**ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ КАЛУЖСКОЙ ОБЛАСТИ**

**«КАЛУЖСКИЙ ТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

**ГАПОУ КО «КТК»**

**КУРСОВАЯ РАБОТА**

**«Разработка алгоритмов работы игры «шахматы» на языке программирования python**

по общепрофессиональной дисциплине:

**«Основы алгоритмизации и программирования»**

специальность: **09.02.07 «Информационные системы и программирование»**

**III курс**

**Выполнил:**

**студент группы 3ИСиП1**

**Клименко З.М.**

**Подпись\_\_\_\_\_\_\_\_\_\_\_\_**

**Руководитель:**

**Грачёв Д.А.**

**Дата защиты\_\_\_\_\_\_\_\_\_**

**Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_**

**г. Калуга, 2021**

**ОГЛАВЛЕНИЕ**

[Введение Ошибка! Закладка не определена.](#_Toc69987406)

[Глава 1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc69987407)

[1.1. Постановка задачи 5](#_Toc69987408)

[1.2 Описание и обоснование выбора и состава технических и программных средств 6](#_Toc69987409)

[ГЛАВА 2 ПРОЕКТНАЯ ЧАСТЬ 12](#_Toc69987410)

[2.1 Информационная модель и ее описание 12](#_Toc69987411)

[2.1.1 Диаграмма потоков данных 13](#_Toc69987412)

[2.2 Разработка алгоритма работы программы 14](#_Toc69987413)

[2.3 Разработка интерфейса программы 14](#_Toc69987414)

[ГЛАВА 3 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА 19](#_Toc69987415)

[3.1 Реализация программного кода 19](#_Toc69987416)

[3.2 Отладка и тестирование программы 27](#_Toc69987417)

[3.3 Руководство пользователя 28](#_Toc69987418)

[ЗАКЛЮЧЕНИЕ 30](#_Toc69987419)

[БИБЛИОГРАФИЯ 31](#_Toc69987420)

**ВВЕДЕНИЕ**

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов, формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL).

Оконное приложение – это класс приложений, использующих для взаимодействия с пользователем элементы графического пользовательского интерфейса, то есть объекты типа: окна, кнопки, поля ввода, элементы контроля и многие другие. При помощи устройств ввода (клавиатуру/мышь/тачпад и прочие), пользователь имеет возможность взаимодействовать с объектами оконного приложения: перемещать, активировать, прокручивать. Примером данного класса являются классические графические приложения, работающие с окнами. В оконных приложениях основным элементом управления является окно или форма, которая в свою очередь содержит другие элементы управления, такие как кнопки, картинки, метки, поля редактирования. Существуют специальные программы для создания оконных приложений.

Целью представленной курсовой работы представляется разработка и осуществление оконного приложения для менеджера стоматологической клиники. Для достижения поставленной цели необходимо решить ряд задач теоретического и практического характера, а именно:

1. На основе теоретического анализа литературы выяснить основные принципы работы стоматологической клиники

2. Изучить предметную область и функции решаемых задач;

3. Создать базу данных для менеджера стоматологической клиники

4. Спроектировать оконное приложение бля управления БД.

5. Провести тестирование, убедиться в отсутствии ошибок

Объектом исследования является процесс разработки АРМ менеджера стоматологической клиники.

Предметом исследования выступает использование полноценной IDE Visual Studio и использование языка программирования C#.

Структура курсовой работы соответствует логике исследования и включает в себя введение, 3 главы, заключение, список литературы.

**ГЛАВА 1 АНАЛИТИЧЕСКАЯ ЧАСТЬ**

* 1. **Постановка задач**

Задача данной работы состоит в том, чтобы разработать автоматизированное рабочее место менеджера стоматологической клиники.

Собой она будет представлять базу данных и оконное приложение. Пользователь запускает приложение и появляется окно которое представляет собой автоматизирование рабочее место, в котором пользователь сможет просматривать базу данных и взаимодействовать с ней.

В приложении будет реализовано:

* Формы с базой данной
* Кнопка добавление
* Кнопка удаления
* Кнопка редактирования
* Панель

Для достижения данных целей я буду использовать язык программирования С#, его библиотеку, полноценную IDE и редактор кода для языка программирования C# - Microsoft visual studio

* 1. **Описание и обоснование выбора, и состава технических и программных средств**

