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

Факультет безопасности информационных технологий

Дисциплина:

«Алгоритмы и структуры данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Блинная сортировка»

Выполнил:
Пивоваров Константин Павлович

_____
(подпись)

Проверил:
Ерофеев Сергей Анатольевич

_____
(отметка о выполнении)

_____
(подпись)

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

1.1 Цель работы

Разработать программу блинной сортировки чисел из файла, записывая их в статический или динамический массив по выбору пользователя.

1.2 Алгоритм блинной сортировки

Для начало введём необходимые для понимания алгоритма понятия.

Неотсортированный подмассив – часть изначального массива, которая не отсортирована, однако количество его элементов по ходу алгоритма постоянно уменьшается, т. к. максимальный элемент такого подмассива перемещается в его конец (если не был там изначально) посредством переворота подмассива.

Переворот подмассива – это изменение порядка элементов подмассива, при котором последний элемент становится первым, а первый — последним.

Неотсортированный подмассив – часть изначального массива, которая отсортирована и постоянно увеличивается, т. к. по ходу алгоритма в начало такого подмассива добавляется максимальный элемент неотсортированного подмассива до тех пор, пока неотсортированный подмассив не опустеет.

Теперь мы готовы полностью описать алгоритм блинной сортировки:

- **Поиск максимального элемента**: находим индекс максимального элемента в неотсортированном подмассиве (изначально им выступает полученный и необработанный массива из файла) простым перебором.
- **Переворот до начала**: если максимальный элемент не находится на первом месте, переворачиваем неотсортированный подмассив от начала до индекса максимального элемента, перемещая максимальный элемент на первую позицию подмассива.
- **Переворот до конца**: затем переворачиваем изначальный массив от начала до конца, перемещая максимальный элемент в конец массива, формируя тем самым отсортированный подмассив.
- **Уменьшение размера подмассива**: уменьшаем размер неотсортированного подмассива, который мы рассматриваем (игнорируем последний элемент, который уже на правильном месте), и повторяем процесс, пока весь исходный массив не будет отсортирован.

1.3 Пример

Рассмотрим массив: [**5, 3, 8, 6, 2]**.

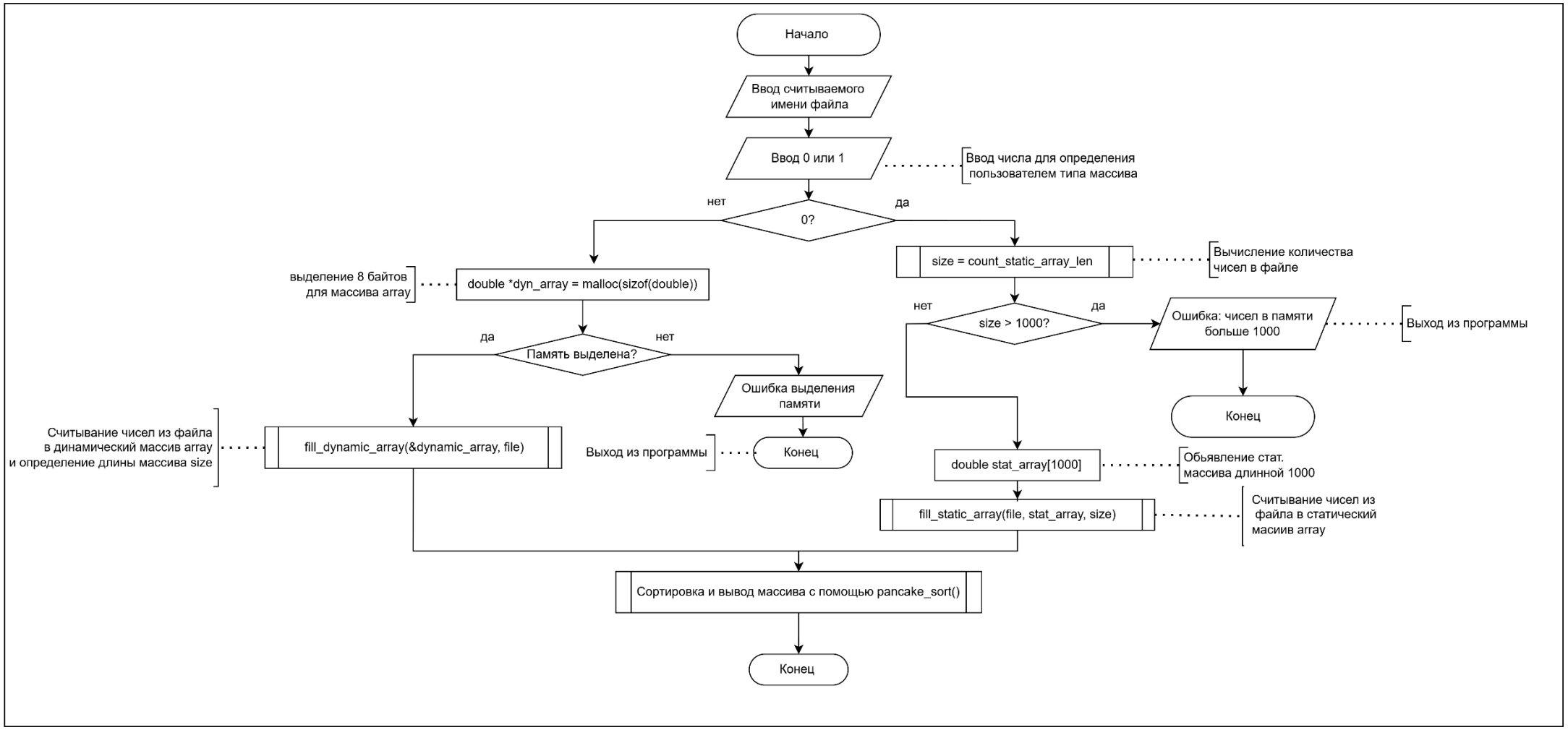
1. **Изначальный массив: [5, 3, 8, 6, 2], где [5, 3, 8, 6, 2] – неотсортированный подмассив, а [] – отсортированный подмассив**
    - Максимальный элемент: **8** (индекс 2).
    - Переворачиваем массив от 0 до 2: **[8, 3, 5, 6, 2]**.
    - Переворачиваем от 0 до 4: **[2, 6, 5, 3, 8]**.
2. **Текущий массив: [2, 6, 5, 3, 8], где [2, 6, 5, 3] – неотсортированный подмассив, а [8] – отсортированный подмассив**
    - Максимальный элемент: **6** (индекс 1).
    - Переворачиваем от 0 до 1: **[6, 2, 5, 3, 8]**.
    - Переворачиваем от 0 до 4: **[3, 5, 2, 6, 8]**.
3. **Текущий массив: [3, 5, 2, 6, 8], где [2, 5, 3] – неотсортированный подмассив, а [6, 8] – отсортированный подмассив**
    - Максимальный элемент: **5** (индекс 1).
    - Переворачиваем от 0 до 1: **[5, 3, 2, 6, 8]**.
    - Переворачиваем от 0 до 3: **[2, 3, 5, 6, 8]**.
4. **Текущий массив: [2, 3, 5, 6, 8], где [2, 3] – неотсортированный подмассив, а [5, 6, 8] – отсортированный подмассив**
    - Максимальный элемент: **3** (индекс 1).
    - Элемент уже на своём месте, переворот не требуется
5. **Текущий массив: [2, 3, 5, 6, 8], где [2] – неотсортированный подмассив, а [3, 5, 6, 8] – отсортированный подмассив**
    - Максимальный элемент: **2** (индекс 0).
    - Элемент уже на своём месте, переворот не требуется
6. **Массив отсортирован: [2, 3, 5, 6, 8].**

