Министерство образования Российской федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

**Пояснительная записка**

к курсовой работа

по курсу «Программирование»

на тему «Игра Шахматы»

Выполнил:

студент группы 24ВВВ1

Будников А.С.

Приняла:

к.т.н., доцент Генералова А.А.

**Оглавление**

**Введение**

«Шахматы» – это одна из самых древних и популярных интеллектуальных игр, сочетающей в себе элементы науки и спорта. С развитием компьютерных технологий шахматы стали важно областью исследования в программировании за счет возможности изучения разработки алгоритмов анализа игровых стратегий и создания пользовательских интерфейсов.

В данной курсовой работе рассматривается разработка шахматной программы на языке программирования Golang, включая реализацию игровой логики и применения алгоритма поиска лучших ходов. Данная работа является актуальной и важной в контексте создания современных интеллектуальных компьютерных игр.

1. **Постановка задачи**

Необходимо разработать программу — игру «Шахматы» с возможностью игры против компьютера.  
 Программа должна быть интуитивно понятной, с реализованными основными правилами шахматной игры, включая начальную расстановку фигур, правила движения, взятия фигур, шаха, мата и патовой ситуации.

Многомодульность программы**.** Необходимо поделить программу на логические модули с четким разделением логики выполнения. Такой подход значительно упростит отладку и тестирование программы, а также повысит расширяемость проекта.

Интерфейс и пользовательское взаимодействие. Программа должна работать в текстовом или графическом режиме. Необходимо определить тип пользовательского интерфейса и разработать его структуру. В случае графического режима – предусмотреть визуальные элементы управления с возможностью управления при помощи компьютерной мыши.Программа должна поддерживать ввод данных с клавиатуры (или мыши), анализировать действия игрока (выбор и перемещение фигур), отображать ходы и информировать о состоянии партии.

Устройство ввода информации **–** клавиатура и мышь. Необходимо реализовать обработку пользовательского ввода таким образом, чтобы все действия, производимые пользователем, были однозначно интерпретируемы.

Необходимо реализовать простой алгоритм игры компьютера, возможным вариантом может стать алгоритм оценки позиции и выбора лучшего хода.

**2. Выбор решения**

При запуске программы на экран пользователя выводится приветственный экран с заставкой и меню, состоящее из 3 пунктов:

1. Запуск игры. На экран выводится пользовательский интерфейс с возможностью начать игру.
2. Настройка. Пользователь изменяет необходимые параметры программы по средством ввода предложенных ему команд.
3. Выход. Программа завершается без сохранения данных.

Разработанная программа состоит из 11 модулей:

1. main.go
2. ui.go
3. custom\_button.go
4. evaluation.go
5. board.go
6. piece.go
7. generator.go
8. move.go
9. minimax.go
10. popcount.go
11. popcount.s

Файл main.go – главный файл программы, служащий для инициализации приложения, вывода главного меню и создания лог-файла.

Файл ui.go нужен для отображения пользовательского интерфейса и обработки событий при взаимодействии пользователя с ним.

Файл custom\_button.go нужен для инициализации кнопки, использующейся в качестве клетки на шахматной доске.

Файл evaluation.go нужен для реализации функций оценки позиции.

Файл board.go нужен для инициализации шахматной доски.

Файл piece.go нужен для инициализации переменных шахматных фигур.

Файл generator.go содержит описание функций, использующихся для генерации возможных ходов фигур.

Файл move.go содержит описание функций, использующихся для перемещения фигур по шахматной доске.

Файл minimax.go содержит описание функций выбора лучшего хода.

Файл popcount.go содержит объявление функции, реализованной на языке Ассемблера.

Файл popcount.s содержит описание функции, объявленной в файле popcount.go.

**3.Описание разработки программы**

В рамках разработки данной программы в качестве основного языка программирования был выбран Go (Golang). Этот язык представляет собой современное и востребованное средство разработки, отличающееся простым и лаконичным синтаксисом, высокой производительностью, а также возможностью взаимодействия с языком Ассемблера. Go ориентирован на эффективную работу с многопоточностью, что делает его особенно удобным для создания масштабируемых прикладных решений. Язык обеспечивает высокоуровневые абстракции при сохранении контроля над управлением памятью и производительностью, предоставляет встроенные средства параллельного выполнения, сборки мусора и строгую типизацию. Кроме того, наличие системы пакетов позволяет структурировать проект на модули, обеспечивая тем самым упорядоченность архитектуры и удобство сопровождения.

Для разработки были использованы среды Visual Studio Code и NeoVim, обладающие широкими возможностями для поддержки Go за счёт расширений и LSP-серверов. Оба инструмента обеспечивают функции статического анализа, отладки и интеллектуальных подсказок, что значительно упрощает и ускоряет процесс программирования.

После запуска программы пользователю отображается заставка и главное меню. Управление осуществляется путём ввода соответствующего номера пункта меню. Начать игру можно, выбрав пункт 1 – «Начать игру», переход к настройкам осуществляется через пункт 2, в котором предоставляется возможность изменения параметров шахматного движка и других опций программы. Завершение работы доступно через пункт 3 – «Выход».

Начальный этап разработки включал реализацию приветственного экрана и главного меню, функционирующего на основе бесконечного цикла и встроенной функции bufio.Reader.Read() для считывания пользовательского ввода из терминала. На основе данной логики позднее была реализована консоль ввода команд, доступная в процессе игры. Блок-схема функции main представлена на соответствующем рисунке.

