Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Глазовский государственный педагогический институт имени В.Г. Короленко»

**ОТЧЕТ**

**по учебной практике**

**ПМ 01 Разработка программных модулей программного обеспечения для компьютерных систем**

**Специальность 09.02.03 Программирование в компьютерных системах**

Выполнил:

студент ОП СПО 33 группы

Шаталов Кирилл

Руководитель практики: ст. преподаватель кафедры

математики и информатики Касаткин К.А.

г.Глазов, 2019

**Оглавление**

**Введение**

Актуальность: В современном обществе все на уровне IT-технологий, в том числе и настольные игры. Также проводятся и шахматные партии в online режиме, следовательно сохраняются записи этих партий. Для этого существуют и создаются шахматные плееры, чтобы была возможность отсмотреть профессиональные игры гроссмейстеров и извлечь из них что-нибудь полезное для себя.

Цель: Разработка шахматного плеера.

Задача: Научиться разрабатывать проекты на языке C++ с применением библиотеки Qt.

**Плеер представляет совокупность методических и программных средств решения следующих задач:**

1. Изображение шахматной доски и шахматных фигур в окне программы

2. Шахматные фигуры можно перемещать мышью

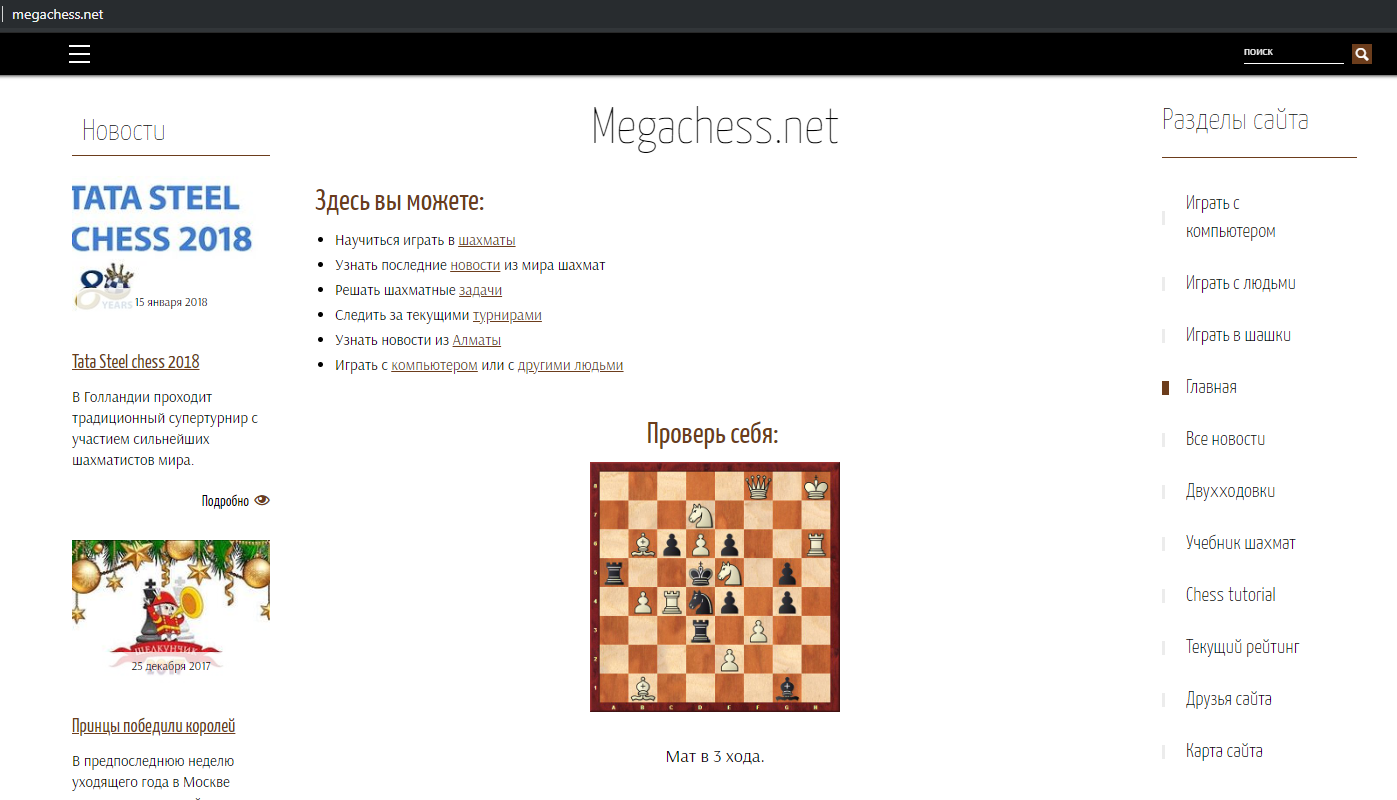
3. Считывание и воспроизведение шахматной партии из файла

