Национальный исследовательский ядерный университет
«МИФИ» Институт интеллектуальных кибернетических
систем

Кафедра №12 «Компьютерные системы и технологии»

ОТЧЕТ

О выполнении лабораторной работы №5

«Работа с массивами структур. Исследование методов сортировки массивов»

Студент: Титов Иван Андреевич.

Преподаватель: Уваров М.П.

Группа: Б23-901

Москва – г .2023

I.Подготовительная часть

Введение

В ходе выполнения лабораторной работы должны быть разработаны:

1. Программа № 1, осуществляющая в диалоговом режиме ввод, сортировку и вывод последовательности данных, которая представляется в виде массива структур.

2. Программа № 2, осуществляющая таймирование сортировки массивов.

Программы № 1 и № 2 должны реализовывать поддержку работы с тремя алгоритмами сортировок: с двумя из состава индивидуального задания и с реализацией алгоритма быстрой сортировки из состава стандартной библиотеки — функцией `qsort()`.

Программа № 1 должна реализовывать следующую функциональность:

1. Ввод массива:

   (a) из стандартного потока ввода потока («с клавиатуры»);

   (b) из текстового файла (с возможностью указания имени файла);

   (c) из бинарного файла (с возможностью указания имени файла).

2. Вывод массива:

   (a) в стандартный поток вывода («на экран»);

   (b) в текстовый файл (с возможностью указания имени файла);

   (c) в бинарный файл (с возможностью указания имени файла).

3. Сортировка массива с возможностью выбора пользователем через диалоговое меню:

   (a) алгоритма сортировки (одного из трёх);

   (b) поля структуры, по которому осуществляется сортировка;

   (c) направления сортировки (по убыванию или по возрастанию).

Программа № 2 должна реализовывать таймирование сортировки с возможностью выбора пользователем через диалоговое меню:

1. алгоритма сортировки (одного из трёх);

2. поля структуры, по которому осуществляется сортировка;

3. направления сортировки (по убыванию или по по возрастанию);

4. количества элементов в генерируемых массивах;

5. количества генерируемых массивов.

Примечания:

1. Взаимодействие программ с пользователем должно быть выстроено с помощью иерархического диалогового меню.

2. Программа № 1 должна осуществлять проверку корректности данных, вводимых пользователем, и, в случае ошибок, выдавать соответствующие сообщения, после чего продолжать работу.

3. Программа № 1 должна осуществлять проверку корректности данных, считываемых из файлов. В случае ошибок формата файла — выдавать соответствующие сообщения в стандартный поток вывода ошибок и продолжать работу, считая что ввод не был выполнен успешно. В случае некорректных данных для конкретных записей — выдавать соответствующие сообщения в поток ошибок, после чего продолжать работу, игнорируя данные записи.

4. Для работы с данными, формат которых описан в индивидуальном задании, должен быть разработан собственный составной тип данных — структура.

5. Для работы с данными, структура которых описана в индивидуальном задании, должен быть разработан формат хранения в текстовом файле.

6. Для работы с данными, структура которых описана в индивидуальном задании, должен быть разработан формат хранения в бинарном файле.
7. Работа с текстовыми файлами должна осуществляться при помощи функций стандартной библиотеки `fopen()`, `fclose()`, `fprintf()`, `fscanf()`.
8. Работа с бинарными файлами должна осуществляться при помощи функций стандартной библиотеки `fopen()`, `fclose()`, `fread()`, `fwrite()`.
9. Логически законченные части алгоритма решения задачи должны быть оформлены в виде отдельных функций с параметрами. Использование глобальных переменных не допускается.
10. Исходные коды программы должны быть логичным образом разбиты на несколько файлов.
11. Программа должна корректным образом работать с памятью, для проверки необходимо использовать соответствующие программные средства, например: valgrind (при тестировании и отладке программы ее необходимо запускать командой вида `valgrind ./lab5`, а при анализе производительности — `./lab5`).

Отчётность по выполнению лабораторной работы должна включать:

1. Блок-схемы алгоритмов сортировки массива.
2. Исходные коды программ.
3. Тестовые наборы для иллюстрации работы программ.
4. Результаты таймирования, содержащие таблицы, графики зависимости времени выполнения сортировок от количества сортируемых элементов и аргументированные выводы об оценке сложности рассмотренных алгоритмов сортировки и её совпадении с теоретическими ожиданиями.

Индивидуальное задание

Структура данных

Автомобиль:

- марка (строка длиной до 16 символов, которая может включать в себя только буквы, дефис и пробелы);
- ФИО владельца (строка произвольной длины);
- пробег (дробное число, соответсвующее величине пробега в тыс. км).

Алгоритмы сортировки

1. Гномья сортировка (Gnome sort).
2. Сортировка выбором (Selection sort).

Цель: организовать программу, имеющую необходимый функционал и отвечающую ряду требований

Функционал:

1. Программа №1:
   А) Хранение массив структур вида:
   -Марка производителя
   -ФИО владельца
   -Значение пробега
   Б) Осуществление ввода структуры с клавиатуры
   В) Осуществление ввода структур из текстового файла

Г) Осуществление ввода структур из бинарного файла
Д) Осуществление вывода структур в терминал
Е) Осуществление вывода структур в текстовый файл
Ё) Осуществление вывода структур в бинарный файл
Ж) Осуществление сортировки массива структур одним из трёх алгоритмов по 1 из 3 параметров, являющихся полями структуры

2. Программа №2:
   А) Создание массива структур со случайными значениями
   Б) Позволять пользователю выбирать параметры сортировки массива структур: по алгоритму сортировки, по типу сортируемого поля, по направлению сортировки

Требования:

1. Логичным образом разбить проект на файлы (заголовочные и с исходным кодом) и функции
2. Не иметь утечек памяти при работе программы (из-за этого, кстати говоря, используется собственная функция oureadline() вместо readline() для считывания строки id с нефиксированной длиной, фукнция помещена в отдельный файл sup.c)
3. Использовать при работе с файлами указанные функции
4. Осуществлять различные проверки корректности вводимых данных при работе программы (в частности, при выборе пунктов меню, при вводе наименования файла, при вводе полей структуры с клавиатуры, при обработке считываемого файла)

II.Алгоритмическая часть

Состав программы 1:

1. Printstruct(const Auto *p, int n) – функция, выводящая в терминал поля заданного массива p, n-го по счёту в выводе.
2. Printarr(const Auto *arr, int len) – функция, вызывающая для каждой структуры arr+i из передаваемого массива длиной len функцию Printstruct.
3. input() – функция, осуществляющая ввод полей структуры (brand, id, run) и осуществляющая их проверку на соответствие формату, возвращает структуру
4. inputtextfile(size_t *k) – функция, осуществляющая чтение текстового файла (ввод его названия, открытие для чтения и обработка ошибок), считывание полей структур (многих или одной) и обработка ошибок несоответствия их формата. Подаваемая на вход величина k отражает количество считанных структур (это необходимо для довыделения памяти для массива структур в основной функции, а также записи считанных структур). Используется функция fscanf
5. inputbinfile(size_t *k) - функция, осуществляющая чтение файла (ввод его названия, открытие для чтения и обработка ошибок), считывание полей структур (многих или одной) и обработка ошибок несоответствия их формата. Подаваемая на вход величина k отражает количество считанных структур (это необходимо для довыделения памяти для массива структур в основной функции, а также записи считанных структур). Используется функция fread
6. outputtextfile (const Auto *arr, int len) – функция. Количество len структур массива структур Auto поштучно записывается в файл, чьё название вводится в начале функции. Расширение созданного файла - .txt. В файл записываются (по порядку):
   - Длина id
   - id
   - brand
   - run

   В данном случае необходимо сначала считывать длину id, чтобы корректно выделить память для id при чтении (напомним, что у неё нефиксированная длина.
7. outputbinfile (const Auto *arr, int len) – функция. Количество len структур массива структур Auto поштучно записывается в файл, чьё название вводится в начале функции. Расширение созданного файла - .bin. В файл записываются (по порядку):
   - Длина id
   - Длина brand
   - id
   - brand
   - run

   В данном случае необходимо сначала считывать длину id, чтобы корректно выделить память для id при чтении (напомним, что у неё

нефиксированная длина. Длина brand же также необходима, так как в бинарной записи невозможно отследить, когда заканчивается строка (даже фиксированного размера). Без длины brand возникают утечки памяти и ошибки (это говоря о том, почему нельзя использовать значение фиксированной длины).

8. main() – основная функция, осуществляющая инициализацию массива структур, вывод в терминал и выбор пунктов меню по средствам switch-case. Осуществляется подключение функций, перечисленных выше (Printstruct() вызывается в printarr())

Состав программы 2:

1. Printstruct(const Auto *p, int n) – функция, выводящая в терминал поля заданного массива p, n-го по счёту в выводе.
2. Printarr(const Auto *arr, int len) – функция, вызывающая для каждой структуры arr+i из передаваемого массива длиной len функцию Printstruct.
3. createAutos(int nstr, int n) – генерация (псевдо)случайных значений для nstr структур и при длине id в генерируемых структурах n. Поле brand всегда генерируется на 17 символов, являющихся заглавными английскими буквами. Id заполняется также заглавными буквами, а run – числами от 1 до 100000.
4. main() – функция осуществляющая инициализацию массива структур, работу с меню, а также ввод значений n и nstr для генерации массива структур

Реализации алгоритмов сортировки:

1. Гномья сортировка gsort(structs,i,sizeof(Auto),comp_s_id)
2. Сортировка выбором csort(structs,i,sizeof(Auto),comp_s_id)

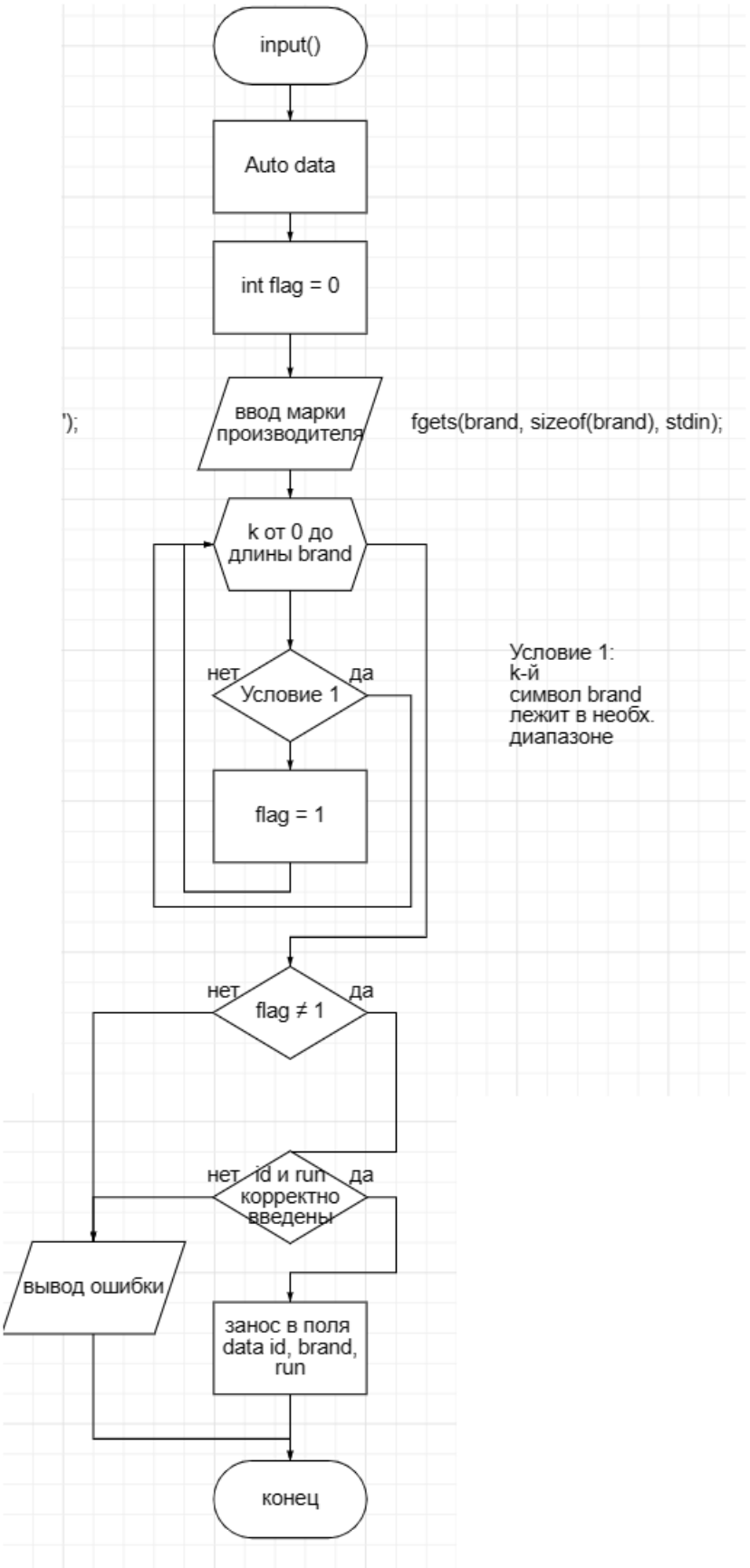
Компараторы:

1. comp_s_b – сорт. с начала по полю brand
2. comp_s_id – сорт. с начала по полю id
3. comp_s_r – сорт. с начала по полю run
4. comp_e_b – сорт. с конца по полю brand
5. comp_e_id – сорт. с конца по полю id
6. comp_e_r – сорт. с конца по полю run

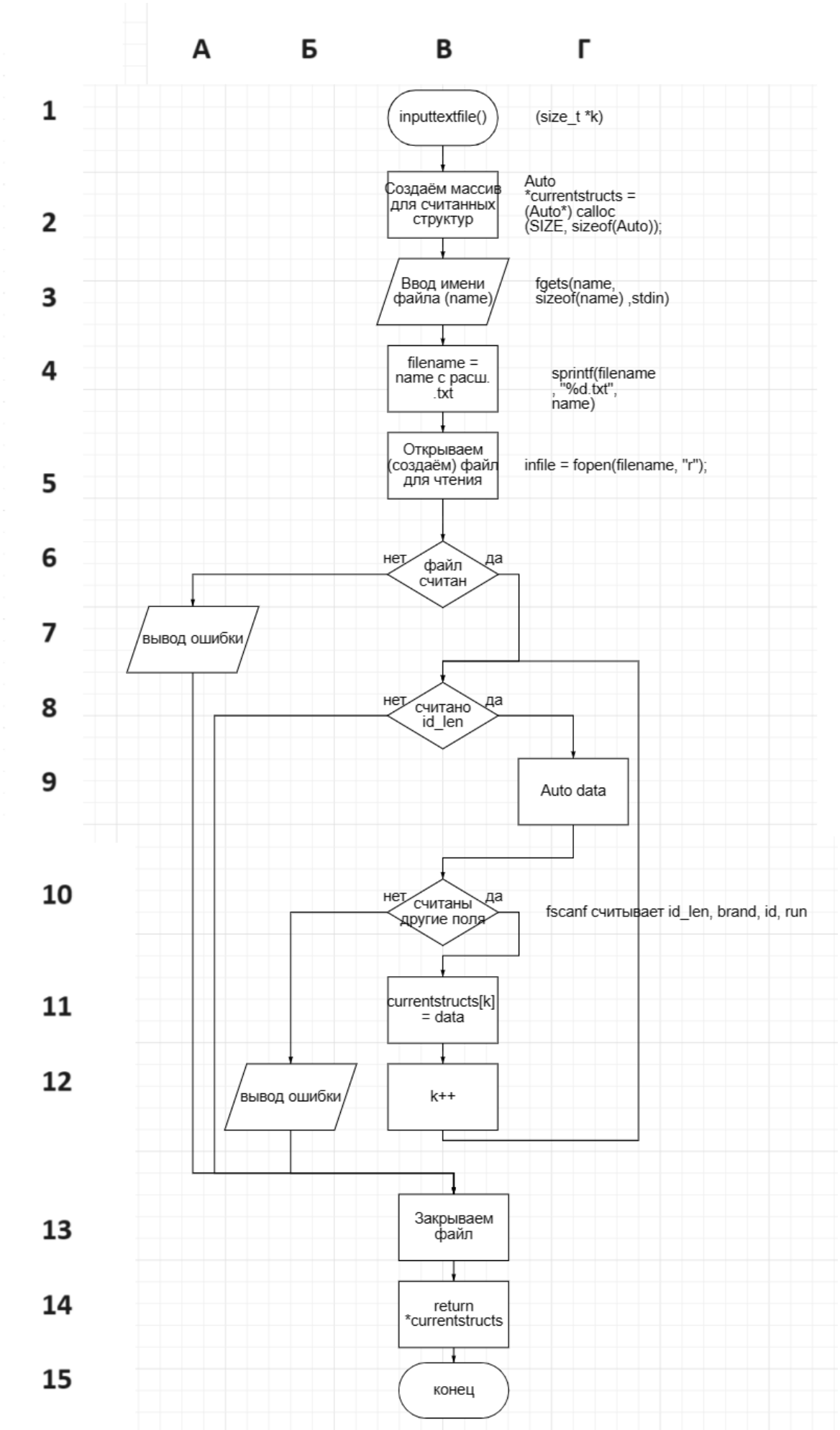
Блок схемы:

Программа 1:

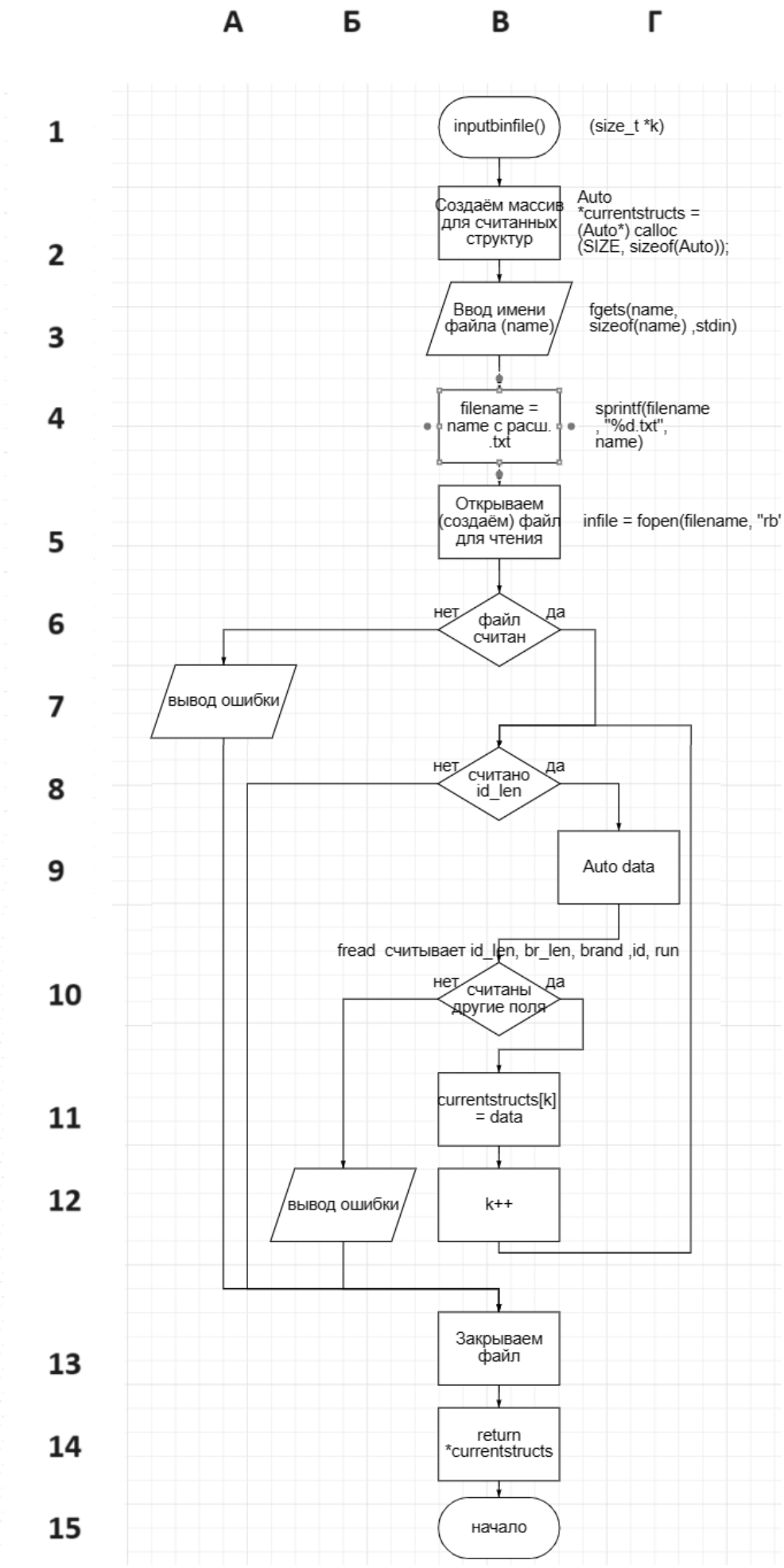
1. Input()



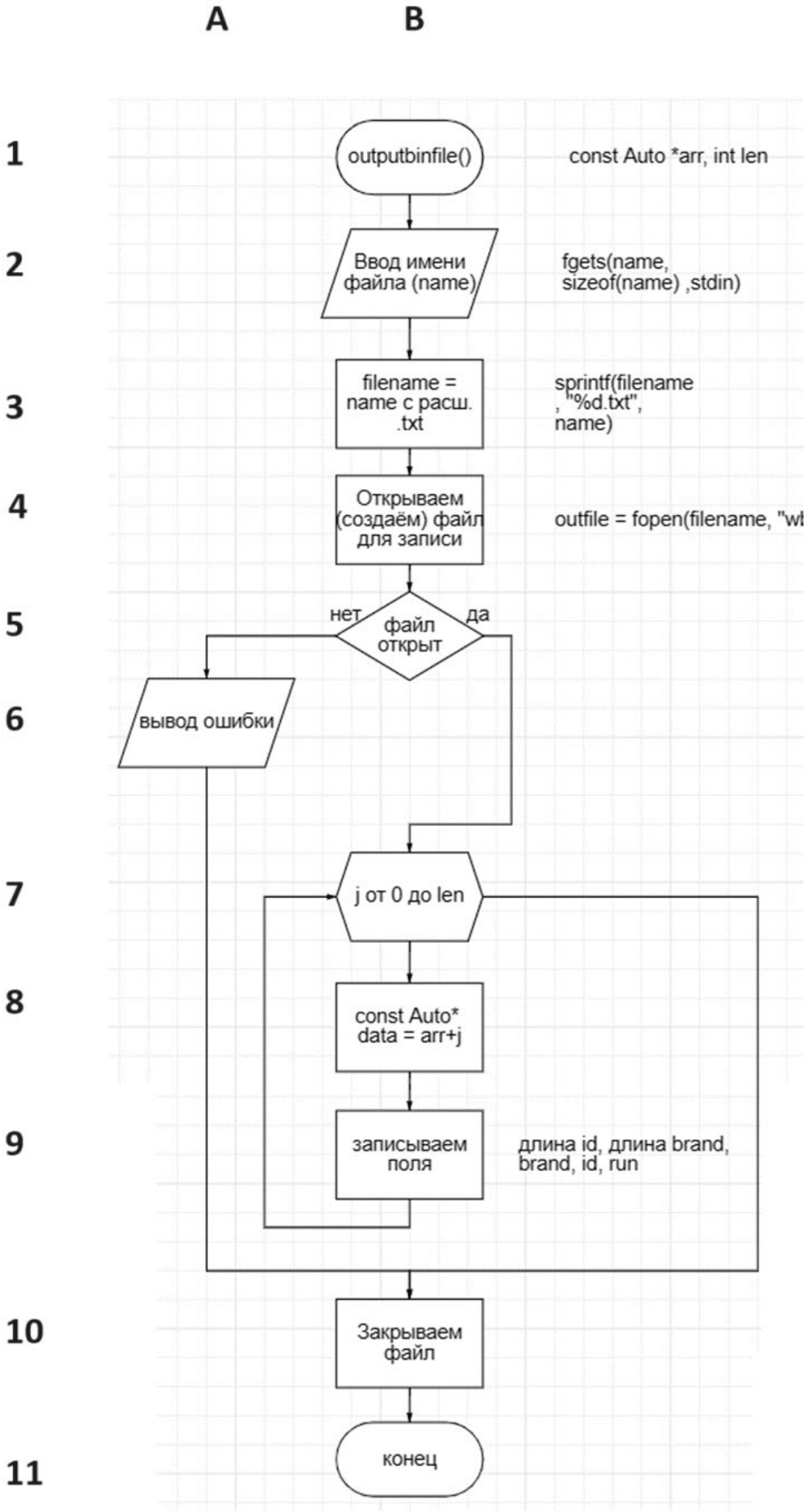
2. Inputtextfile()