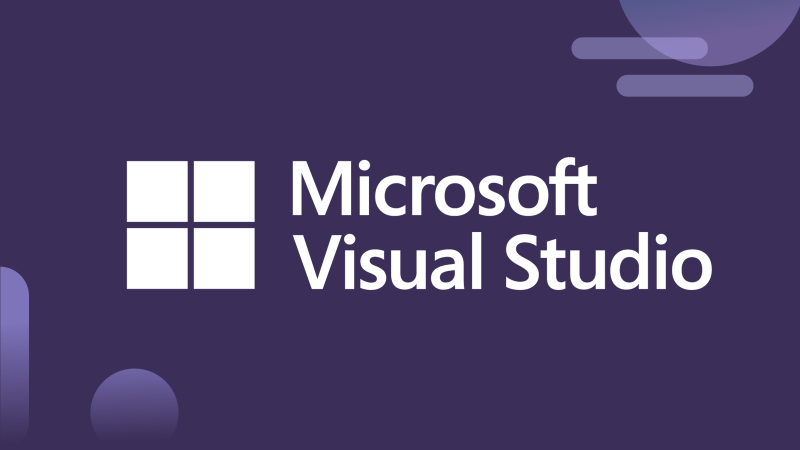
**Техническое обеспечение компьютера:**

* **Минимальные требования:** персональный компьютер с объемом оперативной памяти не менее 1Гб (Гигабайт), жесткого диска не менее 50Гб (Гигабайт), с процессором частотой не менее 2.2 ГГц (Гигагерц).
* **Рекомендуемые требования-** Персональный Компьютер с объемом оперативной памяти 4Гб (Гигабайт), жесткого диска не менее 100Гб (Гигабайт), с процессором частотой не менее 3.3 ГГц (Гигагерц).
* Техническое обеспечение, на котором разрабатывается алгоритм работы игры «шахматы» интегрированная среда разработки PyCharm и sublime Text

**Программное обеспечение:**

• **Microsoft Visual Studio** — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).



*Рисунок 1 Microsoft Visual Stidio*

* **Microsoft word –**текстовый редактор предназначенный ради просмотра, создания и редактирования текстовых документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается компанией Microsoft в содержании пакета Microsoft Office. Программный пакет Microsoft Office приходиться самым прибыльным и известным продуктом на данный момент и самым популярным продуктом из данного семейства является Microsoft Word.



*Рисунок 2 Word*

• **C#** (произносится *си шарп*) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).



*Рисунок 3 C#*

* **Microsoft Visio**-программа, вследствие которой есть возможность для составления графиков, чертежей, диаграмм, блок-схем. способствует при предоставлении графичной информации в элементарном и вразумительном виде.  
   Это отличный графичный редактор, разрешающий действовать с диаграммами и схемами. Программа включает могучий комплект инструментариев, которые будут общеполезны для службы. Приложение располагает возможность использоваться в различной среде. Его используют IT-специалисты, менеджеры, аналитики. Сама программа помещается в пакет Office, а вдобавок сможет употребляться в варианте некоторого компонента.  
  Приложение предрасполагает сильным базой средств. вдобавок от юзера не потребуются какие-нибудь высококлассные сведения в технической или изобразительной области и связанные с этим навыки. Вы можете употреблять готовые шаблоны, фигуры и элементы, вследствие которым и добьетесь необходимого результата.    
   Сориентироваться в программе сумеет аж новичок. Приложение располагает приспособленный и очевидный интерфейс. Пользователь сможет свободно основывать требуемые диаграммы и схемы, необходимые для вывода определенных задач.



*Рисунок 4 Visio*

* **phpMyAdmin** — веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. PhpMyAdmin позволяет через браузер и не только осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных. Приложение пользуется большой популярностью у веб-разработчиков, так как позволяет управлять СУБД MySQL без непосредственного ввода SQL команд.

Приложение распространяется под лицензией GNU General Public License и поэтому многие другие разработчики интегрируют его в свои разработки, например XAMPP, Denwer, AppServ, Open Server.

Проект на данный момент времени локализован на более чем 62 языках.



Рисунок 5 phpMyAdmin

* **Open Server Panel** — это портативная программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий.

Данный программный комплекс включает в себя тщательно подобранный набор серверного программного обеспечения, а также невероятно удобную и продуманную управляющую утилиту, которая обладает мощными возможностями по администрированию и настройке всех доступных компонентов.

**OSPanel** широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях.

Проект существует уже более 10 лет, он успешно зарекомендовал себя как первоклассный, простой и надёжный инструмент, необходимый каждому веб-мастеру. За годы разработки мы постарались реализовать в нём всё самое лучшее, что есть на сегодняшний день в аналогичных программных продуктах.

