Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

Электротехнический факультет

Кафедра “Информационные технологии и автоматизированные системы”

**ОТЧЕТ**

**по учебной практике**

**Часть I**

Визуализация Быстрой сортировки

Выполнил:

студент группы РИС-23-3б

Блинов А. Е.

Проверил:

Доцент кафедры ИТАС

Петренко А. А.

**Часть II**

Анализ приёма абитуриентов 2024 г. кафедры ИТАС

Выполнил:

студент группы РИС-23-3б

Блинов А. Е.

Проверила:

Доцент кафедры ИТАС

Полякова О. А

Пермь 2024

СОДЕРЖАНИЕ

[Часть I. Визуализация быстрой сортировки 3](#_Toc175695397)

[1.1 Анализ алгоритма быстрой сортировки 3](#_Toc175695398)

[1.2 Разработка кода на языке программирования С++ 5](#_Toc175695399)

[1.3 Написание кода быстрой сортировки 6](#_Toc175695400)

[Часть II. Анализ приёма абитуриентов кафедры ИТАС 8](#_Toc175695401)

[2.1 Анализ проходного балла абитуриентов кафедры ИТАС по направлениям: 09.03.04 «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 15.03.06 «Мехатроника и робототехника» 8](#_Toc175695402)

[2.2 Система выявления приоритетов и подачи заявлений абитуриентов на кафедру ИТАС 13](#_Toc175695403)

[2.3 Анализ работы по взаимодействию студентов кафедры ИТАС с абитуриентами 15](#_Toc175695404)

[ЗАКЛЮЧЕНИЕ 17](#_Toc175695405)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc175695406)

[ПРИЛОЖЕНИЕ А 19](#_Toc175695407)

[ПРИЛОЖЕНИЕ Б 21](#_Toc175695408)

[ПРИЛОЖЕНИЕ В 25](#_Toc175695409)

ВВЕДЕНИЕ

Быстрая сортировка (Quicksort) – это известный алгоритм сортировки, основанный на принципе "разделяй и властвуй"[4] и славящийся своей эффективностью. Выбирая опорную точку и разбивая массив вокруг нее, алгоритм помещает опорную точку в правильную отсортированную позицию. При сортировке n элементов эта техника достигает в среднем nlogn сравнений, что делает ее очень быстрой.

Объект исследования: процесс сортировки массива.

Предмет исследования: программа, реализующая алгоритм быстрой сортировки на языке программирования С++.

Цель: реализовать программу на языке программирования С++, демонстрирующую работу алгоритма быстрой сортировки, провести анализ её эффективности.

Задачи:

1.Провести анализ алгоритма быстрой сортировки.

2.Разработать программу на языке программирования С++, реализующую алгоритм.

# 

# Часть I. Визуализация быстрой сортировки

# 1.1 Анализ алгоритма быстрой сортировки

Суть алгоритма заключается в подходе "разделяй и властвуй": он разбивает задачи на более мелкие подзадачи, решает их и объединяет решения. Вначале выбирается опорный элемент, что приводит к разбиению массива на подмассивы. Один подмассив содержит значения, меньшие или равные опорной точке, а другой – большие. Рекурсивно Quicksort сортирует эти подмассивы. Разбиение и сортировка продолжаются до тех пор, пока в каждом подмассиве не останутся отдельные элементы.

Существует несколько вариантов выбора опорного элемента:

* Первый элемент;
* Последний элемент;
* Случайный элемент;
* Элемент по середине.

Алгоритм разделения:

Начинаем с крайнего левого элемента и отслеживаем индексы меньших (или равных) элементов как **i**. Если во время перемещения мы находим элемент меньшего размера, мы меняем текущий элемент на arr[i]. В противном случае мы игнорируем текущий элемент.

**Вывод**

Быстрая сортировка предоставляет эффективное и быстрое решение для сортировки массивов данных. Благодаря применению метода "разделяй и властвуй" и выбору опорного элемента, алгоритм Quicksort способен эффективно справляться с поставленной задачей даже для больших массивов.

# 1.2 Разработка кода на языке программирования С++

Алгоритм быстрой сортировки

В массиве выбирается некоторый элемент, называемый опорным[5]. Затем он помещается в то место массива, где ему полагается быть после упорядочивания всех элементов. В процессе отыскания подходящего места для опорного элемента производятся перестановки элементов так, что слева от них находятся элементы, меньшие опорного, справа – большие (предполагается, что массив сортируется по возрастанию).