Ключевым этапом разработки шахматного движка стало создание двух основных структур: шахматных фигур и игровой доски. Для описания фигур был реализован отдельный модуль, содержащий константы для обозначения типов фигур и их цвета, а также структуру клетки доски с соответствующими полями. Игровая доска была реализована в виде двумерного массива обобщённого типа Board[8][8]Square. Дополнительно были разработаны функции для инициализации доски, получения и перемещения фигур, проверки состояния клетки, а также создания «глубокой копии»[[1]](#footnote-2) доски, необходимой для алгоритма оценки позиции.

Затем был разработан модуль evaluation, содержащий оценочные функции на основе стоимости фигур и оценки позиционного преимущества. В модуле реализована карта соответствия фигура-ценность, массив бонусных значений для центральных клеток и функция оценки безопасности короля. Одна из функций, реализованных в этом модуле, написана на языке Ассемблера в файле popcount.s и предназначена для подсчёта количества фигур заданного цвета. Блок-схемы файлов evaluation.go и popcount.s представлены на рисунке.

Для корректной работы движка были реализованы алгоритмы генерации и валидации ходов, а также перемещения фигур. Для этого созданы файлы move.go и generator.go. В последнем определены ключевые функции генерации возможных ходов для каждой фигуры, оптимизированные за счёт использования универсальных функций для прямолинейных и диагональных перемещений. В move.go описаны следующие функции:

1. abs(x int) int – вспомогательная функция для вычисления абсолютного значения числа;
2. MakeMove(b \*board.Board, m Move) error – функция выполнения хода с учётом правил и проверки безопасности короля;
3. IsKingInCheck(b board.Board, color board.Color) bool – функция, определяющая, находится ли король под шахом.

Блок-схемы файлов move.go и generator.go представлены на соответствующих рисунках.

Следующим этапом стала реализация алгоритмов поиска, сортировки и выбора оптимального хода. В качестве основного алгоритма был использован Minimax с альфа-бета-отсечением, сочетающий простоту реализации с высокой эффективностью. Данный алгоритм был дополнен следующими компонентами: транспозиционной таблицей для хранения оценок, алгоритмом поиска покоя (для устранения «горизонтального эффекта»[[2]](#footnote-3)), функциями сортировки и эвристиками, учитывающими стратегическую важность центральных клеток. Полная реализация представлена в файле minimax.go, блок-схема которого приведена на рисунке.

Заключительным этапом стало создание графического интерфейса, обеспечивающего взаимодействие пользователя с программой. Для этого были реализованы файлы custom\_button.go и ui.go в соответствующем пакете ui. Первый отвечает за обработку событий мыши и выбор клеток на доске, второй — за построение и отрисовку графического интерфейса. В качестве библиотеки был выбран популярный фреймворк[[3]](#footnote-4) Fyne, обеспечивающий удобные средства для создания GUI [[4]](#footnote-5)в Go. Реализованы функции запуска интерфейса, обработки команд, воспроизведения звука при перемещении фигуры и обновления отображения доски.

Блок-схемы файлов custom\_button.go и ui.go приведены на рисунках.

**4.Отладка и тестирование**

В качестве основных инструментов разработки использовались Visual Studio Code и NeoVim. Оба редактора обеспечивают необходимые средства для эффективной работы с языком Go, включая поддержку LSP (Language Server Protocol)[[5]](#footnote-6), статического анализа кода, автодополнения, навигации по проекту и встроенных терминалов. Visual Studio Code был использован как основная среда для структурированной разработки и отладки, а NeoVim — в качестве лёгкого и настраиваемого инструмента для быстрого редактирования кода и работы в терминале.

Отладка производилась с использованием расширений, поддерживающих пошаговое выполнение кода, установку точек останова, анализ значений переменных и состояния памяти во время выполнения. Это позволяло эффективно выявлять и устранять ошибки в логике работы шахматного движка, особенно на этапах взаимодействия с Ассемблером и реализации алгоритмов поиска.

Тестирование осуществлялось поэтапно, в процессе разработки каждого модуля, а также после завершения реализации основных функций программы. В результате тестирования были обнаружены и устранены многочисленные ошибки, связанные с обработкой игровых структур, системой оценки позиций и корректностью исполнения ассемблерных подпрограмм.

**5. Описание программы**

**6.Руководство пользователя**

**Заключение**

**Библиографический список**

1. <https://www.chessprogramming.org>
2. Корнилов Е.Н. «Программирование шахмат и других логических игр.» — СПб.: БХВ-Петербург, 2005.
3. Михалис Цукалос. «Golang для профи», 2020.
4. <https://talkchess.com>
5. <https://habr.com/ru/articles/682122/>
6. <https://chess.fandom.com>
7. Claude E. Shannon, David Levy «Computer Chess Compendium», 1988.

**Листинг программы**

1. совершенно новая копия исходного объекта вместе со всеми вложенными объектами, которые он содержит [↑](#footnote-ref-2)
2. ситуация, при которой поиск останавливается на позиции, которая кажется стабильной, но на самом деле может быть нестабильной из-за потенциальных тактических угроз. [↑](#footnote-ref-3)
3. заготовка, готовая модель в программировании для быстрой разработки, на основе которой можно дописать собственный код [↑](#footnote-ref-4)
4. графический пользовательский интерфейс [↑](#footnote-ref-5)
5. протокол языкового сервера, набор правил, по которому среда программирования связывается с сервером и получает от него инструкции. Протокол нужен, чтобы программы для разработки могли автоматически подставлять данные в код [↑](#footnote-ref-6)