**СОДЕРЖАНИЕ**

1 ЛИСТ ЗАДАНИЯ………………………………………………………………1

2 ВВЕДЕНИЕ……………………………………………………………………1

3 ОБЗОР ЛИТЕРАТУРЫ………………………………………………………1

3.1 Обзор методов и алгоритмов решения поставленной задачи….………1

4 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ………………………………1

4.1. Структура входных и выходных данных………………………………1

4.2. Разработка диаграммы классов. ………………………………………1

4.3. Описание классов. …………………………………………………….1

5 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

5.1 Разработка схем алгоритмов…………………………………………….1

5.2 Разработка алгоритмов…………………………………………….1

6 РЕЗУЛЬТАТЫ РАБОТЫ…………………………………………….1

7 ЗАКЛЮЧЕНИЕ………………………………………………………1

8 ЛИТЕРАТУРА………………………………………………………1

9 ПРИЛОЖЕНИЯ………………………………………………………1

**2 ВВЕДЕНИЕ**

C++ является одним из наиболее популярных языков программирования, который широко используется для разработки приложений, включая системное программное обеспечение, игры, веб-приложения и многое другое. Он был разработан в 1979 году Бьярном Страуструпом как расширение языка программирования C и предоставляет разработчикам мощные инструменты для создания различных типов приложений. Он поддерживает процедурное, объектно-ориентированное и обобщенное программирование, что делает его очень гибким и универсальным языком.

ООП - это парадигма программирования, которая позволяет организовать код в виде объектов, которые взаимодействуют друг с другом. Основными принципами ООП являются наследование, полиморфизм и инкапсуляция.

Наследование позволяет создавать иерархию классов, где дочерние классы наследуют свойства и методы от родительских классов. Это позволяет повторно использовать код и создавать более абстрактные и специализированные классы.

Полиморфизм позволяет использовать один и тот же интерфейс для работы с разными типами объектов. Это достигается с помощью виртуальных функций и переопределения методов в дочерних классах.

Инкапсуляция позволяет скрыть детали реализации класса и предоставить только необходимый интерфейс для работы с ним. Это достигается с помощью модификаторов доступа, таких как public, private и protected.

C++ также поддерживает другие концепции ООП, такие как абстрактные классы, интерфейсы, шаблоны и множественное наследование. Эти возможности позволяют разработчикам создавать гибкие и модульные приложения.

**3 ОБЗОР ЛИТЕРАТУРЫ**

**3.1 Обзор методов и алгоритмов решения поставленной задачи**

Для разработки десктопной игры «Шахматы» необходимо написать игровую логику шахмат и создать пользовательский интерфейс, который включает в себя саму доску с фигурами, отображение возможных ходов для каждый фигуры, очередь хода, возможность для пользователя выбрать режим игры, отображение времени, выделенного на партию.

Для выполнения графического интерфейса была выбрана Qt — библиотека классов C++ и набор инструментального программного обеспечения для создания кросс-платформенных приложений с графическим интерфейсом (GUI).

Для создания и отображения графики главного окна и области шахматной доски можно использовать такие классы Qt, как QWidget и QGraphicsView. Визуальную составляющую функционала взаимодействия с пользователем (меню с кнопками и другие элементы управления) можно реализовать с помощью Qt Designer, а именно Widget Editing Mode, так как в режиме редактирования виджетов объекты можно перетаскивать из окна виджетов главного окна в форму, редактировать, изменять их размер, перетаскивать по форме. Таким же образом реализуется отображение времени и следующий ход. Так же с помощью данных классов и класса QGraphicsItems реализуется отображение графики фигур на доске.

Для реализации игровой логики шахмат необходимо создать классы для

шахматной доски и фигур, определить методы для проверки возможных ходов, наличия шаха, передвижения фигур. Так же необходимо создать класс событий для ведения нотации.

Обзор аналогов, qt, правила самой игры,

**4 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ**

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

**4.1 Структура входных и выходных данных**

Пользователь выбирает режим игры, ходы фигур путем нажатия клавиши мыши, выходными данными являются передвижения фигур, текстовые уведомления пользователя о выигрыше и проигрыше, отображение времени партии.

**4.2 Разработка диаграммы классов**

Диаграмма классов для курсового проекта представлена в приложении 1 и на рис. 1.

