Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**Программное игровое средство «Шахматы»**

|  |  |  |
| --- | --- | --- |
| Студент |  | В.И. Миронюк |
| Руководитель |  | Г.В. Данилова |

Минск 2019

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc7636255)

[1 Анализ аналогов игрового средства 7](#_Toc7636256)

[1.1 Обзор аналогов 7](#_Toc7636257)

[1.2 Постановка задачи 9](#_Toc7636258)

[2 Разработка программного средства 10](#_Toc7636259)

[2.1 Структура программы 10](#_Toc7636260)

[2.2 Интерфейс программного средства 10](#_Toc7636261)

[2.3 Графические ресурсы 13](#_Toc7636262)

[2.4 Игровая логика 14](#_Toc7636263)

[3 Тестрирование программного средства 24](#_Toc7636267)

[4 Руководство пользователя 26](#_Toc7636268)

[4.1 Правила игры 26](#_Toc7636269)

[4.2 Интерфейс программы 26](#_Toc7636270)

[Заключение 28](#_Toc7636271)

[Список использованных источников 29](#_Toc7636272)

# **ВВЕДЕНИЕ**

Шахматы - настольная логическая игра со специальными фигурами на 64-клеточной доске для двух соперников, сочетающая в себе элементы искусства (в части шахматной композиции), науки и спорта.

Время возникновения шахмат неизвестно, и на эту тему постоянно разгораются споры. Все же, ученые сошлись в своих разногласиях, и пришли к выводу, что шахматы возникли в первом веке нашей эры в Северной Индии. Предполагают, что шахматы были прототипом различных воин и битв, но без кровопролития, и поэтому завоевали огромный интерес у правителей древних государств, где они могли сразиться, друг с другом не причинив своему войску ни единой царапинки.

Сложно представить, но история шахматных машин старше, чем история компьютеров. Идея создать машину, играющую в шахматы, датируется ещё восемнадцатым веком. Около 1769 года появился шахматный автомат «Механический турок». Он был предназначен для развлечения королевы Марии-Терезии. Машина действительно неплохо играла - внутри неё находился сильный шахматист, который и делал ходы. Однако, поистнее важной датой в мире шахмат является 11 мая 1997 года. В этот день DeepBlue шахматный суперкомпьютер,  разработанный  компанией IBM,  выиграл матч из 6 партий у чемпиона мира по шахматам Гарри Каспарова. В то время компьютер не умел оценивать позицию и строить игру на основании этой оценки. Рост силы игры достигался исключительно за счет увеличения мощности "железа". Даже алгоритм перебора все еще использовался "брутфорс", то есть перебирались все варианты, но очень быстро. В 2003 году состоялся еще один матч Каспарова против **компьютера** – с Deep Junior, работавшем на 4х-процессорной системе с процессорами Pentium IV 1.9 ГГц и 3 Гб оперативной памяти. Junior - первая программа, демонстрирующая «человечную» игру, и способная пойти на жертву ради инициативы. Матч закончился вничью. В следующем матче Каспаров играл на виртуальной трехмерной доске в 3D-очках, делая ходы при помощи голосовых команд. Этот матч также закончился вничью.

Стоит отметить, что рейтинг лидирющего, на данный момент, шахматного движка «Stockfish» 3447, рейтинг же действующего чемпиона мира Магнуса Карлсена равен 2845. Из этого можно сделать вывод, что человек вряд ли когда-либо сможет обыграть компьютер в шахматы.

Однако, уже совсем скоро шахматные боты, которые просто перебирают комбинации и выбирают выигрышную станут историей. Программа AlphaZero за четыре часа обучилась игре в шахматы, после чего победила Stockfish, который считался лучшим виртуальным шахматным игроком. В AlphaZero внедрили только правила шахмат, но не стратегии и алгоритмы игры. Всему этому программа обучалась сама. Если раньше все шахматные программы запоминали ходы, сделанные людьми, то новое поколение интеллекта получает все навыки, играя само с собой.

Перед началом каждого хода программам давали минуту на размышления. Всего провели 100 игр. AlphaZero победила 25 раз на белых и еще 3 раза на черных. Остальные игры сыграли вничью. Навыки программы компания-создатель назвала «сверхчеловеческими».

Актуальность выбора темы курсовой работы связана с недавними научными исследованиями, доказывающими, что люди, играющие в шахматы на протяжении всей жизни, сохраняют светлость ума и в пожилом возрасте, намного дольше живут, по сравнению с обычными людьми, существенно лучше контролируют своё поведение.

Целью данного проекта является разработка игрового приложения «Шахматы».