4. Запись своей игры в файл

5. Запись каждого хода в формате начальная позиция:конечная позиция, где начальные и конечные позиции в формате номер горизонтали-номер вертикали(нумерация начинается с нуля).

**Обзор аналогов**

**1.megachess.net/sparchess**



На данном сайте вы можете научиться играть в шахматы, посмотреть различные вариации быстрых выигрышей и правильных комбинаций, как более выгодно разменивать фигуры и т.д..

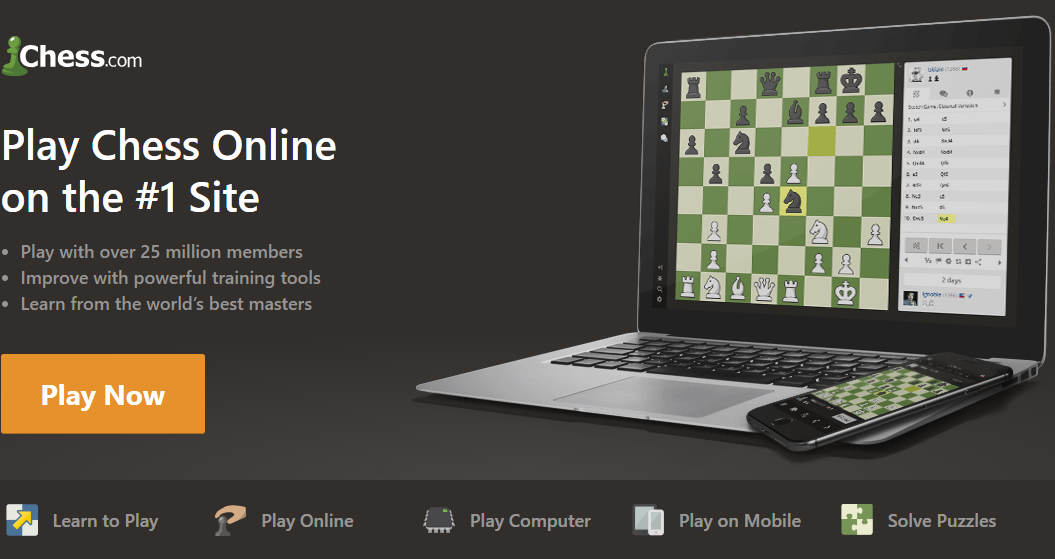
Также можно играть против компьютера или же практиковаться против других шахматистов.

**2.flashplayer.ru>настольные**

Данный сайт – Российская платформа флеш-игр.

На данном сайте вы также можете сыграть и в шахматы, в которых есть своя рейтинговая система подбора соперников.