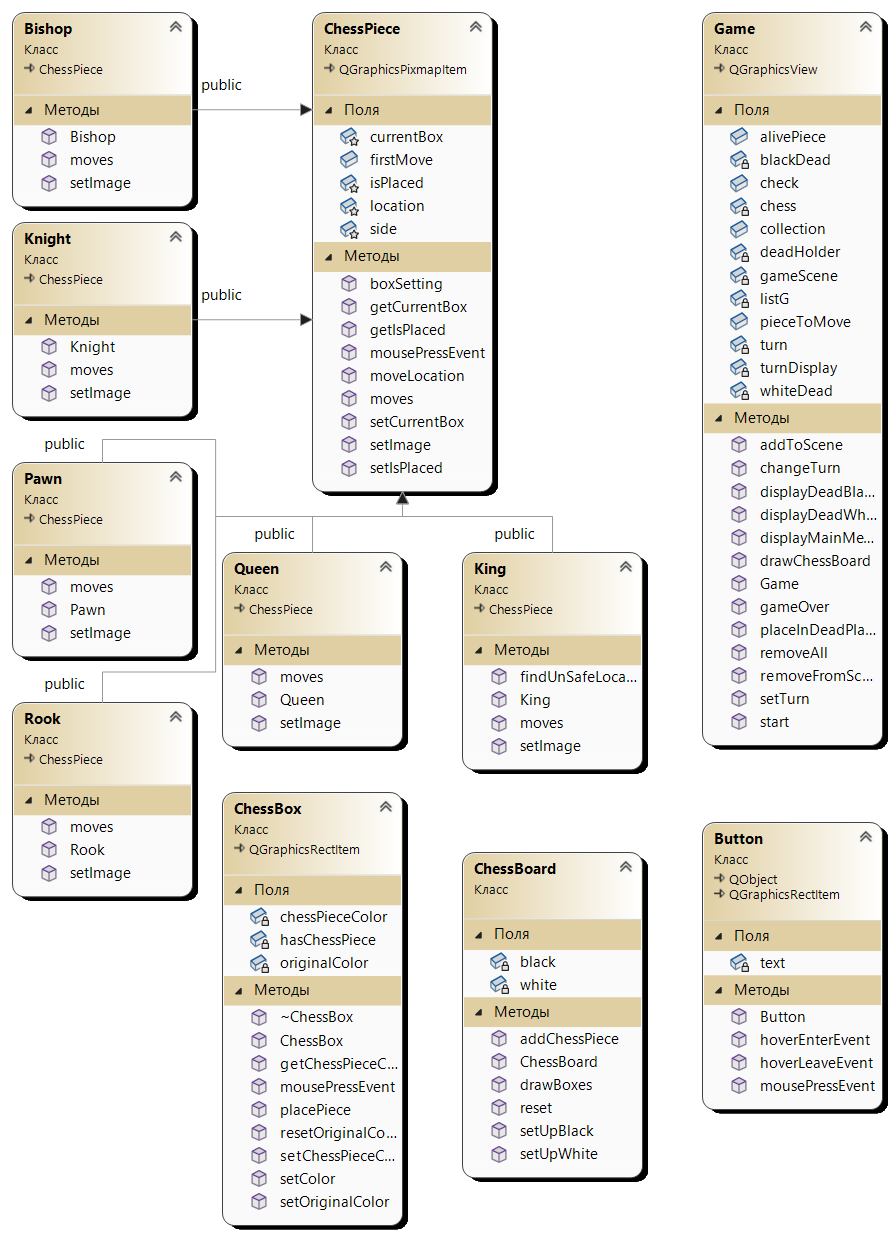


Рис. 1 Диаграмма классов игры

**4.3 Описание классов**

Класс **Game –** класс основного цикла игры.

Описание класса:

public:

Game(QWidget \*parent = 0) – начало игры

void drawChessBoard(); – отображение шахматной доски

void displayDeadWhite(); – отображение списка выбывших белых фигур

void displayDeadBlack(); – отображение списка выбывших черных фигур

void addToScene(QGraphicsItem \*item); – отображение в окне

void removeFromScene(QGraphicsItem \*item); – прекращение отображения в окне

ChessPiece \*pieceToMove; – активная фигура

void setTurn( QString value); – установка хода

void changeTurn(); – смена хода

ChessBox \*collection[8][8]; – ячейки доски

QList <ChessPiece \*> alivePiece; – список оставшихся фигур

void displayMainMenu(); – отображение меню в окне

void gameOver(); – окончание игры

void removeAll(); – прекращение отображения всех объектов

void start(); – начало игры

private:

QGraphicsScene \*gameScene;

QList <ChessPiece \*> whiteDead – список выбывших белых фигур

QList <ChessPiece \*> blackDead – список выбывших черных фигур

QGraphicsRectItem \* deadHolder;

QString turn; – очередь хода

QGraphicsTextItem \* turnDisplay; – графический элемент очереди хода

Класс **ChessBox –** класс графического отображения ячеек.

