num1 == num2) return 0;
    else return 1;
}

int user_cmp_time_incr(const User *u1, const User
*u2) {
    return u1->time - u2->time;
}
int user_cmp_time_decr(const User *u1, const User
*u2) {
    return u2->time - u1->time;
}
```

Файл user.h

```
#ifndef USER_H
```

```
#define USER_H

typedef struct {
    char *name;
    char number[17];
    int time;
} User;

User user_new(char *name, char *number, int time);
void user_print(const User *u);
void user_array_print(const User *arr, int len);
void user_free(User *arr, int len);


int user_cmp_name_incr(const User *u1, const User
*u2);
int user_cmp_name_decr(const User *u1, const User
*u2);

int user_cmp_number_incr(const User *u1, const User
*u2);
int user_cmp_number_decr(const User *u1, const User
*u2);


int user_cmp_time_incr(const User *u1, const User
*u2);
int user_cmp_time_decr(const User *u1, const User
*u2);

#endif
```

Файл writeData.c

```
#include <stdio.h>
#include <stdlib.h>
#include "user.h"

void write_data(FILE *file, User *data, int len) {
    for (int i = 0; i < len; i++) {
        fprintf(file, "%s|%s|%d\n", data[i].name,
data[i].number, data[i].time);
    }
```

```
}
```

Файл writeData.h

```
#ifndef writeData_H
#define writeData_H

void write_data();

#endif
```

Листинг 1. Исходные коды программы lab5_2 (файлы generateData.c, generateData.h, getOptions.c, getOptions.h, main5_2.c, myFun.c, myFun.h, sorts.c, sorts.h, user.c, user.h)

Файл generateData.c

```
#include <stdio.h>
#include <stdlib.h>
#include "user.h"

User *generate_data(int len);
char *generate_name();

User *generate_data(int len) {
    User *data = (User *) malloc(len *
sizeof(User));
    for (int i = 0; i < len; i++) {
        char *name = generate_name();
        char number[17];
        for (int j = 0; j < 16; j++) {
            number[j] = 48 + rand() % 10;
        }
        number[16] = '\0';
        int time = rand();
        data[i] = user_new(name, number, time);
    }

    return data;
}

char *generate_name() {
    char *name = (char *) malloc(1);
    int len_name = 25 + rand() % 11;
```

```
    for (int j = 0; j < len_name; j++) {
        name = realloc(name, (j + 1) *
sizeof(char));
        if ((j == len_name / 3) || (j == (len_name
/ 3) * 2)) {
            name[j] = ' ';
        } else {
            name[j] = 97 + rand() % 26;
        }
    }
    name = realloc(name, (len_name + 1) *
sizeof(char));
    name[len_name] = '\0';

    return name;
}
```

Файл generateData.h

```
#include "user.h"
#ifndef generateData_H
#define generateData_H

User *generate_data(int len);
char *generate_name();

#endif
```

Файл main5_2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "user.h"
#include "getOptions.h"
#include "generateData.h"
#include "sorts.h"

int main() {
    int alg = 0;
    int field_way = 0;
    int check = get_option(&alg, &field_way);
    if (check == 1) {
        printf("Некорректный параметр\n");
```

```
            return 1;
        }

        int q_el = 0;
        printf("Введите количество элементов в
генерируемых массивах: ");
        check = scanf("%d", &q_el);
        if (check != 1) {
            printf("Некорректное значение\n");
            return 1;
        }
        int q_arr = 0;
        printf("Введите количество генерируемых
массивов: ");
        check = scanf("%d", &q_arr);
        if (check != 1) {
            printf("Некорректное значение\n");
            return 1;
        }

        int (*compare)(const User*, const User*);
        choose_cmp(field_way, &compare);

        clock_t start;
        clock_t end;

        double time_do = 0;

        srand(time(NULL));
        for (int i = 0; i < q_arr; i++) {
            User *data_rand = generate_data(q_el);

            start = clock();
            switch(alg) {
                case 1:
                    qsort(data_rand, q_el,
sizeof(User), (int (*)(const void *, const void *))
compare);
                    break;
                case 2:
                    shaker_sort(data_rand, q_el,
compare);
```

```
                    break;
                case 3:
                    insertion_binary(data_rand, q_el,
compare);
                    break;
            }
            end = clock();

            time_do += (double)(end - start) /
CLOCKS_PER_SEC;

            user_free(data_rand, q_el);
            free(data_rand);
        }

        time_do /= q_arr;
        printf("TIME = %lf\n", time_do);

        return 0;
}
```

Файлы getOptions.c, getOptions.h, myFun.c, myFun.h, sorts.c, sorts.h, user.c, user.h приведены в листинге 1.

