

**Министерство образования и науки Кыргызской Республики**  
**Министерство науки и высшего образования Российской Федерации**

Государственное образовательное учреждение  
высшего профессионального образования  
Кыргызско-Российский Славянский университет  
Имени первого Президента Российской Федерации Б. Н. Ельцина  
Естественно-технический факультет  
Кафедра информационных и вычислительных технологий

**КУРСОВАЯ РАБОТА**

По дисциплине: «Алгоритмы и структуры данных»

**Тема: Вариант №3**

**«Демонстрация работы алгоритма сортировки вставками»**

Выполнил: студент группы ЕПИ-4-23 Лосев Данил

Руководитель: Каткова Светлана Николаевна

Оценка: \_\_\_\_\_

Дата: \_\_\_\_\_

**Бишкек-2024**

# Оглавление

Постановка задачи .....	2
Техническое задание .....	2
1. Обзор .....	2
1.1. Введение .....	2
1.2. Назначение программы .....	2
1.3. Цели программы .....	2
1.4. Требования к программе .....	2
1.5. Организация входных и выходных данных.....	2
1.6. Требования к надежности .....	2
1.7. Требования к составу и параметрам технических средств.....	3
1.8. Требования к программной совместимости.....	3
1.9. Платформа, язык программирования .....	3
2. Модель программы.....	3
Архитектура программы .....	4
Форматы файлов .....	4
Тестирование .....	5
Тест №1.....	5
Тест №2.....	5
Тест №3.....	6
Тест №4.....	6
План работы.....	6
Блок-схемы.....	7
Исходный код программы.....	14
Выводы по результатам работы.....	20
Руководство пользователя (приложение 1).....	20
Введение .....	20
Подготовка к работе.....	20
Описание операций .....	21
Аварийные ситуации .....	21
Руководство программиста (приложение 2) .....	21
Назначение и условия применения программы .....	21
Описание среды разработки .....	22
Входные и выходные данные. ....	22
Описание логики работы программы .....	22

## Постановка задачи

Продемонстрируйте работу метода сортировки вставками по возрастанию. Для этого выведите состояние данного массива после каждой вставки на отдельных строках. Если массив упорядочен изначально, не нужно ничего выводить.

### Формат входных данных:

На первой строке кода - целое число  $n$  ( $1 \leq n \leq 100$ ) – количество элементов в массиве. На второй строке задан сам массив: последовательность натуральных чисел, не превышающих  $10^9$ .

### Формат выходных данных:

В выходной файл выведите строки (по количеству вставок) по  $n$  чисел каждая.

## Техническое задание

### 1. Обзор

#### 1.1. Введение

1.1.1. Программа разрабатывается на основе учебного плана кафедры «Информационные и вычислительные технологии».

1.1.2. Наименование системы: **Программа для сортировки массива методом вставок.**

1.1.3. Разработчик: **Данил Лосев, студент группы ЕПИ-4-23, КРСУ.**

#### 1.2. Назначение программы

Программа предназначена для демонстрации работы алгоритма сортировки методом вставок с пошаговым выводом состояния массива на экран и в файл.

#### 1.3. Цели программы

- Реализация и демонстрация алгоритма сортировки вставками.
- Обеспечение пошагового отображения изменений в массиве при каждой вставке.
- Формирование отчетных данных в соответствии с заданным форматом.

#### 1.4. Требования к программе

##### 1.4.1. Требования к функциональным характеристикам:

- Принимает на вход:
  - целое число  $n$  ( $1 \leq n \leq 100$ ), указывающее количество элементов массива;
  - массив из  $n$  натуральных чисел (каждое не превышает  $10^9$ ).
- Сортирует массив методом вставок.
- Выводит состояние массива после каждой вставки в формате строк, каждая строка содержит  $n$  чисел на экран и в файл.
- Если массив изначально отсортирован, ничего не выводить.

#### 1.5. Организация входных и выходных данных

- Входные данные:
  - Первой строкой — число  $n$ ;
  - Второй строкой — элементы массива, разделенные пробелами.
- Выходные данные:
  - Состояние массива после каждой вставки (если применимо), каждая строка содержит  $n$  чисел на экран и в файл.

#### 1.6. Требования к надежности

- Программа корректно обрабатывает массивы любых размеров в пределах допустимых значений  $n$ .

- Обеспечена устойчивость программы к некорректному вводу с соответствующими сообщениями об ошибках.

### **1.7. Требования к составу и параметрам технических средств**

- **Минимальная конфигурация:**
  - Процессор с тактовой частотой 1 ГГц.
  - 512 МБ оперативной памяти.
  - 50 МБ свободного дискового пространства.
- **Рекомендуемая конфигурация:**
  - Процессор с тактовой частотой 2 ГГц или выше.
  - 2 ГБ оперативной памяти.
  - 100 МБ свободного дискового пространства.

### **1.8. Требования к программной совместимости**

Программа совместима с ОС Windows.

### **1.9. Платформа, язык программирования**

- Платформа: консольное приложение.
- Язык программирования: C++.

## **2. Модель программы**

Программа представляет из себя консольное приложение с простым и интуитивно понятным интерфейсом. При запуске программы появляется начальный экран где у пользователя просят ввести размер массива (В случае если пользователь вводит неправильный размер высвечивается ошибка и у пользователя просят ввести размер ещё раз).

```
Enter the size of array ( 1 ≤ size ≤ 100 ): 101

ERROR: You entered the wrong size!!!
Please try again
```

```
Enter the size of array ( 1 ≤ size ≤ 100 ): 5_
```

В случае если пользователь вводит размер правильно, то его переводит в новый раздел, где просят ввести сам массив (В случае если пользователь вводит неправильный массив высвечивается ошибка и у пользователя просят ввести размер ещё раз).

```
The size of array: 5
Enter the array ( natural numbers not exceeding 10^9 ): -1 цу 345 132423
ERROR: Invalid input (e.g., number exceeds the limit or wrong format)!!!
```

Если пользователь ввёл правильный массив, то начинается процесс сортировки (Если массив изначально отсортирован, то программа об этом скажет)

```
The size of array: 5
Enter the array ( natural numbers not exceeding 10^9 ): 1 2 3 4 5

Array is sorted

Sorted array: [1] [2] [3] [4] [5]
```

```

The size of array: 5
Enter the array ( natural numbers not exceeding 10^9 ): 123 456 13 75675 213

Your array: [123] [456] [13] [75675] [213]

Sorting...
123      [456]      13      75675      213
[13] ← 123 ← 456      75675      213
13      123      456      [75675]      213
13      123      [213] ← 456 ← 75675

Sorted array: [13] [123] [213] [456] [75675]

```

Также данная сортировка сохраняется в файл

После этого программа предлагает попробовать ещё раз

```

Try again? ( 1=yes , 0=no ): 0_

```

## Архитектура программы

Ниже представлены **класс**, реализуемый в программе, его атрибуты и методы

Название:	DYNAMIC_ARRAY
Атрибуты:	bool isError = false bool isSorted = true int *cArray int cSize
Методы:	DYNAMIC_ARRAY(int fSize) ~DYNAMIC_ARRAY() void deleteArray() int get(int flIndex) void set(int flIndex, int fElement) bool getError() void isSort() void fillTheArray(const std::string flInput) void insertionSortArray() void printArray(char fOpen = ' ', char fClose = ' ') void printArrayInSort(int flIndexOfInsertElement, int fStart) void printArrayInSortToFile(int flIndexOfInsertElement, int fStart, std::string fName = FILE_NAME)

## Форматы файлов

В программе используются следующие форматы файлов:

### 1. Исходный файл программы:

- **Имя файла:** TaskVariantThreeClass.cpp
- **Тип файла:** текстовый файл с кодом на языке C++.
- **Содержание:**
  - Полный исходный код программы, реализующий сортировку массива методом вставок.