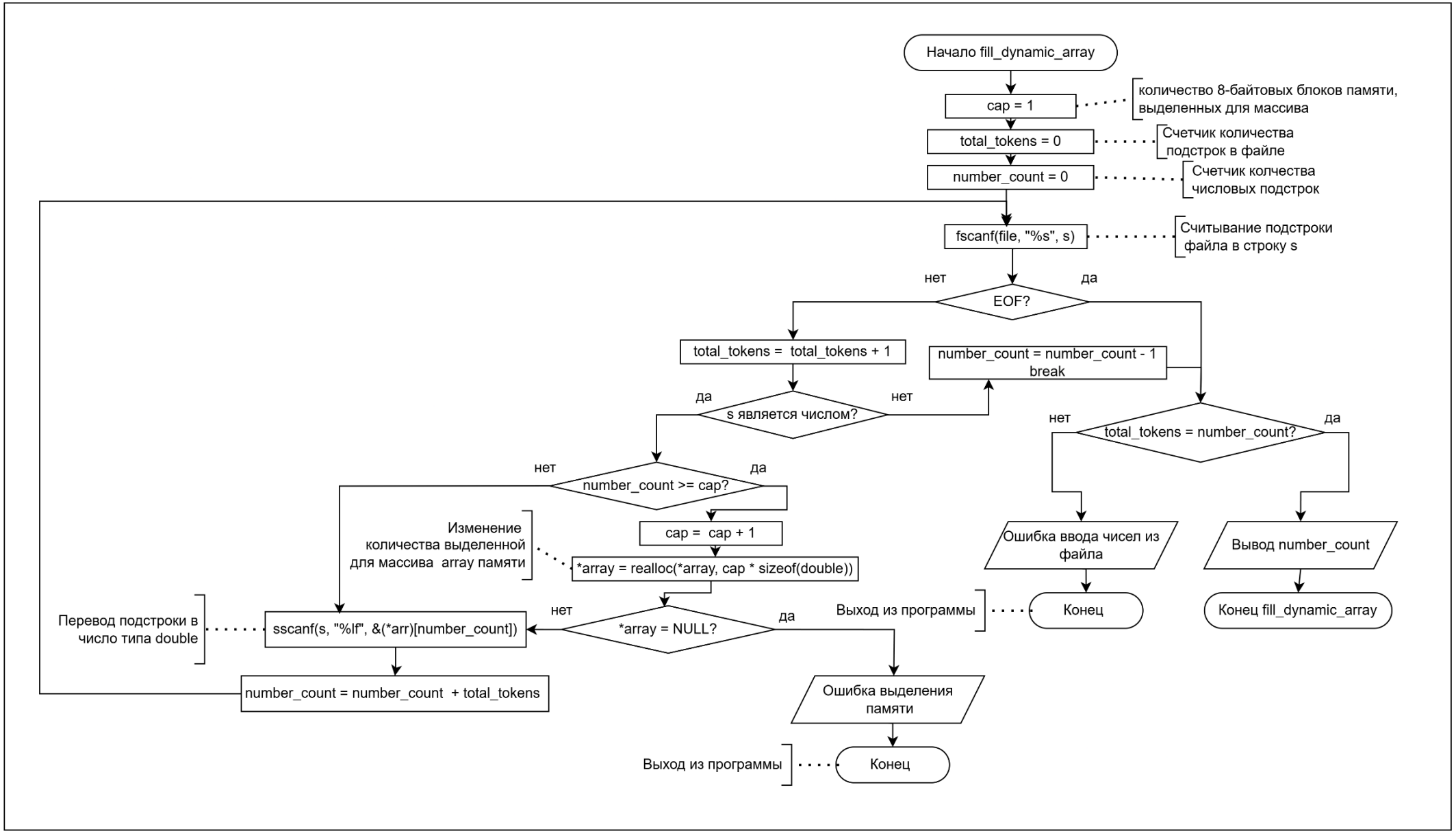
2 ТАБЛИЦА ИСПОЛЬЗУЕМЫХ ПЕРЕМЕННЫХ

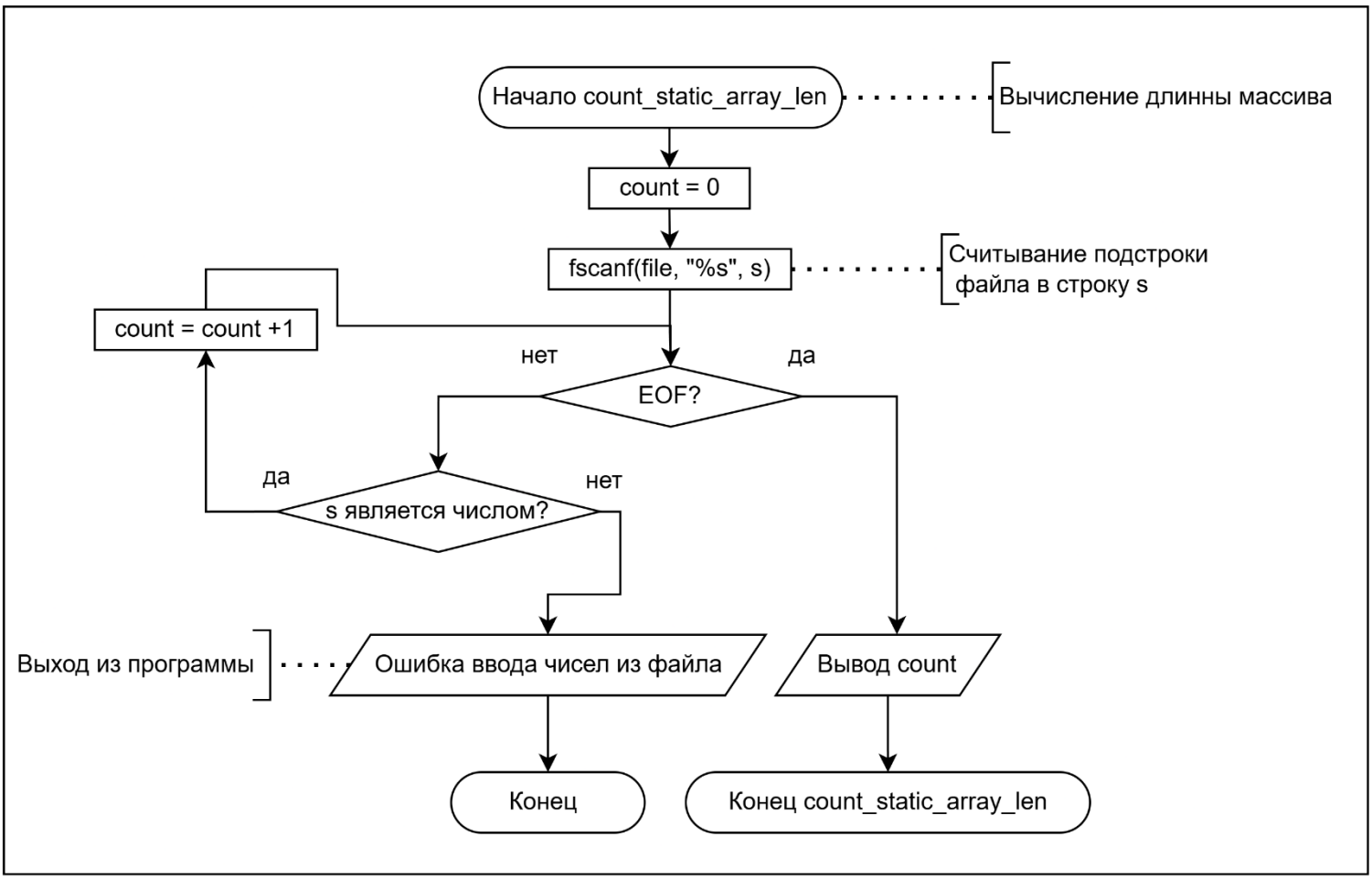
Имя переменной	Тип переменной	Диапазон значений	Роль в программе
count	int	$[0; 2^{31}-1]$	Хранит длину статического массива
array (в функции get_dynamic_array)	double **	$[0, 2^{64}-1]$	Хранит указатель на указатель на динамический массив (существует только в функции fill_dynamic_array)
cap	int	$[0; 2^{31}-1]$	Хранит число, определяющее, сколько 8 байтовых блоков памяти надо выделить для динамического массива
total_tokens	int	$[0; 2^{31}-1]$	Хранит количество строк в файле, разделённых пробелом
number_count	int	$[0; 2^{31}-1]$	Хранит количество строк в файле, которые представляют собой числа и разделены пробелом
s	char[100]	[-128; 127]	Хранит строку, содержащую число из файла
stat_array	double[]	$[-1,79 * 10^{308}; 1,79 * 10^{308}]$	Статический массив для хранения чисел
dyn_array	double *	$[0, 2^{64}-1]$	Динамический массив для хранения чисел
s	char[10]	[0; 255]	Хранит вводимую пользователем строку с выбранным типом массива
filename	char[256]	[0; 255]	Хранит имя файла, с которого происходит считывание чисел в массив
file	FILE *	$[0, 2^{64}-1]$	Хранит файловый указатель на файл, из которого считываются числа в массив
size	int	$[0; 2^{31}-1]$	Хранит длину массива (динамического или статического)
token	*char	$[0, 2^{64}-1]$	Строка(токен), полученная в ходе применения strtok к строке s (существует только в функции is_valid_number)
digit_found	int	0 или 1	Определяет наличие цифр во введённой строке
decimal_found	int	0 или 1	Определяет наличие десятичной точки во введённой строке