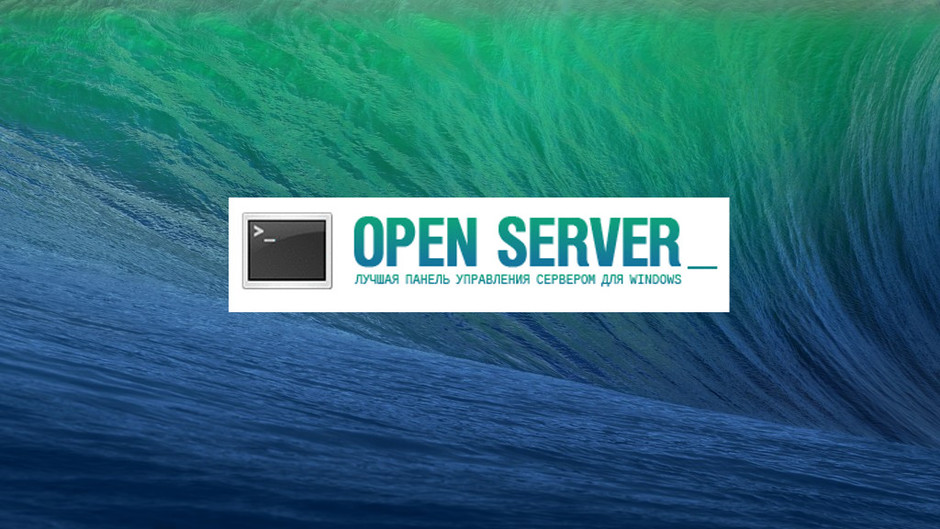


Рисунок 6 phpMyAdmin

**ГЛАВА 2 ПРОЕКТНАЯ ЧАСТЬ**

**2.1 информационная модель и её описание**

Информационная модель — модель объекта, представленная в виде информации, описывающей существенные для данного рассмотрения параметры и переменные величины объекта, связи между ними, входы и выходы объекта и позволяющая путём подачи на модель информации об изменениях входных величин моделировать возможные состояния объекта. Информационные модели нельзя потрогать или увидеть, они не имеют материального воплощения, потому что строятся только на информации. Информационная модель — совокупность информации, характеризующая существенные свойства и состояния объекта, процесса, явления, а также взаимосвязь с внешним миром.

Информационные модели делятся на описательные и формальные.

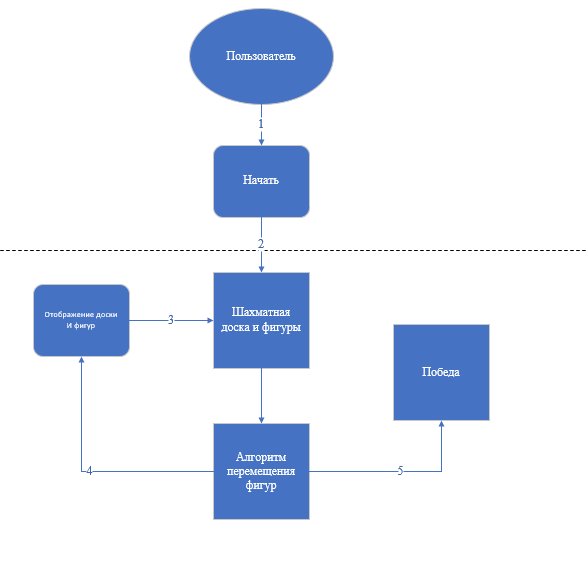
Описательные информационные модели — это модели, созданные на естественном языке (т.е. на любом языке общения между людьми: английском, русском, китайском, мальтийском и т.п.) в устной или письменной форме.

Формальные информационные модели — это модели, созданные на формальном языке (т.е. научном, профессиональном или специализированном). Примеры формальных моделей: все виды формул, таблицы, графы, карты, схемы и т.д.

Хроматические (информационные) модели — это модели, созданные на естественном языке семантики цветовых концептов и их онтологических предикатов (т.е. на языке смыслов и значений цветовых канонов, репрезентативно воспроизводившихся в мировой культуре). Примеры хроматических моделей: "атомарная" модель интеллекта (АМИ), межконфессиональная имманентность религий (МИР), модель аксиолого-социальной семантики (МАСС) и др., созданные не базе теории и методологии хроматизма.

Типы информационных моделей

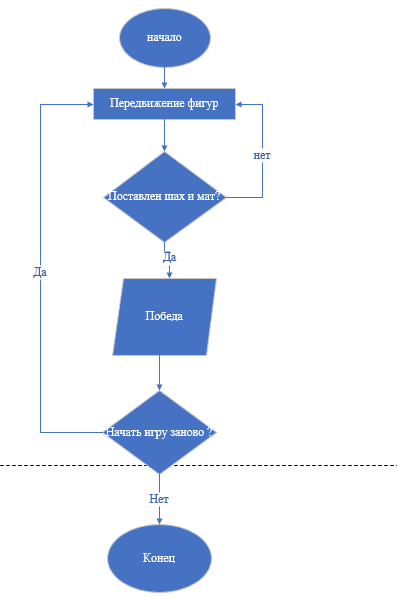
1. Моделирование отклика системы на внешнее воздействие
2. Классификация внутренних состояний системы
   1. **1 диаграмма потоков данных**



*Рисунок 7 DFD*

* 1. **разработка алгоритма работы программы**

Блок схема – вариация схемы которая описывает алгоритм, в котором каждые шаги содержат определённые блоки и между собой соединяющей линией, которая показывает последовательность шагов.