### 2. Подключённые библиотек:

- **<cstdlib>** Подключение библиотеки для выполнения системных команд, например, очистки консоли

- **<fstream>** Подключение библиотеки для работы с файлами
- **<iostream>** Подключение библиотеки для работы с вводом и выводом в консоль
- **<limits>** Подключение библиотеки для определения пределов чисел, используется в проверке ввода
- **<regex>** Подключение библиотеки для работы с регулярными выражениями
- **<sstream>** Подключение библиотеки для работы с потоками строк, преобразование строк в числа
- **<string>** Подключение библиотеки для работы со строками
- **<synchapi.h>** Подключение библиотеки для использования функции Sleep для пауз в выполнении
- **<windows.h>** Подключение библиотеки для работы с функциями Windows, например, времени системы

### 3. Файл вывода:

- **Имя файла:** output.txt
- **Тип файла:** текстовый файл.
- **Содержание:**
  - Состояние массива после каждой вставки в процессе сортировки.
  - Каждый результат представлен строкой, содержащей элементы массива, разделенные пробелами.
- **Пример содержимого:**

```
Insertion step: 1: 123 [456] 13 75675 213
Insertion step: 2: [13] ← 123 ← 456 75675 213
Insertion step: 3: 13 123 456 [75675] 213
Insertion step: 4: 13 123 [213] ← 456 ← 75675
```

- **Особенности:**
  - Файл создается программой автоматически.

## Тестирование

### Тест №1

**Ситуация:** Ввод размера массива и самого массива в неправильном диапазоне

**Ожидание:** Программа должна сообщить о неправильном вводе

**Проверка:**

```
Enter the size of array ( 1 ≤ size ≤ 100 ): 101
```

```
ERROR: You entered the wrong size!!!
Please try again
```

```
The size of array: 5
```

```
Enter the array ( natural numbers not exceeding 10^9 ): 1000000001
```

```
ERROR: Invalid input (e.g., number exceeds the limit or wrong format)!!!
```

**Вывод:** Программа успешно вывела сообщение об ошибке

### Тест №2

**Ситуация:** Ввод массива размером в 1 элемент

**Ожидание:** Вывод сообщения что массив отсортирован

**Проверка:**

```
The size of array: 1
Enter the array ( natural numbers not exceeding 10^9 ): 100

Array is sorted

Sorted array: [100]
```

**Вывод:** Программа успешно вывела информацию о том, что массив отсортирован

### Тест №3

**Ситуация:** Ввод отсортированного массива

**Ожидание:** Вывод сообщения что массив отсортирован

**Проверка:**

```
The size of array: 5
Enter the array ( natural numbers not exceeding 10^9 ): 1 2 3 4 5

Array is sorted

Sorted array: [1] [2] [3] [4] [5]
```

**Вывод:** Программа успешно вывела информацию о том, что массив отсортирован

### Тест №4

**Ситуация:** Ввод неотсортированного массива

**Ожидание:** Вывод статуса сортировки в каждый момент вставки

**Проверка:**

```
The size of array: 5
Enter the array ( natural numbers not exceeding 10^9 ): 312 5643 234 67 2

Your array: [312] [5643] [234] [67] [2]

Sorting ...
312    [5643]    234    67    2
[234] ← 312 ← 5643    67    2
[67]  ← 234 ← 312 ← 5643    2
[2]   ← 67  ← 234 ← 312 ← 5643

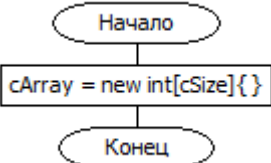
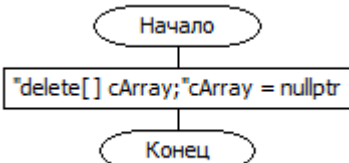
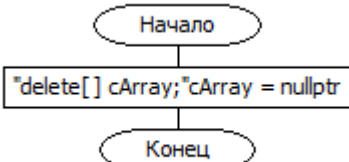
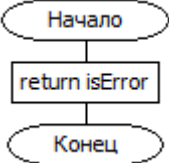
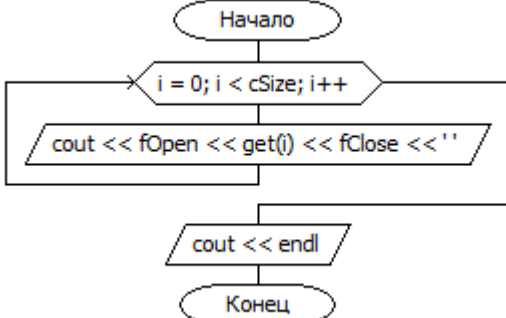
Sorted array: [2] [67] [234] [312] [5643]
```

**Вывод:** Программа вывела статус сортировки в каждый момент вставки

## План работы

1. Анализ задачи и постановка требований (18.11.2024)
2. Разработка технического задания (19.11.2024)
3. Написание кода программы (20.11.2024 – 23.11.2024)
4. Тестирование программы (24.11.2024)
5. Отладка кода (24.11.2024)
6. Написание отчета (26.11.2024)
7. Первая проверка курсовой работы преподавателем (30.11.2024)
8. Сдача работы (07.12.2024)

## Блок-схемы

<pre>DYNAMIC_ARRAY::DYNAMIC_ARRAY(int fSize) : cSize(fSize) {     cArray = new int[cSize]{}; };</pre>	 <pre> graph TD     Start([Начало]) --&gt; Process[cArray = new int[cSize]{}]     Process --&gt; End([Конец])     </pre>
<pre>DYNAMIC_ARRAY::~~DYNAMIC_ARRAY() {     delete[] cArray;     cArray = nullptr; }</pre>	 <pre> graph TD     Start([Начало]) --&gt; Process["delete[] cArray; cArray = nullptr"]     Process --&gt; End([Конец])     </pre>
<pre>void DYNAMIC_ARRAY::deleteArray() {     delete[] cArray;     cArray = nullptr; }</pre>	 <pre> graph TD     Start([Начало]) --&gt; Process["delete[] cArray; cArray = nullptr"]     Process --&gt; End([Конец])     </pre>
<pre>bool DYNAMIC_ARRAY::getError() {     return isError; }</pre>	 <pre> graph TD     Start([Начало]) --&gt; Process[return isError]     Process --&gt; End([Конец])     </pre>
<pre>void DYNAMIC_ARRAY::printArray(char fOpen, char fClose) {     for (int i = 0; i &lt; cSize; i++)     {         cout &lt;&lt; fOpen &lt;&lt; get(i) &lt;&lt; fClose &lt;&lt; ' ';     }     cout &lt;&lt; endl; }</pre>	 <pre> graph TD     Start([Начало]) --&gt; Loop{i = 0; i &lt; cSize; i++}     Loop --&gt; Body[/cout &lt;&lt; fOpen &lt;&lt; get(i) &lt;&lt; fClose &lt;&lt; ' '/]     Body --&gt; Loop     Loop --&gt; EndPrint[/cout &lt;&lt; endl/]     EndPrint --&gt; End([Конец])     </pre>