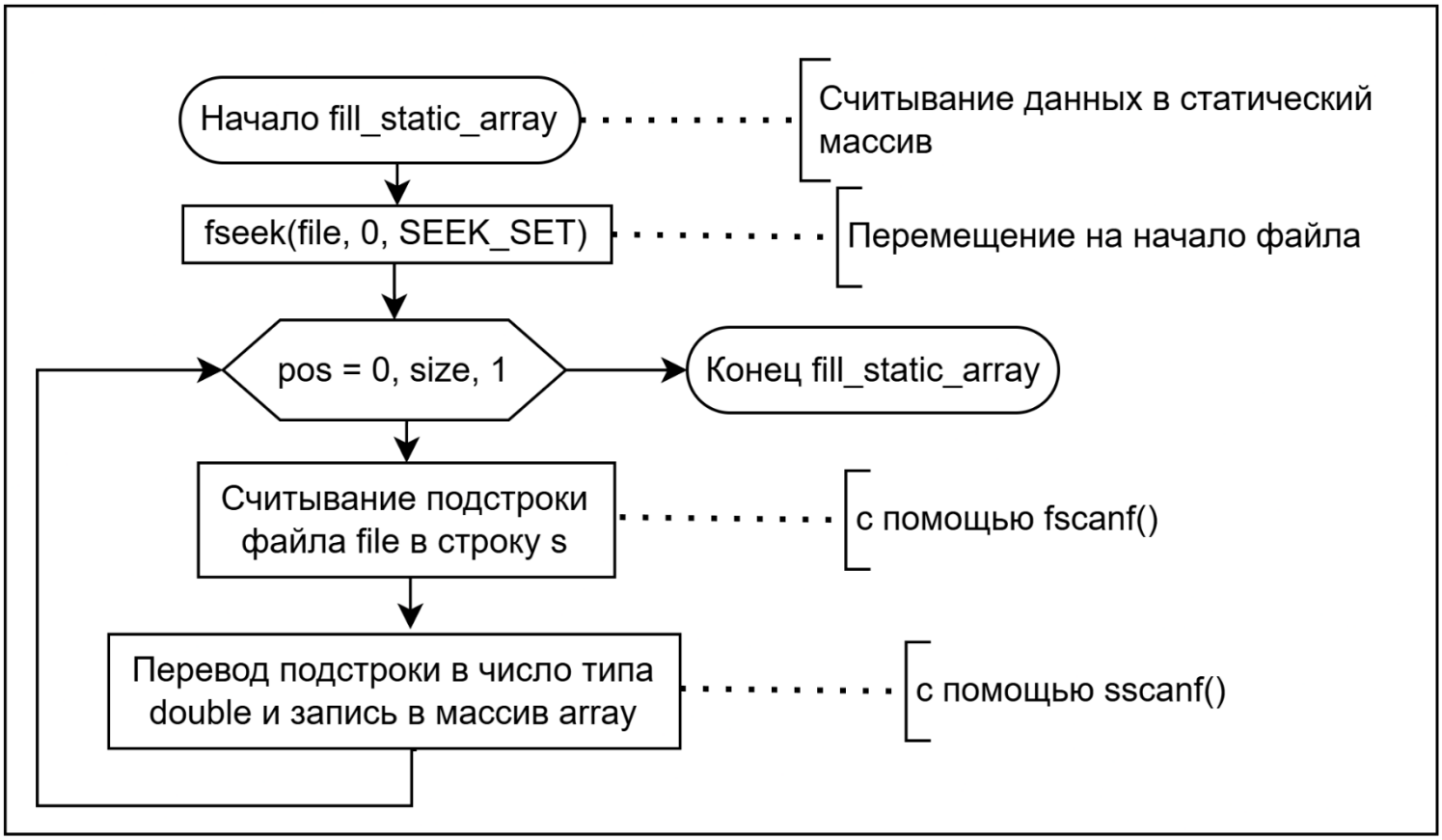
pos	int	[0; 2^{31}-1]	Хранит текущую позицию строки token (существует только в функции is_valid_number)
curr_size	int	[0; 2^{31}-1]	Хранит кол-во чисел, находящихся до отсортированного подмассива
max_index	int	[0; 2^{31}-1]	Хранит индекс максимального элемента в неотсортированном подмассиве
buffer	double	[-1,79 * 10^{308}; 1,79 * 10^{308}]	Хранит значение элемента массива, чтобы произвести замену элементов в массиве
start	int	[0; 2^{31}-1]	Хранит индекс первого элемента неотсортированного подмассива
end	int	[0; 2^{31}-1]	Хранит индекс первого элемента отсортированного подмассива