*Рисунок 8 блок-схема*

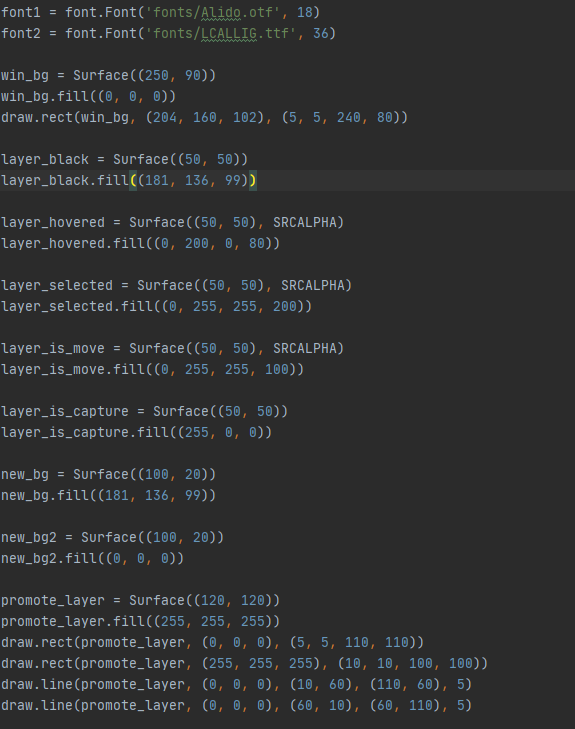
Здесь продемонстрирован алгоритм работы программы. Пользователи запускают игру и начинают передвижение фигур так как в этом суть игры. Когда король одной из сторон получает шах и мат, то объявляется победитель, а именно какая из сторон одержала победу: «чёрные» или «белые».

**2.3 разработка интерфейса программы**



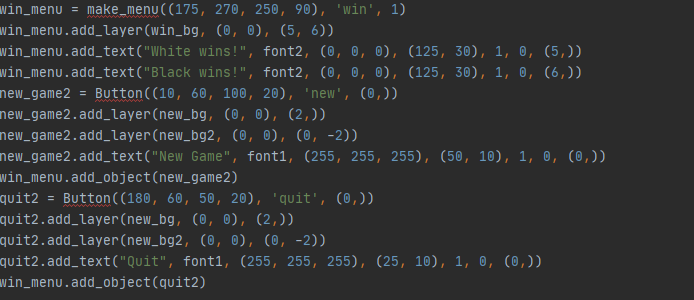
*Рисунок 9 загрузка фигур*

Здесь происходит загрузка шахматных фигур в виде картинок формата png.



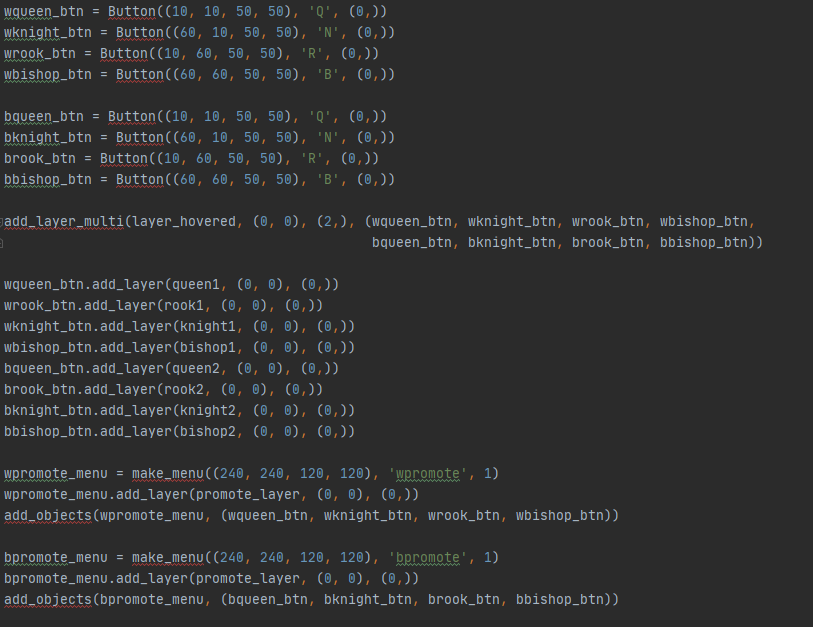
*Рисунок 10 создание доски*

Здесь происходит создание шахматного игрового поля, распределения цветов для кнопок и клеток, а также загрузка шрифтов и их настройка.