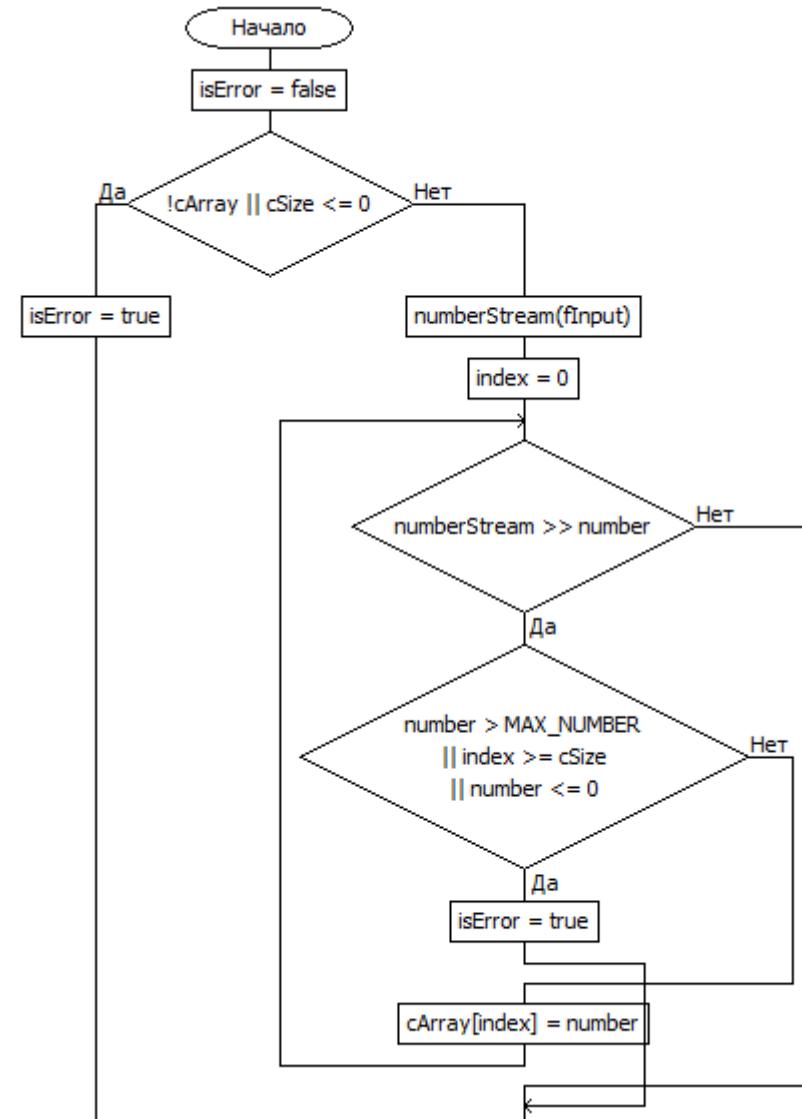
```

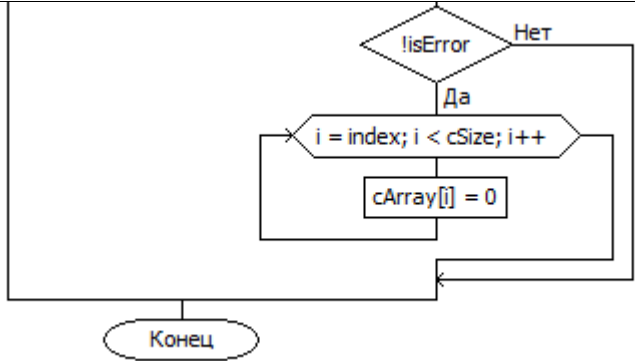
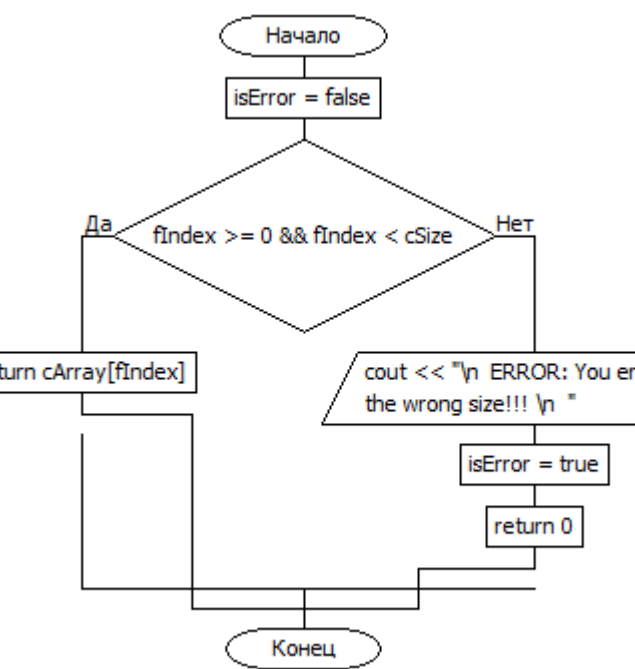
void DYNAMIC_ARRAY::fillTheArray(const string fInput)
{
    isError = false;
    if (!cArray || cSize <= 0)
    {
        isError = true;
    }
    else
    {
        istringstream numberStream(fInput);
        int number;
        int index = 0;

        while (numberStream >> number)
        {
            if (number > MAX_NUMBER || index >= cSize || number <= 0)
            {
                isError = true;
                break;
            }
            cArray[index] = number;
            ++index;
        }

        if (!isError)
        {
            for (int i = index; i < cSize; i++)
            {
                cArray[i] = 0;
            }
        }
    }
}