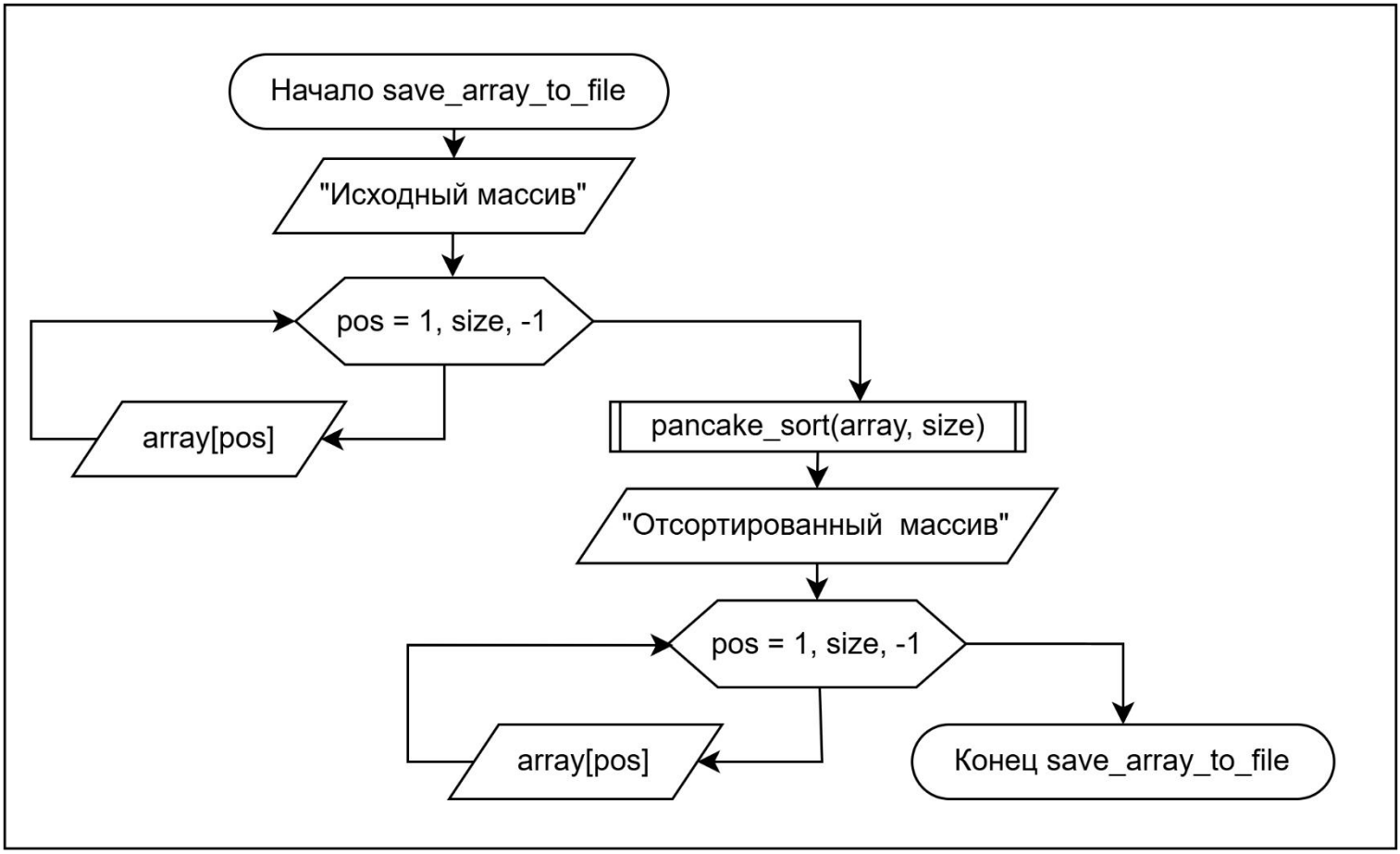
3 БЛОК-СХЕМЫ

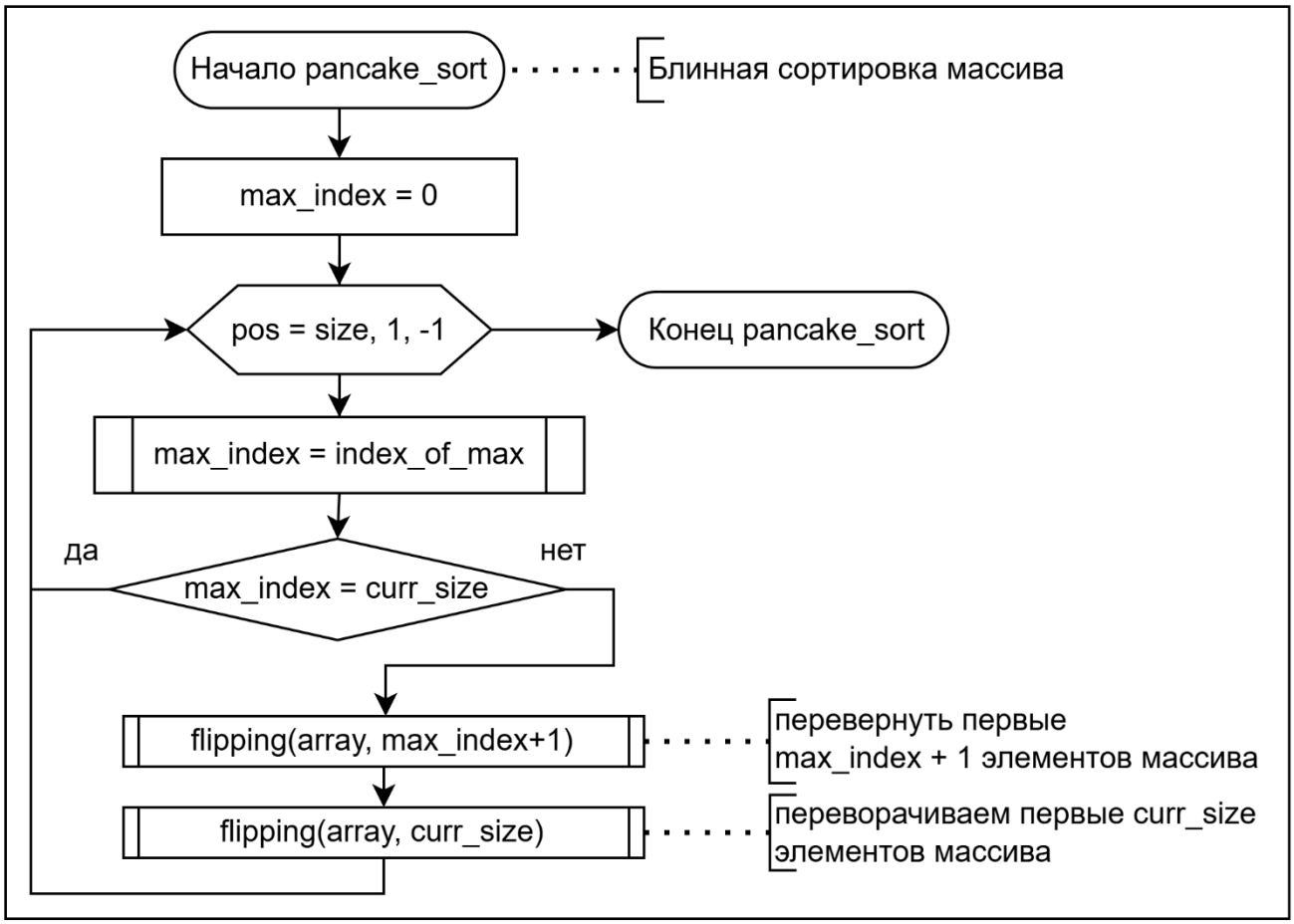

Начало
Ввод считываемого имени файла
Ввод 0 или 1
Ввод числа для определения пользователем типа массива
нет
0?
да
выделение 8 байтов для массива array
double *dyn_array = malloc(sizof(double))
size = count_static_array_len
Вычисление количества чисел в файле
да
Память выделена?
нет
нет
size > 1000?
да
Ошибка: чисел в памяти больше 1000
Выход из программы
Ошибка выделения памяти
Конец
Считывание чисел из файла в динамический массив array и определение длины массива size
fill_dynamic_array(&dynamic_array, file)
Выход из программы
Конец
double stat_array[1000]
Обьявление стат. массива длинной 1000
fill_static_array(file, stat_array, size)
Считывание чисел из файла в статический масиив array
Сортировка и вывод массива с помощью pancake_sort()
Конец