5. Описание тестовых примеров

Сравнив результаты со скриншотов из 2 и 3 столбцов таблицы 2 (см. раздел «скриншоты»), можно сделать вывод, что все сортировки возвращают корректный результат (1-3 – qsort, 4-6 – shaker sort, 7-9 – insertion sort with binary search)

Таблица 1. Сравнительная таблица для сортировок qsort, shaker sort, insertion sort with bunary search

Кол-во элементов	Qsort, время сортировки (с)	Shaker sort, время сортировки (с)	Insertion sort with binary search, время сортировки (с)
2000	0.00094	0.044971	0.004972
4000	0.00213	0.184183	0.020196
6000	0.00327	0.429555	0.044635
8000	0.0045	0.73546	0.079931
10000	0.00592	1.218395	0.122633
12000	0.00722	1.701997	0.169179

14000	0.00879	2.404154	0.227756
16000	0.00973	3.178473	0.300293
18000	0.01127	4.022978	0.38315
20000	0.01289	5.034891	0.491043
22000	0.0143	5.933204	0.564155
24000	0.01556	7.194229	0.665426
26000	0.01744	8.431107	0.806135
28000	0.01889	9.976463	0.926999
30000	0.01939	11.68079	1.06065

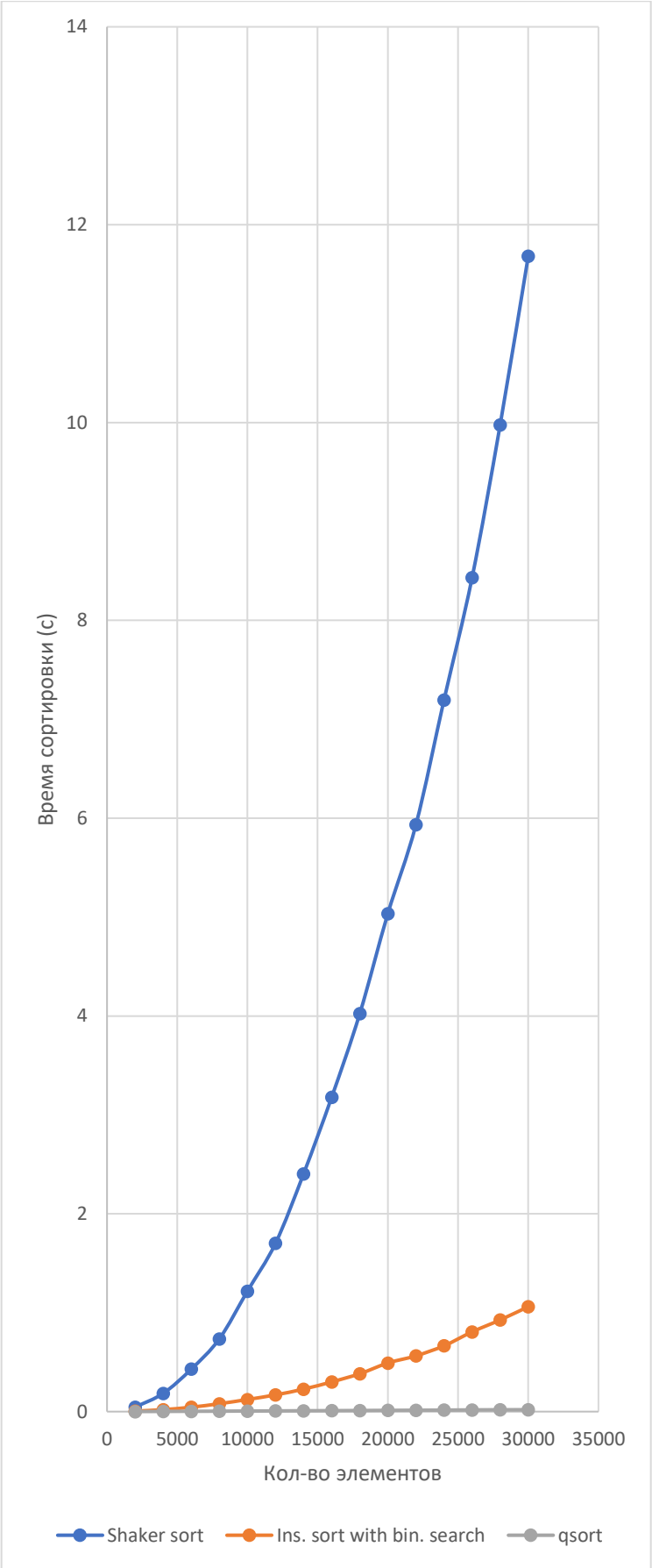

Рисунок 3. Сравнительный график времени сортировок qsort, shaker sort, insertion sort with binar search