```



	 <pre> graph TD     Start([Начало]) --&gt; Init[isError = false]     Init --&gt; Cond{findex &gt;= 0 &amp;&amp; findex &lt; cSize}     Cond -- Да --&gt; Ret[return cArray[findex]]     Ret --&gt; End([Конец])     Cond -- Нет --&gt; Print[/cout &lt;&lt; "\n ERROR: You entered the wrong size!!! \n "/]     Print --&gt; SetTrue[isError = true]     SetTrue --&gt; Ret0[return 0]     Ret0 --&gt; End </pre>
<pre> int DYNAMIC_ARRAY::get(int fIndex) {     isError = false;     if (fIndex &gt;= 0 &amp;&amp; fIndex &lt; cSize)     {         return cArray[fIndex];     }     else     {         cout &lt;&lt; "\n ERROR: You entered the wrong size!!! \n ";         isError = true;         return 0;     } } </pre>	 <pre> graph TD     Start([Начало]) --&gt; Init[isError = false]     Init --&gt; Cond{findex &gt;= 0 &amp;&amp; findex &lt; cSize}     Cond -- Да --&gt; Ret[return cArray[findex]]     Ret --&gt; End([Конец])     Cond -- Нет --&gt; Print[/cout &lt;&lt; "\n ERROR: You entered the wrong size!!! \n "/]     Print --&gt; SetTrue[isError = true]     SetTrue --&gt; Ret0[return 0]     Ret0 --&gt; End </pre>

```

void DYNAMIC_ARRAY::insertionSortArray()
{
    isSort();
    if (isSorted == false)
    {
        cout << "\nYour array: ";
        printArray('[', ']');
        cout << "\nSorting...\n";

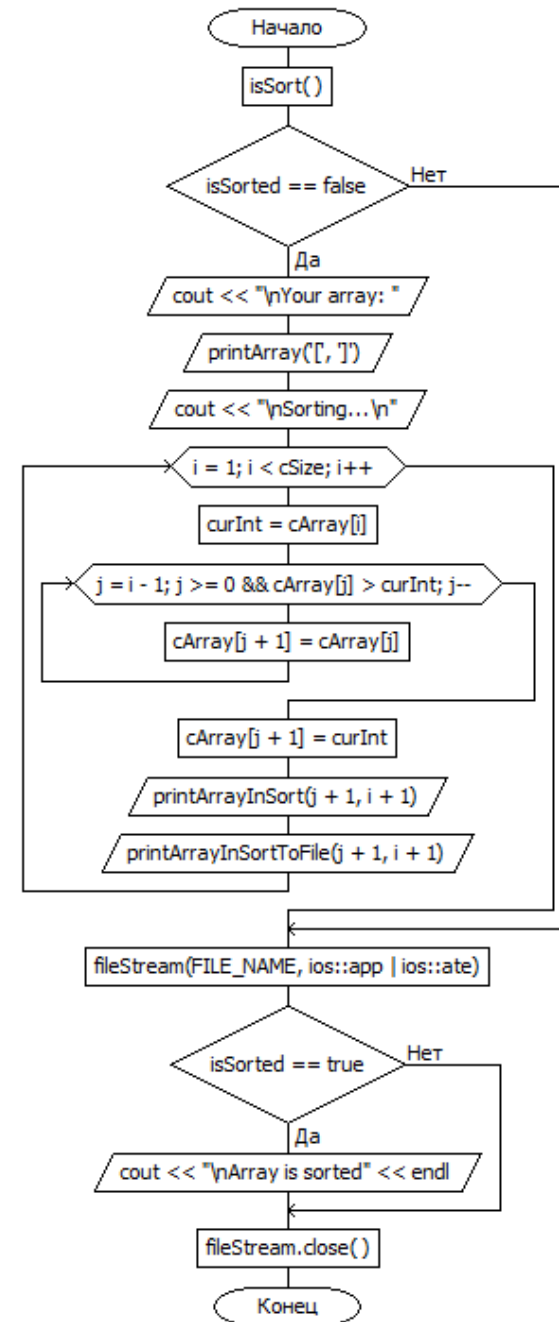
        for (int i = 1; i < cSize; i++)
        {
            int curInt = cArray[i];
            int j;

            for (j = i - 1; j >= 0 && cArray[j] > curInt; j--)
            {
                cArray[j + 1] = cArray[j];
            }
            cArray[j + 1] = curInt;

            printArrayInSort(j + 1, i + 1);
            printArrayInSortToFile(j + 1, i + 1);
        }

        ofstream fileStream(FILE_NAME, ios::app | ios::ate);
        if (isSorted == true)
        {
            cout << "\nArray is sorted" << endl;
            fileStream << "\nArray is sorted" << endl;
        }
        fileStream << endl;
        fileStream.close();
    }
}

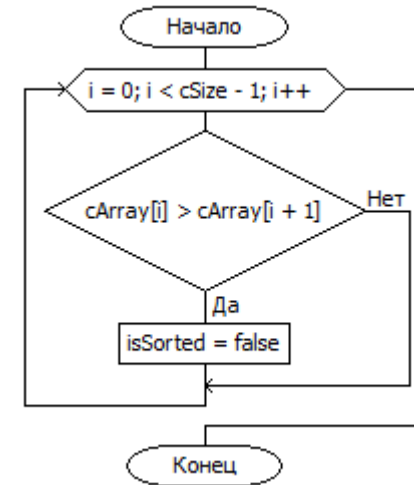
```



```

void DYNAMIC_ARRAY::isSort()
{
    for (int i = 0; i < cSize - 1; i++)
    {
        if (cArray[i] > cArray[i + 1])
        {
            isSorted = false;
        }
    }
}

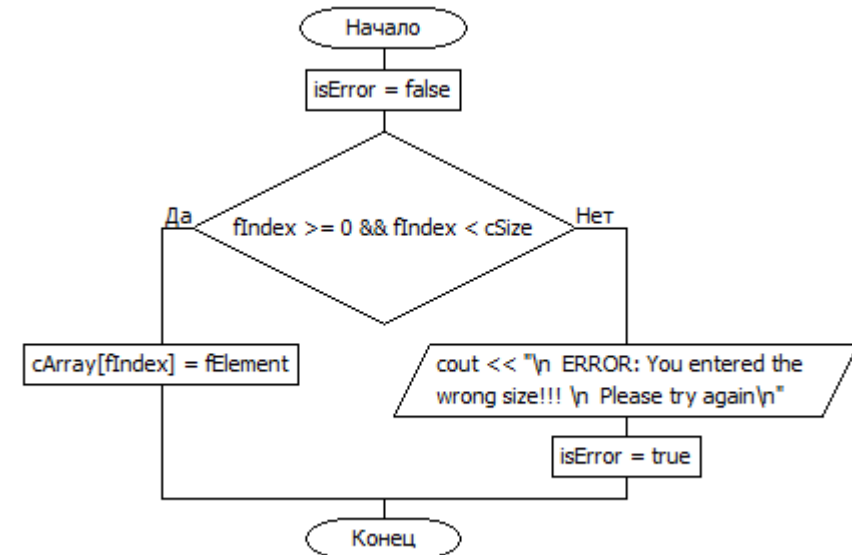
```



```

void DYNAMIC_ARRAY::set(int fIndex, int fElement)
{
    isError = false;
    if (fIndex >= 0 && fIndex < cSize)
    {
        cArray[fIndex] = fElement;
    }
    else
    {
        cout << "\n ERROR: You entered the wrong size!!! \n Please try
again\n";
        isError = true;
    }
}

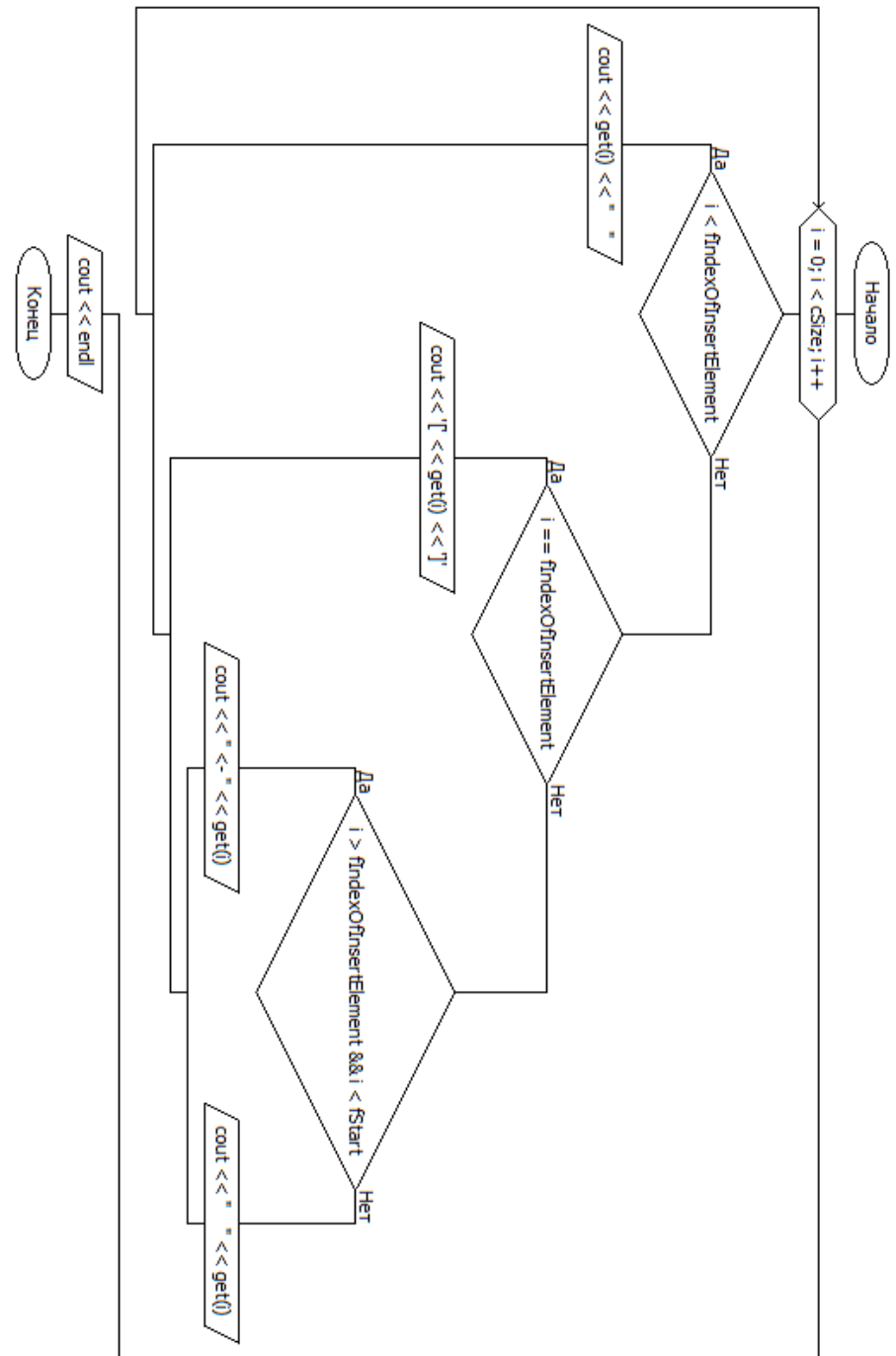
```



```

void DYNAMIC_ARRAY::printArrayInSort(int fIndexOfInsertElement, int
fStart)
{
    for (int i = 0; i < cSize; i++)
    {
        if (i < fIndexOfInsertElement)
        {
            cout << get(i) << "  ";
        }
        else if (i == fIndexOfInsertElement)
        {
            cout << '[' << get(i) << ']';
        }
        else if (i > fIndexOfInsertElement && i < fStart)
        {
            cout << " <- " << get(i);
        }
        else
        {
            cout << "  " << get(i);
        }
    }
    cout << endl;
}