1. **АНАЛИЗ АНАЛОГОВ ПРОГРАММНОГО СРЕДСТВА**
   1. **Обзор аналогов**

На данный момент существует огромное количество ресурсов и приложений для игры в шахматы, самыми популярными из них являются «Chess.com», «Lichess.com», «Titan chess» и др.

Chess.com - сайт для игры в шахматы онлайн, форум, шахматная социальная сеть, а также одноимённая компания, владеющая данным сайтом. По данным Alexa.com, является самым популярным шахматным сайтом в мире. Домен chess.com был зарегистрирован в 1995 году компанией Aficionado, располагавшейся в американском штате Калифорния и занимавшейся продажей [шахматного программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D1%85%D0%BC%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA) под [брендом](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B5%D0%BD%D0%B4) Chess Mentor. По замыслу создателей, он представляет из себя своеобразный шахматный онлайн-дом, в котором объединены несколько функций: игра в безопасном и дружелюбном месте, поиск друзей, возможность сохранять сыгранные партии, делиться друг с другом шахматными историями, идеями, а также обучать друг друга. В [2014 году](https://ru.wikipedia.org/wiki/2014_%D0%B3%D0%BE%D0%B4) было заявлено, что на сайте с момента его запуска было проведено более 1 миллиарда партий. Внешний вид сайта Chess.com указан на рисунке 1.1.



Рисунок 1.1 - Внешний вид сайта «Chess.com»

Lichess.org - шахматный интернет-сервер, созданный французским программистом Тибо Дюплесси. Интересным является то, что программное обеспечение, используемое на сервере, является открытым и распространяемым под свободной лицензией AGPL. Основной функцией сайта является игра в шахматы онлайн или по переписке против других игроков с использованием различных вариантов контроля времени.



Рисунок 1.2 - Внешний вид сайта «Lichess.org»

Titan chess шахматная компьютерная игра, разработанная компанией  Oberon Games для Microsoft и изданная Microsoft Studios в 2006 году. Является одной из стандартных игр операционных систем Windows Vista (начиная с редакций «Домашняя расширенная») и Windows 7.



Рисунок 1.2 - Внешний вид сайта компьютерной игры «Titan chess»

* 1. **Постановка задачи**

В рамках данного курсового проекта планируется разработка игрового программного средства «Шахматы», адаптированного под запуск на современных операционных системах, в частности, версиях ОС Windows 7 и старше.

В игровом средстве планируется реализовать следующие функции:

* игра для двоих игроков;
* сохранение истории ходов и матчей;
* определение Шаха и Мата.

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки Embarcadero Delphi 10.3.

1. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА
2. 1. Структура программы

В данном приложении необходимо использовать восемнадцать структурных блоков, такое количество обусловлено принципом “на каждый класс отдельной блок”:

– unitBishop – модуль, отвечающий за проверку корректности хода слона;

– unitPawn – модуль, отвечающий за проверку корректности хода пешки;

– unitQueen – модуль, отвечающий за проверку корректности хода ферзя;

– FormHelp – модуль, отвечающий за окно помощи пользователю;

– unitRook – модуль, отвечающий за проверку корректности хода ладьи;

– unitHorse – модуль, отвечающий за проверку корректности хода лошади;

– unitKing– модуль, отвечающий за проверку корректности хода короля;

– unitHistory – модуль, отвечающий за работу с историей ходов;

– unitTypes – модуль, отвечающий за типы данных;

– unitHistory – модуль, отвечающий за работу с историей ходов;

– unitCell– модуль, отвечающий за обработку нажатия на клетку;

– unitBoard– модуль, отвечающий за действия, связанные с игровым полем;

– unitPieces– модуль, отвечающий за класс фигур;

– FormMain – модуль, отображающий главное окно программы;

– FormMat – модуль, отображающий информацию о мате;

– FormMatchesTable – модуль, отображающий сохраненные матчи;

–FormMatchName – модуль, отображающий главное окно для сохранения матча;

– FormMainMenu – модуль, отображающий главное меню.

* 1. Интерфейс программного средства

Интерфейс играет немаловажную роль в программе. Грамотно сконструированный интерфейс упрощает освоение программы, а неудачный - наоборот снижает эффективность работы, вызывает у клиентов разочарование. В связи с этим, необходимо тщательно продумывать интерфейс приложения.

В качестве главного меню выступает окно, в котором пользователю предоставляется выбор необходимой опции:

– «Resume game»;

– «Recovery match»;

– «Save match»;

– «Help»;

– «Exit».