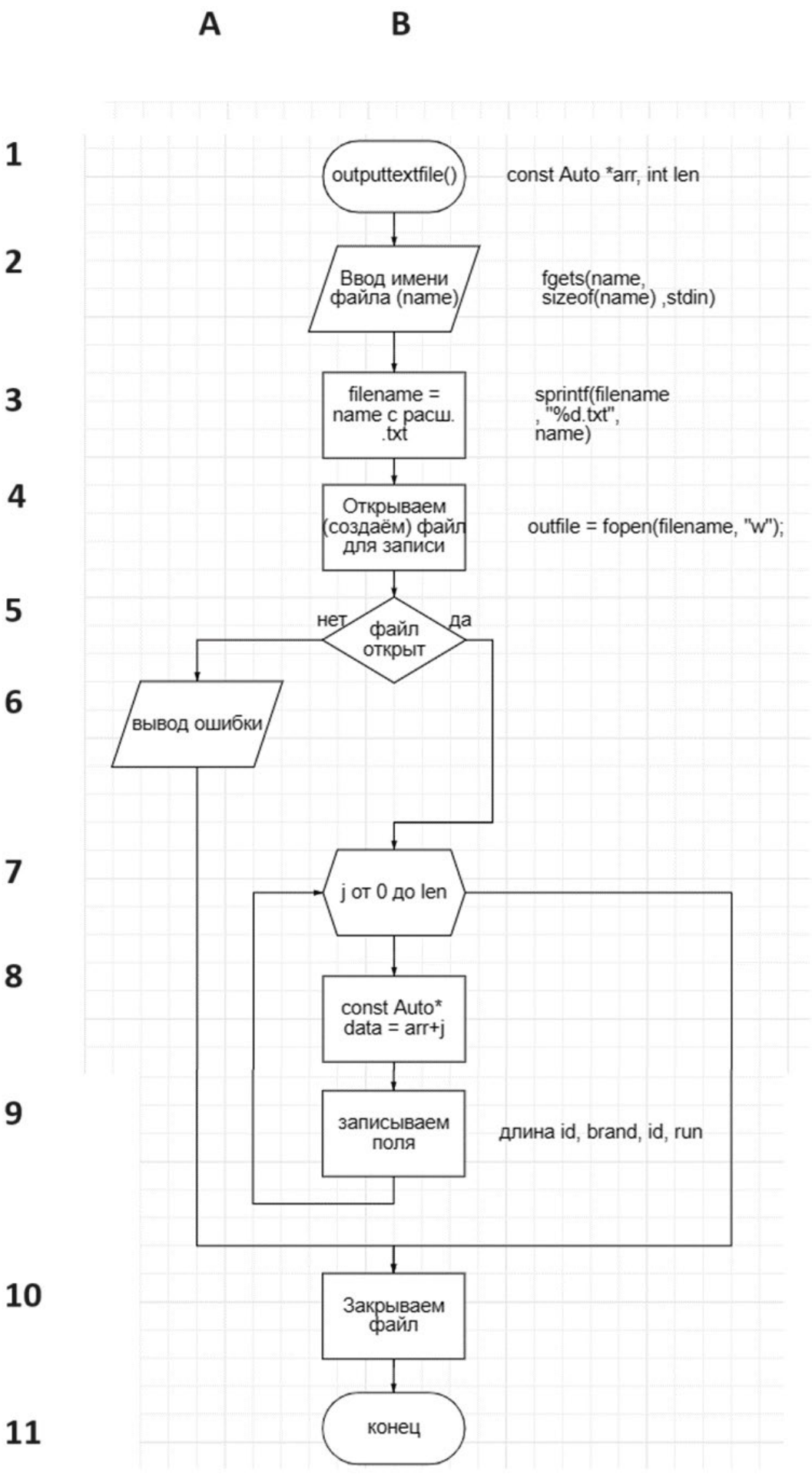
3. Inputbinfile()



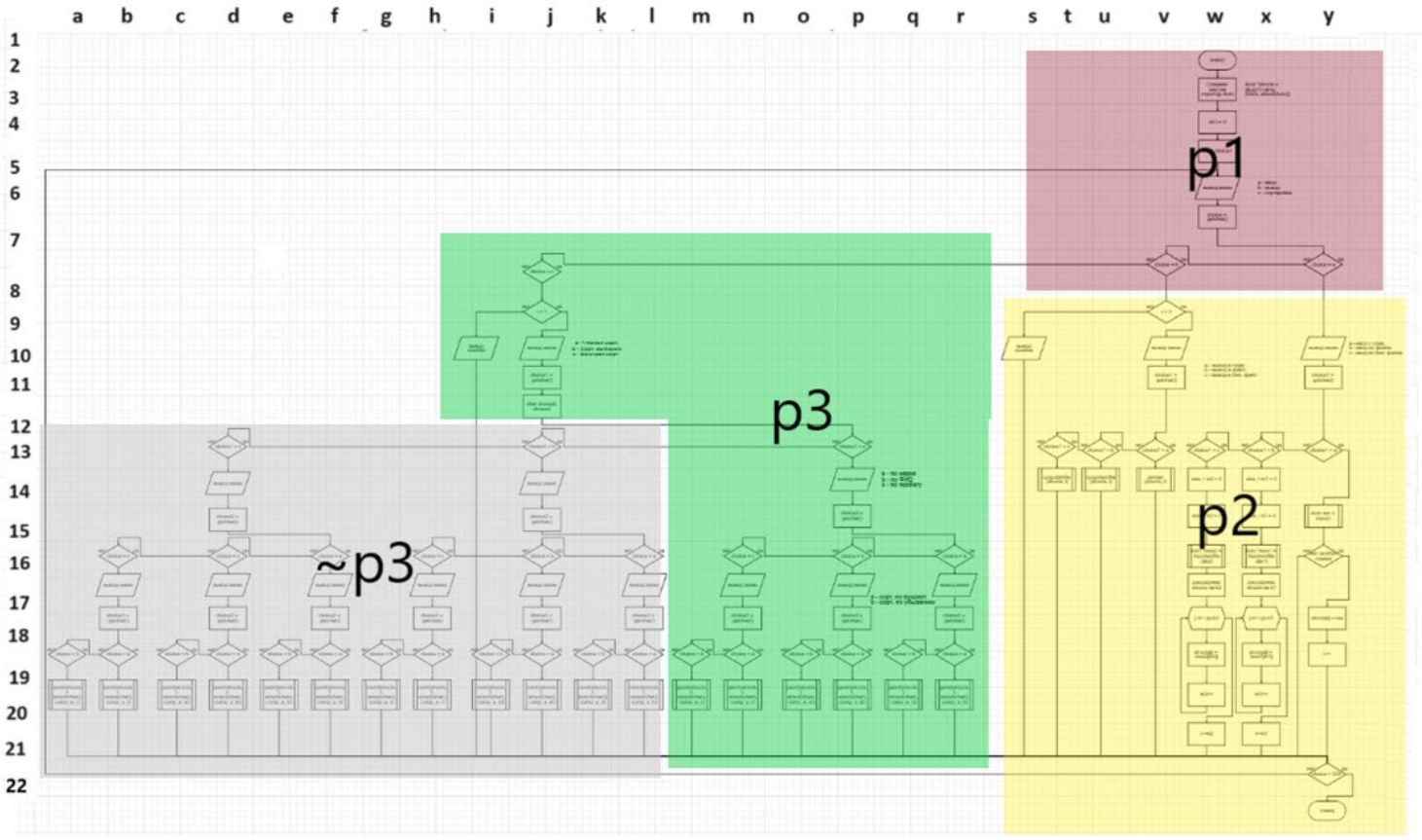
4. Outputbinfile()



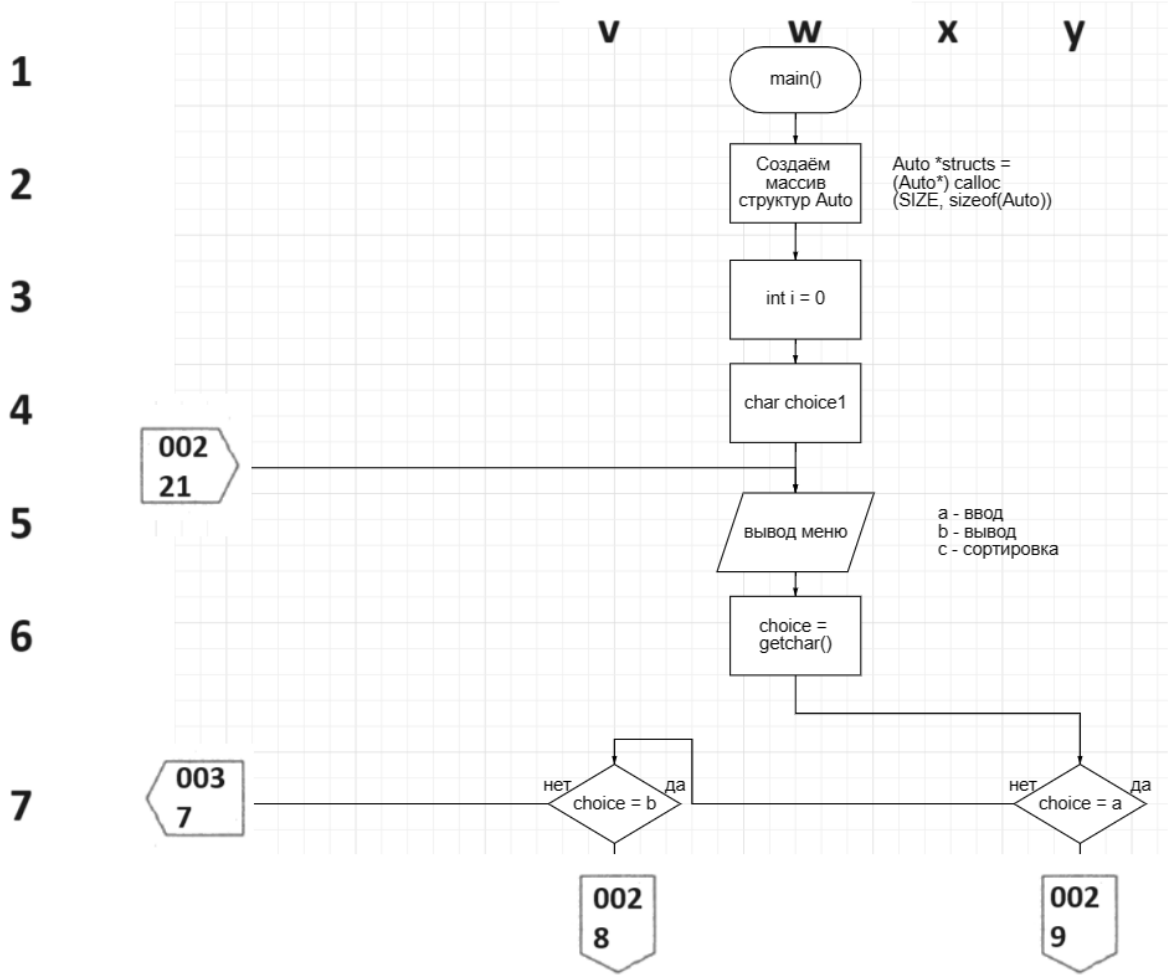
5. Outputtextfile()



