МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

на курсовую работу

по дисциплине «Алгоритмы и структуры данных» Тема «Компьютерная логическая игра “Канадские шашки”»

Р.02069337.23/709-05

Листов 36

Руководитель разработки:

Кандидат технических наук, доцент кафедры «ИВК»

Шишкин Вадим Викторинович

« » 2024г.

Исполнитель:

студент гр. ИСТбд-22

Гончаров Тимур Евгеньевич

« » 2024г.

2024 г.

Содержание

Аннотация 2

Техническое задание 3

Пояснительная записка 8

Руководство программиста 18

Текст программы 27

**Аннотация**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**на курсовую работу**

**по дисциплине «Алгоритмы и структуры данных» Тема** «Компьютерная логическая игра “Канадские шашки”»

Р.02069337.23/709-05

Листов 5

Исполнитель:

студент гр. ИСТбд-22

Гончаров Тимур Евгеньевич

« » 2024г.

2024 г.

##### Введение

Название приложения «Канадские шашки», условное обозначение «*Canadian Checkers*».

Канадские шашки являются вариантом международных шашек, но имеют существенные отличия в размере игрового поля и количестве игровых фигур. Игра ведется на доске размером 12×12 клеток, между двумя соперниками (игрок против игрока). В начальной позиции у каждого игрока по 30 шашек, расположенных на чёрных клетках первых пяти горизонталей с каждой стороны. Основной целью игры является взятие всех шашек противника или запереть шашки противника.

Правила игры предусматривают, что простые шашки могут ходить только по диагонали вперед на одно поле. Если шашка противника находится на соседней клетке, и за ней имеется свободная клетка, игрок может взять эту шашку. При достижении последней горизонтали обычная шашка имеет два типа ходов. Первый - Обычная шашка превращается в дамку и ход переходит другому игроку. Второй – Обычная шашка имеет возможность продолжить взятие шашки противника, но при этом дамкой не становится. Взятие шашек противника является обязательным, причем при наличии нескольких вариантов взятия игрок обязан выбрать тот, при котором будет взято максимальное количество шашек противника.

Функциональные возможности:

* реализация авторизации и регистрации;
* простая и интуитивно понятная графика игрового поля;
* реализация пошагового игрового процесса;
* возможность выбора направления движения шашек;
* автоматическое удаление взятых шашек с игрового поля;
* автоматическое завершение игры.

##### Основания для разработки

Учебная программа по дисциплине «Алгоритмы и структуры данных» по специальности 09.03.02 «Информационные системы и технологии».

##### Требования к программе или программному изделию

* 1. **Функциональное назначение**

Программа предназначена для реализации настольной игры «Канадские шашки» между двумя игроками на одном компьютере. Основной целью является предоставление игрокам удобного и интуитивно понятного интерфейса для игры, обучения правилам, а также анализа игровых стратегий.

Перечень автоматизируемых процессов:

* автоматизация расстановки и перемещения шашек на игровом поле;
* проверка корректности ходов и соблюдения правил игры;
* автоматическое удаление взятых шашек с доски;
* ведение счета взятых шашек для каждого игрока;
* подсчет очков взятых шашек и определение победителя.

Основная группа пользователей – игроки, желающие сыграть друг с другом на одном устройстве.

* 1. **Требования к функциональным характеристикам**
     1. **Требования к структуре приложения**

Приложение должно быть разработано в виде одного модуля с дополнительными информационными файлами при необходимости.

* + 1. **Требования к составу функций приложения**

Основные функции приложения:

* регистрация/авторизация пользователя;
* отображение навигационного меню пользователя;
* отрисовка игрового поля;
* взаимодействие с пользователем;
* интерактивные прием, проверка правильности и отрисовка хода пользователя;
* проверка окончания игры;
* информирование пользователя об окончании игры и победителе.
  + 1. **Требования к организации информационного обеспечения, входных и выходных данных**

В приложении должен быть реализован графический интерфейс взаимодействия с пользователем. Логин и пароль пользователя должны вводиться с клавиатуры. Логины и пароли зарегистрированных пользователей должны храниться в отдельном файле или базе данных в зашифрованном виде. Пояснительные информационные сообщения для пользователя должны выводиться справа игрового поля по ходу игры.