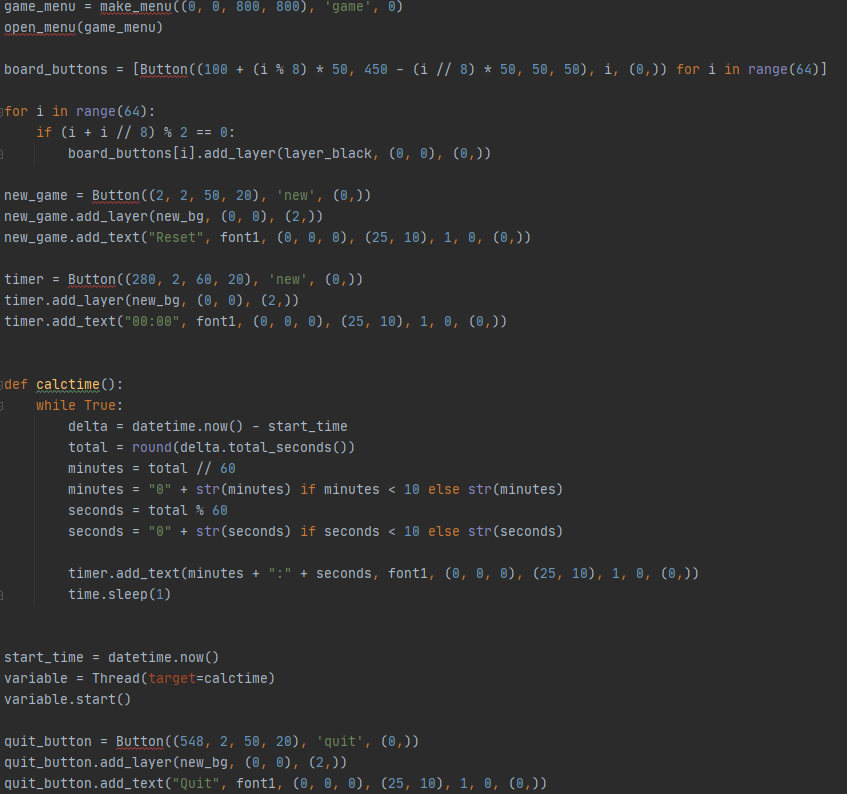
*Рисунок 11 меню победы*

Здесь создаётся окно меню победы, которое появляется соответственно в момент, когда одна из сторон одержит победу. Также содержит в себе кнопки.



*Рисунок 12 окно выбора*

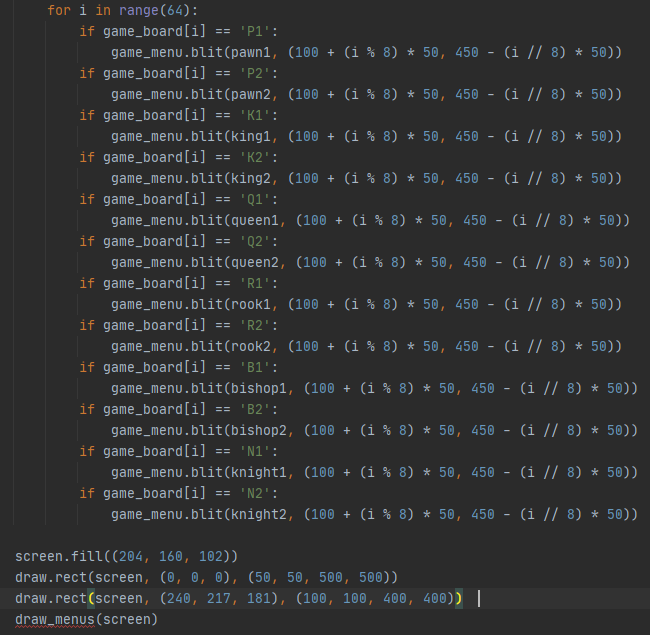
Здесь создаётся меню выбора, которое показывается тогда, когда пешка доходит до конца игрового поля. По правилам игры пешка в таком случае может поменять себя на другую фигуру, а именно на: короля, ладью, коня и ферзя.



*Рисунок 13 интерфейс*

Здесь создаётся сам интерфейс, а именно:

* размер доски
* кнопка «new game»
* кнопка «quit»
* таймера



*Рисунок 14 отрисовка фигур*

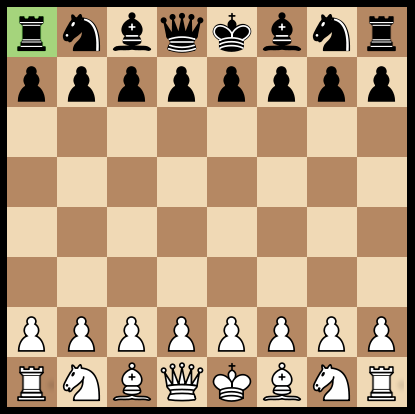
Здесь происходит сама отрисовка всех фигур на шахматной доске для обеих сторон. В нижней части, когда происходит распределение цветов шахматной доски и внешней области.

**ГЛАВА 3 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА**

**3.1 Реализация программного кода**

Для того, чтобы наиболее полно отразить особенности управления игрой необходимо разработать ходы для все фигур, так как в игре присутствуют 6 различных фигур и всего 16 фигур на каждую сторону.

