Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**КУРСОВАЯ РАБОТА**

**Моделирование решения Кубика Рубика 3x3**

по дисциплине «Структуры данных»

Выполнили

студенты гр. 5151003/30002 Д. А. Плотников

И. А. Штарев

Руководитель

асс. преподавателя И. Д. Панков

«07» июня 2024 г.

Санкт-Петербург

2024

**СОДЕРЖАНИЕ**

1. [ВВЕДЕНИЕ 3](#_Toc167918191)
2. [1. ОПИСАНИЕ РЕШЕНИЯ 4](#_Toc167918192)
3. [1.1. ГРАФИЧЕСКАЯ ЧАСТЬ 5](#_Toc167918193)

# ВВЕДЕНИЕ

Кубик Рубика – культовая головоломка, покорившая умы миллионов людей по всему миру. С виду простая конструкция, состоящая из цветных квадратов, таит в себе невероятное количество комбинаций и алгоритмов, позволяющих достичь заветного результата – собрать все цвета на своих гранях.

Данная курсовая работа посвящена разработке алгоритма, способного решить кубик Рубика, состояние которого задаётся пользователем. В работе рассматриваются различные подходы к решению головоломки, анализируются их преимущества и недостатки.

В рамках исследования был разработан и реализован алгоритм, основанный на ЗАКОНЧИТЬ.

Практическая значимость данной работы заключается в создании универсального решения для сборки кубика Рубика, доступного для использования пользователями с различным уровнем подготовки. Программа может быть использована как в развлекательных целях, так и в качестве инструмента для изучения алгоритмов решения головоломки.

# 1. ОПИСАНИЕ РЕШЕНИЯ

Приложение «Rubik Solver» используется для решения кубика Рубика с состоянием, заданным пользователем из файла или вручную (через интерфейс ввода состояния кубика).

После ввода состояния кубика пользователю предлагается нажать на кнопку «решить» для сохранения в файл последовательности поворотов, необходимых для решения заданного им кубика. Решение можно загрузить и из заранее подготовленного файла с последовательностью поворотом.

После задания состояния кубика и получения необходимой последовательности действий пользователю предлагается просмотреть решение пошагово, вызывая каждый новый шаг очередным нажатием кнопки «следующий шаг». Повороты кубика на экране сопровождаются выведением информации о соответствующем повороте.

После просмотра всех шагов пользователю сообщается, что решение завершено.

# 1.1. ГРАФИЧЕСКАЯ ЧАСТЬ

Для отрисовки графической части разработанного решения была использована библиотека OpenGL в связке со вспомогательными библиотеками GLFW и GLAD. Самый очевидный ход при отрисовке окон в OpenGL – одинарная буферизация. При ней все изменения в кадре сразу же отображаются в экранном буфере, что может привести к мерцанию экрана. Для разрешения этих ситуаций в решении использована двойная буферизация. При ней вместо одного буфера кадра используются два:

* Задний буфер (back buffer): скрытый от пользователя буфер, где происходит рисование объектов.
* Передний буфер (front buffer): буфер, который в данный момент отображается на экране.

Все операции рисования происходят в заднем буфере, невидимом для пользователя. Как только кадр полностью отрисован в заднем буфере, происходит мгновенный обмен (swap) между передним и задним буферами. Задний буфер становится видимым, а передний становится доступным для рисования следующего кадра. При двойной буферизации достигается целостность кадров: пользователь видит только полностью отрисованные кадры.

# 1.2. ПРЕДСТАВЛЕНИЕ КУБИКА И ЕГО ОТРИСОВКА

Кубик представлен структурой Cube, хранящей в себе шесть статических массивов длины 9 и типа color (перечисление, отвечающее за цвет). При запуске программы экземпляр этой структуры создается в виде глобальной переменной. Хотя использование глобальных переменных и является небезопасной практикой, в данной случае решение об их использовании является оправданным, так как это обеспечивает удобный доступ к редактированию кубика из любого места программы.

Отрисовка кубика происходит в изометрии: пользователю видны три смежных стороны, чего вполне достаточно для полноценного отображения любого поворота. За отрисовку кубика отвечают функции drawSide и drawSquare. Первая принимает на вход информацию о том, какую сторону следует отрисовать (левую, правую или верхнюю) и массив цветов, соответствующих этой стороне. Вторая вызывается первой девять раз, по каждому на каждую клетку. Эта функция принимает на вход координаты угла клетки и три компоненты цвета этой клетки.