* 1. **Требования к надежности**

Приложение должно обеспечивать устойчивую работу при любых корректных действиях пользователей.

* 1. **Требования к информационной и программной совместимости**

Операционная система: *Windows* 10/11.

Язык программирования: *Python* (рекомендуется версия 3.8+).

Среда разработки: *PyCharm*.

Библиотеки:

* tkinter *–* используется для создания графического интерфейса пользователя;
* hashlib – используется для хеширования данных,
* enum – используется для определения типов,
* Python Imaging Library(PIL) - используется для работы с изображениями,
* pathlib – используется для работы с путями к изображениям,
* time – используется для создания анимации,
* json – используется для хранения данных пользователей
  1. **Требования к маркировке и упаковке**

Требования к маркировке и упаковке не предъявляются.

* 1. **Требования к транспортированию и хранению**
     1. **Условия транспортирования**

Требования к условиям транспортирования не предъявляются.

* + 1. **Условия хранения**

Обеспечение свободного доступа к проекту в репозитории до окончания срока учебы.

* + 1. **Сроки хранения**

Срок хранения – до окончания срока учебы».

##### Требования к программной документации

В состав программной документации должны входить:

• Техническое задание;

• Руководство пользователя;

• Пояснительная записка;

##### Стадии и этапы разработки

Этап 1: Анализ требований к проекту.

Этап 2: Разработка Технического задания.

Этап 3: Проектирование архитектуры программы.

Этап 4: Реализация программы.

Этап 5: Тестирование и исправление ошибок.

Этап 6: Документирование.

Этап 7: Сдача программного продукта и приемка.

##### Порядок контроля и приемки

Контроль и приемка программного обеспечения осуществляются в соответствии с планом тестирования, который включает проверку следующих функциональных возможностей:

• Корректная работа системы регистрации, авторизации, шифрования, а также процесса игры, включая отображение доски и обычных шашек, их перемещение;

• Проверка победителя по окончания игры

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

# Курсовая работа

**По дисциплине «Алгоритмы и структуры данных»**

**Тема** «Компьютерная логическая игра “Канадские шашки”»

# Пояснительная записка

Р.02069337.23/709-05

Листов 9

Исполнитель:

студент гр. ИСТбд-22

Гончаров Тимур Евгеньевич

« » 2024г.

2024 г.

##### Введение

Название приложения «Канадские шашки», условное обозначение «*Canadian Checkers*».

Разработанное приложение представляет собой полноценную реализацию игры в канадские шашки с графическим пользовательским интерфейсом. В основе приложения лежит система авторизации пользователей, которая обеспечивает безопасный доступ к игре. Данные пользователей хранятся в JSON-файле с применением хеширования паролей для обеспечения безопасности.

Игровой процесс построен на классических правилах канадских шашек, где игра ведется на доске размером 12x12 клеток. Приложение полностью поддерживает все правила игры, включая обязательное взятие шашек противника, превращение простых шашек в дамки при достижении последней линии, а также специальные правила ходов для дамок (Первый – Обычная шашка превращается в дамку и ход переходит другому игроку. Второй – Обычная шашка имеет возможность продолжить взятие шашки противника, но при этом дамкой не становится). Игровая логика реализует автоматическое определение всех возможных ходов и проверку их корректности.

##### Проектная часть

* 1. **Постановка задачи на разработку приложения**

Разработка приложения выполнялась по следующим задачам:

* + - * Разработка системы авторизации пользователей, которая должна обеспечивать безопасную регистрацию новых игроков и вход существующих пользователей в систему. Система включает в себя хранение учетных данных в JSON-файле с обязательным хешированием паролей для обеспечения безопасности. Также требуется реализовать проверку корректности введенных данных и обработку ошибок при авторизации.
      * Создание базовой игровой логики канадских шашек. Все правила игры, включая движение обычных шашек и дамок, обязательного взятия шашек противника, превращение обычных шашек в дамки при достижении последней линии доски. Разработка алгоритмов проверки возможных ходов и определения победителя партии.
      * Разработка графического пользовательского интерфейса с использованием библиотеки Tkinter. Интерфейс должен включать адаптивное игровое поле размером 12x12 клеток, которое будет корректно масштабироваться при изменении размера окна.
      * Создание информационной панели для отображения текущего состояния игры, включая очередность хода и количество взятых шашек.
      * Реализация визуальных эффектов и анимации шашек при перемещении, подсветки возможных ходов и выделение выбранной шашки.
      * Разработка системы управления игровым процессом, включающую возможность начать новую игру, сдаться, выйти из приложения.
  1. **Математические методы**

В основе игры лежит ориентированный граф G = (V, E), где V - множество вершин, представляющих позиции на игровом поле, а E - множество ребер, представляющих возможные ходы. Игровое поле моделируется в виде двумерной прямоугольной сетки размером 12×12.

Множество вершин V определяется как:

V = {vi,j | i, j ∈ {0, 1, ..., 11}}, где i - номер строки, j - номер столбца.

Каждой вершине vi,j соответствует состояние si,j из множества состояний S:

S = {empty, white\_checker, black\_checker, white\_queen, black\_queen}

Множество ребер E состоит из упорядоченных пар вершин и определяется правилами ходов:

E = {(vi,j, vk,l) | существует допустимый ход из позиции (i,j) в позицию (k,l)}

Веса ребер w(e) для e ∈ E определяются следующим образом:

w(e) = 1 для простого хода

w(e) = 2 для хода со взятием одной шашки

w(e) > 2 для последовательного взятия нескольких шашек

Направления движения задаются векторами смещения:

D = {(-1,-1), (1,-1), (-1,1), (1,1)}, где каждая пара представляет смещение по координатам (Δi, Δj)

Для простых шашек множество достижимых вершин определяется как:

Reach(vi,j) = {vk,l | |k-i| = 1, |l-j| = 1, направление соответствует цвету шашки}

Для дамок множество достижимых вершин расширяется:

ReachQueen(vi,j) = {vk,l | k-i = ±n, l-j = ±n, n ∈ {1,...,11}, путь свободен}

Функция взятия шашек определяется как:

Capture(vi,j) = {vk,l | существует шашка противника на пути из vi,j в vk,l}

Условие превращения в дамку:

Transform(vi,j) = true, если:

* для белой шашки: j = 0
* для черной шашки: j = 11

Множество возможных ходов для позиции определяется как:

Moves(vi,j) = {

Capture(vi,j), если Capture(vi,j) ≠ ∅

Reach(vi,j), если Capture(vi,j) = ∅

}

Состояние игры в любой момент времени описывается подграфом G' = (V, E'), где E' ⊆ E - множество допустимых ходов в текущей позиции. После каждого хода происходит пересчет множества E' с учетом новых позиций шашек и правил игры.

Победное состояние достигается, когда для одного из игроков:

Win(player) = true, если:

Moves(vi,j) = ∅ для всех шашек противника, или

На поле не осталось шашек противника

**Обоснование выбора модели:**

* Графовая модель является оптимальным выбором для реализации канадских шашек благодаря нескольким ключевым преимуществам:
  + Естественным образом отражает структуру игрового поля и возможные перемещения шашек, где вершины представляют позиции, а ребра - допустимые ходы.
  + Эффективна при реализации сложных игровых механик, таких как взятие нескольких шашек за один ход, где последовательность взятий может быть представлена как путь в графе. Динамическая природа графа позволяет легко обновлять состояние игры после каждого хода, пересчитывая множество допустимых ребер.
  + Упрощает поиск возможных ходов и проверку правил игры. Алгоритмы обхода графа позволяют эффективно находить все доступные перемещения для каждой шашки.
  1. **Архитектура и алгоритмы**
     1. **Архитектура**

Приложение построено с использованием архитектуры на основе объектно-ориентированного программирования (ООП), где основным объектом является игровое поле. Архитектура приложения представлена на рисунке 1.