Тем самым массив разбивается на две части:

* неотсортированные элементы слева от опорного элемента;
* неотсортированные элементы справа от опорного элемента.

Чтобы отсортировать эти два меньших подмассива, алгоритм рекурсивно вызывает сам себя.

Если требуется сортировать больше одного элемента, то нужно:

* выбрать в массиве опорный элемент, чаще всего — крайний слева;
* переупорядочить массив, помещая опорный элемент на его окончательное место;
* отсортировать рекурсивно элементы слева от опорного;
* отсортировать рекурсивно элементы справа от опорного.
  1. **Написание кода быстрой сортировки**

В программе подключается библиотека iostream[1], которая необходима для ввода и вывода данных. Рассмотрим функции, использующиеся в данной программе:

1. void quickSort(int\* arr, int startIndex, int endIndex)

Функция сортировки массива

Параметры: int\* arr – массив целых чисел, который нужно отсортировать, int startIndex – начальный индекс, int endIndex – конечный индекс.

2. int main()

Главная функция программы. Создает массив целых чисел, вызывает функцию quickSort для его сортировки, затем выводит отсортированный массив на экран. Возвращает 0, что указывает на успешное завершение программы.

В программе подключается всего лишь одна библиотека, что говорит о том, что для реализации быстрой сортировки нам не нужны познания каких-либо сложных библиотек языка С++.

**1.4 Визуализация быстрой сортировки**

Для разработки визуализации быстрой сортировки используем библиотеку SFML. SFML — это библиотека, предназначенная для разработки приложений с использованием графики, аудио, видео и сетевых функций.

Использование SFML для визуализации сортировки в лабораторной работе позволяет эффективно продемонстрировать, как работает алгоритм, обеспечивает гибкость и простоту в освоении проекта в будущем.

**Вывод**

Был разработан код, реализующий быструю сортировку, а также была реализована визуализация работы данной сортировки.

**Часть II. Анализ приёма абитуриентов кафедры ИТАС**

* 1. **Анализ проходного балла абитуриентов кафедры ИТАС по направлениям: 09.03.04 «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 15.03.06 «Мехатроника и робототехника»**

Во время учебной практики студентами первого курса проведена работа по сбору и анализу статистических данных о приеме абитуриентов на направления 09.03.04 «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 15.03.06 «Мехатроника и робототехника».

Для того чтобы помочь абитуриентам в выборе специальностей кафедра ИТАС выделила группу студентов, которые предоставляли подробную информацию о направлениях подготовки, учебных дисциплинах и профессиях, которые абитуриенты смогут освоить по окончании четырех курсов обучения на кафедре. Помощь заключалась в прямых телефонных контактах студентов кафедры ИТАС с абитуриентами.

В процессе общения с абитуриентами стало очевидно, что многие из них сталкиваются с непростым выбором будущей специальности. Некоторым из них недоставало информации о выбранных направлениях, и они принимали решения наугад, основываясь на поверхностных знаниях. Телефонные консультации помогли многим абитуриентам лучше понять суть направлений, на которые они собирались подать документы, и возможные перспективы, которые открываются перед ними после окончания учебы. Оказалось, что почти 20% абитуриентов, с которыми контактировали студенты кафедры ИТАС, намерены поменять выставленные приоритеты. Сложность в выборе приоритетов могла возникнуть из-за большого количества направлений, связанных с информационными технологиями, а также из-за недостаточного понимания системы приоритетов.