Рисунок 2.1 - Внешний вид главного меню

При нажатии кнопки «Resume», главное меню исчезает и вместо него появляется доска с фигурами в оставленной игроками позиции.

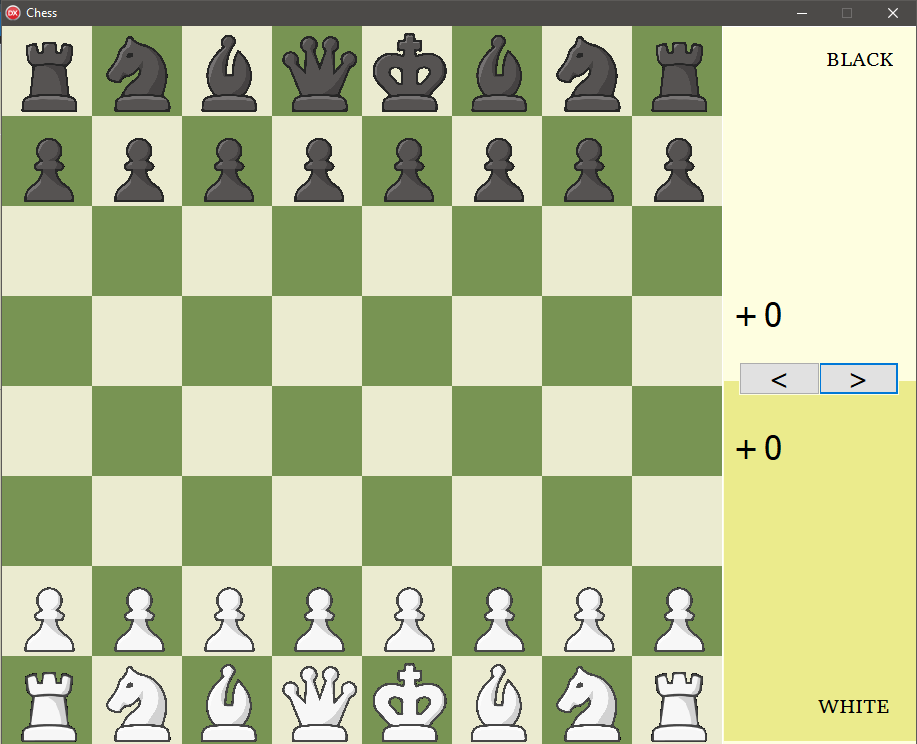


Рисунок 2.2 - Внешний вид доски с фигурами в начальной позиции

В окне с доской и фигурами доступны следующе опции:

– «Menu», вызов главного меню;

– « < », ход назад;

– « > », ход вперед.

Подменю опции «Recovery match» содержит следующие опции:

– «Delete», удаление выбранного матча;

– «Back», возврат в главное меню приложения;

– «Recovery match», восстановление сохраненного матча.

Внешний вид окна опции «Recovery match» представлен на рисунке 2.3.

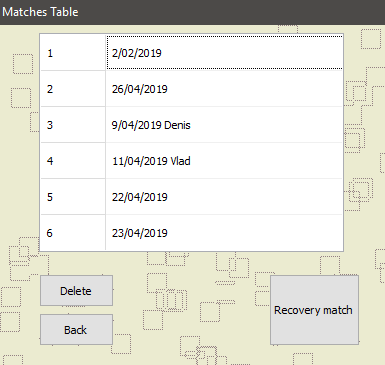


Рисунок 2.3 - Внешний вид окна опции «Recovery match»

Для выбора матча необходимо навести на него курсором или с помощью стрелок управления перевести указатель на него, выбранный матч находится в рамке.

При нажатии на кнопку 'Recovery match' происходит загрузка матча, а также для удобства происходит автоматическое закрытие окна главного меню.

В приложении предусмотрена ситуация ложного нажатия на кнопку 'Delete', и, чтобы избежать безвозвратной потери данных, происходит повторной запрос, с просьбой подтвердить это действие.

Подменю опции «Save match» содержит следующие опции:

– «Back», возврат в главное меню;

– «Save match», сохранение матча с именем записанного в специальную область;

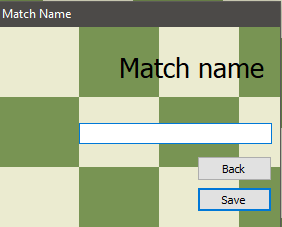


Рисунок 2.4 - Внешний вид окна опции «Save match»

Опция «Help» содержит информацию о авторе, а также руководство пользователя.