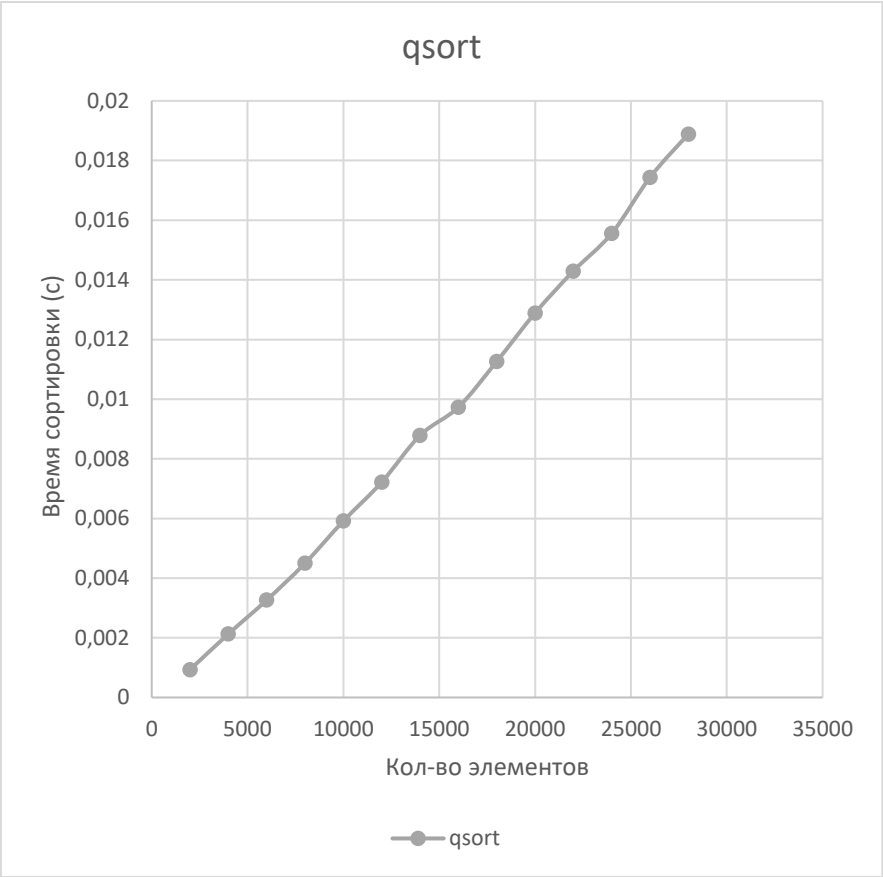

Рисунок 4. Зависимость времени сортировки qsort от кол-ва элементов

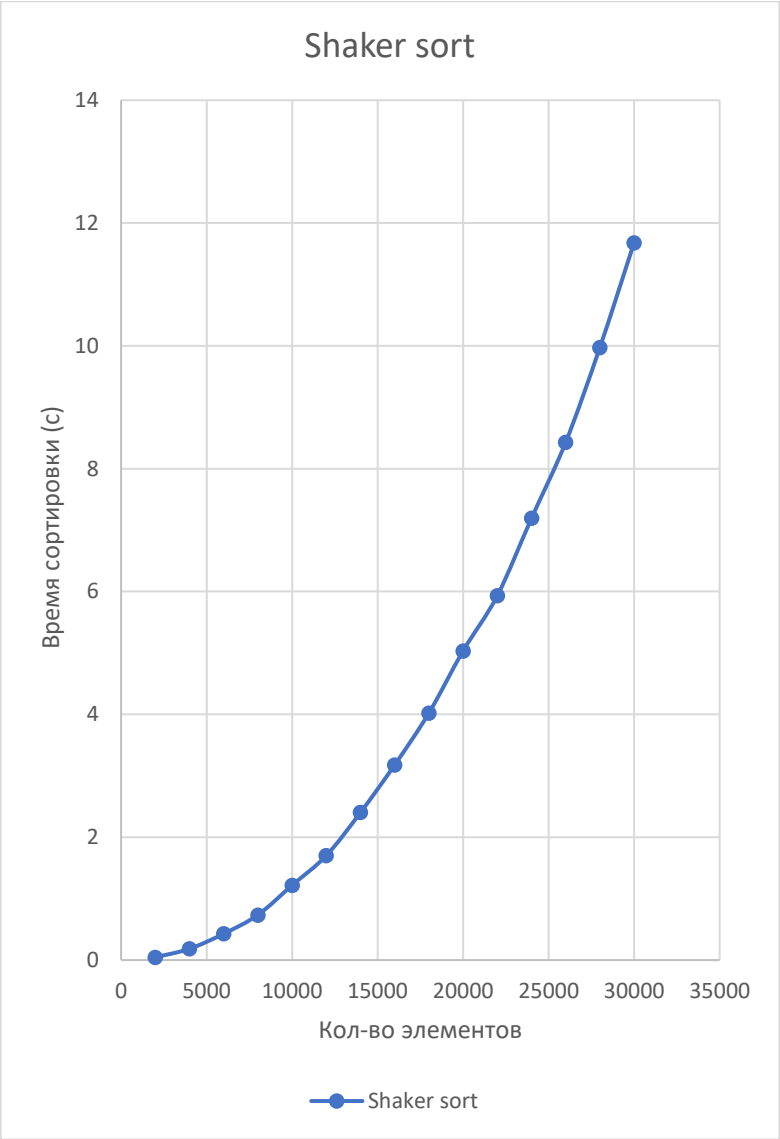

Рисунок 5. Зависимость времени сортировки shaker sort от кол-ва элементов

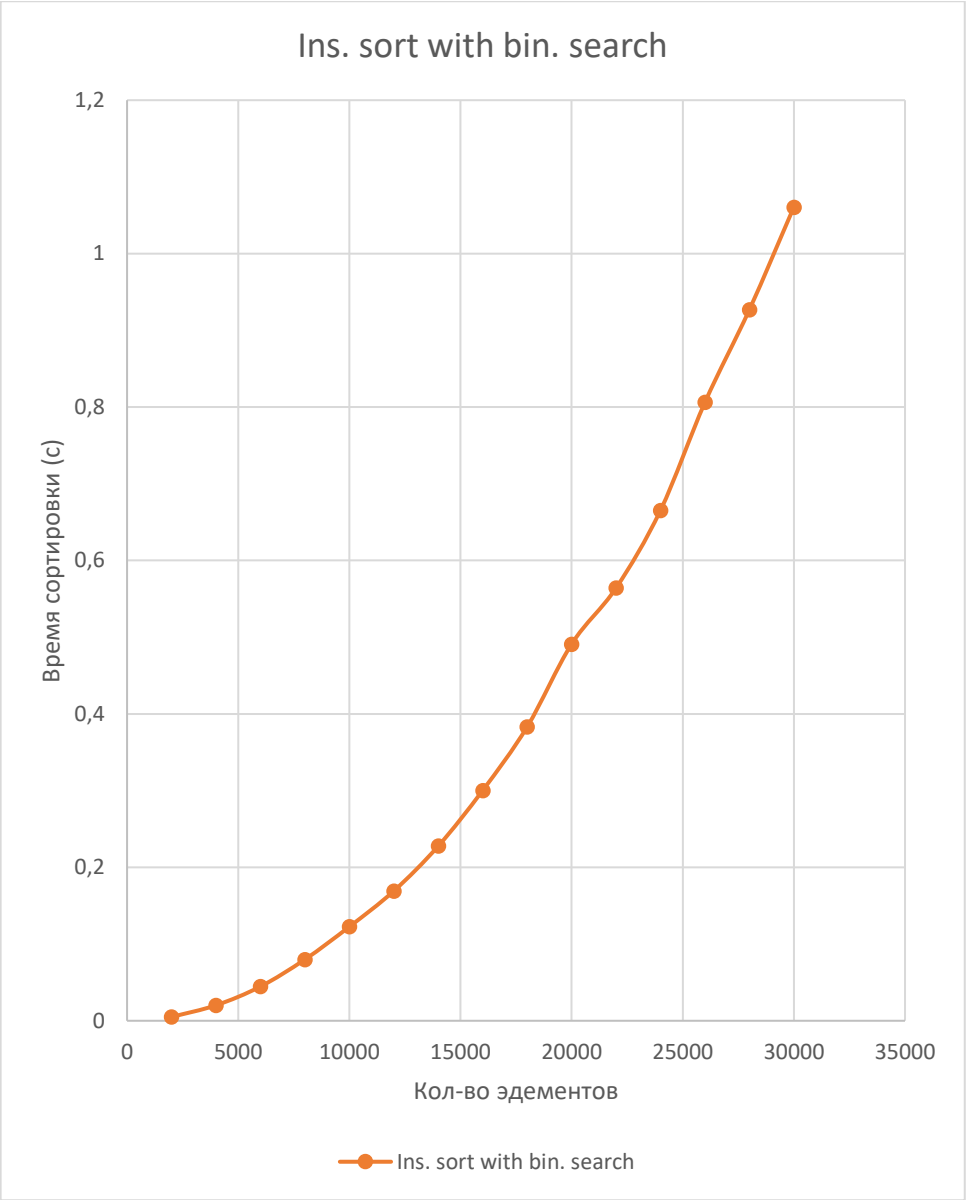

Рисунок 6. Зависимость времени сортировки insertion sort with binary search от времени

6. Скриншоты

samos.dozen.mephi.ru - PuTTY

```
 1 Ivanov Petr Sergeevich|+793487374ы897659|273469532
 2 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
 3
 4 Smirnova Elena Igorevna|8876345868+564675|10835
 5 Eltseva Maria Nicolaevna|+776398497648759847532 46|392874
 6 Fedorova Arina Evgenevna|+702992976597658|
 7 Kirsanov Pavel Petrovich|8 734 123 9876|08347503
 8 Verholomova Alica Vasilevna|23847552
 9                        |230948572|83475013948
10 Belousov Alexsandr Michailovich|+737876597659765|50938225
11 |Semenov Anton Egorovich|29920187548
12 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
13 Saveliev Igor Vladimirovich|703984275|01s98475243
14 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
15 Romanova Anna P@vlovna|230948572|83475013948
16
```