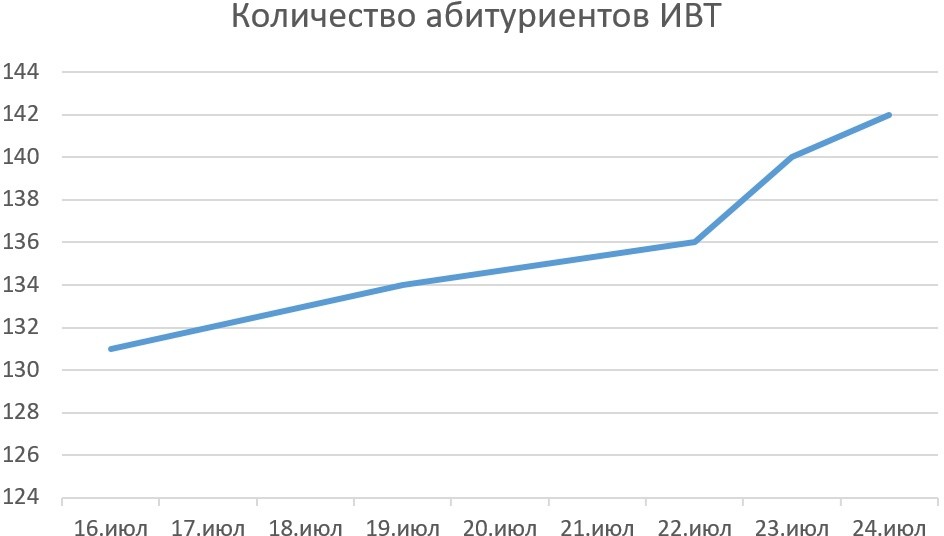
Абитуриенты с высокими баллами стремятся поступать в ведущие вузы Санкт-Петербурга и Москвы. В частности, четверть абитуриентов с баллами выше 260, рассматривают именно эти города для продолжения своего образования. Именно поэтому ПНИПУ является для таких абитуриентов вариантом запаса.

В ходе телефонных консультаций абитуриенты, выбравшие первым приоритетом направление «Программная инженерия», добавляли вторым приоритетом направление «Информатика и вычислительная техника» как второй вариант на случай, если не пройдут по первому. Особое внимание абитуриенты проявили к уникальному решению кафедры ИТАС, которое заключается в совместном обучении на двух направлениях – ПИ и ИВТ – в течение первых двух курсов. Анализ звонков приемной комиссии выявил, что абитуриенты активно интересовались этой возможностью. Важным моментом для них стало то, что обе специальности получают одинаковую базовую подготовку по программированию и другим дисциплинам. Абитуриенты также проявили интерес к возможности сменить направление в конце второго курса. Этот фактор оказался особенно привлекательным для тех, кто не проходил по баллам на «Программную инженерию».

На момент начала обзвона, среди абитуриентов, рассматривавших Пермский национальный исследовательский политехнический университет, 57 человек выбрали «Программная инженерия» (ПИ) в качестве первого приоритета, 13 человек отдали предпочтение «Информатике и вычислительной технике» (ИВТ), а направление «Мехатроника и робототехника» (МИР) оказалось на первом месте только у двух абитуриентов. Эти данные охватывают диапазон баллов от 217 до 288, и всего проанализированы 176 абитуриента. Из них 41 человек изначально планировали поступать в другие вузы, в то время как 40 человек выбирали ПНИПУ, но на другие кафедры.

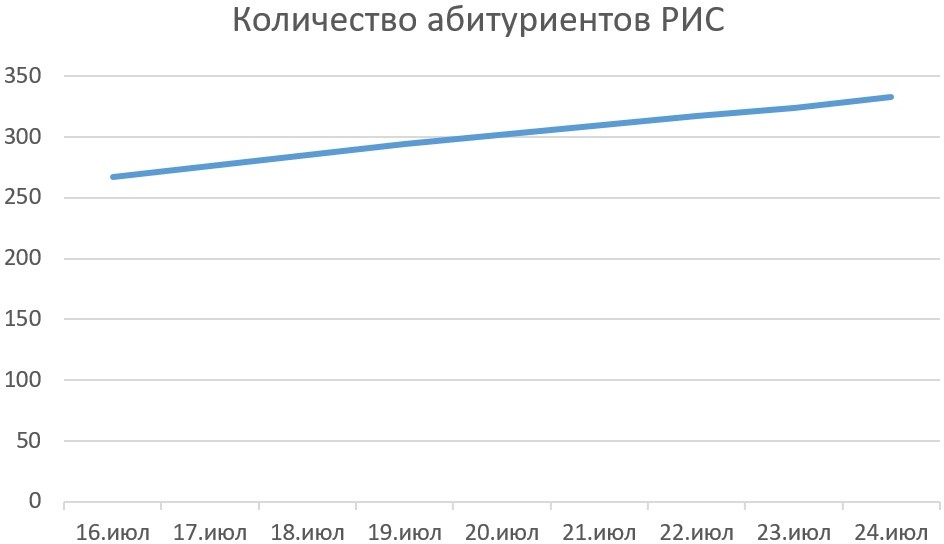
По мере проведения консультаций ситуация начала меняться. К 22 июля количество абитуриентов, выбравших ПИ в качестве первого приоритета, возросло до 66 человек, ИВТ до 19, а МИР — до 7 человек. На этом этапе 10 абитуриентов уже подали оригиналы документов в ПНИПУ, что свидетельствует о серьезности их намерений, а 16 человек изменили свои приоритеты, выбрав направления кафедры ИТАС.