* 1. Графические ресурсы

Графические ресурсы включают в себя изображения конкретных шахматных фигур определенного цвета. В ходе работы с графическими ресурсами необходимо было прибегнуть к использованию стороннего программного обеспечения, а именно к программам компании Adobe Inc: Adobe Photoshop и Adobe Illustrator. Это мера была вынужденной в связи с отсутствием изображений фигур в необходимом формате (.BMP). Этот формат отличается невероятно малыми размерами даже в сжатом (заархивированном) виде, в связи с этим, он вызывает наименьшую нагрузку на систему.

После поиска подходящего по стилю изображения фигуры необходимо было переводить растровое изображение в векторное с помощью программы Adobe Illustrator, после чего, вручную вырезать задний фон и конвертировать в .BMP –формат с помощью программы Adobe Photoshop,

Загрузка изображения происходит при создании объекта фигуры. Процедура создания объекта фигуры приведена ниже.

//Вызывается при создании объекта класса короля

constructor TKing.Create(PieceColor: TPlayerColor);

begin

inherited; //Происходит вызов конструктора класса-родителя

if PieceColor = White then //Определение цвета короля

begin

Code := WKingCode; //Задание персонального номера фигуры

//Загрузка изображения

Picture.LoadFromResourceName(Hinstance,'bmpWhiteKing')

end

else

begin

Code := BKingCode;

Picture.LoadFromResourceName(Hinstance, 'bmpBlackKing');

end;

end;

В конструкторе Create вызывается родительский метод Create при помощи ключевого слова «inherited», код родительского метода Create приведен ниже.

constructor TPiece.Create(PieceColor: TPlayerColor);

begin

Color := PieceColor; //Установка цвета фигуры

Picture := TBitmap.Create; //Создание объекта класса TBitmap

Picture.Transparent := True;//Установка прозрачности

end;

После вызова родительского метода Create происходит загрузка изображений с помощью метода класса BMP, LoadFromResourceName(), в связи с чем, приложение не привязано к какому-либо файлу с изображениями, что, в свою очередь, делает программу более безопасной.

Так как каждый объект класса TPiece имеет свое изображение, память для которого выделяется вручную, необходимо освобождать эту пямять при удалении объекта фигуры, это и происходит в деструкторе класса TPiece. Поскольку реальное удаление фигур происходит только при закрытии программы, в процессе они сохраняются для возможности сохранении истории, деструктор класса TPiece вызывается только в конце работы приложения. Код деструктора приведен ниже.

destructor TPiece.Destroy;

begin

Picture.Free; //Освобождение памяти

inherited; //Вызов родительского деструктора

end;

* 1. Игровая логика

Работу данной части программы обеспечивают модули unitBishop, unitPawn, unitQueen, unitRook , unitHorse, unitKing, unitPieces, PlayerLogic.

2.4.1 Проверка корректности хода

Модули в названии которых присутствуют имена фигур unitPawn, unitHorse и др, отвечают за проверку хода конкретной фигуры. В зависимости от фигуры проверка хода разбивается на 3 или 4 этапа.

1 этап - это проверка возможности фигуры переместиться на финальную клетку.

2 этап - это проверка на то, свободен ли путь между начальной клеткой и финальной клеткой.

3 этап - это проверка финальной клетки на наличие там союзной фигуры.

4 этап - это проверка на то опасен ли этот ход для союзного короля, если да - то ход не верен. Для примера, ниже приведен код проверки хода слона.

function TBishop.IsCorrectPieceMove(const StartPos, FinalPos:

TCoordinateOnBoard): Boolean;

var

IsCorrectMove: Boolean;

Begin

//Проверка на то, может ли слон переместиться на финальную // клетку

IsCorrectMove := (Abs(StartPos.X-FinalPos.X) =

Abs(StartPos.Y - FinalPos.Y));

//Вызов функции которая проверяет свободен ли путь по // которому слон может перейти на финальную клетку

Result := IsCorrectMove and IsCleanWay(StartPos,FinalPos);

end;

//Функция для проверяет, свободен ли путь по которому должен // пройти слон

function TBishop.IsCleanWay(StartPos,FinalPos: TCoordinateOnBoard): Boolean;

var

i, j: TCoordinates;

Increment: ShortInt;

begin

Result := True;

if ((StartPos.X > FinalPos.X) and (StartPos.Y > FinalPos.Y)) or

((StartPos.X < FinalPos.X) and (StartPos.Y < FinalPos.Y)) then

begin

// Если слон идет по главной диагонали.

Increment := 1;

j := Min(StartPos.Y, FinalPos.Y) + Increment;

end

else

begin

// Если слон идет по побочной диагонали диагонали.