Рисунок 7. Входной файл input.txt

Таблица 2. Запуск программы lab5_1 при разных наборах входных данных

№	Скриншоты входного потока	Скриншоты выходного файла файла
1	[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1 ==10954== Memcheck, a memory error detector ==10954== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al. ==10954== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info ==10954== Command: ./lab5_1 ==10954== Выберите алгоритм сортировки: 1 - qsort 2 - Shaker sort 3 - Insertion sort with binary search Введите команду: 1 Выберите поле, по которому осуществляется сортировка: 1 - ФИО 2 - номер телефона 3 - время последнего звонка Введите команду: 1 Выберите направление сортировки: 1 - по возрастанию 2 - по убыванию Введите команду: 1 Введите имя входного текстового файла: input.txt Введите имя выходного текстового файла: output.txt Некорректные данные в 1 строке файла Некорректные данные в 3 строке файла Некорректные данные в 4 строке файла Некорректные данные в 5 строке файла Некорректные данные в 6 строке файла Некорректные данные в 8 строке файла Некорректные данные в 9 строке файла Некорректные данные в 11 строке файла Некорректные данные в 13 строке файла Некорректные данные в 15 строке файла ==10954== ==10954== HEAP SUMMARY: ==10954== in use at exit: 204,229 bytes in 221 blocks ==10954== total heap usage: 2,853 allocs, 2,632 frees, 400,208 bytes allocated ==10954== ==10954== LEAK SUMMARY: ==10954== definitely lost: 0 bytes in 0 blocks ==10954== indirectly lost: 0 bytes in 0 blocks ==10954== possibly lost: 0 bytes in 0 blocks ==10954== still reachable: 204,229 bytes in 221 blocks ==10954== suppressed: 0 bytes in 0 blocks ==10954== Rerun with --leak-check=full to see details of leaked memory ==10954== ==10954== For lists of detected and suppressed errors, rerun with: -s ==10954== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0) [kruglikova.mv@unix:~/inf/lab5/lab5_1]$	samos.dozen.mephi.ru - PuTTY 1 Belousov Alexsandr Michailovich|+737876597659765|50938225 2 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742 3 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847 4 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723 5 Kirsanov Pavel Petrovich|8 734 123 9876|8347503 6

2

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==11083== Memcheck, a memory error detector
==11083== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==11083== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==11083== Command: ./lab5_1
==11083==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 1
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 2
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 2
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==11083==
==11083== HEAP SUMMARY:
==11083==     in use at exit: 204,107 bytes in 219 blocks
==11083==   total heap usage: 2,852 allocs, 2,633 frees, 400,177 bytes allocated
==11083==
==11083== LEAK SUMMARY:
==11083==    definitely lost: 0 bytes in 0 blocks
==11083==    indirectly lost: 0 bytes in 0 blocks
==11083==      possibly lost: 0 bytes in 0 blocks
==11083==    still reachable: 204,107 bytes in 219 blocks
==11083==         suppressed: 0 bytes in 0 blocks
==11083== Rerun with --leak-check=full to see details of leaked memory
==11083==
==11083== For lists of detected and suppressed errors, rerun with: -s
==11083== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

samos.dozen.mephi.ru - PuTTY

```
1 Belousov Alexsandr Michailovich|+737876597659765|50938225
2 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
3 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
4 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
5 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
6
```

3

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==11256== Memcheck, a memory error detector
==11256== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==11256== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==11256== Command: ./lab5_1
==11256==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 1
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 3
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 2
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==11256==
==11256== HEAP SUMMARY:
==11256==     in use at exit: 204,229 bytes in 221 blocks
==11256==   total heap usage: 2,859 allocs, 2,638 frees, 400,342 bytes allocated
==11256==
==11256== LEAK SUMMARY:
==11256==    definitely lost: 0 bytes in 0 blocks
==11256==    indirectly lost: 0 bytes in 0 blocks
==11256==      possibly lost: 0 bytes in 0 blocks
==11256==    still reachable: 204,229 bytes in 221 blocks
==11256==         suppressed: 0 bytes in 0 blocks
==11256== Rerun with --leak-check=full to see details of leaked memory
==11256==
==11256== For lists of detected and suppressed errors, rerun with: -s
==11256== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

samos.dozen.mephi.ru - PuTTY

```
1 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
2 Belousov Alexsandr Michailovich|+737876597659765|50938225
3 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
4 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
5 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
6
```

4

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==11500== Memcheck, a memory error detector
==11500== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==11500== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==11500== Command: ./lab5_1
==11500==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 2
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 1
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 2
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==11500==
==11500== HEAP SUMMARY:
==11500==     in use at exit: 204,229 bytes in 221 blocks
==11500==   total heap usage: 2,853 allocs, 2,632 frees, 400,208 bytes allocated
==11500==
==11500== LEAK SUMMARY:
==11500==    definitely lost: 0 bytes in 0 blocks
==11500==    indirectly lost: 0 bytes in 0 blocks
==11500==      possibly lost: 0 bytes in 0 blocks
==11500==    still reachable: 204,229 bytes in 221 blocks
==11500==         suppressed: 0 bytes in 0 blocks
==11500== Rerun with --leak-check=full to see details of leaked memory
==11500==
==11500== For lists of detected and suppressed errors, rerun with: -s
==11500== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

samos.dozen.mephi.ru - PuTTY

```
1 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
2 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
3 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
4 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
5 Belousov Alexsandr Michailovich|+737876597659765|50938225
6
```