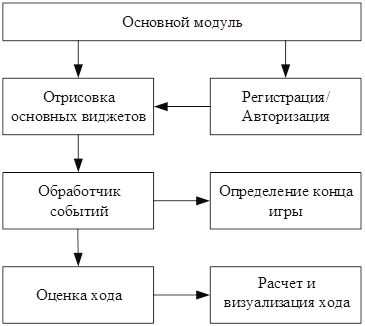


Рисунок 1. Архитектура приложения «Канадские шашки»

Основные функции приложения:

* handle\_move — обрабатывает ход игрока, проверяет его корректность, выполняет перемещение шашки и обновляет состояние игры. После выполнения хода проверяет возможность дополнительных взятий;
* get\_required\_moves\_list — анализирует текущую позицию и возвращает список всех обязательных ходов со взятием для текущего игрока. Проверяет возможности взятия как для обычных шашек, так и для дамок;
* get\_optional\_moves\_list — формирует список всех возможных ходов без взятия для текущего игрока в текущей позиции. Учитывает различные правила движения для простых шашек и дамок;
* animate\_move — выполняет анимацию перемещения шашки от начальной до конечной позиции. Обеспечивает плавное визуальное отображение хода;
* check\_for\_game\_over — проверяет условия завершения игры: отсутствие возможных ходов у игрока или полное уничтожение шашек одной из сторон. При выполнении условий объявляет победителя;
* draw\_possible\_moves — отображает на игровом поле все возможные ходы для выбранной шашки, включая как обычные перемещения, так и ходы со взятием. Помогает игроку визуально оценить доступные варианты;
* handle\_player\_turn — управляет очередностью ходов, переключает активного игрока после завершения хода, проверяет наличие дополнительных ходов при множественном взятии.
  + 1. **Алгоритм «Поиска возможных ходов»**

На рисунке 2 представлен алгоритм поиска возможных ходов.

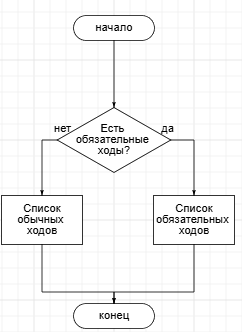


Рисунок 2. Алгоритм «Поиска возможных ходов»

* + 1. **Алгоритм «Проверки взятия шашек»**

На рисунке 3 представлен алгоритм авторизации пользователя

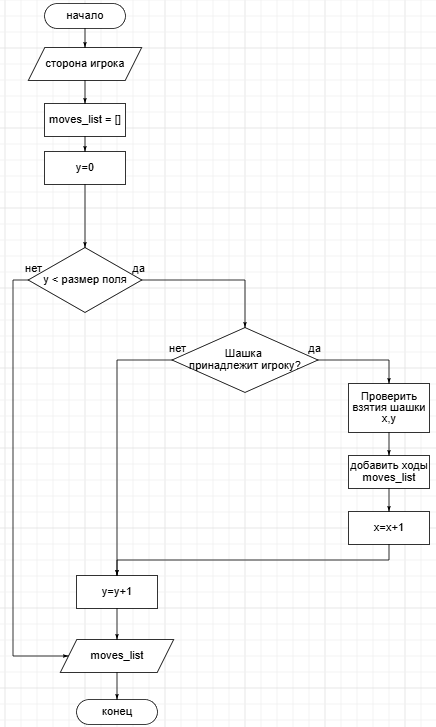


Рисунок 3. Алгоритм «Проверки взятия шашек»

* + 1. **Алгоритм «Авторизации»**

На рисунке 4 представлен алгоритм авторизации пользователя

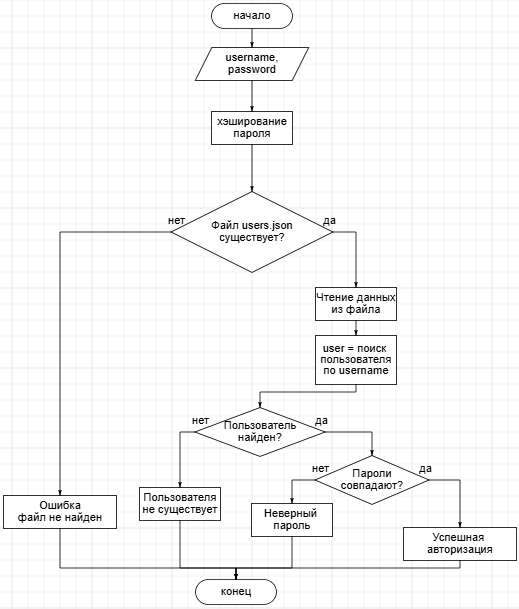


Рисунок 4. Алгоритм «Авторизация»