Increment := -1;

j := Max(StartPos.Y, FinalPos.Y) + Increment;

end;

// Идем по «всему пути» слона

for i := Min(StartPos.X, FinalPos.X) + 1 to Max(StartPos.X,

FinalPos.X) - 1 do

begin

if BoardObj.Board[j, i].Piece <> nil then

begin

// Если на пути есть фигура

Result := False; //Ход невозможен

Break; //Выходим из цикла

end;

Inc(j, Increment);

end;

// Если финальная клетка не пустая то смотрим союзная ли это //фигура или нет

if BoardObj.Board[FinalPos.Y, FinalPos.X].Piece <> nil then

Result:=(Result)and((BoardObj.Board[FinalPos.Y,

FinalPos.X].Piece.Color <> Color))

end;

Каждая функция проверки хода принимает 2 входных аргумента: начальную клетку и конечную, в результате этого мы можем описать этот метод как абстрактный метод класса TPiece, после чего, учитывая, что все классы фигур наследуются от класса TPiece, описывать метод IsCorrectPieceMove непосредственно в них, что, в свою очередь, является примером полиморфизма.

2.4.2 Совершение хода

После проверки на корректность хода, если ход корректен, происходит задание необходимых параметров, перестановка фигур и отрисовка хода. Код совершающий эти действия приведен ниже.

// Задание необходимых параметров для хода

PlayerObj.SetFieldsBeforeMove(IsCastlingMove, Self);

// Перестановка фигур

BoardObj.MakeMove(IsCastlingMove, TempPiece, Self);

// Отрисовка хода

BoardObj.DrawMove(IsCastlingMove, Self);

// Задание необходимых параметров для продолжения игры

PlayerObj.SetFieldsAfterMove(Self);

* + 1. Данные

В качестве структуры для хранения данных используется статический массив, каждый элемент которого ячейка шахматного поля - объект класса TCell. Класс TCell наследуется от класса TPaintBox.

За создание игрового поля отвечает метод CreatePieces. Код метода CreatePieces приведен ниже.

procedure TBoardClass.CreatePieces;

var

Y, X: TCoordinates;

Cell: TCell;

TempPiece: TPiece;

begin

TempPiece := TPiece.Create(white); //Создание временной фигуры

for Y := 1 to AmountCells do

for X := 1 to AmountCells do

begin

Cell := TCell.Create(Parent); //Создание ячейки

Cell.SetFields(X, Y); //Заполнение полей объекта ячейки

Cell.Piece := TempPiece.CreatePiece(Y, X);

Board[Y,X] := Cell; //Задание конкретной ячейки

end;

TempPiece.Free;

end;

После инициализации игрового поля устанавливаются начальные значения текущего состояния игры, а также вызывается процедура прорисовки игрового поля. Фрагмент кода, выполняющий данные действия, имеет вид:

BoardObj.DrawBoard; //Отрисовать доску

PlayerObj := TPlayerLogicClass.Create; // Создать класс логики

PlayerObj.SetFields // Установ. флаги в нач положение

В начале игры все ячейки игрового поля являются невыделенными, поэтому их цвет соответствует стандартному цвету шахматной клетки, при нажатии на какую-либо ячейку вызывается метод класса TCell – Click, в этом методе описан многоуровневый алгортим который определяет, какое именно действие хотел совершить пользователь, в конце метода в соответсвии с результатом работы алгоритма устанавливаются необходимые флаги и вызывается метод Draw класса TCell, который проверяет состояние клетки (находится ли в ней фигура, выделена ли клетка и др.), и отрисовывает клетку соответствующим необходимым цветом.

Необходимо отметить, что отрисовка ячейки устроена таким образом, что, при необходимости можно изменить цвет игрового поля внеся минимум изменений в код программы, это в очередной раз доказывает гибкость данного программного средства. Код метода Draw имеет следущий вид:

procedure TCell.Draw;

const

// Перечисление стандартных цветов

GreenColor = $00539478;

WhiteColor = $00D0EBEB;

Yellow\_White\_Color = $0083F6F6;

Yellow\_Green\_Color = $0042CBBC;

begin

Parent := BoardObj.Parent;

// Если клетка четная то цвет соответствует

// белому цвету в классических шахматах

if not Odd(CoordinatesOnBoard.X + CoordinatesOnBoard.Y) then

if Selected then

// Если ячейка выделена то обозначаем ее

// не стандартным цветом

Color := Yellow\_Green\_Color

else

Color := GreenColor

else

// Если клетка нечетная то цвет соответствует

// черному цвету цвету в классических шахматах

if Selected then

Color := Yellow\_White\_Color

else

Color := WhiteColor;

Paint;

end;

В случае мата происходит вызов окна Мата в котором пользователю предоставляется возможность сохранить матч, начать новый матч, просмотреть историю этого матча.