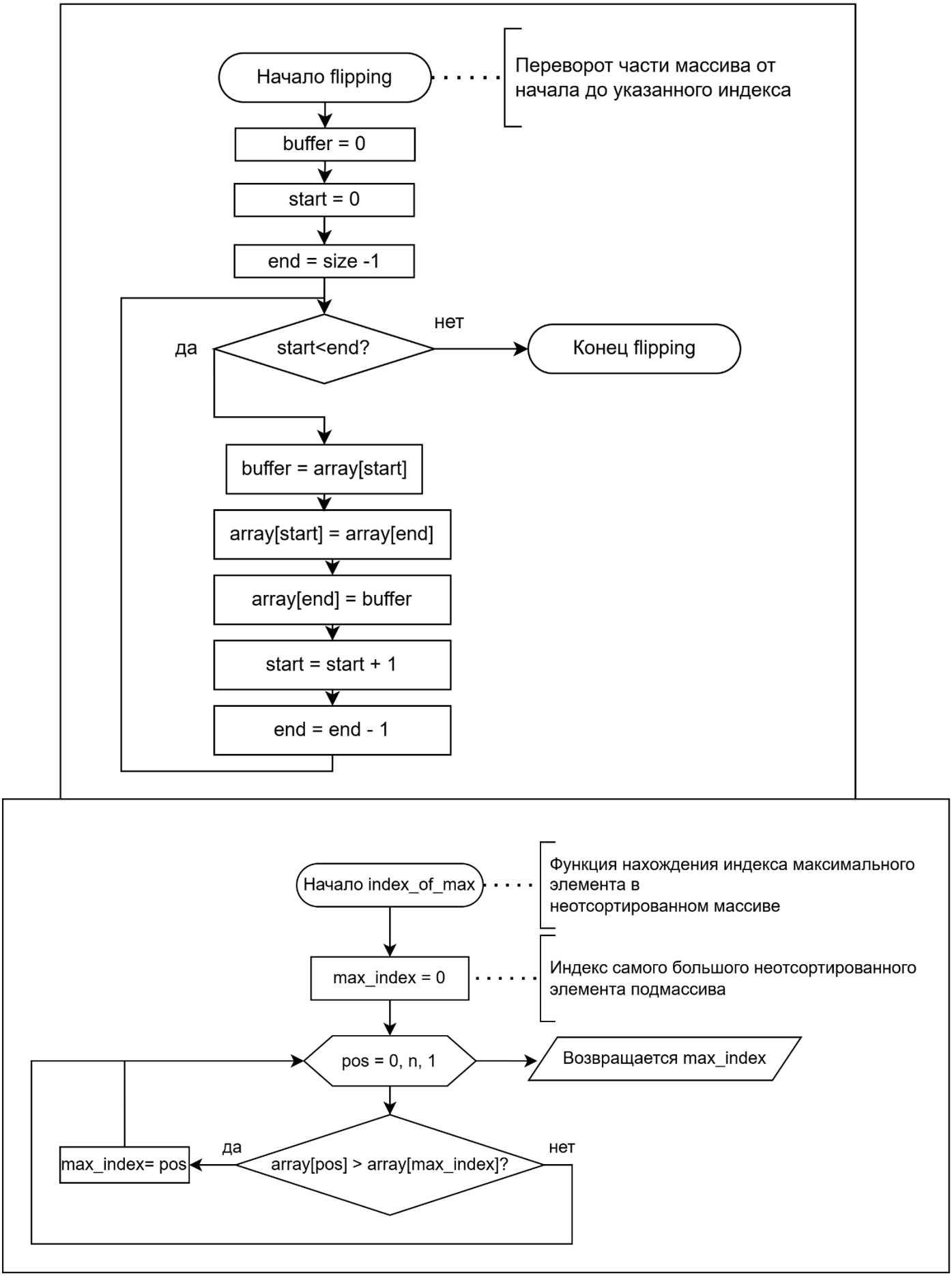

Начало fill_dynamic_array
cap = 1
количество 8-байтовых блоков памяти, выделенных для массива
total_tokens = 0
Счетчик количества подстрок в файле
number_count = 0
Счетчик колчества числовых подстрок
fscanf(file, "%s", s)
Считывание подстроки файла в строку s
нет
да
EOF?
total_tokens = total_tokens + 1
number_count = number_count - 1
break
да
нет
s является числом?
нет
да
total_tokens = number_count?
нет
да
number_count >= cap?
Изменение количества выделенной для массива array памяти
cap = cap + 1
*array = realloc(*array, cap * sizeof(double))
Ошибка ввода чисел из файла
Вывод number_count
Выход из программы
Конец
Конец fill_dynamic_array
Перевод подстроки в число типа double
sscanf(s, "%lf", &(*arr)[number_count])
нет
*array = NULL?
да
number_count = number_count + total_tokens
Ошибка выделения памяти
Выход из программы
Конец


Начало count_static_array_len
Вычисление длинны массива
count = 0
fscanf(file, "%s", s)
Считывание подстроки файла в строку s
count = count +1
нет
да
EOF?
да
нет
s является числом?
Выход из программы
Ошибка ввода чисел из файла
Вывод count
Конец
Конец count_static_array_len


Начало fill_static_array
Считывание данных в статический массив
fseek(file, 0, SEEK_SET)
Перемещение на начало файла
pos = 0, size, 1
Конец fill_static_array
Считывание подстроки файла file в строку s
с помощью fscanf()
Перевод подстроки в число типа double и запись в массив array
с помощью sscanf()


Начало save_array_to_file
"Исходный массив"
pos = 1, size, -1
array[pos]
pancake_sort(array, size)
"Отсортированный массив"
pos = 1, size, -1
array[pos]
Конец save_array_to_file


Начало pancake_sort
Блинная сортировка массива
max_index = 0
pos = size, 1, -1
Конец pancake_sort
max_index = index_of_max
да
нет
max_index = curr_size
flipping(array, max_index+1)
перевернуть первые
max_index + 1 элементов массива
flipping(array, curr_size)
переворачиваем первые curr_size
элементов массива


Начало flipping
Переворот части массива от начала до указанного индекса
buffer = 0
start = 0
end = size -1
да
start<end?
нет
Конец flipping
buffer = array[start]
array[start] = array[end]
array[end] = buffer
start = start + 1
end = end - 1
Начало index_of_max
Функция нахождения индекса максимального элемента в неотсортированном массиве
max_index = 0
Индекс самого большого неотсортированного элемента подмассива
pos = 0, n, 1
Возвращается max_index
max_index= pos
да
array[pos] > array[max_index]?
нет

4 КОД ПРОГРАММЫ С КОММЕНТАРИЯМИ