**1.4. Тестирование**

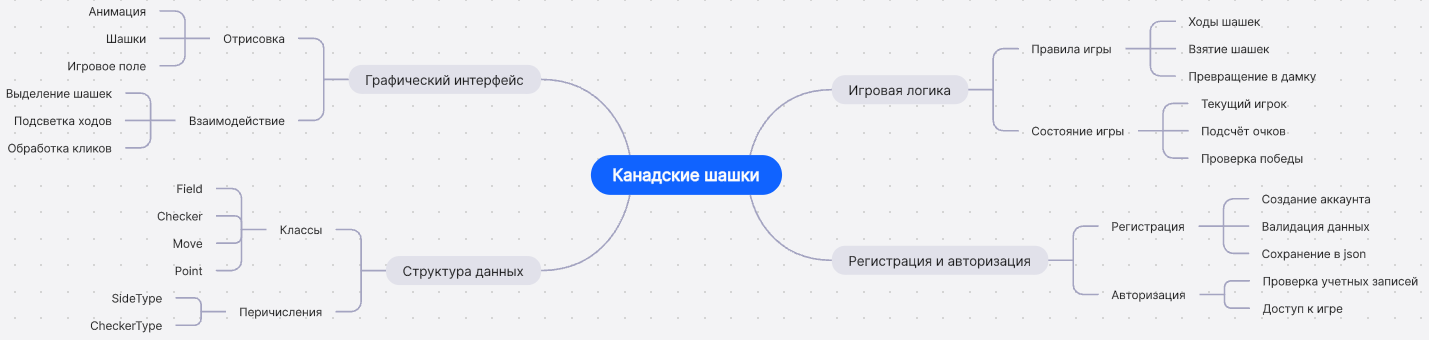


Рисунок 4. Mind Map

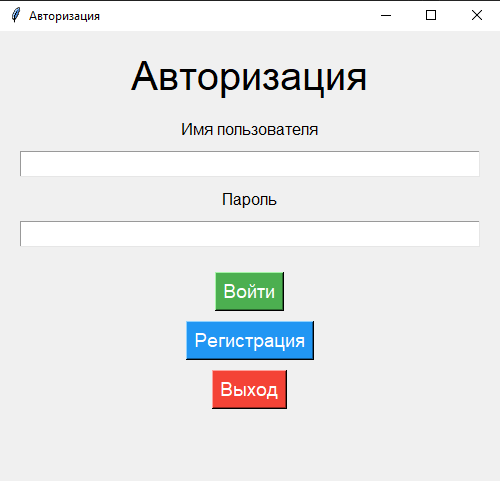


Рисунок 5. Окно «Авторизация»

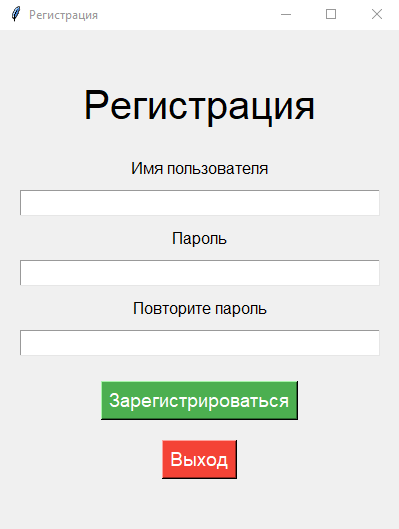


Рисунок 6. Окно «Регистрация»

**1.4.1. Описание отчета о тестировании**

Отчет о тестировании приложения "Канадские шашки" содержит информацию о проведенных тестах, сценариях тестирования, полученных результатах и выводах.