* Пешка – 8 фигур
* Слон – 2 фигуры
* Король – 1 фигура
* Ферзь (дама) – 1 фигура
* Конь – 2 фигуры
* Ладья – 2 фигуры



*Рисунок 15 все фигуры на доске*

Для физического представления данных следует продумать структуру каждой фигуры.

if game\_board[selected] == 'P1':  
 if game\_board[selected + 8] == None:  
 moves.append(selected + 8)  
 if selected // 8 == 1 and game\_board[selected + 16] == None:  
 moves.append(selected + 16)  
 if selected % 8 != 0 and ((game\_board[selected + 7] != None and game\_board[selected + 7][1] == '2') or \  
 (game\_board[selected - 1] == 'P2' and en\_passent == selected - 1)):  
 captures.append(selected + 7)  
 if selected % 8 != 7 and ((game\_board[selected + 9] != None and game\_board[selected + 9][1] == '2') or \  
 (game\_board[selected + 1] == 'P2' and en\_passent == selected + 1)):  
 captures.append(selected + 9)

здесь продемонстрировано логика нахождения возможных ходов и их подсветка. По сути, всё реализовано при помощи координат. Происходит проверка может ли пойти пешка дальше, находиться ли она в углу, сделан ли ход первым игроком. Если перед пешкой нет никаких преград и следующая клетка пустая, то она может пойти дальше. Чтобы понять, что пешка находиться в углу происходит деление с остатком. Если получиться ноль значит в углу, а это может произойти при значении 7 или же 9. Также всё представлено для пешки второго игрока только наоборот. Будет уже не + 8, а -8 так как это и есть координаты.

if game\_board[selected][0] in 'RQ':  
 x, y = selected % 8, selected // 8  
 for i in range(x + 1, 8, 1):  
 if game\_board[i + y \* 8] == None   
 moves.append(i + y \* 8)  
 elif game\_board[i + y \* 8][1] != str(turn   
 captures.append(i + y \* 8)  
 break  
 else:  
 break  
 for i in range(x - 1, -1, -1):  
 if game\_board[i + y \* 8] == None:  
 moves.append(i + y \* 8)  
 elif game\_board[i + y \* 8][1] != str(turn):  
 captures.append(i + y \* 8)  
 break  
 else:  
 break  
 for i in range(y + 1, 8, 1):  
 if game\_board[x + i \* 8] == None:  
 moves.append(x + i \* 8)  
 elif game\_board[x + i \* 8][1] != str(turn):  
 captures.append(x + i \* 8)  
 break  
 else:  
 break  
 for i in range(y - 1, -1, -1):  
 if game\_board[x + i \* 8] == None:  
 moves.append(x + i \* 8)  
 elif game\_board[x + i \* 8][1] != str(turn):  
 captures.append(x + i \* 8)  
 break  
 else:  
 break

Здесь продемонстрирована такая же структура только для ладьи и ферзя. Находятся вместе, потому что имеют схожие ходы и проще было бы реализовать их вместе. Здесь добавлен **х** это уже будет являться ходом по вертикали.

i = selected + 7  
while (i - 7) % 8 != 0 and (i - 7) // 8 != 7:   
 if game\_board[i] == None:   
 moves.append(i)  
 elif game\_board[i][1] != str(turn):   
 captures.append(i)  
 break  
 else:  
 break  
 i += 7  
i = selected + 9  
while (i - 9) % 8 != 7 and (i - 9) // 8 != 7:  
 if game\_board[i] == None:  
 moves.append(i)  
 elif game\_board[i][1] != str(turn):  
 captures.append(i)  
 break  
 else:  
 break  
 i += 9  
i = selected - 7  
while (i + 7) % 8 != 7 and (i + 7) // 8 != 0:  
 if game\_board[i] == None:  
 moves.append(i)  
 elif game\_board[i][1] != str(turn):  
 captures.append(i)  
 break  
 else:  
 break  
 i -= 7  
i = selected - 9  
while (i + 9) % 8 != 0 and (i + 9) // 8 != 0:  
 if game\_board[i] == None:  
 moves.append(i)  
 elif game\_board[i][1] != str(turn):  
 captures.append(i)  
 break  
 else:  
 break  
 i -= 9

Здесь показана структура для слона и другая половина ферзя. Они также имеют схожие ходы только уже по диагонали. (I - 9) = ходы в левую часть координат. Верно, и обратное. Ферзь объединяет в себе ходы ладьи и слона, что делает его самой сильной фигурой имея возможность передвигаться на неограниченное количество клеток как по вертикали, горизонтали и по диагонали по всей доске если конечно же нет преград в виде других фигур и отдельные случаи если королю поставлен шах. Цикл while используется в Python для неоднократного исполнения определенной инструкции до тех пор, пока заданное условие остается истинным. Этот цикл позволяет программе перебирать блок кода. Сначала программа оценивает условие цикла while. Если оно истинное, начинается цикл, и тело while исполняется.

if game\_board[selected][0] == 'N':  
 x, y = selected % 8, selected // 8  
 if x >= 2 and y <= 6:   
 if game\_board[(x - 2) + (y + 1) \* 8] == None:

moves.append((x - 2) + (y + 1) \* 8)  
 elif game\_board[(x - 2) + (y + 1) \* 8][1] != str(turn): captures.append((x - 2) + (y + 1) \* 8)  
 if x >= 1 and y <= 5:  
 if game\_board[(x - 1) + (y + 2) \* 8] == None:  
 moves.append((x - 1) + (y + 2) \* 8)  
 elif game\_board[(x - 1) + (y + 2) \* 8][1] != str(turn):  
 captures.append((x - 1) + (y + 2) \* 8)  
 if x <= 6 and y <= 5:  
 if game\_board[(x + 1) + (y + 2) \* 8] == None:  
 moves.append((x + 1) + (y + 2) \* 8)  
 elif game\_board[(x + 1) + (y + 2) \* 8][1] != str(turn):  
 captures.append((x + 1) + (y + 2) \* 8)  
 if x <= 5 and y <= 6:  
 if game\_board[(x + 2) + (y + 1) \* 8] == None:  
 moves.append((x + 2) + (y + 1) \* 8)  
 elif game\_board[(x + 2) + (y + 1) \* 8][1] != str(turn):  
 captures.append((x + 2) + (y + 1) \* 8)  
 if x <= 5 and y >= 1:  
 if game\_board[(x + 2) + (y - 1) \* 8] == None:  
 moves.append((x + 2) + (y - 1) \* 8)  
 elif game\_board[(x + 2) + (y - 1) \* 8][1] != str(turn):  
 captures.append((x + 2) + (y - 1) \* 8)  
 if x <= 6 and y >= 2:  
 if game\_board[(x + 1) + (y - 2) \* 8] == None:  
 moves.append((x + 1) + (y - 2) \* 8)  
 elif game\_board[(x + 1) + (y - 2) \* 8][1] != str(turn):  
 captures.append((x + 1) + (y - 2) \* 8)  
 if x >= 1 and y >= 2:  
 if game\_board[(x - 1) + (y - 2) \* 8] == None:  
 moves.append((x - 1) + (y - 2) \* 8)  
 elif game\_board[(x - 1) + (y - 2) \* 8][1] != str(turn):  
 captures.append((x - 1) + (y - 2) \* 8)  
 if x >= 2 and y >= 1:  
 if game\_board[(x - 2) + (y - 1) \* 8] == None:  
 moves.append((x - 2) + (y - 1) \* 8)  
 elif game\_board[(x - 2) + (y - 1) \* 8][1] != str(turn):  
 captures.append((x - 2) + (y - 1) \* 8)

Здесь представлена структура коня. Единственная фигура, которая может перескакивать другие фигуры ходя буквой «Г». Добавлено значение Y для упрощения кода.

if game\_board[selected][0] == 'K':  
 x, y = selected % 8, selected // 8  
 attacked = attacked\_spaces(1 + turn % 2, game\_board)  
 if selected == 3 and wcastle % 2 == 0 and \  
 game\_board[1] == None and game\_board[2] == None and \  
 not 1 in attacked and not 2 in attacked and not 3 in attacked:  
 moves.append(1)  
 if selected == 3 and -1 < wcastle < 2 and \  
 game\_board[4] == None and game\_board[5] == None and game\_board[6] == None and \  
 not 3 in attacked and not 4 in attacked and not 5 in attacked:  
 moves.append(5)  
 if selected == 59 and bcastle % 2 == 0 and \  
 game\_board[58] == None and game\_board[57] == None and \  
 not 57 in attacked and not 58 in attacked and not 59 in attacked:  
 moves.append(57)  
 if selected == 59 and -1 < bcastle < 2 and \  
 game\_board[60] == None and game\_board[61] == None and game\_board[62] == None and \  
 not 59 in attacked and not 60 in attacked and not 61 in attacked:  
 moves.append(61)  
 if x >= 1 and y <= 6:   
 if game\_board[(x - 1) + (y + 1) \* 8] == None:

moves.append((x - 1) + (y + 1) \* 8)  
 elif game\_board[(x - 1) + (y + 1) \* 8][1] != str(turn): captures.append((x - 1) + (y + 1) \* 8)  
 if y <= 6:  
 if game\_board[x + (y + 1) \* 8] == None:  
 moves.append(x + (y + 1) \* 8)  
 elif game\_board[x + (y + 1) \* 8][1] != str(turn):  
 captures.append(x + (y + 1) \* 8)  
 if x <= 6 and y <= 6:  
 if game\_board[(x + 1) + (y + 1) \* 8] == None:  
 moves.append((x + 1) + (y + 1) \* 8)  
 elif game\_board[(x + 1) + (y + 1) \* 8][1] != str(turn):  
 captures.append((x + 1) + (y + 1) \* 8)  
 if x <= 6:  
 if game\_board[(x + 1) + y \* 8] == None:  
 moves.append((x + 1) + y \* 8)  
 elif game\_board[(x + 1) + y \* 8][1] != str(turn):  
 captures.append((x + 1) + y \* 8)  
 if x <= 6 and y >= 1:

Здесь всё также, как и для предыдущих фигур, разве что происходит проверка может ли король быть захваченным и взятым после хода. Так как цель игры заключается в том, чтобы поставить шах и мат (захватить) королю соперника. По своей сути король совершает такие же ходы, как и ферзь, но с самой главной разницей в том, что имеет ограниченный выбор передвижения на одну соседнюю клетку.

selected-это число, представляющее, где находится выбранная фигура  
moves-это список мест, куда может переместиться выбранная фигура  
captures-это список мест, где выбранный фрагмент может быть захвачен  
game\_board-это список каждого места на доске и говорит, какая фигура там существует  
turn-это число 1 или 2, представляющее, чей это ход.