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>

// Переворачивает элементы массива от начала до позиции n-1
void flipping(double* array, int size)
{
    double buffer = 0;
    int left = 0;
    int right = size - 1;
    while (left < right)
    {
        buffer = array[left];
        array[left] = array[right];
        array[right] = buffer;
        left++;
        right--;
    }
}

// Находит индекс максимального значения в массиве длиной n
int index_of_max(double* array, int size)
{
    int max_index = 0;
    for (int j = 0; j < size; j++)
    {
        if (array[j] > array[max_index])
        {
            max_index = j;
        }
    }
    return max_index;
}

// Выполняет блинную сортировку массива
void pancake_sort(double* array, int size)
{
    for (int curr_size = size; curr_size > 1; curr_size--)
    {
        int max_index = index_of_max(array, curr_size);
        if (max_index != curr_size - 1)
        {
            flipping(array, max_index + 1);
            flipping(array, curr_size);
        }
    }
}

// Проверяет, является ли строка корректным представлением числа
int is_valid_number(char* token)
{
    int pos = 0;
    int digit_found = 0;
    int decimal_found = 0;

    while (isspace(token[pos]))
    {
        pos++;
    }
    if (token[pos] == '-')
    {
        pos++;
    }
```

```
    while (token[pos] != '\0' && !isspace(token[pos]))
    {
        if (isdigit(token[pos]))
        {
            digit_found = 1;
        }
        else if (token[pos] == '.')
        {
            if (decimal_found)
                return 0;
            decimal_found = 1;
        }
        else if (token[pos] == 'e')
        {
            pos++;
            if (token[pos] == '+' || token[pos] == '-')
            {
                pos++;
                while (token[pos] != '\0' && !isspace(token[pos]))
                {
                    if (isdigit(token[pos]))
                    {
                        digit_found = 1;
                    }
                    else
                    {
                        return 0;
                    }
                    pos++;
                }
            }
            else
            {
                return 0;
            }
        }
        else
        {
            return 0;
        }
        if (token[pos] != '\0' && !isspace(token[pos]))
            pos++;
    }
    while (isspace(token[pos]))
    {
        pos++;
    }
    return (token[pos] == '\0' && digit_found) ? 1 : 0;
}