**3.chess.com**



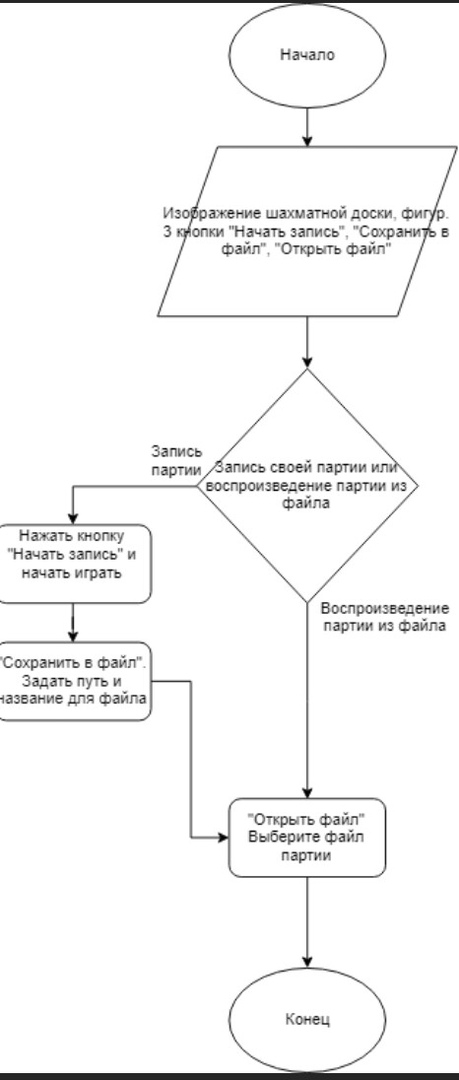
Наверно самый популярный сайт по шахматам.

На нем вы можете изучить теорию, играть онлайн, играть против компьютера и так же на мобильном устройстве.

**Постановка задачи и требования, предъявляемые к программе**

Была поставлена задача создания шахматного плеера, который должен воспроизводить записанные шахматные партии. Программа должна считывать ходы из файла, после чего фигуры должны двигаться.

**Описание алгоритма работы**



Приложение работает по показанной на рисунке блок-схеме. Когда вы открываете программу, вы видите перед собой шахматную доску, шахматные фигуры и 3 кнопки. При нажатии на кнопку “Начать запись” вы можете начать двигать фигуры и играть партию, при завершении вы нажимаете на кнопку “завершить запись”, далее при нажатии на кнопку “Сохранить в файл” вы указываете путь, куда вам нужно сохранить свою партию. Теперь вы можете выбрать свою партию, либо уже записанную, нажав на кнопку “Открыть файл”, затем начнется воспроизведение выбранной вами партии.

**Конвертирование партий:**

Конвертирование партий происходит вручную. Находится последовательность ходов и воспроизводится вручную, как шахматная партия, после чего запись сохраняется.

**Выбор и обоснование выбора средства разработки, исходный код программы**

Технические средства:

4.1. Персональный компьютер стандартной конфигурации.

4.2. Операционная система Windows 10 Professional x64.

4.3. Язык программирования C++.

4.4. Библиотека Qt.

Мною была выбрана среда разработки Visual Studio, как более удобная при работе с программным кодом. Также выбран язык C++ с применением библиотеки Qt. Выбрано это было по причине того, что во время учебных занятий мы достаточно много занимались с этим и мною было решено использовать этот язык программирования для разработки.

**Исходный код программы:**

1. **Файл main.cpp**

#include <Qapplication>

#include “mainwindow.h”

int main(int argc, char \*argv[]) {

Qapplication app(argc, argv);

MainWindow mainWin;

mainWin.show();

return app.exec();

}

1. **Файл mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QmainWindow>

#include <Qlabel>

#include <QpushButton>

#include <QfileDialog>

class MainWindow : public QmainWindow {

Q\_OBJECT

public:

MainWindow();

protected:

void mouseMoveEvent(QmouseEvent \*event);

void mousePressEvent(QmouseEvent \*event);

void mouseReleaseEvent(QmouseEvent \*event);

void paintEvent(QpaintEvent \*event);

void startGame(Qstring filename);

void ndgame(Qstring status);

void resetBoard();

void nextChess(int n);

private slots:

void openFile();

void saveFile();

void startRecord();

void stopRecord();

void moveChess();

private:

Qlabel \*position;

QpushButton \*openButton;

QpushButton \*saveButton;

QpushButton \*startRecordButton;

QpushButton \*stopRecordButton;

};

#endif

1. **Файл test.pro**

TEMPLATE = app

TARGET = test

INCLUDEPATH += .