```

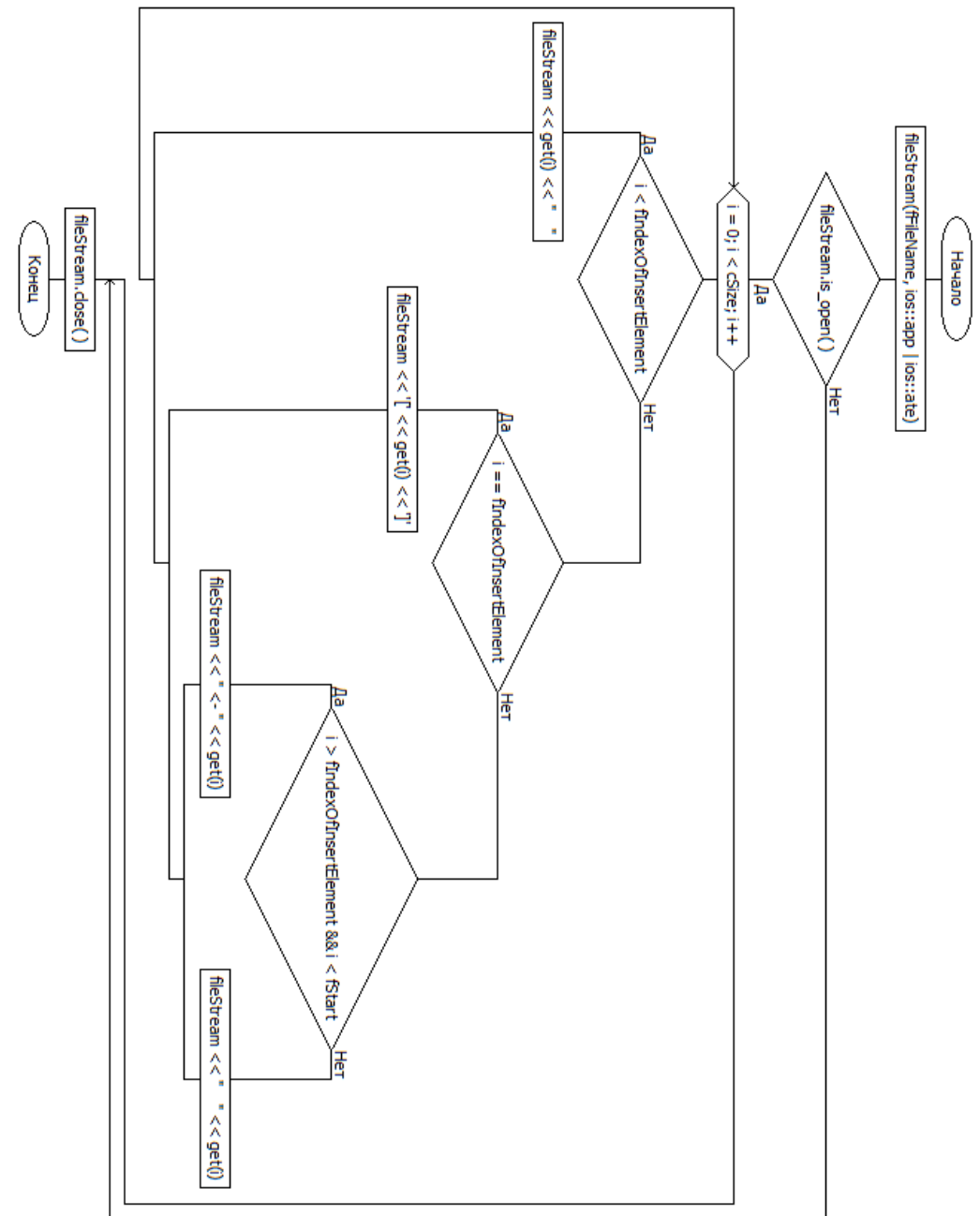


```

void DYNAMIC_ARRAY::printArrayInSortToFile(int
fIndexOfInsertElement, int fStart, string fFileName)
{
    ofstream fileStream(fFileName, ios::app | ios::ate);
    if (fileStream.is_open())
    {
        fileStream << "Insertion step: " << fStart - 1 << ": ";

        for (int i = 0; i < cSize; i++)
        {
            if (i < fIndexOfInsertElement)
            {
                fileStream << get(i) << " ";
            }
            else if (i == fIndexOfInsertElement)
            {
                fileStream << '[' << get(i) << ']';
            }
            else if (i > fIndexOfInsertElement && i < fStart)
            {
                fileStream << " <- " << get(i);
            }
            else
            {
                fileStream << " " << get(i);
            }
        }
        fileStream << endl;
    }
    fileStream.close();
}