5

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==11556== Memcheck, a memory error detector
==11556== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==11556== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==11556== Command: ./lab5_1
==11556==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 2
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 2
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 1
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==11556==
==11556== HEAP SUMMARY:
==11556==     in use at exit: 204,229 bytes in 221 blocks
==11556==   total heap usage: 2,853 allocs, 2,632 frees, 400,208 bytes allocated
==11556==
==11556== LEAK SUMMARY:
==11556==    definitely lost: 0 bytes in 0 blocks
==11556==    indirectly lost: 0 bytes in 0 blocks
==11556==      possibly lost: 0 bytes in 0 blocks
==11556==    still reachable: 204,229 bytes in 221 blocks
==11556==         suppressed: 0 bytes in 0 blocks
==11556== Rerun with --leak-check=full to see details of leaked memory
==11556==
==11556== For lists of detected and suppressed errors, rerun with: -s
==11556== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

samos.dozen.mephi.ru - PuTTY

```
1 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
2 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
3 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
4 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
5 Belousov Alexsandr Michailovich|+737876597659765|50938225
6
```

6

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==11614== Memcheck, a memory error detector
==11614== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==11614== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==11614== Command: ./lab5_1
==11614==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 2
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 3
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 1
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==11614==
==11614== HEAP SUMMARY:
==11614==     in use at exit: 204,229 bytes in 221 blocks
==11614==   total heap usage: 2,853 allocs, 2,632 frees, 400,208 bytes allocated
==11614==
==11614== LEAK SUMMARY:
==11614==    definitely lost: 0 bytes in 0 blocks
==11614==    indirectly lost: 0 bytes in 0 blocks
==11614==      possibly lost: 0 bytes in 0 blocks
==11614==    still reachable: 204,229 bytes in 221 blocks
==11614==         suppressed: 0 bytes in 0 blocks
==11614== Rerun with --leak-check=full to see details of leaked memory
==11614==
==11614== For lists of detected and suppressed errors, rerun with: -s
==11614== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

samos.dozen.mephi.ru - PuTTY

```
1 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
2 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
3 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
4 Belousov Alexsandr Michailovich|+737876597659765|50938225
5 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
6
```

7

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==11778== Memcheck, a memory error detector
==11778== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==11778== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==11778== Command: ./lab5_1
==11778==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 3
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 1
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 2
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==11778==
==11778== HEAP SUMMARY:
==11778==     in use at exit: 204,229 bytes in 221 blocks
==11778==   total heap usage: 2,853 allocs, 2,632 frees, 400,208 bytes allocated
==11778==
==11778== LEAK SUMMARY:
==11778==    definitely lost: 0 bytes in 0 blocks
==11778==    indirectly lost: 0 bytes in 0 blocks
==11778==      possibly lost: 0 bytes in 0 blocks
==11778==    still reachable: 204,229 bytes in 221 blocks
==11778==         suppressed: 0 bytes in 0 blocks
==11778== Rerun with --leak-check=full to see details of leaked memory
==11778==
==11778== For lists of detected and suppressed errors, rerun with: -s
==11778== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

samos.dozen.mephi.ru - PuTTY

```
1 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
2 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
3 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
4 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
5 Belousov Alexsandr Michailovich|+737876597659765|50938225
6
```

8

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==11914== Memcheck, a memory error detector
==11914== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==11914== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==11914== Command: ./lab5_1
==11914==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 3
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 2
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 1
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==11914==
==11914== HEAP SUMMARY:
==11914==     in use at exit: 204,107 bytes in 219 blocks
==11914==   total heap usage: 2,852 allocs, 2,633 frees, 400,177 bytes allocated
==11914==
==11914== LEAK SUMMARY:
==11914==    definitely lost: 0 bytes in 0 blocks
==11914==    indirectly lost: 0 bytes in 0 blocks
==11914==      possibly lost: 0 bytes in 0 blocks
==11914==    still reachable: 204,107 bytes in 219 blocks
==11914==         suppressed: 0 bytes in 0 blocks
==11914== Rerun with --leak-check=full to see details of leaked memory
==11914==
==11914== For lists of detected and suppressed errors, rerun with: -s
==11914== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

```
samos.dozen.mephi.ru - PuTTY
1 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
2 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
3 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
4 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
5 Belousov Alexsandr Michailovich|+737876597659765|50938225
6
```