QT+=widgets

# Input

HEADERS += mainwindow.h

SOURCES += main.cpp mainwindow.cpp

1. **Файл mainwindow.cpp**

#include <QtGui>

#include “mainwindow.h”

#include <QmessageBox>

#include <Qtimer>

#include <QtWidgets>

const Qstring imagesDir = “images/”;

ndgamet INTERVAL = 10; // Интервал движения шахмат (меньше – быстрее)

const Qstring tempfileName = “.temp”;

ndgamet squareWidth = 60;

ndgamet squareHeight = 60;

ndgamet boardWidth = 8;

ndgamet boardHeight = 8;

ndgamet boardMarginX = 40;

ndgamet boardMarginY = 100;

int mouseX = -1;

int mouseY = -1;

int boardX = -1;

int boardY = -1;

bool isPressed = false;

int chessPressedX = -1;

int chessPressedY = -1;

bool isMovedNow = false;

int movedX = -1;

int movedY = -1;

int movedEndX = -1;

int movedEndY = -1;

int aX = -1;

int aY = -1;

Qstring movedChess = “”;

Qtimer \*timer;

Qvector<Qvector<int> > chessQueque;

int nextChessN = -1;

bool isRecordNow = false;

Qfile tempfile(tempfileName);

Qstring board[boardHeight][boardWidth] = {

{“b5”, “b2”, “b0”, “b4”, “b1”, “b0”, “b2”, “b5”},

{“b3”, “b3”, “b3”, “b3”, “b3”, “b3”, “b3”, “b3”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“w3”, “w3”, “w3”, “w3”, “w3”, “w3”, “w3”, “w3”},

{“w5”, “w2”, “w0”, “w4”, “w1”, “w0”, “w2”, “w5”}

};

MainWindow::MainWindow() {

setWindowTitle(tr(“Шахматы”));

resize(560, 630);

position = new Qlabel(this);

position->setGeometry(boardWidth \* squareWidth + boardMarginX, boardHeight \* squareHeight + boardMarginY, 100, 20);

setMouseTracking(true);

startRecordButton = new QpushButton(“Начать запись”, this);

stopRecordButton = new QpushButton(“Завершить запись”, this);

startRecordButton->setGeometry(boardMarginX, 10, 150, 50);

stopRecordButton->setGeometry(boardMarginX, 10, 150, 50);

stopRecordButton->setVisible(false);

saveButton = new QpushButton(“Сохранить в файл”, this);

openButton = new QpushButton(“Открыть файл”, this);

saveButton->setGeometry(boardMarginX + 170, 10, 145, 50);

openButton->setGeometry(boardMarginX + 170 + 165, 10, 145, 50);

saveButton->setEnabled(false);

connect(openButton, SIGNAL(clicked()), this, SLOT(openFile()));

connect(saveButton, SIGNAL(clicked()), this, SLOT(saveFile()));

connect(startRecordButton, SIGNAL(clicked()), this, SLOT(startRecord()));

connect(stopRecordButton, SIGNAL(clicked()), this, SLOT(stopRecord()));

}

void MainWindow::openFile() {

Qstring filename = QfileDialog::getOpenFileName(this, “Выберите файл для открытия”, NULL, “Chess-file (\*.chess)”);

if (!filename.isNull()) {

startGame(filename);

}

}

void MainWindow::saveFile() {

Qstring filename = QfileDialog::getSaveFileName(this, “Выберите место сохранения файла”, NULL, “Chess-file (\*.chess)”);

if (!filename.isNull()) {

if (filename.indexOf(“.chess”, 0, Qt::CaseInsensitive) == -1) {

filename += “.chess”;

}

if (Qfile::copy(tempfileName, filename)) {

QmessageBox::information(this, “Сохранено”, “Файл сохранен в “ + filename);

} else {

QmessageBox::warning(this, “Ошибка сохранения”, “Файл не сохранен”);

}

}

}

void MainWindow::startRecord() {

resetBoard();

isRecordNow = true;

stopRecordButton->setVisible(true);

startRecordButton->setVisible(false);

saveButton->setDisabled(true);

openButton->setDisabled(true);

if (!tempfile.open(QIODevice::WriteOnly)) {

QmessageBox::warning(this, «Ошибка записи», «Временный файл недоступен»);

stopRecord();