К моменту завершения обзвона наблюдался дальнейший рост интереса к направлениям кафедры ИТАС. Количество абитуриентов, выбравших ПИ в качестве первого приоритета, увеличилось до 76 человек, ИВТ — до 20, а МИР остался на уровне 7 человек. В этот период 34 абитуриента уже подали оригиналы документов, закрепив свой выбор. Важно отметить, что 35 человек, которые изначально рассматривали другие направления или кафедры, изменили свои приоритеты в пользу кафедры ИТАС.



*Рисунок 2.1 - График подачи заявлений на направление ИВТ как*

*высший приоритет*



*Рисунок 2.2 - График подачи заявлений на направление ПИ как высший*

*приоритет*

Значительно меньший прирост абитуриентов, выбравших направление «Информатика и вычислительная техника» высшим приоритетом по сравнению с приростом тех, кто выбрал направление «Программная инженерия» (Рисунок 2.1), объясняется тем, что большинство либо сразу знали, что хотят поступать на ИВТ и поэтому не меняли свой выбор, либо для них ИВТ является запасным вариантом, и его ставят вторым приоритетом после ПИ. Количество абитуриентов, выбравших ИВТ как второй приоритет после ПИ, составляет 67 человек, что примерно 9-10% людей, подавших документы на это направление. Это свидетельствует о том, что консультации помогли абитуриентам лучше разобраться в связанных направлениях и сделать более осознанный выбор.

Чаще всего вместе с Пермским национальным исследовательским политехническим университетом для поступления рассматривают Высшую Школу Экономики (ВШЭ), Пермский государственный национальный исследовательский университет (ПГНИУ) и Уральский Федеральный университет (УрФУ). Примерно 10% абитуриентов рассматривают ВШЭ, причем все они имеют баллы выше 240. ПГНИУ рассматривают около 5% абитуриентов, а УрФУ – 4%. Эти данные отражают разнообразие выбора абитуриентов и конкурентоспособность нашего университета среди других ведущих образовательных институтов региона.

В итоге, около 50 человек выразили желание скорректировать свои приоритеты: кто-то решил выбрать направления кафедры ИТАС в качестве основного, кто-то добавил дополнительные направления с этой кафедры для подстраховки, а кто-то поменял местами уже выбранные направления. Этот результат свидетельствует об эффективности проведенной работы и значимости консультационной поддержки, оказанной студентами ПНИПУ.

## **2.2 Система выявления приоритетов и подачи заявлений абитуриентов на кафедру ИТАС**

Студентам кафедры ИТАС поручили задачу по привлечению в Пермский национальный исследовательский политехнический университет как можно большего числа способных и заинтересованных в изучении информационных технологий абитуриентов.

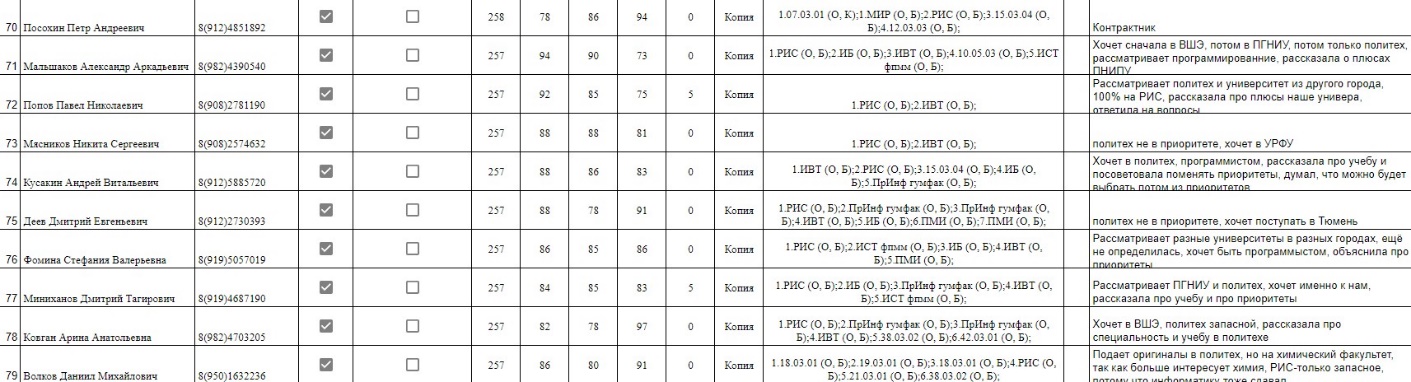
На первом собрании консультантов кафедры ознакомили с рекомендациями о том, что лучше донести информацию студентам и как наиболее продуктивно выстраивать разговор, а также распределили работу по сменам. В утренней и дневной смене работали по 6 человек. Для звонков абитуриентам консультантам выделили служебные телефоны и помещения. Звонки начинались не ранее 12:00 и не позднее 17:30, чтобы не доставлять дискомфорт абитуриентам.