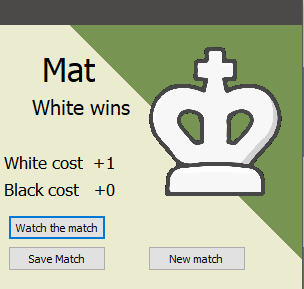


Рисунок 2.5 – Внешний вид окна опции Мата

* 1. Работа с историей.

Работа с историей ходов и матчей реализованы в модулях unitHistory,

PlayerLogic и FormMatchesTable.

## **2.5.1 Работа с историей ходов**

Каждый ход является объектом класса THistoryClass, описание этого класса приведено ниже.

THistoryClass = class

public

KilledPiece, MovedPiece: TPiece;

CastlingState: Byte;

StartPos, FinalPos: TCoordinateOnBoard;

Constructor Create(const parKilledPiece, parMovedPiece:

TPiece; const parStartPos, parFinalPos: TCoordinateOnBoard; const parCastlingState: Byte);

end;

Для каждого кода сохраняется фигура, которую убили, фигура, которая походила, начальная и финальная позиция «ходящей» фигуры, а также состояние, связанное с возможностью совершить рокировку.

Для удобного взаимодействия с историей ходов на главной странице расположены кнопки «<», для хода назад и «>», для хода вперед. При нажатии на кнопку «<» вызывается метод TurnBack класса TPlayerLogicClass,

Код метода с необходимыми комментариями приведен ниже.

procedure TPlayerLogicClass.TurnBack;

var

KilledPiece, MovedPiece: TPiece;

StartPos, FinalPos: TCoordinateOnBoard;

Begin

// Если первый ход, то нельзя нажать назад

if IsNotFirstMove(CurStepIndex) then

begin

//Отчищаем выделенную ячейку

BoardObj.ClearStartCell;

//Берем параметры для дальнейшего перемещения

KilledPiece := TurnHistory[CurStepIndex].KilledPiece;

MovedPiece := TurnHistory[CurStepIndex].MovedPiece;

StartPos := TurnHistory[CurStepIndex].StartPos;

FinalPos := TurnHistory[CurStepIndex].FinalPos;

//Если это был ход, при котором пешка превратилась в

// королеву

if ItWasTranformMoveToPawn(MovedPiece, BoardObj.Board

[FinalPos.Y,FinalPos.X].Piece)then

DeleteTransformQueen(FinalPos);

// Совершаем ход

MakeHistoryMove(KilledPiece,MovedPiece,StartPos,FinalPos);

// Задаем параметры для продолжения игры

SetFieldsFromPreviousTurn

(TurnHistory[PlayerObj.CurStepIndex]);

end;

end;

## **2.5.2 Работа с историей матчей**

Работа с историей матчей реализована в модуле PlayerLogic, а визуальная часть в модуле FormMatchesTable.

Для сохранения матча используется типизированный файл, который открывается при нажатии на кнопку «Save match» или «Recovery match». Тип файла TMatch хранит в себе все поля необходимые для восстановления матча. Тип TMatch выглядит следующим образом:

TMatch = Record

Name: string[20]; //Имя матча

BoardSnaphot: TBoardSnaphot; //Информация о доске

CurStepIndex: Integer; //Текущий индекс хода

WhoseTurn: TPlayerColor; //Текущий ход

IsShahState, IsMatState: Boolean; //Состояние шаха и мата

CastlingState: integer; //Состояние рокировки

//Позиции короля

WhiteKingPos, BlackKingPos: TCoordinateOnBoard;

End;

При нажатии на кнопку «Save match» вызывается метод SaveMatch, который собирает всю необходимую информацию о матче и передает ее в метод SaveToFile. В методе SaveToFile происходит вызов метода OpenFilе запись информации в файл, если файл не поврежден. Если файл был поврежден пользователем или OC, метод OpenFile создаст новый файл и вернет код ошибки. Пользователю будет передано соответствующее сообщение об ошибке.



Рисунок 2.6 – Блок-схема метода SaveToFile

Отрисовка таблицы сохраненных матчей происходит в модуле FormMatchesTable и состоит из двух методов: FillFixedCol и FillMatchesCol.

1. ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

В ходе тестирования приложения были выявлены некоторые недостатки проверки рокировки.