```



## Исходный код программы

```
#include <cstdlib> // Подключение библиотеки для выполнения системных команд, например,
очистки консоли
#include <fstream> // Подключение библиотеки для работы с файлами
#include <iostream> // Подключение библиотеки для работы с вводом и выводом в консоль
#include <limits> // Подключение библиотеки для определения пределов чисел, используется в
проверке ввода
#include <regex> // Подключение библиотеки для работы с регулярными выражениями
#include <sstream> // Подключение библиотеки для работы с потоками строк, преобразование
строк в числа
#include <string> // Подключение библиотеки для работы со строками
#include <synchapi.h> // Подключение библиотеки для использования функции Sleep для пауз в
выполнении
#include <windows.h> // Подключение библиотеки для работы с функциями Windows, например,
времени системы

// Объявление глобальных констант для ограничения размеров массива и записи данных
const int MAX_NUMBER = 1000000000; // Ограничение на максимальное значение элементов
массива
const int MAX_ARRAY_SIZE = 100; // Ограничение на максимальный размер массива
const int MIN_ARRAY_SIZE = 1; // Ограничение на минимальный размер массива
const std::string FILE_NAME = "output.txt"; // Название файла для записи данных

// Определение класса для работы с динамическими массивами
class DYNAMIC_ARRAY
{
private:
    bool isError = false; // Флаг ошибок, возникающих при работе с массивом
    bool isSorted = true; // Флаг проверки упорядоченности массива
    int *cArray; // Указатель на динамический массив
    int cSize; // Текущий размер массива

public:
    DYNAMIC_ARRAY(int fSize); // Конструктор для выделения памяти под массив
    ~DYNAMIC_ARRAY(); // Деструктор для очистки памяти
    void deleteArray(); // Метод для удаления динамического массива
    int get(int fIndex); // Метод для получения элемента по индексу
    void set(int fIndex, int fElement); // Метод для установки элемента по индексу
    bool getError(); // Метод для проверки наличия ошибок
    void isSort(); // Метод для проверки массива на упорядоченность
    void fillTheArray(const std::string fInput); // Метод для заполнения массива из строки
    void insertionSortArray(); // Метод для сортировки массива вставками
    void printArray(char fOpen = ' ', char fClose = ' '); // Метод для вывода массива в
консоль
    void printArrayInSort(int fIndexofInsertElement, int fStart); // Метод для вывода
отсортированного массива
    void printArrayInSortToFile(int fIndexofInsertElement, int fStart,
                                std::string fFileName = FILE_NAME); // Метод для записи
отсортированного массива в файл
};

// Основная функция программы
int main()
{
    bool isProgram = true, sizeError = false; // Флаги продолжения программы и ошибок
размера
    while (isProgram) // Главный цикл программы
    {
```

```

int arraySize = 0; // Переменная для хранения размера массива
do
{
    std::system("cls"); // Очистка консоли перед вводом
    sizeError = false; // Сброс флага ошибок
    std::cout << "Enter the size of array ( 1 <= size <= 100 ): "; // Запрос
размера массива
    std::cin >> arraySize; // Ввод размера
массива
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Очистка
буфера ввода

    // Проверка корректности ввода размера массива
    if (arraySize > MAX_ARRAY_SIZE || arraySize < MIN_ARRAY_SIZE ||
std::cin.fail())
    {
        std::cin.clear(); // Сброс флага ошибки ввода
        std::cin.ignore(); // Очистка оставшегося ввода
        std::cout << " ERROR: You entered the wrong size!!! \n Please try
again\n"; // Сообщение об ошибке
        Sleep(2000); // Задержка перед повторным вводом
        sizeError = true; // Установка флага ошибки
    }
} while (sizeError); // Повтор запроса до корректного ввода

DYNAMIC_ARRAY array(arraySize); // Создание объекта класса с заданным размером
массива

std::string input; // Переменная для строки ввода значений массива
std::regex pattern("^\\d+(\\s+\\d+)*$"); // Регулярное выражение для проверки
формата ввода
bool isSymbol = false; // Флаг корректности ввода

do
{
    std::system("cls"); // Очистка консоли перед вводом
    std::cout << "The size of array: " << arraySize << std::endl
        << "Enter the array ( natural numbers not exceeding 10^9 ): "; //
Запрос значений массива
    std::getline(std::cin, input); // Ввод строки значений массива
    array.fillTheArray(input); // Заполнение массива
    isSymbol = std::regex_match(input, pattern); // Проверка строки на
соответствие формату

    // Проверка корректности ввода или наличия ошибок
    if (array.getError() || !isSymbol)
    {
        std::cout
            << " ERROR: Invalid input (e.g., number exceeds the limit or wrong
format)!!!\n"; // Сообщение об
// ошибке
        Sleep(2000); // Задержка перед повторным вводом
    }
} while (array.getError() || !isSymbol); // Повтор, пока ввод некорректен

array.insertionSortArray(); // Сортировка массива методом вставок
std::cout << std::endl;
std::cout << "Sorted array: ";