В качестве инструментов выбраны: Google таблицы для структурированного представления информации и учета абитуриентов, с которыми состоялся разговор, Telegram для удобной коммуникации между консультантами и руководителем. Таблица содержит: ФИО, номер телефона, количество баллов, приоритеты для поступления, комментарии консультантов по содержанию разговору с абитуриентом, а также информацию для пометки процесса работа. Именно такая система учета и анализа данных увидала эффективность работы, исключила повторяющиеся звонки и облегчила делегирование задач для большего количества звонков абитуриентов.

К работе приступили 16.07.2024. Консультанты записывали часть разговоров, для дальнейшего обсуждения с коллегами и руководством.

На следующий день провели собрание, для прослушивания уже совершенных разговоров и на их основе построены последующее обсуждение сильных и слабых моментов, выявлены удачные подходы к работе, а также консультанты делились опытом друг с другом.

На основе полученной информации и опыта, далее звонки абитуриентам проходили более успешно и эффективно. Работа завершена 25.07.2024.



*Рисунок 2.3 - Структура таблицы анализа*

## **2.3 Анализ работы по взаимодействию студентов кафедры ИТАС с абитуриентами**

Деятельность по обзвону абитуриентов принесла пользу поступающим: консультанты кафедры ИТАС помогли определиться с выбором приоритетов по направлениям, если такая помощь действительно нужна; напомнили про ключевые даты подачи согласия и документов; познакомили с вузом и кафедрой. Вместе с этим собраны данные по каждому абитуриенту благодаря комментарию к звонку. По итогам проведенной работы сформированы следующие статистические данные:

Статистика в отношении абитуриентов, поступивших на направление ПИ:

- Изначально собирались поступать на направление: 25 человек (46%). К таким относятся Сурнин Кирилл (275 баллов ЕГЭ) и Вицев Максим (270 баллов).

- Изначально стоял приоритет на другое направление ПНИПУ, но поменяли основное направление на ПИ: 10 (19%). Морозова Надежда (261 балл) выбирала между направлениями ИВТ и ПИ, после телефонной консультации решила поступать на ПИ.

- Выбирали между несколькими ВУЗами, включая ПНИПУ: 13 (24%). Например, Меньшиков Арсений (275 баллов) и Иванов Семен (263 балла) выбирали между несколькими вузами и в итоге поступили в ПНИПУ.

- Не удалось связаться: 6 (11%)

Статистика в отношении абитуриентов, поступивших на направление ИВТ:

- Изначально собирались поступать на направление: 19 человек (45%). Оглезнев Никита (259 баллов) сразу подал оригиналы на кафедру ИТАС

- Изначально стоял приоритет на другое направление ПНИПУ, но поменяли основное направление на ИВТ: 4 (10%). Среди таких абитуриентов: Балтаева Алина (250 баллов)

- Выбирали между несколькими ВУЗами, включая ПНИПУ: 4 (10%). Балтаева Алина также выбирала между несколькими вузами и выбрала ПНИПУ.

- Не удалось связаться: 15 (35%)

Статистика в отношении абитуриентов, поступивших на направление МИР:

- Изначально собирались поступать на направление: 20 человек (47%). Васильев Сергей (263 балла) сразу решил поступать на это направление.

- Изначально стоял приоритет на другое направление ПНИПУ, но поменяли основное направление на МИР: 10 (23%). Старинов Артем (202 балла) выбирал между направлениями ПНИПУ.

- Выбирали между несколькими ВУЗами, включая ПНИПУ: 9 (21%). Например, Чуян Даниил (223 балла) выбирал между УрФУ и ПНИПУ.