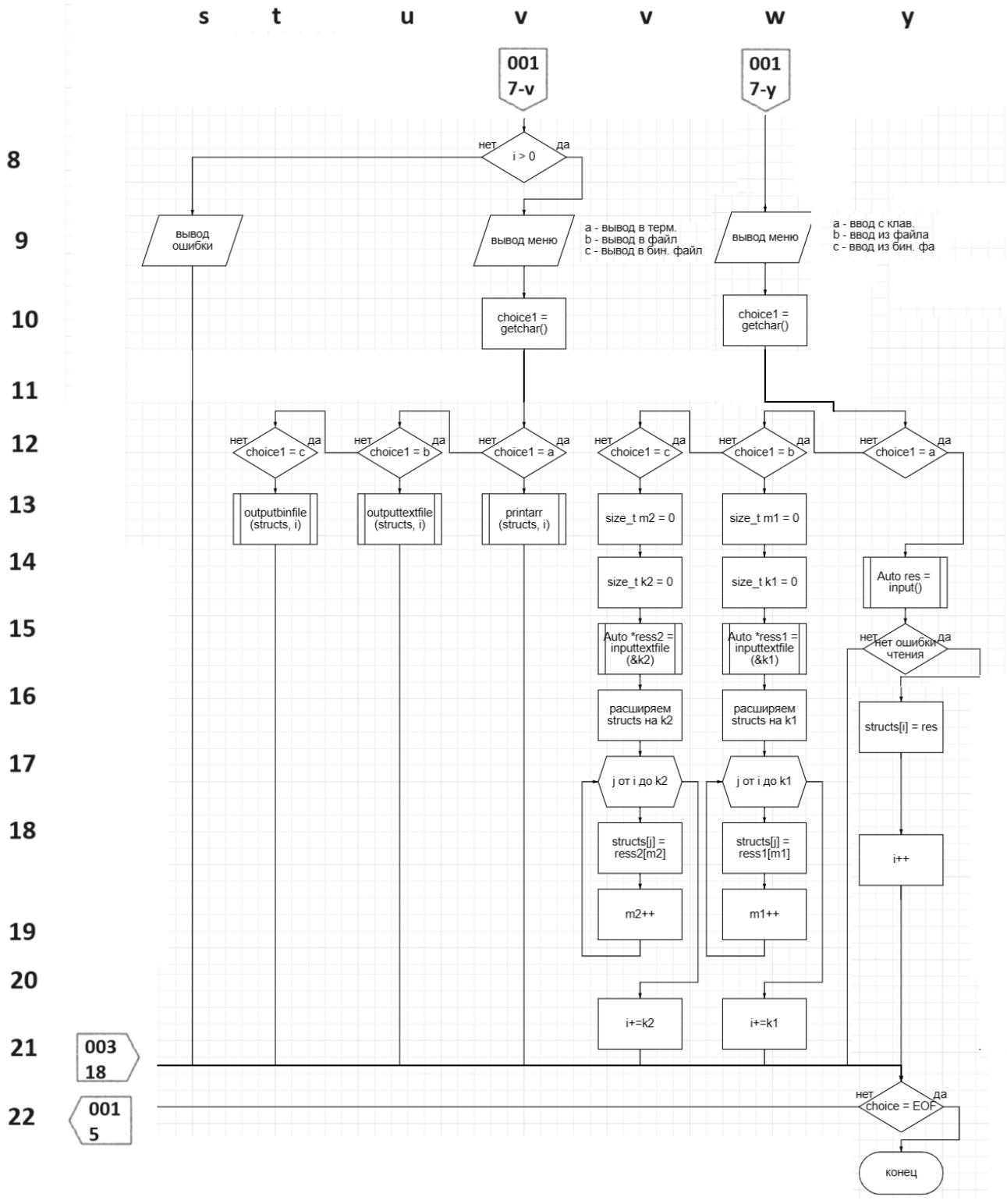
6. **main():**
    1. Общий вид (где p1 – верхняя часть блок-схемы, p2 – ввод и вывод, p3 – выбор и осуществление сортировки, ~p3 – аналогично p3 (в межстр. соединителях обоз. аналогично p4 = 004)



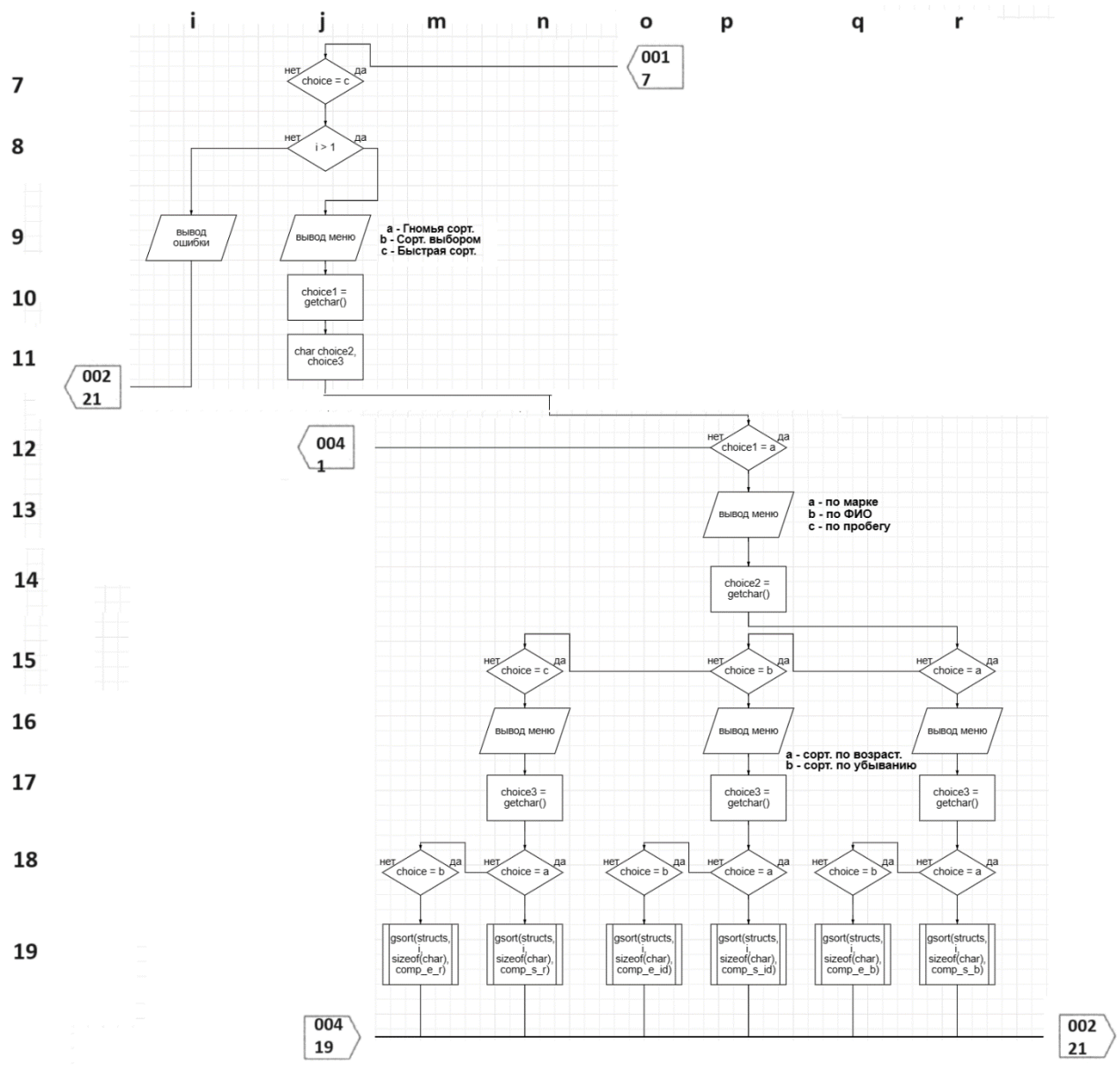
2. P1:



3. P2:

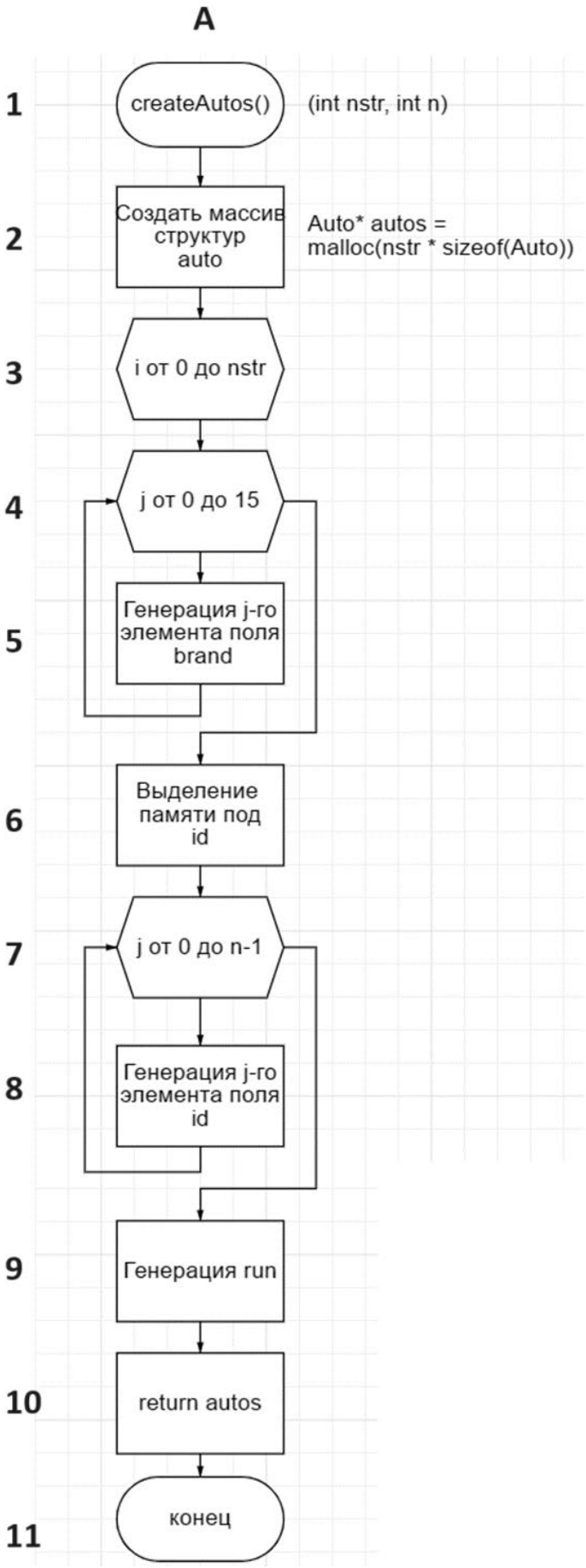

s
t
u
v
v
w
y
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
001
7-v
001
7-y
нет
i > 0
да
вывод ошибки
вывод меню
a - вывод в терм.
b - вывод в файл
c - вывод в бин. файл
вывод меню
a - ввод с клав.
b - ввод из файла
c - ввод из бин. фа
choice1 = getchar()
choice1 = getchar()
choice1 = c
choice1 = b
choice1 = a
choice1 = c
choice1 = b
choice1 = a
outputbinfile (structs, i)
outputtextfile (structs, i)
printarr (structs, i)
size_t m2 = 0
size_t m1 = 0
size_t k2 = 0
size_t k1 = 0
Auto res = input()
Auto *ress2 = inputtextfile (&k2)
Auto *ress1 = inputtextfile (&k1)
нет ошибки чтения
расширяем structs на k2
расширяем structs на k1
structs[i] = res
j от i до k2
j от i до k1
structs[j] = ress2[m2]
structs[j] = ress1[m1]
i++
m2++
m1++
i+=k2
i+=k1
003
18
001
5
choice = EOF
конец

4. P3:

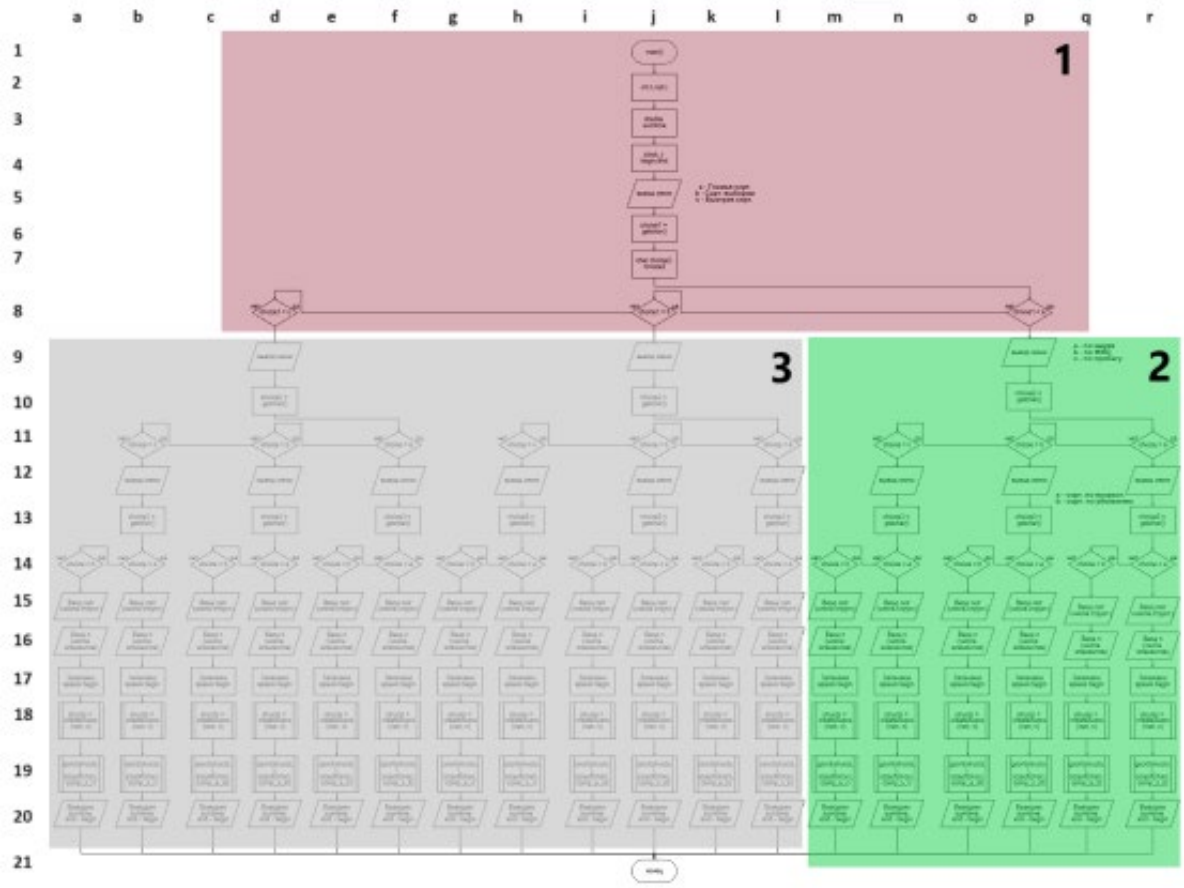

i
j
m
n
o
p
q
r
7
8
9
10
11
12
13
14
15
16
17
18
19
нет
да
choice = c
i > 1
вывод ошибки
вывод меню
a - Гномья сорт.
b - Сорт. выбором
c - Быстрая сорт.
choice1 = getchar()
char choice2, choice3
001 7
002 21
004 1
004 19
002 21
choice1 = a
a - по марке
b - по ФИО
c - по пробегу
choice2 = getchar()
choice = b
choice = a
a - сорт. по возраст.
b - сорт. по убыванию
choice3 = getchar()
gsort(structs, i, sizeof(char), comp_e_r)
gsort(structs, i, sizeof(char), comp_s_r)
gsort(structs, i, sizeof(char), comp_e_id)
gsort(structs, i, sizeof(char), comp_s_id)
gsort(structs, i, sizeof(char), comp_e_b)
gsort(structs, i, sizeof(char), comp_s_b)