**3.2 Отладка и тестирование программы**

После того как разработка игры была завершена необходимо провести тестирование игры. Для начала была запущена сама игра и проверка всё ли было установлено. Была обнаружена ошибка в виде отсутствия самого таймера. Оказалось, что забыл прописать одну цифру в скобках, а именно: time.sleep(1)

После этого была совершена проверка на: движения шахматных фигур, поочередность хода команд, могут ли двигаться фигуры, когда королю поставлен шах, работоспособность кнопок, появления окна победы и окна выбора фигур, когда пешка доходит до конца.

После завершения теста игры, произошла смена цветов на более приятные для зрительного осмотра. Здесь применялся сайт с онлайн пипеткой для определения точного кода цвета.

Затем файл с игрой был переделан в exe файл. Всё было хорошо, но оказалось, что файл не сможет запуститься если на компьютере не будет языка python. В файле с расширением bat была прописана ещё одна строка для установки pyinstaller.

Для теста со стороны папка с игрой была переделана в архив дабы иметь возможность кому-либо её отправить. Также в папке был создан документ Readme для краткой инструкции по запуску самой игры. Таким образом игра была отправлена моему другу и с его стороны жалоб по недоработке не было.

**3.3. Руководство пользователя**

Пользователь запускает игру. Перед ним игровое поле, если он играет за «белых», то начинает ход первым. В случае, когда наступает момент хода определённого игрока и он не может совершить ход какой-либо фигурой, то нужно проверить не поставлен ли этому игроку шах и его король под угрозой. В таком случае необходимо избавиться от угрозы короля разными способами:

* Увести короля
* Нейтрализовать вражескую фигуру, которая потенциально может «съесть» короля
* Преградить путь потенциальной убийце короля другой фигурой

Когда пешка доходит до конца своего пути появиться меню выбора кем она теперь может стать, а именно:

* Ферзь
* Конь
* Слон
* Ладья



*Рисунок 16 выбор фигур*

В случае победы одной из сторон показывается сообщение о том какая из сторон победила



*Рисунок 17 победа «белых»*

Чтобы выйти из игры достаточно нажать кнопку «quit».

Для того, чтобы начать игру по новой нажать кнопку «Reset» или «New Game»

**ЗАКЛЮЧЕНИЕ**

В курсовой работе был проведен анализ поставленной проблемы с последовательным её решением. Был рассмотрен и усвоен теоретический материал, что позволило более качественно и быстро справиться с поставленными целями и задачами.

Целью курсовой работы являлось разработка алгоритма работы «шахматы» на языке программирования Python. Можно сделать вывод, что цель достигнута в результате выполнения всех поставленных задач и работ, была создана игра «шахматы» и проведено тестирование.

Были реализованы следующие элементы

* шахматная доска
* игровые фигуры
* таймера
* список «съеденных» фигур и их количество
* кнопки
* окно победы для двух сторон
* подсвечивание ходов и фигур

Была проведена большая работа в ходе, которой удалось закрепить свои навыки по:

* языку программирования Python
* использованию программы Visio
* Использованию интегрированной среды Pycharm

Игра «шахматы» имеет полностью рабочий алгоритм и при желании его можно усовершенствовать.

**БИБЛИОГРАФИЯ**

1. Зэд шоу «Лёгкий способ выучить Python». Зэд шоу. - Москва**: СИНТЕГ, 2020. - 481**c.
2. Кори альтхофф «сам себе программист». Как научиться программировать / Альтхофф, Н.Н. и. - М.: Горячая Линия Телеком, **2020.** **-** 240 c.
3. Марк лутц. Python. Изучаем Python / Марк лутц. - М.: Бином-Пресс**, 2019**. - 590 c.
4. Тимошок, Т. В. Шахматы 2002. Самоучитель / Т.В. Тимошок. - М.: Диалектика, **2017**. - 352 c.
5. Официальный учебный курс по Python. Python 3 2003 (+ CD-ROM). - М.: Эком, Бином. Лаборатория знаний, **2016**. - 528 c.
6. Епанешников, А. М. Практика создания игр на Python / А.М. Епанешников, В.А. Епанешников. - Москва: **Гостехиздат**, **2016**. - 440 c.
7. Гурвиц, Г. Python 3. Разработка приложений на реальном примере / Г. Гурвиц. - М.: БХВ-Петербург, **2018**. - 496 c.
8. Архангельский, А.Я. Программирование в Python 3; Бином - М., **2018. - 258** c.
9. Дарахвелидзе, П.Г.; Марков, Python наиболее полное руководство; БХВ-Петербург - М**., 2018. - 234**c.
10. Шумаков, П.В. Python 3 и разработка игр; Нолидж - М., **2018. -** 704 c.
11. Сухарев, М.В. Основы Python. Профессиональный подход; Наука - М., **2018**. - 600 c.
12. Марков, Е.П.; Никифоров, В.В. Python 2005 для .NET; БХВ-Петербург - М., **2017**. - 896 c.