Описание класса:

public:

ChessBox(QGraphicsItem\* parent = 0); – отображение прямоугольника

~ChessBox(); – деструктор

void mousePressEvent(QGraphicsSceneMouseEvent\* event); – обработка нажатий мыши

void setColor(QColor color); – установка цвета

void placePiece(ChessPiece\* piece); – положение ячейки

void resetOriginalColor(); – сброс цвета ячейки

void setOriginalColor(QColor value); – восстановление цвета ячейки

QString getChessPieceColor(); – получение цвета ячейки

void setChessPieceColor(QString value); – установка цвета фигуры в ячейке

private:

QColor originalColor; – цвет ячейки

bool hasChessPiece; –смена цвет ячейки

QString chessPieceColor; – цвет фигуры в ячейке

Класс **ChessBoard –** класс графического отображения ячеек.

Описание класса:

public:

ChessBoard();– конструктор, установка всех фигур

void drawBoxes(int x, int y); – отображение ячейки

void setUpWhite(); – установка всех белых фигур

void setUpBlack(); – установка всех черных фигур

void addChessPiece(); – изменение полей количества фигур

private:

QList <ChessPiece\*> white; – список белых ячеек

QList <ChessPiece\*> black; – список белых ячеек

Класс **ChessPiece** **–** базовыйкласс фигуры.

Описание класса:

public:

void mousePressEvent(QGraphicsSceneMouseEvent\* event); – обработка нажатий мыши

virtual void setImage() = 0; – установка изображения фигуры

bool getIsPlaced(); – проверка на установку фигуры

void setIsPlaced(bool value); – проверка на установку фигуры

QList <ChessBox\*> moveLocation(); – список перемещений

virtual void moves() = 0; – перемещение

bool firstMove; – первый ход

Класс **Button** **–** базовыйкласс фигуры.

Описание класса:

public:

Button(QString name, QGraphicsItem\* parent = NULL); – конструктор

void mousePressEvent(QGraphicsSceneMouseEvent\* event); – обработка нажатий мыши

void hoverEnterEvent(QGraphicsSceneHoverEvent\* event); – обработка нажатий клавиши Enter

void hoverLeaveEvent(QGraphicsSceneHoverEvent\* event); – обработка прекращения нажатий

private:

QGraphicsTextItem\* text; – надпись

Класс **Bishop** **–**класс фигуры слона, наследует базовый класс фигуры.

Описание класса:

public:

Bishop(QString team, QGraphicsItem\* parent = 0); – установка фигуры

void setImage(); – установка изображения фигуры

void moves(); – ход фигуры

Класс **King** **–**класс фигуры короля, наследует базовый класс фигуры.

Описание класса:

public:

King(QString team, QGraphicsItem\* parent = 0); – установка фигуры

void setImage(); – установка изображения фигуры

void findUnSafeLocation(); – поиск позиции шаха

void moves(); – ход фигуры

Класс **Knight** **–**класс фигуры коня, наследует базовый класс фигуры.

Описание класса:

public:

Knight(QString team, QGraphicsItem\* parent = 0); – установка фигуры

void setImage(); – установка изображения фигуры

void moves(); – ход фигуры

Класс **Pawn** **–**класс фигуры пешки, наследует базовый класс фигуры.

Описание класса:

public:

Pawn(QString team, QGraphicsItem\* parent = 0); – установка фигуры

void setImage(); – установка изображения фигуры

void moves(); – ход фигуры

Класс **Queen** **–**класс фигуры королевы, наследует базовый класс фигуры.

Описание класса:

public:

Queen(QString team, QGraphicsItem\* parent = 0); – установка фигуры

void setImage(); – установка изображения фигуры

void moves(); – ход фигуры

Класс **Rook** **–**класс фигуры королевы, наследует базовый класс фигуры.

Описание класса:

public:

Rook(QString team, QGraphicsItem\* parent = 0); – установка фигуры

void setImage(); – установка изображения фигуры

void moves(); – ход фигуры