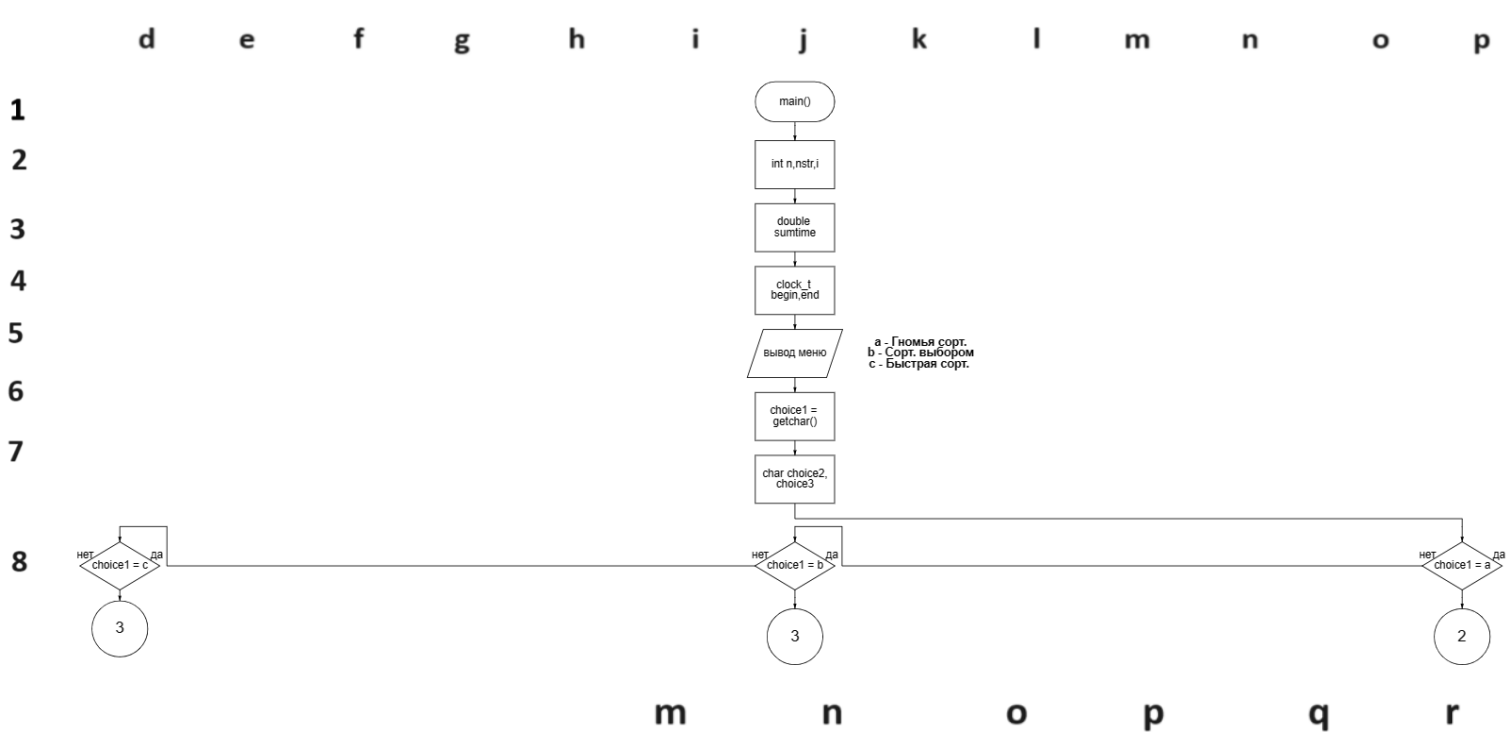
Программа 2:

1. **createAutos()**



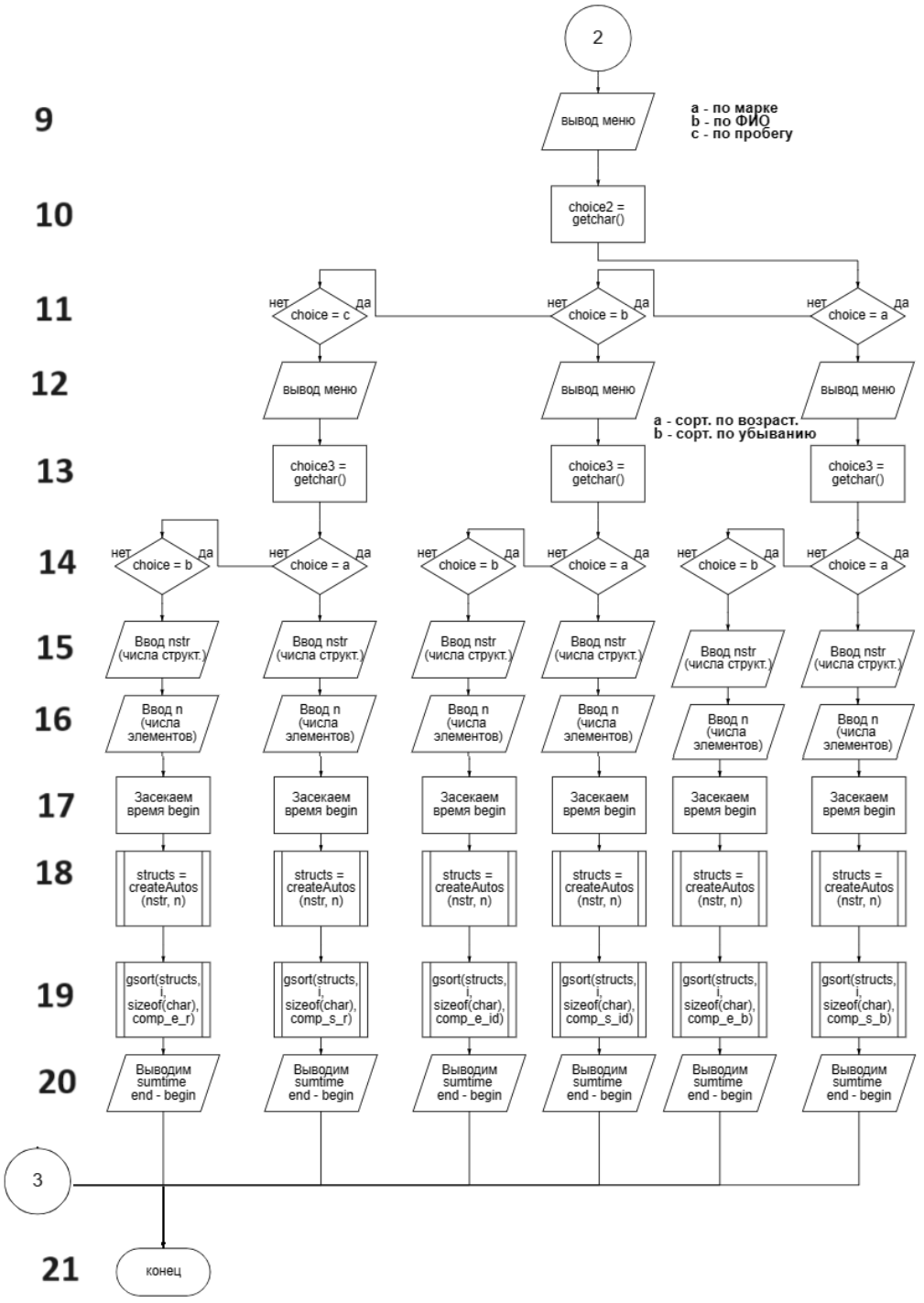
2. **main()**
    1. Общий вид (цифрами помечены соединители)





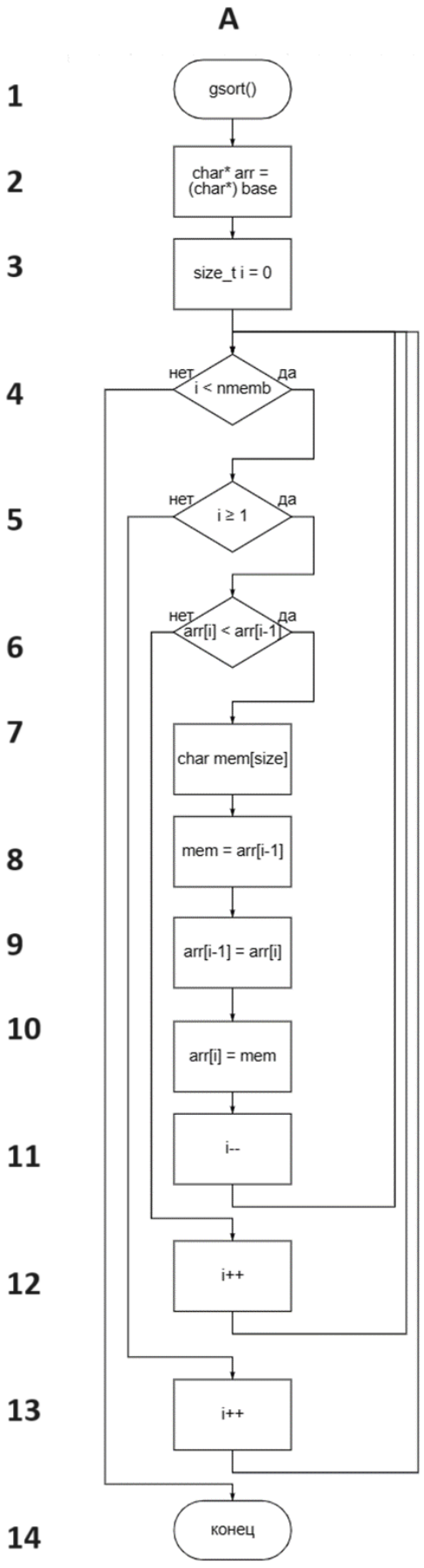
2. верх блок-схемы (1) и блок-схема одного из выборов алгоритма сортировки (2) – в данном случае гномьей

(3) аналогично (2)

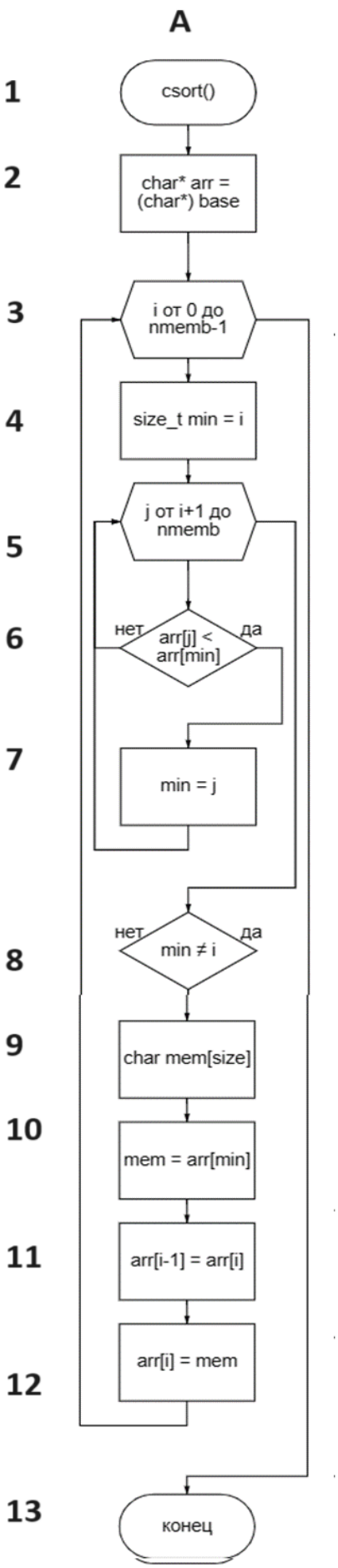


Блок-схемы функций сортировок:

1. **Гномья сортировка**



2. **Сортировка выбором**



III. Код

1. Printarr() и printstruct()

```
void printstruct(const Auto *p, int n){
        printf("---\n");
        printf("|%d|\n",n+1);
        printf("---\n");
        printf("Производитель: %s\nВладелец: %s\nПробег: %lf тыс. км.\n", p->brand, p->id,
p->run);
}

void printarr(const Auto *arr, int len){
        for (int i = 0; i < len; i++){
                printstruct(arr+i, i);
        }
}
```

2. Input()

```
Auto input(){
        Auto data;
        char brand[17], *id;
        double run;
        int flag = 0;
        printf("Введите марку производителя\n");
        fgets(brand, sizeof(brand), stdin);
        brand[strcspn(brand, "\n")] = 0;
        for (int k = 0; k<strlen(brand); k++){
                if (((brand[k] < 'a' || brand[k] > 'z'))&&(brand[k] < 'A' || brand[k] >
'Z') && brand[k] != '-' && brand[k] != ' '){
                        flag = 1;
                }
        }
        if (flag == 1){
                printf("Ошибка ввода\n");
                data.id = NULL;
                return data; //обработка ошибки
        }
        printf("Введите ФИО\n");
        id = oureadline(NULL);
        printf("Введите пробег\n");
        if (scanf("%lf",&run) <= 0){
                printf("Ошибка ввода\n");
                free(id);
                data.id = NULL;
                return data; //обработка ошибки
        }
        strncpy(data.brand, brand, sizeof(data.brand));
        data.id = id;
        data.run = run;
        return data;
}
```

3. Inputtextfile()

```
Auto* inputtextfile(size_t *k){
        Auto *currentstructs = (Auto*) calloc(SIZE, sizeof(Auto));
        FILE *infile = NULL;
        size_t id_len;
        char name[100], filename[104];
        int flag;
        printf("Введите имя файла\n");
        fgets(name, sizeof(name), stdin);
        name[strcspn(name, "\n")] = 0;
        sprintf(filename, "%s.txt", name);
        infile = fopen(filename, "r");
        if (infile == NULL) {
```

```
                fprintf(stderr,"Ошибка чтения файла\n");
                currentstructs[0].id = NULL;
                return currentstructs;
        }
        while ((flag = fscanf(infile, "%zu", &id_len)) != EOF) {
                if (flag == 1){
                        Auto data; int flag1 = 0;
                        data.id = calloc(id_len + 1, sizeof(char));
                        data.id[id_len] = '\0';
                        flag = fscanf(infile, " %[^\n] %[^\n] %lf", data.brand, data.id,
&data.run);
                        for (int j = 0; j<strlen(data.brand); j++){
                                if (((data.brand[j] < 'a' || data.brand[j] >
'z'))&&(data.brand[j] < 'A' || data.brand[j] > 'Z') && data.brand[j] != '-' &&>
flag1 = 1;
                                }
                        }
                        if (flag == 3 && flag1 != 1) {
                                currentstructs = realloc(currentstructs, sizeof(Auto) *
(SIZE + *k + 1));
                                currentstructs[*k] = data;
                                (*k)++;
                        }
                        else {
                                free(data.id);
                                fprintf(stderr, "Ошибка чтения %zu-й записи файла %s\n",
*k+1, filename);
                                break;
                        }
                }
                else{
                        fprintf(stderr, "Ошибка чтения %zu-й записи файла %s\n", *k+1,
filename);
                        break;
                }
        }
        fclose(infile);
        return currentstructs;
}
```

4. Inputbinfile()

```
Auto* inputbinfile(size_t *k){
        Auto *currentstructs = (Auto*) calloc(SIZE, sizeof(Auto));
        FILE *infile = NULL;
        size_t id_len;
        char name[100], filename[104];
        int flag2;
        printf("Введите имя файла\n");
        size_t br_len;
        fgets(name, sizeof(name), stdin);
        name[strcspn(name, "\n")] = 0;
        sprintf(filename, "%s.bin", name);
        infile = fopen(filename, "rb");
        if (infile == NULL) {
                printf("Ошибка чтения файла. Убедитесь, что он существует и имеет
необходимое расширение\n");
                currentstructs[0].id = NULL;
                return currentstructs;
        }
        while ((flag2 = fread(&id_len, sizeof(size_t), 1, infile)) != 0) {
                if (flag2 == 1){
                        Auto data; int flag1 = 0;
                        data.id = calloc(id_len + 1, sizeof(char));
                        data.id[id_len] = '\0';
```

```
                        if (fread(&br_len, sizeof(size_t), 1, infile)>=1 &&
fread(data.brand, sizeof(char), br_len, infile) >= 1 && fread(data.id, sizeof(c>
data.brand[br_len] = '\0';
                                for (int j = 0; j<strlen(data.brand); j++){
                                        if (((data.brand[j] < 'a' || data.brand[j] >
'z'))&&(data.brand[j] < 'A' || data.brand[j] > 'Z') && data.brand[j] !>
flag1 = 1;
                                        }
                                }
                                if (flag1 == 1){
                                        free(data.id);
                                        fprintf(stderr, "Ошибка чтения %zu-й записи
файла %s\n", *k+1, filename);
                                        break;
                                }
                                currentstructs = realloc(currentstructs, sizeof(Auto) *
(*k + SIZE + 1));
                                data.brand[br_len] = '\0';
                                currentstructs[*k] = data;
                                (*k)++;
                        }
                        else {
                                free(data.id);
                                fprintf(stderr, "Ошибка чтения %zu-й записи файла %s\n",
*k+1, filename);
                                break;
                        }
                }
                else{
                        fprintf(stderr, "Ошибка чтения %zu-й записи файла %s\n", *k+1,
filename);
                        break;
                }
        }
        fclose(infile);
        return currentstructs;
}
```

5. Outputtextfile()

```
void outputtextfile (const Auto *arr, int len){
        FILE *outfile = NULL;
        char name[100], filename[104];
        printf("Введите имя файла\n");
        fgets(name,sizeof(name),stdin);
        name[strcspn(name, "\n")] = 0;
        sprintf(filename,"%s.txt",name);
        outfile = fopen(filename, "w");
        if(outfile == NULL) {
                fprintf(stderr,"Ошибка открытия файла\n");
        }
        for (int j = 0; j<len; j++){
                const Auto* data = arr+j;
                fprintf(outfile, "%zu\n",strlen(data->id));
                fprintf(outfile, "%s\n",data->brand);
                fprintf(outfile, "%s\n",data->id);
                fprintf(outfile, "%lf\n",data->run);
                fprintf(outfile, "\n");
        }
        fclose(outfile);
}
```

6. Outputbinfile()

```
void outputbinfile (const Auto *arr, int len){
        FILE *outfile = NULL;
        char name[100], filename[104];
        printf("Введите имя файла\n");
```

```
        fgets(name,sizeof(name),stdin);
        name[strcspn(name, "\n")] = 0;
        sprintf(filename,"%s.bin",name);
        outfile = fopen(filename, "wb");
        if (outfile == NULL) {
                printf("Ошибка открытия файла\n");
        }
        for (int j = 0; j<len; j++){
                const Auto* data = arr+j;
                size_t id_len = strlen(data->id);
                size_t br_len = strlen(data->brand);
                fwrite(&id_len, sizeof(size_t), 1, outfile);
                fwrite(data->id, sizeof(char), id_len, outfile);
                fwrite(&br_len, sizeof(size_t),1,outfile);
                fwrite(data->brand, sizeof(char), br_len, outfile);
                fwrite(&data->run, sizeof(double), 1, outfile);
        }
        fclose(outfile);
}
```

7. Main()

```
void main(){
        Auto *structs = (Auto*) calloc(SIZE, sizeof(Auto));
        int i = 0;
        char choice;
        do{
                char choice1;
                printf("a) - ввод записи\nb) - вывод записей\nc) - сортировка структур\n");
                choice = getchar();
                scanf("%*[^\n]");
                scanf("%*c");
                switch(choice){
                        case 'a':
                                FILE *infile = NULL;
                                printf("a) - ввод с клавиатуры\nb) - ввод из файла\nc) - ввод из бин. файла\nd) - вовзрат в
главное меню\n");
                                choice1 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch(choice1){
                                        case 'a':
                                                Auto res = input();
                                                if (res.id!= NULL){
                                                        structs[i] = res; i++;
                                                }
                                                printf("Запись взята на хранение\n");
                                                scanf("%*[^\n]");
                                                scanf("%*c");
                                                structs = realloc(structs, sizeof(Auto) * (SIZE + i));
                                                break;
                                        case 'b':
                                                size_t m1 = 0;
                                                size_t k1 = 0;
                                                Auto *ress1 = inputtextfile(&k1);
                                                structs = realloc(structs, sizeof(Auto) * (i + k1 + SIZE));
                                                if (ress1[0].id != NULL){
                                                        for (size_t j = i; j<=k1;j++){
                                                                structs[j] = ress1[m1];
                                                                m1++;
                                                        }
                                                }
                                                free(ress1);
                                                printf("Из файла извлечено %zu записей\n", k1);
                                                i+=k1;
                                                break;
                                         case 'c':
                                                size_t m2 = 0;
                                                size_t k2 = 0;
                                                Auto *ress2 = inputbinfile(&k2);
                                                structs = realloc(structs, sizeof(Auto) * (i + k2 + SIZE));
                                                if (ress2[0].id != NULL){
                                                        for (int j = i; j<=k2; j++){
                                                                structs[j] = ress2[m2];
                                                                m2++;
                                                        }
                                                 }
                                                 free(ress2);
                                                 printf("Из бинарного файла извлечено %zu записей\n", k2);
                                                 i+=k2;
                                                 break;
                                        case 'd':
                                                printf("Осуществлён выход в главное меню\n");
                                                break;
                                        default: printf("Ошибка ввода\n"); break;
                                }
                                break;
```

```
                        case 'b':
                                if (i < 1){
                                        printf("Нет ни одной записи для вывода\n");
                                        break;
                                }
                                FILE *outfile = NULL;
                                printf("a) - вывод в терминал\nb) - вывод в файл\nc) - вывод в бин. файл\nd) - вовзрат в
главное меню\n");
                                choice1 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch(choice1){
                                        case 'a':
                                                printf("Следующие записи находятся на хранении:\n");
                                                printarr(structs, i);
                                                break;
                                        case 'b':
                                                outputtextfile(structs, i);
                                                break;
                                        case 'c':
                                                outputbinfile(structs, i);
                                                break;
                                        case 'd':
                                                printf("Осуществлён выход в главное меню\n");
                                                break;
                                        default: fprintf(stderr,"Ошибка ввода\n"); break;
                                }
                                break;
                        case 'c':
                                if (i <= 1){
                                        printf("Недостаточно записей для сортировки\n");
                                        break;
                                }
                                char choice2,choice3;
                                printf("a) - гномья сортировка\nb) - сортировка выбором\nc) - быстрая сортировка\n");
                                choice1 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice1){
                                        case 'a':
                                                printf("a) - сорт. по марке\nb) - сорт. по ФИО\nc) - сорт. по пробегу\n");
                                                choice2 = getchar();
                                                scanf("%*[^\n]");
                                                scanf("%*c");
                                                switch (choice2){
                                                        case 'a':
                                                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                                                choice3 = getchar();
                                                                scanf("%*[^\n]");
                                                                scanf("%*c");
                                                                switch (choice3){
                                                                        case 'a':
                                                                                gsort(structs,i,sizeof(Auto),comp_s_b);
                                                                                break;
                                                                        case 'b':
                                                                                gsort(structs,i,sizeof(Auto),comp_e_b);
                                                                                break;
                                                                        default: printf("Ошибка ввода\n"); break;
                                                                }
                                                                break;
                                                        case 'b':
                                                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                                                choice3 = getchar();
                                                                scanf("%*[^\n]");
                                                                scanf("%*c");
                                                                switch (choice3){
                                                                        case 'a':
                                                                                gsort(structs,i,sizeof(Auto),comp_s_id);
                                                                                break;
                                                                        case 'b':
                                                                                gsort(structs,i,sizeof(Auto),comp_e_id);
                                                                                break;
                                                                        default: printf("Ошибка ввода\n"); break;
                                                                }
                                                                break;
                                                        case 'c':
                                                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                                                choice3 = getchar();
                                                                scanf("%*[^\n]");
                                                                scanf("%*c");
                                                                switch (choice3){
                                                                        case 'a':
                                                                                gsort(structs,i,sizeof(Auto),comp_s_r);
                                                                                break;
                                                                        case 'b':
                                                                                gsort(structs,i,sizeof(Auto),comp_e_r);
                                                                                break;
                                                                        default: printf("Ошибка ввода\n"); break;
                                                                }
                                                                break;
                                                        default: printf("Ошибка ввода\n"); break;
                                                }
                                                break;
                                        case 'b':
                                                printf("a) - сорт. по марке\nb) - сорт. по ФИО\nc) - сорт. по пробегу\n");
```

```
                choice2 = getchar();
                scanf("%*[^\n]");
                scanf("%*c");
                switch (choice2){
                        case 'a':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                csort(structs,i,sizeof(Auto),comp_s_b);
                                                break;
                                        case 'b':
                                                csort(structs,i,sizeof(Auto),comp_e_b);
                                                break;
                                        default: printf("Ошибка ввода\n"); break;
                                }
                                break;
                        case 'b':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                csort(structs,i,sizeof(Auto),comp_s_id);
                                                break;
                                        case 'b':
                                                csort(structs,i,sizeof(Auto),comp_e_id);
                                                break;
                                        default: printf("Ошибка ввода\n"); break;
                                }
                                break;
                        case 'c':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                csort(structs,i,sizeof(Auto),comp_s_r);
                                                break;
                                        case 'b':
                                                csort(structs,i,sizeof(Auto),comp_e_r);
                                                break;
                                        default: printf("Ошибка ввода\n"); break;
                                }
                                break;
                        default: printf("Ошибка ввода\n"); break;
                }
                break;
        case 'c':
                printf("a) - сорт. по марке\nb) - сорт. по ФИО\nc) - сорт. по пробегу\n");
                choice2 = getchar();
                scanf("%*[^\n]");
                scanf("%*c");
                switch (choice2){
                        case 'a':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                qsort(structs,i,sizeof(Auto),comp_s_b);
                                                break;
                                        case 'b':
                                                qsort(structs,i,sizeof(Auto),comp_e_b);
                                                break;
                                        default: printf("Ошибка ввода\n"); break;
                                }
                                break;
                        case 'b':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                qsort(structs,i,sizeof(Auto),comp_s_id);
                                                break;
                                        case 'b':
                                                qsort(structs,i,sizeof(Auto),comp_e_id);
                                                break;
                                        default: printf("Ошибка ввода\n"); break;
                                }
                                break;
                        case 'c':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
```

```
                                                                qsort(structs,i,sizeof(Auto),comp_s_r);
                                                                break;
                                                        case 'b':
                                                                qsort(structs,i,sizeof(Auto),comp_e_r);
                                                                break;
                                                        default: printf("Ошибка ввода\n"); break;
                                                }
                                                break;
                                        default: printf("Ошибка ввода\n"); break;
                                }
                                break;
                        case 'd':
                                printf("Осуществлён выход в главное меню\n");
                                break;
                        default:
                                fprintf(stderr,"Ошибка ввода\n"); break;
                        }
                        break;
                case EOF: printf("Завершение работы\n"); break;
                default: printf("Ошибка выбора\n"); break;
                }
        } while(choice != EOF);
        for (int j = 0; j<=i; j++){
                free(structs[j].id);
        }
        free(structs);
}
```