Как известно, рокировку можно совершать только в случае, если ни одна из фигур, принимающих участие в рокировке, не совершала ни одного хода. Следствием этого является необходимость хранить данные о всех 6 фигурах (4 ладьи и 2 короля). До стадии тестирования состояние рокировки хранилось только в объектах класса короля, что позволяла игрокам ходить ладьей, после чего возвращать ее на начальную позицию и делать с ней рокировку, что не соответствовало стандартным правилам шахмат. Выходом являлась переработка способа хранения данных о рокировке.

Было создано поле класса TPlayerLogicClass типа Byte, каждый бит которого содержал информацию о соответствующей фигуре. Значение бита эквивалентное единице означало то, что фигура еще не ходила, соответственно рокировка с ней разрешена. За получение информации о определенном бите отвечают методы CheckKingCastlingState, CheckLongRookCastlingState, CheckShortRookCastlingState. Для примера ниже приведен код метода CheckKingCastlingState.

function TPlayerLogicClass.CheckKingCastlingState(const Y:

TCoordinates): Boolean;

const

WKingMask = 1; //00000001

BKingMask = 8; //00001000

begin

//Определение цвета короля,

//по его начальному расположению

if Y = 1 then

Result := (PlayerObj.CastlingState and WKingMask) <> 0

else

Result := (PlayerObj.CastlingState and BKingMask) <> 0

end;

За изменение состояния определенных фигур отвечает метод класса TPlayerLogic, ChangeCastlingState.

В нем происходит определение того, какая фигура ходит в данный момент. Если этой фигурой является ладья или король, то происходит определение цвета данной фигуры и изменение состояния рокировки у данной фигуры. Состояние рокировки изменяется путем применения операций AND и NOT с маской текущей фигуры, благодаря этому нам удается изменить состояние определённого бита в переменно CastlingState.

// Изменение состояние белого короля

PlayerObj.CastlingState := PlayerObj.CastlingState

and (not WKingMask)

// Изменение состояние черной левой ладьи

PlayerObj.CastlingState := PlayerObj.CastlingState

and (not BLeftRookMask)

Данный метод позволяет задействовать минимальное количество памяти, поскольку хранится лишь одна переменная типа Byte, а не 6 переменных типа Boolean.

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
   1. Правила игры

Правила соответствуют классическим правилам игры «Шахматы». Цель игры - поставить мат королю соперника. Мат происходит, когда король попадает под шах и не может от него уйти. Защититься от шаха можно тремя способами:

* отступить на другое поле (не рокировкой);
* закрыться от шаха другой фигурой;
* взять фигуру, напавшую на короля.

Если король не может избежать мата - партия завершена. Обычно короля не берут и не снимают с доски, партия просто объявляется законченной.

На доске присутствуют 6 разновидностей фигур. Каждая из фигур ходит по-своему. Фигуры не могут перепрыгивать через другие фигуры (за исключением коня) или вставать на поле, где уже стоит фигура того же цвета. Однако они могут встать на место фигуры противника, которая считается взятой. Фигуры обычно размещают так, чтобы они угрожали фигурам противника взятием (встать на поле, где стояла взятая фигура, заменив её), защищали свои собственные фигуры, которым грозит взятие, или контролировали важные поля на доске. Полный список правил описан по ссылке «https://www.chess.com/ru/kak-igrat-v-shakhmaty». Для выделения конкретной фигуры необходимо нажать левой кнопкой мыши по ней. Выделенная на данный момент клетка с фигурой подсвечивается желтым цветом, для того, чтобы отменить выделение необходимо нажать левой кнопкой мыши на выделенную фигуру. Если нажатие происходит на фигуру, цвета, противоположному цвету игрока, который ходит на данный момент, фигура не выделяется. Чтобы совершить ход, необходимо выделить фигуру и нажать на клетку, в которую необходимо походить, при возможности такого хода, фигура будет перемещена в выбранную клетку.

* 1. Интерфейс программы

Главное окно программы, изображенно на рисунке 4.1.

Для удобства, игра начинается сразу же после запуска приложения. Для перехода в главное меню необходимо нажать кнопку «Esc» и выбрать в нем необходимую функцию.

Для корректной работы программы также необходим файл с сохранёнными матчами. В случае отсутствия файл будет создан автоматически. При этом все сохраненные матчи будут утеряны.