// Считает количество чисел в файле для статического массива
int count_static_array_len(FILE* stream)
{
    int count = 0;
    char buffer[100];
    while (fscanf(stream, "%s", buffer) != EOF)
    {
        if (is_valid_number(buffer))
            count++;
        else
        {
            fprintf(stderr, "Ошибка ввода чисел из файла\n");
            exit(EXIT_FAILURE);
        }
    }
    return count;
}

// Считывает данные из файла в статический массив
void fill_static_array(FILE* stream, double* array, int size)
```

```
{
    fseek(stream, 0, SEEK_SET);
    char buffer[100];
    for (int m = 0; m < size; m++)
    {
        fscanf(stream, "%s", buffer);
        sscanf(buffer, "%le", &array[m]);
    }
}

// Выводит массив в файл до и после сортировки
void save_array_to_file(double* array, int size)
{
    FILE* output_file = fopen("tt.txt", "w");
    fprintf(output_file, "Исходный массив:\n");
    for (int t = 0; t < size; t++)
    {
        fprintf(output_file, "%lf\n", array[t]);
    }
    pancake_sort(array, size);
    fprintf(output_file, "Отсортированный массив:\n");
    for (int t = 0; t < size; t++)
    {
        fprintf(output_file, "%lf\n", array[t]);
    }
    printf("\nДанные записаны в файл: %s\n", "tt.txt");
    fclose(output_file);
}

// Считывает данные из файла в динамический массив
int fill_dynamic_array(double** array, FILE* stream)
{
    char buffer[100];
    int total_tokens = 0;
    int number_count = 0;
    int capacity = 1;

    while (fscanf(stream, "%s", buffer) != EOF)
    {
        total_tokens++;
        if (is_valid_number(buffer))
        {
            if (number_count >= capacity)
            {
                capacity++;
                *array = realloc(*array, capacity * sizeof(double));
                if (*array == NULL)
                {
                    fprintf(stderr, "Ошибка выделения памяти.\n");
                    exit(EXIT_FAILURE);
                }
            }
            sscanf(buffer, "%le", &(*array)[number_count]);
            number_count++;
        }
        else
        {
            number_count = -1;
            break;
        }
    }
    if (number_count != total_tokens)
    {
        fprintf(stderr, "Ошибка ввода чисел из файла\n");
        exit(EXIT_FAILURE);
    }
    return number_count;
}

int main()
```

```
{
    int array_choice;
    printf("Эта программа сортирует числа из указанного файла и записывает их в
tt.txt.\nЧисла должны быть разделены пробелами.\nВведите имя файла: ");
    char file_name[256];
    scanf("%s", file_name);

    FILE* input_file = fopen(file_name, "r");
    if (input_file == NULL)
    {
        printf("\nОшибка при открытии файла.\n");
        return EXIT_FAILURE;
    }

    printf("\nВыберите тип массива (0 — статический, 1 — динамический): ");
    scanf("%d", &array_choice);
    int array_size;

    if (array_choice == 1)
    {
        double* dyn_array = malloc(sizeof(double));
        array_size = fill_dynamic_array(&dyn_array, input_file);
        save_array_to_file(dyn_array, array_size);
        free(dyn_array);
    }
    else if (array_choice == 0)
    {
        array_size = count_static_array_len(input_file);
        if (array_size > 1000)
        {
            fprintf(stderr, "\nОшибка: чисел в файле больше 1000\n");
            exit(EXIT_FAILURE);
        }
        double stat_array[1000];
        fill_static_array(input_file, stat_array, array_size);
        save_array_to_file(stat_array, array_size);
    }
    else
    {
        fprintf(stderr, "\nОшибка: неверный выбор массива\n");
        exit(EXIT_FAILURE);
    }

    fclose(input_file);
    return 0;
}
```

5 ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был реализован алгоритм блинной сортировки чисел из файла. Была сделана блок-схема, иллюстрирующая работу алгоритма, и программа, которая сортирует числа из файла и записывает их в файл tt.txt. Для реализации алгоритма была написана программа на языке С на виртуальной машине. Программа была протестирована на различных входных данных, а именно:

Тест 1:

Ввод:

Введите имя файла: 1.txt

Выберите тип массива, введя число 0 или 1, где:

0 - Статический массив (его длина - 1000 элементов)

1 - Динамический массив

Ваш выбор: 1

Вывод:

Данные записаны в файл: tt.txt

Тест 2:

Ввод:

Введите имя файла: faaf

Вывод:

Ошибка при открытии файла.

Тест 3:

Ввод:

Введите имя файла: 1.txt

Выберите тип массива, введя число 0 или 1, где:

0 - Статический массив (его длина - 1000 элементов)

1 - Динамический массив

Ваш выбор: 2

Вывод:

Ошибка: некорректный выбор массива

Тест 4:

Ввод:

Введите имя файла: 1.txt

Выберите тип массива, введя число 0 или 1, где:

0 - Статический массив (его длина - 1000 элементов)

1 - Динамический массив

Ваш выбор: fafafasf

Вывод:

Ошибка: некорректный выбор массива

Тест 5:

Ввод:

Введите имя файла: 1.txt

Выберите тип массива, введя число 0 или 1, где:

0 - Статический массив (его длина - 1000 элементов)

1 - Динамический массив

Ваш выбор: 1fafaf1

Вывод:

Ошибка: некорректный выбор массива

Тест 6:

Ввод:

Введите имя файла: t1.txt

Выберите тип массива, введя число 0 или 1, где:

0 - Статический массив (его длина - 1000 элементов)

1 - Динамический массив

Ваш выбор: 0

Вывод:

Ошибка: чисел в файле больше 1000