8. createAutos()

```
Auto* createAutos(int nstr, int n) {
    srand(time(NULL));
    Auto* autos = malloc(nstr * sizeof(Auto));
    for (int i = 0; i < nstr; i++) {
        for (int j = 0; j < 16; j++) {
            autos[i].brand[j] = 'A' + rand() % 26;
        }
        autos[i].brand[16] = '\0';
        autos[i].id = malloc((n + 1) * sizeof(char));
        for (int j = 0; j < n; j++) {
            autos[i].id[j] = 'A' + rand() % 26;
        }
        autos[i].id[n] = '\0';
        autos[i].run = rand() % 100000 + 1; //от 1 до 100000
    }
    return autos;
}
```

9. main() (2 программа)

```
int main() {
        clock_t begin, end;
        double sumtime;
        int n, nstr,i;
        Auto* structs;
        char choice2,choice3;
        printf("a) - гномья сортировка\nb) - сортировка выбором\nc) - быстрая сортировка\n");
        char choice1 = getchar();
        scanf("%*[^\n]");
        scanf("%*c");
        switch (choice1){
                case 'a':
                        printf("a) - сорт. по марке\nb) - сорт. по ФИО\nc) - сорт. по пробегу\n");
                        choice2 = getchar();
                        scanf("%*[^\n]");
                        scanf("%*c");
                        switch (choice2){
                                case 'a':
                                        printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                        choice3 = getchar();
                                        scanf("%*[^\n]");
                                        scanf("%*c");
                                        switch (choice3){
                                                case 'a':
                                                        printf("Введите число генерируемых структур\n");
                                                        scanf("%d", &nstr);
                                                        printf("Введите число элементов структур\n");
                                                        scanf("%d", &n);
                                                        structs = createAutos(nstr, n);
                                                        begin = clock();
                                                        gsort(structs,nstr,sizeof(Auto),comp_s_b);
```

```
                                        end = clock();
                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                        //printarr(structs, nstr);
                                        for (int i = 0; i < nstr; i++) {
                                            free(structs[i].id);
                                        }
                                        free(structs);
                                        break;
                                case 'b':
                                        printf("Введите число генерируемых структур\n");
                                        scanf("%d", &nstr);
                                        printf("Введите число элементов структур\n");
                                        scanf("%d", &n);
                                        structs = createAutos(nstr, n);
                                        begin = clock();
                                        gsort(structs,nstr,sizeof(Auto),comp_e_b);
                                        end = clock();
                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                        for (int i = 0; i < nstr; i++) {
                                                free(structs[i].id);
                                        }
                                        free(structs);
                                        break;
                                default:
                                        printf("Ошибка ввода\n");
                                        break;
                        }
                        break;
                case 'b':
                        printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                        choice3 = getchar();
                        scanf("%*[^\n]");
                        scanf("%*c");
                        switch (choice3){
                                case 'a':
                                        printf("Введите число генерируемых структур\n");
                                        scanf("%d", &nstr);
                                        printf("Введите число элементов структур\n");
                                        scanf("%d", &n);
                                        structs = createAutos(nstr, n);
                                        begin = clock();
                                        gsort(structs,nstr,sizeof(Auto),comp_s_id);
                                        end = clock();
                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                        for (int i = 0; i < nstr; i++) {
                                                free(structs[i].id);
                                        }
                                        free(structs);
                                        break;
                                case 'b':
                                        printf("Введите число генерируемых структур\n");
                                        scanf("%d", &nstr);
                                        printf("Введите число элементов структур\n");
                                        scanf("%d", &n);
                                        structs = createAutos(nstr, n);
                                        begin = clock();
                                        gsort(structs,nstr,sizeof(Auto),comp_e_id);
                                        end = clock();
                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                        for (int i = 0; i < nstr; i++) {
                                                free(structs[i].id);
                                        }
                                        free(structs);
                                        break;
                                default:
                                        printf("Ошибка ввода\n");
                                        break;
                        }
                        break;
                case 'c':
                        printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                        choice3 = getchar();
                        scanf("%*[^\n]");
                        scanf("%*c");
                        switch (choice3){
```

```
                                        case 'a':
                                                printf("Введите число генерируемых структур\n");
                                                scanf("%d", &nstr);
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                gsort(structs,nstr,sizeof(Auto),comp_s_r);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        case 'b':
                                                printf("Введите число генерируемых структур\n");
                                                scanf("%d", &nstr);
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                gsort(structs,nstr,sizeof(Auto),comp_e_r);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        default:
                                                printf("Ошибка ввода\n");
                                                break;
                                        }
                                        break;
                                default:
                                        printf("Ошибка ввода\n");
                                        break;
                        }
                        break;
                case 'b':
                        printf("a) - сорт. по марке\nb) - сорт. по ФИО\nc) - сорт. по пробегу\n");
                        choice2 = getchar();
                        scanf("%*[^\n]");
                        scanf("%*c");
                        switch (choice2){
                                case 'a':
                                        printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                        choice3 = getchar();
                                        scanf("%*[^\n]");
                                        scanf("%*c");
                                        switch (choice3){
                                                case 'a':
                                                        printf("Введите число генерируемых структур\n");
                                                        scanf("%d", &nstr);
                                                        printf("Введите число элементов структур\n");
                                                        scanf("%d", &n);
                                                        structs = createAutos(nstr, n);
                                                        begin = clock();
                                                        csort(structs,nstr,sizeof(Auto),comp_s_b);
                                                        end = clock();
                                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                        for (int i = 0; i < nstr; i++) {
                                                                free(structs[i].id);
                                                        }
                                                        free(structs);
                                                        break;
                                                case 'b':
                                                        printf("Введите число генерируемых структур\n");
                                                        scanf("%d", &nstr);
                                                        printf("Введите число элементов структур\n");
                                                        scanf("%d", &n);
                                                        structs = createAutos(nstr, n);
                                                        begin = clock();
                                                        csort(structs,nstr,sizeof(Auto),comp_e_b);
                                                        end = clock();
                                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
```

```
                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                        for (int i = 0; i < nstr; i++) {
                                                free(structs[i].id);
                                        }
                                        free(structs);
                                        break;
                                default:
                                        printf("Ошибка ввода\n");
                                        break;
                                }
                                break;
                        case 'b':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                printf("Введите число генерируемых структур\n");
                                                scanf("%d", &nstr);
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                csort(structs,nstr,sizeof(Auto),comp_s_id);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        default:
                                                printf("Ошибка ввода\n");
                                                break;
                                        }
                                        break;
                        case 'c':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                printf("Введите число генерируемых структур\n");
                                                scanf("%d", &nstr);
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                csort(structs,nstr,sizeof(Auto),comp_s_r);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        case 'b':
                                                printf("Введите число генерируемых структур\n");
                                                scanf("%d", &nstr);
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                csort(structs,nstr,sizeof(Auto),comp_s_r);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        default:
                                                printf("Ошибка ввода\n");
                                                break;
```

```
                                        }
                                        break;
                                default:
                                        printf("Ошибка ввода\n");
                                        break;
                        }
                        break;
                case 'c':
                        printf("a) - сорт. по марке\nb) - сорт. по ФИО\nc) - сорт. по пробегу\n");
                        choice2 = getchar();
                        scanf("%*[^\n]");
                        scanf("%*c");
                        switch (choice2){
                                case 'a':
                                        printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                        choice3 = getchar();
                                        scanf("%*[^\n]");
                                        scanf("%*c");
                                        switch (choice3){
                                                case 'a':
                                                        printf("Введите число генерируемых структур\n");
                                                        scanf("%d", &nstr);
                                                        printf("Введите число элементов структур\n");
                                                        scanf("%d", &n);
                                                        structs = createAutos(nstr, n);
                                                        begin = clock();
                                                        qsort(structs,nstr,sizeof(Auto),comp_s_b);
                                                        end = clock();
                                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                        for (int i = 0; i < nstr; i++) {
                                                                free(structs[i].id);
                                                        }
                                                        free(structs);
                                                        break;
                                                case 'b':
                                                        printf("Введите число генерируемых структур\n");
                                                        scanf("%d", &nstr);
                                                        printf("Введите число элементов структур\n");
                                                        scanf("%d", &n);
                                                        structs = createAutos(nstr, n);
                                                        begin = clock();
                                                        qsort(structs,nstr,sizeof(Auto),comp_e_b);
                                                        end = clock();
                                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                        for (int i = 0; i < nstr; i++) {
                                                                free(structs[i].id);
                                                        }
                                                        free(structs);
                                                        break;
                                                default:
                                                        printf("Ошибка ввода\n");
                                                        break;
                                        }
                                        case 'b':
                                        printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                        choice3 = getchar();
                                        scanf("%*[^\n]");
                                        scanf("%*c");
                                        switch (choice3){
                                                case 'a':
                                                        printf("Введите число генерируемых структур\n");
                                                        scanf("%d", &nstr);
                                                        printf("Введите число элементов структур\n");
                                                        scanf("%d", &n);
                                                        structs = createAutos(nstr, n);
                                                        begin = clock();
                                                        qsort(structs,nstr,sizeof(Auto),comp_s_id);
                                                        end = clock();
                                                        sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                        printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                        for (int i = 0; i < nstr; i++) {
                                                                free(structs[i].id);
                                                        }
                                                        free(structs);
                                                        break;
                                                case 'b':
                                                        printf("Введите число генерируемых структур\n");
                                                        scanf("%d", &nstr);
```

```
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                csort(structs,nstr,sizeof(Auto),comp_e_id);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        default:
                                                printf("Ошибка ввода\n");
                                                break;
                                }
                                break;
                        case 'c':
                                printf("a) - сорт. с начала\nb) - сорт. с конца\n");
                                choice3 = getchar();
                                scanf("%*[^\n]");
                                scanf("%*c");
                                switch (choice3){
                                        case 'a':
                                                printf("Введите число генерируемых структур\n");
                                                scanf("%d", &nstr);
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                qsort(structs,nstr,sizeof(Auto),comp_s_r);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        case 'b':
                                                printf("Введите число генерируемых структур\n");
                                                scanf("%d", &nstr);
                                                printf("Введите число элементов структур\n");
                                                scanf("%d", &n);
                                                structs = createAutos(nstr, n);
                                                begin = clock();
                                                qsort(structs,nstr,sizeof(Auto),comp_e_id);
                                                end = clock();
                                                sumtime = (double)(end - begin)/CLOCKS_PER_SEC;
                                                printf("Итоговое время выполнения: %0.11f\n",
sumtime);
                                                for (int i = 0; i < nstr; i++) {
                                                        free(structs[i].id);
                                                }
                                                free(structs);
                                                break;
                                        default:
                                                printf("Ошибка ввода\n");
                                                break;
                                }
                        }
                        break;
                default:
                        printf("Ошибка ввода\n");
                        break;
        }
}
```