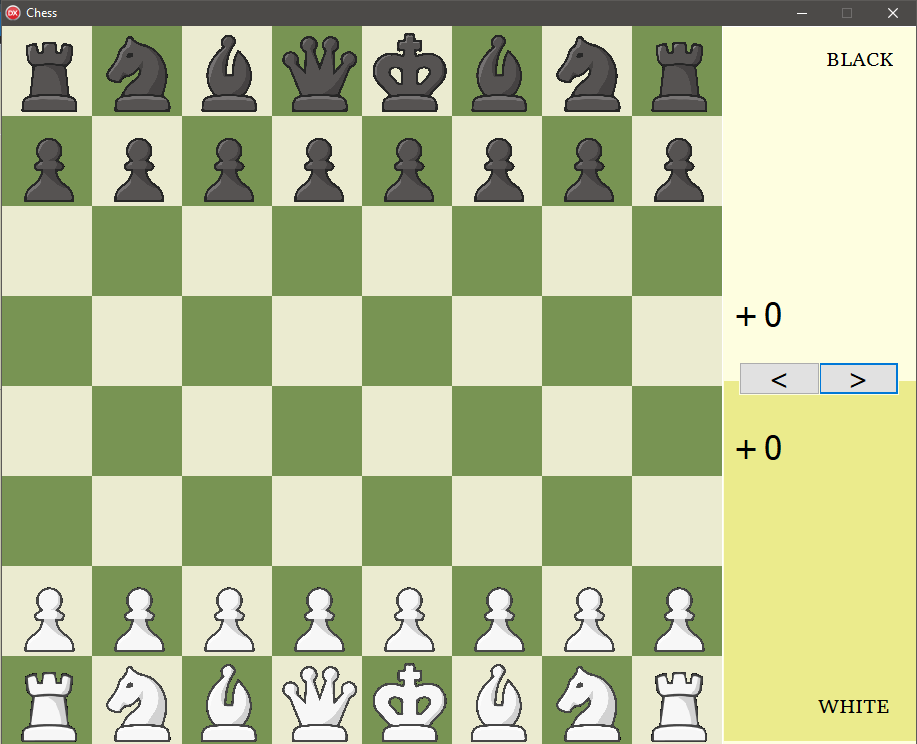


Рисунок 4.1 – Главное окно программы

ЗАКЛЮЧЕНИЕ

На сегодняшний день популярность компьютерных игр с каждым днем становится все больше. Для каждой настольной игры, уже существуют компьютерные аналоги, которые позволяют наслаждаться любыми играми в дороге, путешествии имея с собой лишь смартфон или ноутбук. Конечно, как и везде, здесь есть и негативная сторона, нынешняя молодежь становится полностью зависимой от своих девайсов.

В рамках данного курсового проекта было разработано игровое программное средство «Шахматы», Игра шахматы способствует развитию логического мышления, внимания, позволяет развить способность прогнозирования на несколько шагов вперед.

Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

* игра для двоих игроков;
* сохранение истории ходов и матчей;
* определение Шаха и Мата.

Для успешного создания игровое программного средства «Шахматы», необходимо было изучить возможности языка Delphi, Adobe Photoshop и Adobe Illustrator.

Существует большое число возможностей для дальнейшего улучшения приложения. Одним из самых простых направлений является введение режима игры для 3, 4 и 5 игроков, на данный момент, такие виды шахмат стремительно набирают популярность. Также, одной из идеи для дальнейшего развития приложения является адаптирование данного приложение для Android и IOS - устройств.

Данное приложение поспособствует не только приятному времяпрепровождению, но и позволит активно развить как логическое, так и абстрактное мышления.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. RAD Studio Product Documentation – Embarcadero Technologies [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://docs.embarcadero.com/products/rad_studio>.
2. Программирование на языке Delphi / Д. А Фараонов, В. В. Delphi 6. Учебный курс: / В. В. Фараонов. – СПб: изд. С. В. Малгачёва, 2001. – 231 с.
3. Уилсон, С. Принципы проектирования и разработки программного обеспечения, yчебн. курс. – СПб, 2003.
4. Глухова, Л. А. Основы алгоритмизации и программирования: Лаб. практикум для студ. спец. I-40 01 01 «Программное обеспечение информационных технологий» дневной формы обуч. В 4 ч. / Л. А. Глухова, Е. Е. Фадеева, Е. П. Фадеева. – Минск: БГУИР, 2007. – Ч. 3. – 51 с.
5. Сурков [и др.]. – учеб. пособие. – Режим доступа: http://www.rsdn.ru/?summary/3165.xml, – 2005.
6. Орлов, С. А. Технологии разработки программного обеспечения: учеб. Пособие. – СПб, 2003.
7. Свободная энциклопедия Википедия. – Статья «Шахматы для программистов». Электронные данные.  Режим доступа: https://ru.wikipedia.org/wiki/Шахматы, – 22.02.2019