- Не удалось связаться: 5 (11%)

# ЗАКЛЮЧЕНИЕ

В ходе разработки программы быстрой сортировки был изучен алгоритм работы сортировки. Была реализована программа на языке С++, демонстрирующая работу быстрой сортировки. Был проведен анализ эффективности данной программы. Были выполнены поставленные задачи по проведению анализа алгоритма сортировки, а также была разработана программа, реализующая алгоритм.

Также, консультанты за время своей работы сделали выводы, что количество подавших заявления на поступление на данные направления также увеличилось, что говорит о повышенном интересе к данным специальностям. На всех направлениях наблюдается повышение проходных баллов по сравнению с прошлым годом, из чего следует, что можно предполагать более высокий уровень навыков абитуриентов, чем ранее ожидалось. В целом, результаты работы комиссии по приему на обучение свидетельствуют о хорошо выполненных консультациях по поступлению на учебные программы. Поэтому работу, проведенную студентами кафедры ИТАС, можно считать успешной.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб.: Питер, 2021. – 263 с.
2. Quick Sort Algorithm // Scaler Topics [Электронный ресурс]. URL – <https://www.scaler.com/topics/data-structures/quick-sort-algorithm/> (дата обращения: 25.06.2024)
3. Tutorials // Learn SFML [Электронный ресурс]. URL – <https://www.sfml-dev.org/learn.php> (дата обращения: 25.06.2024)
4. 114 стандартных алгоритмов // RaidoProg [Электронный ресурс]. URL – <https://radioprog.ru/post/1460> (дата обращения: 25.06.2024)
5. Быстрая сортировка // Хабр [Электронный ресурс]. URL –<https://habr.com/ru/companies/otus/articles/524948/> (дата обращения: 25.06.2024)
6. Quick Sort Algorithm // GeeksForgeeks [Электронный ресурс]. URL –<https://www.geeksforgeeks.org/quick-sort-algorithm/> (дата обращения: 25.06.2024)

**ПРИЛОЖЕНИЕ А**

**Код быстрой сортировки С++**

#include <iostream>

using namespace std;

static void quickSort(int\* arr, int startIndex, int endIndex) {

if (startIndex < endIndex) {

int leftIndex = startIndex, rightIndex = endIndex;

int pivotElem = arr[(leftIndex + rightIndex) / 2];

do {

while (arr[leftIndex] < pivotElem) leftIndex++;

while (arr[rightIndex] > pivotElem) rightIndex--;

if (leftIndex <= rightIndex) {

int tmp = arr[leftIndex];

arr[leftIndex++] = arr[rightIndex];

arr[rightIndex--] = tmp;

}

} while (leftIndex < rightIndex);

quickSort(arr, startIndex, rightIndex);

quickSort(arr, leftIndex, endIndex);

}

return;

}

int main() {

system("chcp 1251 > Null");

srand(time(0));

int size = 10;

int\* arr = new int[size] {};

for (int i = 0; i < size; i++) { arr[i] = rand() % 101 - 10; }

cout << "\nBefore sort: ";

for (int i = 0; i < size; i++) { cout << arr[i] << " "; }

cout << endl;

quickSort(arr, 0, size - 1);

cout << "\nAfter sort: ";

for (int i = 0; i < size; i++) { cout << arr[i] << " "; }

cout << endl;

return 0;

}

**ПРИЛОЖЕНИЕ Б**

**Визуализация SFML - код**

Код:

#include <SFML/Graphics.hpp>

#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>

constexpr float PI = 3.14159265f;

static void displayArray(sf::RenderWindow& window, std::vector<int>& arr, const std::string& label, std::pair<int, int> swapIndices) {

window.clear(sf::Color::Black);

sf::Font font;

font.loadFromFile("arialmt.ttf");

sf::Text labelText;

labelText.setFont(font);

labelText.setCharacterSize(20);

labelText.setString(label);

labelText.setFillColor(sf::Color::White);

labelText.setStyle(sf::Text::Bold);

labelText.setPosition((window.getSize().x - labelText.getLocalBounds().width) / 2.f, 10.f);

window.draw(labelText);

sf::Text text;

text.setFont(font);

text.setCharacterSize(20);

text.setFillColor(sf::Color::White);

float startX = (window.getSize().x - arr.size() \* 40) / 2.0f;

float startY = 100.0f;

float offsetX = 40.0f;

float offsetY = 30.0f;

float arrowSize = 8.0f;

for (size\_t i = 0; i < arr.size(); i++) {

text.setString(std::to\_string(arr[i]));

text.setPosition(startX + i \* offsetX, startY);

window.draw(text);