}

}

void MainWindow::stopRecord() {

isRecordNow = false;

stopRecordButton->setVisible(false);

startRecordButton->setVisible(true);

saveButton->setDisabled(false);

openButton->setDisabled(false);

tempfile.close();

}

void MainWindow::startGame(Qstring filename) {

saveButton->setEnabled(false);

openButton->setEnabled(false);

startRecordButton->setEnabled(false);

stopRecordButton->setEnabled(false);

resetBoard();

Qfile file(filename);

if (file.open(QIODevice::ReadOnly)) {

Qstring gameRaw;

gameRaw = file.readAll();

if (gameRaw.trimmed() != “”) {

QstringList items = gameRaw.split(“\n”);

for (int i = 0; i < items.length(); i++) {

Qstring item = items[i];

if (item == “”) {

break;

}

QstringList oneItem = item.split(“:”);

if (oneItem.length() != 2) {

ndgame(«Ошибка воспроизведения – неверный формат файла»);

return;

}

Qstring before = oneItem[0];

Qstring after = oneItem[1];

QstringList beforePosition = before.split(“-“);

QstringList afterPosition = after.split(“-“);

if (beforePosition.length() != 2 || afterPosition.length() != 2) {

ndgame(“Ошибка воспроизведения – неверный формат файла”);

return;

}

int beforeX = beforePosition[0].toInt();

int beforeY = beforePosition[1].toInt();

int afterX = afterPosition[0].toInt();

int afterY = afterPosition[1].toInt();

Qvector<int> quequeItem;

quequeItem.append(beforeX);

quequeItem.append(beforeY);

quequeItem.append(afterX);

quequeItem.append(afterY);

chessQueque.append(quequeItem);

}

nextChess(0);

} else {

ndgame(«Ошибка воспроизведения – файл пустой»);

}

file.close();

} else {

ndgame(«Ошибка воспроизведения – не удалось открыть файл»);

}

}

void MainWindow::moveChess() {

if (movedX != movedEndX) {

if (movedX > movedEndX) {

movedX -= 1;

} else {

movedX += 1;

}

}

if (movedY != movedEndY) {

if (movedY > movedEndY) {

movedY -= 1;

} else {

movedY += 1;

}

}

if (movedX == movedEndX && movedY == movedEndY) {

timer->stop();

isMovedNow = false;

board[aY][aX] = movedChess;

nextChess(nextChessN);

}

update();

}

void MainWindow::nextChess(int n) {

if (nextChessN >= chessQueque.size()) {

update();

ndgame(«Воспроизведение игры завершено. Количество ходов: « + Qstring::number(n));

return;

}

Qvector<int> item = chessQueque.at(n);

isMovedNow = true;

int beforeX = item[0];

int beforeY = item[1];

int afterX = item[2];

int afterY = item[3];

movedChess = board[beforeY][beforeX];

board[beforeY][beforeX] = “”;

movedX = boardMarginX + squareWidth \* beforeX;

movedY = boardMarginY + squareHeight \* beforeY;

movedEndX = boardMarginX + squareWidth \* afterX;

movedEndY = boardMarginY + squareHeight \* afterY;

aX = afterX;

aY = afterY;

nextChessN = n + 1;

timer = new Qtimer(this);

connect(timer, SIGNAL(timeout()), this, SLOT(moveChess()));

timer->start(INTERVAL);

}

void MainWindow::ndgame(Qstring status) {

saveButton->setEnabled(false);

openButton->setEnabled(true);

startRecordButton->setEnabled(true);

stopRecordButton->setEnabled(true);

chessQueque.clear();

nextChessN = -1;

QmessageBox::information(this, “Игра завершена”, status);

}

void MainWindow::mouseMoveEvent(QmouseEvent \*event) {

mouseX = event->x();

mouseY = event->y();

boardX = (mouseX – boardMarginX) / squareWidth;

boardY = (mouseY – boardMarginY) / squareHeight;

if (isRecordNow) {

if (mouseX – boardMarginX < 0 || mouseX > boardMarginX + squareWidth \* boardWidth)

boardX = -1;

if (mouseY – boardMarginY < 0 || mouseY > boardMarginY + squareHeight \* boardHeight)

boardY = -1;

if (isPressed) {

update();