```



```

        array.printArray('[', ']); // Вывод отсортированного массива в консоль
        Sleep(2000);                // Задержка перед завершением цикла
        array.deleteArray();         // Удаление динамического массива
        std::cout << "\nTry again? ( 1=yes , 0=no ): "; // Запрос продолжения программы
        std::cin >> isProgram;       // Чтение решения пользователя
    }
    return 0; // Возврат успешного завершения программы
}

// Реализация конструктора класса для инициализации массива
DYNAMIC_ARRAY::DYNAMIC_ARRAY(int fSize) : cSize(fSize)
{
    cArray = new int[cSize]{}; // Выделение памяти для массива заданного размера
};

// Реализация деструктора для освобождения памяти
DYNAMIC_ARRAY::~~DYNAMIC_ARRAY()
{
    delete[] cArray; // Удаление массива
    cArray = nullptr; // Обнуление указателя
}

// Метод для удаления массива
void DYNAMIC_ARRAY::deleteArray()
{
    delete[] cArray; // Освобождение памяти массива
    cArray = nullptr; // Обнуление указателя для предотвращения утечек
}

// Метод для получения элемента по индексу
int DYNAMIC_ARRAY::get(int fIndex)
{
    isError = false; // Сброс флага ошибки
    if (fIndex >= 0 && fIndex < cSize) // Проверка индекса на корректность
    {
        return cArray[fIndex]; // Возврат значения элемента
    }
    else
    {
        std::cout << "\n ERROR: You entered the wrong size!!! \n "; // Сообщение об
ошибке
        isError = true; // Установка флага
ошибки
        return 0; // Возврат нуля при
ошибке
    }
}

// Метод для установки элемента по индексу
void DYNAMIC_ARRAY::set(int fIndex, int fElement)
{
    isError = false; // Сброс флага ошибки
    if (fIndex >= 0 && fIndex < cSize) // Проверка индекса на корректность
    {
        cArray[fIndex] = fElement; // Установка значения элемента
    }
    else
    {

```

```

        std::cout << "\n ERROR: You entered the wrong size!!! \n Please try again\n"; //
Сообщение об ошибке
        isError = true; // Установка флага ошибки
    }
}

// Метод для проверки наличия ошибок
bool DYNAMIC_ARRAY::getError()
{
    return isError; // Возврат состояния флага ошибок
}

// Метод для проверки массива на упорядоченность
void DYNAMIC_ARRAY::isSort()
{
    for (int i = 0; i < cSize - 1; i++) // Итерация по элементам массива
    {
        if (cArray[i] > cArray[i + 1]) // Проверка на нарушение порядка
        {
            isSorted = false; // Установка флага неупорядоченности
        }
    }
}

void DYNAMIC_ARRAY::fillTheArray(const std::string fInput)
{
    isError = false; // Инициализируется флаг ошибки
    if (!cArray || cSize <= 0) // Проверяется корректность инициализации массива
    {
        isError = true; // Устанавливается ошибка, если массив не выделен или размер
некорректен
    }
    else
    {
        std::istringstream numberStream(fInput); // Создается поток для обработки входной
строки
        int number; // Объявляется переменная для хранения числа
        int index = 0; // Устанавливается начальный индекс

        while (numberStream >> number) // Чтение чисел из строки
        {
            // Проверка условий корректности числа и индекса
            if (number > MAX_NUMBER || index >= cSize || number <= 0)
            {
                isError = true; // Если данные некорректны, устанавливается ошибка
                break; // Прерывается цикл обработки
            }
            cArray[index] = number; // Число сохраняется в массиве
            ++index; // Переход к следующему элементу
        }

        // Если ошибок нет, заполняются оставшиеся элементы массива нулями
        if (!isError)
        {
            for (int i = index; i < cSize; i++)
            {
                cArray[i] = 0; // Остальные элементы заполняются нулями
            }
        }
    }
}

```

```

    }
}

void DYNAMIC_ARRAY::insertionSortArray()
{
    isSort(); // Проверяется, отсортирован ли массив
    if (isSorted == false) // Если массив не отсортирован
    {
        std::cout << "\nYour array: ";
        printArray([' ', '']); // Вывод исходного массива
        std::cout << "\nSorting...\n";

        for (int i = 1; i < cSize; i++) // Перебираются элементы массива
        {
            int curInt = cArray[i]; // Сохраняется текущий элемент
            int j; // Индекс для поиска позиции вставки

            // Сдвиг элементов вправо для освобождения места
            for (j = i - 1; j >= 0 && cArray[j] > curInt; j--)
            {
                cArray[j + 1] = cArray[j]; // Перемещение элементов
            }
            cArray[j + 1] = curInt; // Текущий элемент вставляется на правильное место

            printArrayInSort(j + 1, i + 1); // Отображение промежуточного состояния
массива
            printArrayInSortToFile(j + 1, i + 1); // Сохранение состояния в файл
        }
    }

    // Проверка, если массив отсортирован
    std::ofstream fileStream(FILE_NAME, std::ios::app | std::ios::ate);
    if (isSorted == true)
    {
        std::cout << "\nArray is sorted" << std::endl; // Вывод подтверждения сортировки
        fileStream << "\nArray is sorted" << std::endl; // Запись в файл
    }
    fileStream << std::endl;
    fileStream.close(); // Закрывается файл
}

void DYNAMIC_ARRAY::printArray(char fOpen, char fClose)
{
    for (int i = 0; i < cSize; i++) // Итерация по всем элементам массива
    {
        std::cout << fOpen << get(i) << fClose << ' '; // Вывод каждого элемента с
указанными символами
    }
    std::cout << std::endl; // Завершение строки вывода
}

void DYNAMIC_ARRAY::printArrayInSort(int fIndexOfInsertElement, int fStart)
{
    for (int i = 0; i < cSize; i++) // Проход по массиву
    {
        if (i < fIndexOfInsertElement) // Элементы до вставки
        {
            std::cout << get(i) << "    ";
        }
    }
}

```

```

        else if (i == fIndexOfInsertElement) // Текущий вставляемый элемент
        {
            std::cout << '[' << get(i) << '>';
        }
        else if (i > fIndexOfInsertElement && i < fStart) // Сдвинутые элементы
        {
            std::cout << " <- " << get(i);
        }
        else // Остальные элементы массива
        {
            std::cout << "    " << get(i);
        }
    }
    std::cout << std::endl; // Завершение строки
}

void DYNAMIC_ARRAY::printArrayInSortToFile(int fIndexOfInsertElement, int fStart,
std::string fFileName)
{
    std::ofstream fileStream(fFileName, std::ios::app | std::ios::ate); // Открытие файла
    в режиме добавления данных
    if (fileStream.is_open()) // Проверка на успешное открытие файла
    {
        fileStream << "Insertion step: " << fStart - 1 << ":    "; // Запись текущего шага
        сортировки

        for (int i = 0; i < cSize; i++) // Обход всех элементов массива
        {
            if (i < fIndexOfInsertElement) // Элементы до вставляемого
            {
                fileStream << get(i) << "    "; // Запись без изменений
            }
            else if (i == fIndexOfInsertElement) // Вставляемый элемент
            {
                fileStream << '[' << get(i) << '>'; // Выделение скобками для наглядности
            }
            else if (i > fIndexOfInsertElement && i < fStart) // Сдвинутые элементы
            {
                fileStream << " <- " << get(i); // Указание стрелки для сдвинутых
элементов
            }
            else // Остальные элементы массива
            {
                fileStream << "    " << get(i); // Обычная запись с отступом
            }
        }
        fileStream << std::endl; // Переход на новую строку после записи текущего
состояния
    }
    fileStream.close(); // Закрытие файла для завершения операции
}