}

if (swapIndices.first != -1 && swapIndices.second != -1) {

float x1 = startX + swapIndices.first \* offsetX + offsetX / 2;

float x2 = startX + swapIndices.second \* offsetX + offsetX / 2;

sf::VertexArray line(sf::Lines, 2);

line[0].position = sf::Vector2f(x1, startY);

line[1].position = sf::Vector2f(x2, startY);

float angle = atan2(0, x2 - x1);

sf::ConvexShape arrow1;

arrow1.setPointCount(3);

arrow1.setPoint(0, sf::Vector2f(x1, startY));

arrow1.setPoint(1, sf::Vector2f(x1 + arrowSize \* cos(angle - PI / 6), startY + arrowSize \* sin(angle - PI / 6)));

arrow1.setPoint(2, sf::Vector2f(x1 + arrowSize \* cos(angle + PI / 6), startY + arrowSize \* sin(angle + PI / 6)));

arrow1.setFillColor(sf::Color::Red);

sf::ConvexShape arrow2;

arrow2.setPointCount(3);

arrow2.setPoint(0, sf::Vector2f(x2, startY));

arrow2.setPoint(1, sf::Vector2f(x2 - arrowSize \* cos(angle - PI / 6), startY - arrowSize \* sin(angle - PI / 6)));

arrow2.setPoint(2, sf::Vector2f(x2 - arrowSize \* cos(angle + PI / 6), startY - arrowSize \* sin(angle + PI / 6)));

arrow2.setFillColor(sf::Color::Red);

window.draw(line);

window.draw(arrow1);

window.draw(arrow2);

}

window.display();

}

static void quickSort(std::vector<int>& arr, int startIndex, int endIndex, sf::RenderWindow& window) {

if (startIndex < endIndex) {

int pivot = arr[endIndex];

int i = startIndex - 1;

for (int j = startIndex; j < endIndex; j++) {

if (arr[j] < pivot) {

i++;

std::swap(arr[i], arr[j]);

displayArray(window, arr, "Sorting...", std::make\_pair(i, j));

sf::sleep(sf::milliseconds(1000));

}

}

std::swap(arr[i + 1], arr[endIndex]);

displayArray(window, arr, "Sorting...", std::make\_pair(i + 1, endIndex));

sf::sleep(sf::milliseconds(1000));

quickSort(arr, startIndex, i, window);

quickSort(arr, i + 2, endIndex, window);

}

}

int main() {

sf::RenderWindow window(sf::VideoMode(800, 600), "Quick Sort Visualization");

window.setFramerateLimit(10);

int size = 10;

std::vector<int> arr(size);

for (int i = 0; i < size; i++) { arr[i] = rand() % 101 - 15; }

displayArray(window, arr, "Before Sort", std::make\_pair(-1, -1));

sf::sleep(sf::seconds(1));

quickSort(arr, 0, arr.size() - 1, window);

displayArray(window, arr, "After Sort", std::make\_pair(-1, -1));

while (window.isOpen()) {

sf::Event event;

while (window.pollEvent(event)) {

if (event.type == sf::Event::Closed) {

window.close();

}

}

}

return 0;