}

}

}

void MainWindow::mousePressEvent(QmouseEvent \*event) {

boardX = (mouseX – boardMarginX) / squareWidth;

boardY = (mouseY – boardMarginY) / squareHeight;

chessPressedX = boardX;

chessPressedY = boardY;

if (board[chessPressedY][chessPressedX] != “”) {

if (event->button() == Qt::LeftButton && isRecordNow) {

isPressed = true;

mouseX = event->x();

mouseY = event->y();

}

} else {

isPressed = false;

}

}

void MainWindow::mouseReleaseEvent(QmouseEvent \*) {

isPressed = false;

if (boardX >= 0 && boardY >= 0 && chessPressedX >= 0 && chessPressedY >= 0 && (chessPressedX != boardX || chessPressedY != boardY) && isRecordNow && board[chessPressedY][chessPressedX] != “”) {

Qchar selfColor = board[chessPressedY][chessPressedX][0];

Qchar newColor = board[boardY][boardX][0];

if (selfColor != newColor) {

board[boardY][boardX] = board[chessPressedY][chessPressedX];

board[chessPressedY][chessPressedX] = “”;

QtextStream out(&tempfile);

out << chessPressedX << “-“ << chessPressedY << “:” << boardX << “-“ << boardY << “\n”;

}

}

update();

}

void MainWindow::paintEvent(QpaintEvent \*) {

Qpainter p(this);

int color = 1;

for (int i = 0; i < boardWidth; i++)

{

for (int j = 0; j < boardHeight; j++)

{

if (color)

p.setBrush(Qt::white);

else

p.setBrush(Qt::black);

int sqX = boardMarginX + j \* squareWidth;

int sqY = boardMarginY + i \* squareHeight;

p.drawRect(sqX, sqY, squareWidth, squareHeight);

if (board[i][j] != “”) {

Qstring image = imagesDir + board[i][j] + “.png”;

p.drawImage(sqX, sqY, Qimage(image).scaled(squareWidth, squareHeight));

}

color = !color;

}

color = !color;

}

if (isPressed && chessPressedX >= 0 && chessPressedY >= 0) {

Qstring image = imagesDir + board[chessPressedY][chessPressedX] + “.png”;

p.drawImage(mouseX – (squareWidth / 2), mouseY – (squareHeight / 2), Qimage(image).scaled(squareWidth, squareHeight));

}

if (isMovedNow) {

Qstring image = imagesDir + movedChess + “.png”;

p.drawImage(movedX, movedY, Qimage(image).scaled(squareWidth, squareHeight));

}

Qstring horizontalChars[boardWidth] = {“A”, “B”, “C”, “D”, “E”, “F”, “G”, “H”};

Qstring verticalChars[boardHeight] = {“1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”};

for (int i = 0; i < boardWidth; i++) {

p.save();

p.translate(squareWidth / 2 + boardMarginX + i \* squareWidth, boardMarginY – 20);

p.rotate(180);

p.drawText(0, 0, horizontalChars[i]);

p.restore();

p.drawText(squareWidth / 2 + boardMarginX + i \* squareWidth, boardMarginY + squareHeight \* boardHeight + 20, horizontalChars[i]);

}

for (int i = 0; i < boardHeight; i++) {

p.save();

p.drawText(boardMarginX – 20, boardMarginY + squareHeight \* boardHeight – i \* squareHeight – squareHeight / 2, verticalChars[i]);

p.translate(boardMarginX + squareHeight \* boardHeight + 20, boardMarginY + squareHeight \* boardHeight – i \* squareHeight – squareHeight / 2);

p.rotate(180);

p.drawText(0, 0, verticalChars[i]);

p.restore();

}

}

void MainWindow::resetBoard() {

Qstring newboard[8][8] = {

{“b5”, “b2”, “b0”, “b4”, “b1”, “b0”, “b2”, “b5”},

{“b3”, “b3”, “b3”, “b3”, “b3”, “b3”, “b3”, “b3”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“”, “”, “”, “”, “”, “”, “”, “”},