10. Gsort()

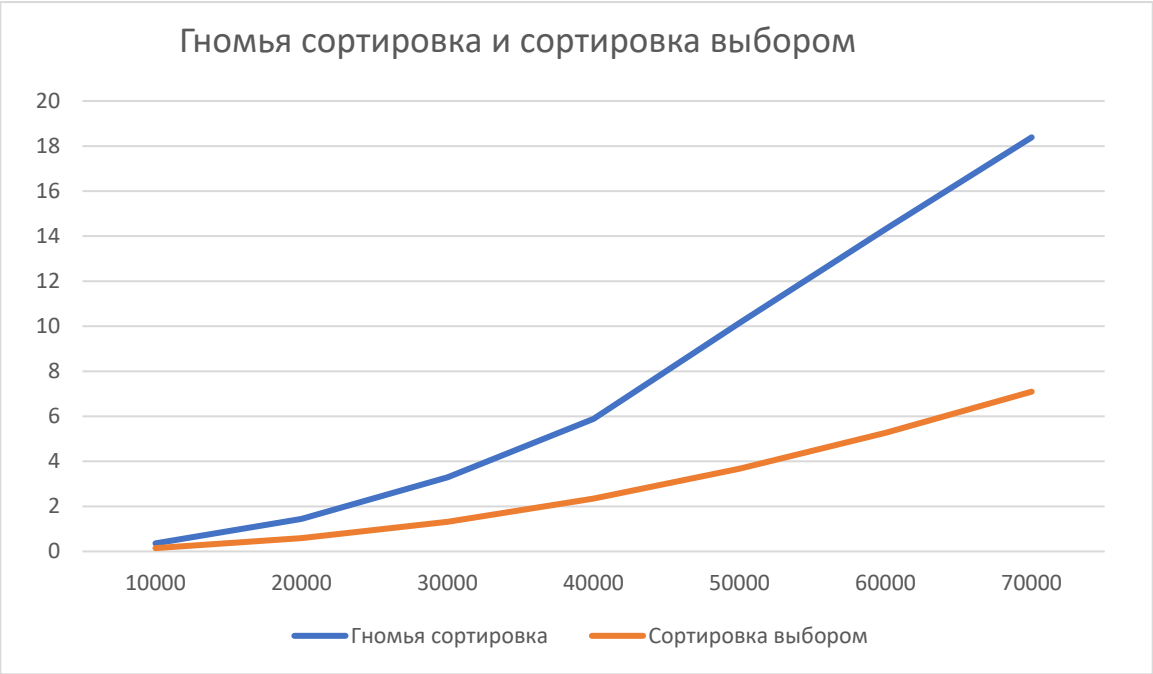
```
void gsort(void *base, size_t nmemb, size_t size, int (*compar)(const void *, const void
*)){
        char* arr = (char*) base;
        size_t i = 0;
        while (i < nmemb){
                if (i >= 1){
                        if (compar(&arr[i*size],&arr[(i-1)*size])<0){
                                char mem[size];
```

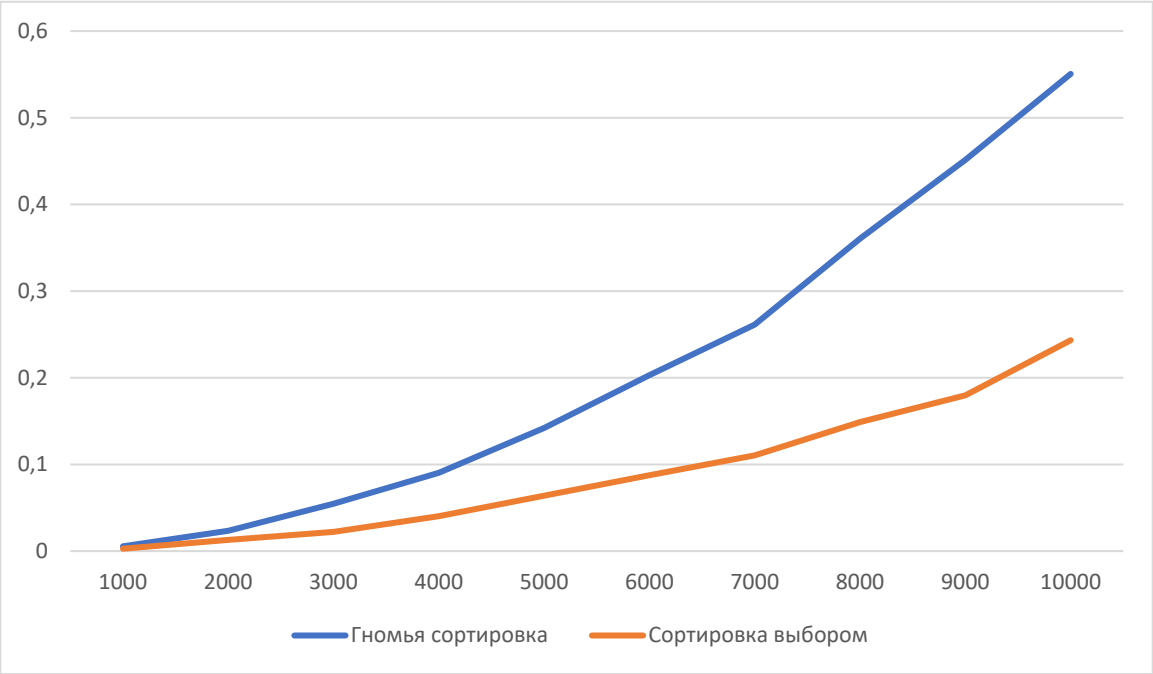
```
                                memcpy(mem, &arr[(i-1)*size], size);
                                memcpy(&arr[(i-1)*size], &arr[i*size], size);
                                memcpy(&arr[i*size], mem, size);
                                i--;
                        }
                        else i++;
                }
                else i++;
        }
}
```

11. Csort()

```
void csort(void *base, size_t nmemb, size_t size, int (*compar)(const void *, const void
*)){
        char* arr = (char*) base;
        for (int i = 0; i < nmemb - 1; i++){
                size_t min = i;
                for (int j = i+1;j < nmemb; j++){
                        if (compar(&arr[j*size], &arr[min*size])<0){
                                min = j;
                        }
                }
                if (min != i){
                        char mem[size];
                        memcpy(mem, &arr[min*size], size);
                        memcpy(&arr[min*size], &arr[i*size], size);
                        memcpy(&arr[i*size], mem, size);
                }
        }
}
```

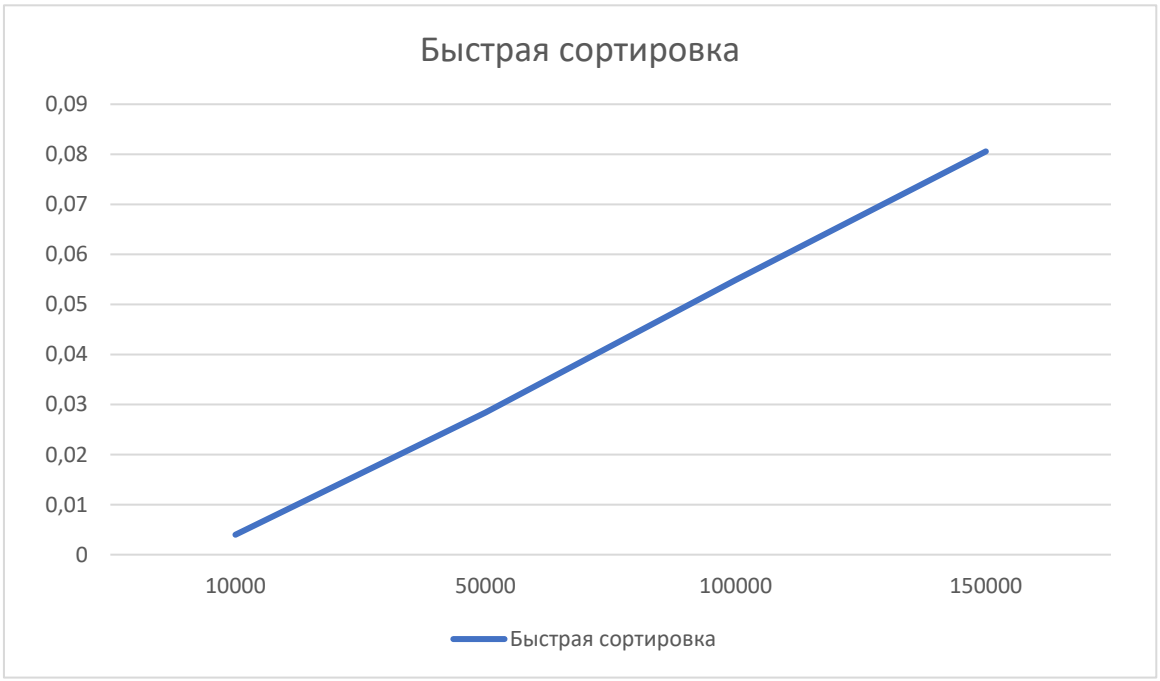
IV. Тесты





1. n = 100 (число элементов поля id), nstr перечислены на оси OX, OY – время выполнения сортировки в секундах.
   Полученные результаты говорят о том, что гномья сортировка и сортировка выбором имеют эффективность O(n^2), что соответствует теоретическим данным. Стоит отметить, что хотя алгоритмы гномьей сортировки и сортировки выбором сопоставимы, но сортировка выбором всё же существенно быстрее.

2. n= 1000(число элементов поля id), nstr перечислены на оси OX, OY – время выполнения сортировки в секундах.
Полученные результаты говорят о том, что эффективность быстрой сортировки O(n) или O(nlogn). Если рассмотреть выборочные значения теста, будет получен следующий график:



Что более явно показывает эффективность быстрой сортировки O(n) или O(nlogn). Теории больше соответствует O(nlogn)

V. Вывод

Работая над лабораторной работой №5, автор освоил работу со структурами: их инициализацией, применением для создания «записей» об одноформатных данных. Были освоены чтение и запись текстовых и бинарных файлов в C, алгоритмы сортировок, функции-компараторы.

В ходе анализа полученных при тестированиях во второй программе был сделан вывод о том, что, как и следовало ожидать, гномья сортировка и сортировка выбором имеют эффективность O(n^2), при этом сортировка выбором эффективней. Быстрая сортировка имеет эффективность O(nlogn), что также соответствует теории.

Помимо того, немаловажным является разработка алгоритма, т.к. в силу всё возрастающих размеров программ, требуется корректным образом и наглядно представить алгоритм, описывающий структуру программы. Особенно важно здесь деление программы на части – отдельные подсистемы большой системы.

Выполненная работа неидеальна. Слабостью программы можно назвать способ выбора отдельных пунктов меню. Эту часть программы, вероятно, можно сократить, отказавшись от switch-case. По крайней мере, эта часть хоть и работает (и работает эффективно), но является довольно громоздкой.