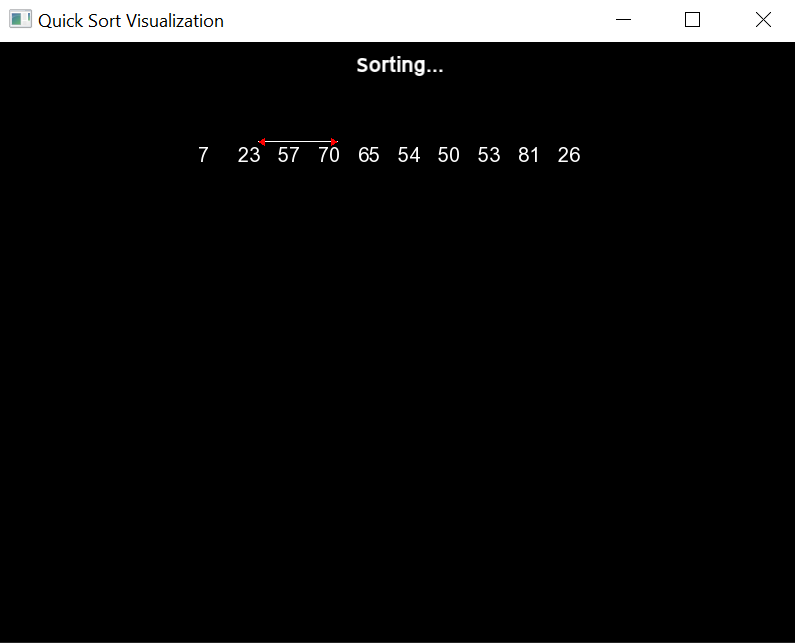
}

**ПРИЛОЖЕНИЕ В**

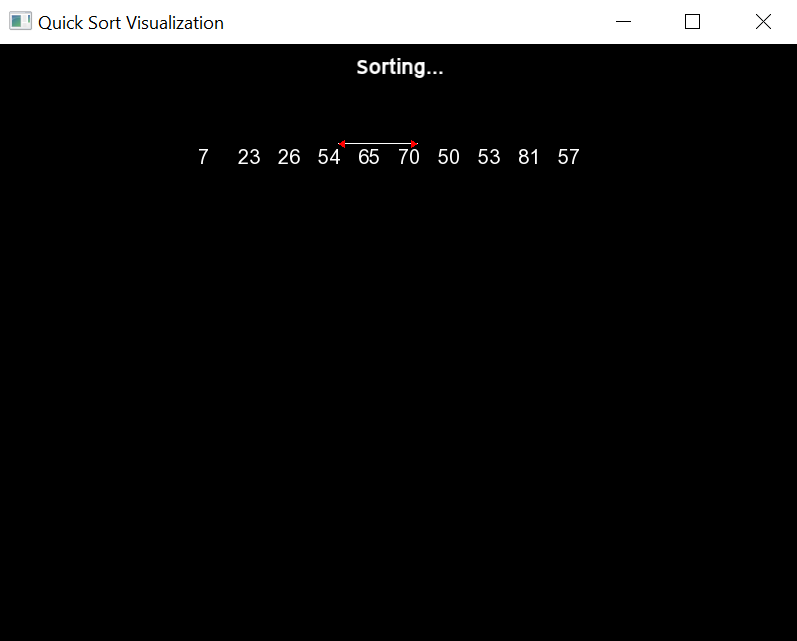
**Визуализация SFML**



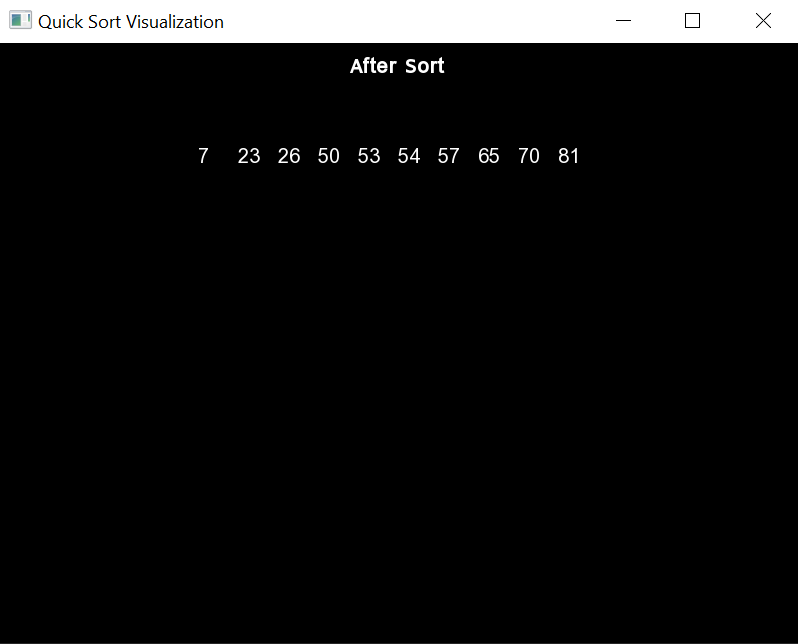
*Рисунок 1 – изначальный массив*



*Рисунок 2 – процесс сортировки массива*

**

*Рисунок 2.1 – процесс сортировки массива*

**

*Рисунок 3 – отсортированный массив*