{“w3”, “w3”, “w3”, “w3”, “w3”, “w3”, “w3”, “w3”},

{“w5”, “w2”, “w0”, “w4”, “w1”, “w0”, “w2”, “w5”}

};

for (int i = 0; i < boardWidth; i++) {

for (int j = 0; j < boardHeight; j++) {

board[i][j] = newboard[i][j];

}

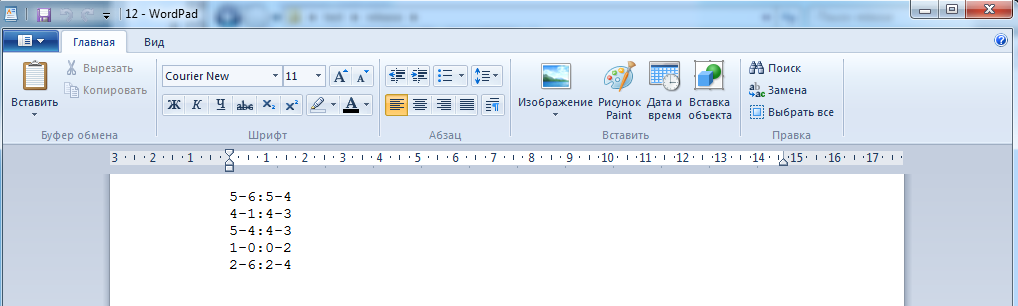
}

update();

}

**Тестирование программы**

В ходу тестирования были созданы несколько файлов для отсмотрения того, как записываются проделанные ходы.



**Результат тестирования:**

В конечном итоге были созданы три файла для воспроизведения с самыми популярными и красивыми играми профессиональных игроков в шахматы.

1. Adolf Anderssen vs Jean Dufresne
2. Adolf Anderssen vs Lionel Adalbert Bagration Felix Kieseritzky
3. Paul Morphy vs Duke Karl

**Руководство по установке**

**Руководство пользователя**

Для пользователя плеер достаточно прост и понятен, если он хочет записать свою игру, то все, что ему нужно сделать - это нажать на кнопку “Начать запись”, сыграть шахматную партию, или же просто сделать несколько ходов. Далее закончить запись и выбрать, куда он хочет сохранить запись игры. Для воспроизведения нужно лишь нажать на кнопку “Открыть файл” и выбрать нужную ему шахматную партию, и партия начнет воспроизводиться.

**Заключение:**

В ходе прохождения учебной практики были получены знания по приминению библиотеки Qt и применению QGraphicScene.

Был разработан плеер шахматных партий так, как и требовалось от поставленной задачи.

Был оформлен документ “Техническое задание”.

Сделана презентация, отчет и защитное слово для защиты.

В конечном итоге стартовое окно плеера выглядит так:

**Список литературы:**

1. Класс QGraphicsScene. Шлее М.Е. Qt Профессиональное программирование на C++ [электронный ресурс] URL: <https://books.google.ru/books?id=RasUCAAAQBAJ&pg=PA332&lpg=PA332&dq=qgraphicsscene+connect+qpushbutton&source=bl&ots=ulgpUlqkGt&sig=ACfU3U3kuvgLUwdewYbqlGxy7hftA90hzA&hl=ru&sa=X&ved=2ahUKEwjNpe-Us73gAhVRyaYKHeC7DloQ6AEwB3oECAMQAQ#v=onepage&q=qgraphicsscene%20connect%20qpushbutton&f=false>
2. Каркас графического представления: QGraphicsScene, QGraphicsItem, QGraphicsView [электронный ресурс]. URL : <http://doc.crossplatform.ru/qt/4.4.3/graphicsview.html>
3. Qt/C++ - Урок 018. QGraphicsItem - наследование и СЛОТы [электронный ресурс] URL: <http://doc.crossplatform.ru/qt/4.4.3/graphicsview.html>
4. Qt/C++ - Урок 051. QMediaPlayer – Аудио плеер на Qt [электронный ресурс]. URL: <https://evileg.com/ru/post/155/>
5. Шахматы на Qt C++ [электронный ресурс]. URL: <https://www.youtube.com/watch?v=4KcE1GcJarQ>