9

```
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$ valgrind ./lab5_1
==12084== Memcheck, a memory error detector
==12084== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==12084== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==12084== Command: ./lab5_1
==12084==
Выберите алгоритм сортировки:
1 - qsort
2 - Shaker sort
3 - Insertion sort with binary search
Введите команду: 3
Выберите поле, по которому осуществляется сортировка:
1 - ФИО
2 - номер телефона
3 - время последнего звонка
Введите команду: 3
Выберите направление сортировки:
1 - по возрастанию
2 - по убыванию
Введите команду: 2
Введите имя входного текстового файла: input.txt
Введите имя выходного текстового файла: output.txt
Некорректные данные в 1 строке файла
Некорректные данные в 3 строке файла
Некорректные данные в 4 строке файла
Некорректные данные в 5 строке файла
Некорректные данные в 6 строке файла
Некорректные данные в 8 строке файла
Некорректные данные в 9 строке файла
Некорректные данные в 11 строке файла
Некорректные данные в 13 строке файла
Некорректные данные в 15 строке файла
==12084==
==12084== HEAP SUMMARY:
==12084==     in use at exit: 204,229 bytes in 221 blocks
==12084==   total heap usage: 2,853 allocs, 2,632 frees, 400,208 bytes allocated
==12084==
==12084== LEAK SUMMARY:
==12084==    definitely lost: 0 bytes in 0 blocks
==12084==    indirectly lost: 0 bytes in 0 blocks
==12084==      possibly lost: 0 bytes in 0 blocks
==12084==    still reachable: 204,229 bytes in 221 blocks
==12084==         suppressed: 0 bytes in 0 blocks
==12084== Rerun with --leak-check=full to see details of leaked memory
==12084==
==12084== For lists of detected and suppressed errors, rerun with: -s
==12084== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
[kruglikova.mv@unix:~/inf/lab5/lab5_1]$
```

```
samos.dozen.mephi.ru - PuTTY
1 Ivanichuk Alexsandr Vladimirovich|8938 942 34 56 |72093847
2 Belousov Alexsandr Michailovich|+737876597659765|50938225
3 Kirsanov Pavel Petrovich|8 734 123 9876|8347503
4 Goncharova Anastasia Dmitrievna|+7 925 790 63 47|198742
5 Kirsanov Nicolay Petrovich|+7 34567898 21 |-7394723
6
```

7. Выводы

По расчетам каждая из сортировок имеет свою среднюю сложность алгоритма:

1. Qsort основана на том, что массив рекурсивно разбивается на части и каждая из частей сортируется. Количество рекурсивных разбиений не больше чем O(log n), количество операций на каждом уровне рекурсии не более чем O(n). Значит средняя сложность алгоритма qsort составляет O(n * log n). График на рис. 4 соответствует этой сложности

2. Shaker sort основана на том, что отсортированные элементы постепенно накапливаются в начале и в конце массива: за каждую итерацию цикла наибольший элемент перемещается в конец неотсортированной части массива, а наименьший – в начало. Таким образом, при первой итерации сложность алгоритма составит не более чем n(n – 1), при второй (n – 2)(n – 3) и тд. Значит при каждой итерации цикла сложность на данном этапе будет составлять O(n^2). Значит средняя сложность сортировки также составит O(n^2). График на рис. 5 соответствует этой сложности
3. Insertion sort with binary search основана на том, что слева от рассматриваемого элемента, в отсортированной части массива, ищется место, на которое должен быть вставлен рассматриваемый элемент. Всего в массиве n элементов, чтобы поставить каждый из них на место нужно не более n операций, значит сложность алгоритма O(n^2). График на рис. 6 соответствует этой сложности