```

## Выводы по результатам работы

В ходе выполнения работы были достигнуты следующие результаты:

1. Реализована программа для сортировки массива методом вставок. Программа полностью соответствует поставленным требованиям:
  - Принимает входные данные в виде числа элементов массива и самого массива.
  - Выполняет сортировку массива методом вставок.
  - Пошагово отображает изменения в массиве, сохраняя их в файл output.txt.
  - Не производит лишнего вывода, если массив изначально отсортирован.
2. Проведено тестирование программы на различных наборах данных. Результаты тестов подтвердили корректность работы алгоритма в каждом из случаев, включая граничные и нетипичные ситуации (например, массив длиной 1, массив с одинаковыми элементами).
3. Программа разработана с использованием языка C++ и протестирована на платформе консоли. Выбранный подход показал свою эффективность и соответствие требованиям надежности.
4. Файловая организация программы позволяет сохранять результаты работы для последующего анализа. Использование текстового файла output.txt делает вывод данных удобным и понятным.

### Заключение:

Работа продемонстрировала применение алгоритма сортировки вставками и его пошаговую визуализацию. Программа может быть использована как обучающий инструмент для изучения принципов работы алгоритмов сортировки.

## Руководство пользователя (приложение 1)

### Введение

Данная программа предназначена для демонстрации работы алгоритма сортировки массива методом вставок. Пользователь вводит массив данных, и программа пошагово сортирует его, отображая изменения в файле output.txt и на экране. Программа подходит для учебных целей и анализа работы алгоритма.

---

### Подготовка к работе

1. **Требования к системе:**
  - Компьютер с установленной ОС Windows, Linux или macOS.
  - Компилятор C++ (например, GCC, Clang или MSVC).
2. **Установка:**
  - Скачайте файл программы TaskVariantThreeClass.cpp.
  - Убедитесь, что в системе установлен компилятор.
3. **Компиляция программы:**
  - Для компиляции используйте следующую команду в терминале или командной строке:  
`g++ -o TaskVariantThreeClass TaskVariantThreeClass.cpp`
  - Выполненный код создаст исполняемый файл TaskVariantThreeClass.exe.
4. **Запуск программы:**
  - Введите:  
`./TaskVariantThreeClass`  
(для Windows: TaskVariantThreeClass.exe).

 TaskVariantThreeClass.exe	28.11.2024 17:57	Приложение	1 304 КБ
---	------------------	------------	----------

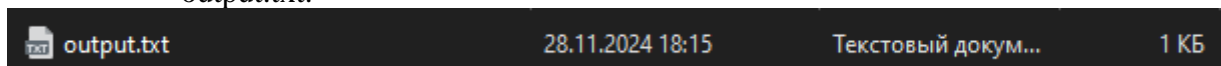
## Описание операций

### 1. Ввод данных:

- При запуске программа запросит два значения:
  - Число  $n$  — количество элементов массива.
  - Сам массив из  $n$  натуральных чисел, разделенных пробелами.

### 2. Процесс выполнения:

- Программа выполнит сортировку массива методом вставок.
- Изменения в массиве после каждой вставки будут записаны в файл output.txt.



### 3. Вывод результатов:

- Если массив был изначально отсортирован, файл output.txt добавиться информация .
- Если происходили изменения, в файле output.txt будут записаны промежуточные состояния массива.

---

## Аварийные ситуации

### 1. Некорректный ввод:

- Если пользователь вводит нечисловое значение размера , программа завершится с ошибкой.
- Решение: перезапустите программу и введите данные корректно.

### 2. Выход за пределы допустимых значений:

- Программа проверяет, чтобы  $n$  было в пределах от 1 до 100. Если значение выходит за эти рамки, будет выведено сообщение об ошибке.
- Решение: введите  $n$  в пределах допустимого диапазона.

### 3. Ошибка записи в файл output.txt:

- Если файл не может быть создан (например, из-за отсутствия прав), программа завершится с ошибкой.
- Решение: убедитесь, что у вас есть права на запись в папку, где находится программа.

### 4. Прерывание работы программы:

- Если программа завершится преждевременно, файл output.txt может содержать только частичные результаты.
- Решение: перезапустите программу с корректным вводом данных.

---

Данное руководство поможет быстро освоить использование программы и эффективно справляться с возможными проблемами.

## Руководство программиста (приложение 2)

### Назначение и условия применения программы

#### 1.1. Назначение программы

Программа предназначена для демонстрации работы алгоритма сортировки массива методом вставок. Она используется для обучения и анализа принципов работы данного алгоритма.

#### 1.2. Функции, выполняемые программой

Прием входных данных: числа массива и его размер.

Сортировка массива методом вставок.

Пошаговый вывод промежуточных состояний массива в файл output.txt и на экран.

### **1.3. Условия, необходимые для выполнения программы**

#### **1.3.1. Объем оперативной памяти**

##### **Минимальная конфигурация:**

- 512 МБ оперативной памяти.
- 50 МБ свободного дискового пространства.

##### **Рекомендуемая конфигурация:**

- 2 ГБ оперативной памяти.
- 100 МБ свободного дискового пространства.

#### **1.3.2. Требования к составу периферийных устройств**

- Устройство ввода (клавиатура).
- Монитор для отображения результата выполнения программы.

#### **1.3.3. Минимальные требования**

- Операционная система: Windows 7 и выше, Linux, macOS.
- Компилятор C++ (например, GCC 5.0+, Clang 6.0+, MSVC).
- 1 ГБ свободного места на диске.

#### **1.4. Требования к персоналу (программисту)**

- Знание основ алгоритмов и структур данных.
- Опыт работы с языком программирования C++.
- Умение работать с текстовыми редакторами и компиляторами.

---

### **Описание среды разработки**

- Для разработки программы использовалась следующая среда:
- Язык программирования: C++ (стандарт C++17).
- Компилятор: GCC версии 14.2.0.
- Среда разработки: NeoVim (сборка LazyVim).
- Операционная система: Windows 10.
- Дополнительно использовались:
- Инструменты для тестирования: терминал для запуска программы, файловый редактор для анализа output.txt.

---

### **Входные и выходные данные.**

#### **Входные данные**

- Целое число  $n$  ( $1 \leq n \leq 100$ ) — количество элементов массива.
- Последовательность из  $n$  натуральных чисел, каждое из которых не превышает  $10^9$ .

#### **Выходные данные**

- Пошаговый вывод на экран состояния массива
- Файл output.txt, содержащий пошаговый вывод состояния массива на каждой итерации сортировки.

### **Описание логики работы программы**

#### **Логика работы программы**

- Программа считывает входные данные.
- Производится сортировка массива методом вставок.
- Каждой итерация вставки выводится на экран и записывается в файл output.txt.
- После завершения сортировки программа предлагает повторить ввод.