**1.4.2. Цель тестирования**

Цель тестирования – проверить корректность работы всех функций приложения, выявить ошибки и дефекты, обеспечить стабильную работу приложения и соответствие его заданным требованиям.

**1.4.3. Методика тестирования**

Для тестирования приложения "Канадские шашки" применялись следующие методы:

* **Функциональное тестирование:** Проверка корректности работы каждой функции приложения.
* **Тестирование пользовательского интерфейса:** Проверка удобства использования интерфейса.
* **Тестирование на граничных значениях:** Проверка корректности работы приложения на границах допустимых значений параметров.
* **Тестирование производительности:** Проверка скорости работы приложения, особенно при большом количестве ходов.

**1.4.4. Проведенные тесты**

**Тест 1: Создание новой игры**

* **Описание теста:** Проверка возможности запуска новой игры, очистка игрового поля, установка начальных условий.
* **Результат:** Тест пройден успешно.

**Тест 2: Ход игрока**

* **Описание теста:** Выбор игроком клетки шашки и его размещение на другую клетку, проверка обновления состояния игрового поля.
* **Результат:** Тест пройден успешно.

**Тест 3: Ход компьютера**

* **Описание теста:** Проверка корректности выбора клетки компьютером, размещение шашки компьютера в клетку, проверка обновления состояния игрового поля.
* **Результат:** Тест пройден успешно.

**Тест 4: Проверка победы**

* **Описание теста:** Проверка на наличие возможных ходов пользователем или компьютера, выведение сообщения о победе.
* **Результат:** Тест пройден успешно.

**Тест 5: Проверка ничьей**

* **Описание теста:** Проверка на наличии возможных ходов пользователем и компьютером, выведение сообщения о ничьей.
* **Результат:** Тест пройден успешно.

**Тест 6: Возможность повтора игры**

* **Описание теста:** Проверка возможности начать новую игру после завершения текущей.
* **Результат:** Тест пройден успешно.

**1.4.5. Выводы**

В ходе тестирования не было выявлено критических ошибок. Приложение "Канадские шашки" работает стабильно, корректно выполняет все заявленные функции и имеет удобный и понятный интерфейс.

##### Источники, использованные при разработке

* 1. Вирт Н. Алгоритмы + структуры данных = программы: Пер. с англ. / Н. Вирт. – Москва : Мир, 1985. – 406 с.
  2. Архитектура программного обеспечения [Электронный ресурс]. – Режим доступа: URL: [https://dic.academic.ru/dic.nsf/ruwiki/14691](https://dic.academic.ru/dic.nsf/ruwiki/146913)3 (дата обращения : 01.12.2024)
  3. Назина, О.Е. Что такое тестирование. Курс молодого бойца / О.Е. Назина.  Санкт-Петербург : БХВ-Петербург, 2022.  592 с.: ил.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | |  | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | |  | | Руководство программиста  на курсовую работу  по дисциплине «Алгоритмы и структуры данных»  Тема «Канадские шашки» | | | | | | |
|  | | |  | |  | | | |
|  | | Исполнитель  студент гр. ИСТбд-22  Гончаров Т. Е  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | |
|  | | 2024 | | | | | | |

**1. Назначение и условия применения программы**

**1.1. Назначение и функции, выполняемые приложением**

**Назначение:**

Приложение "Крестики-нолики" представляет собой компьютерную реализацию классической настольной игры, известной как "Крестики-нолики". Оно позволяет игрокам (человеку или компьютеру) соревноваться в стратегическом построении своих символов на игровом поле.

**Функции:**

* **Создание новой игры:** Инициализация нового игрового поля 3x3, очистка предыдущих данных игры.
* **Ход игрока:** Игрок выбирает клетку на поле, чтобы разместить свой символ (крестик).
* **Ход компьютера:** Компьютер выбирает клетку для определения наиболее подходящей клетки для размещения символа (нолика).
* **Проверка победы:** Проверяет каждую строку, столбец и диагональ на наличие трех одинаковых символов, состоящих из крестиков или ноликов.
* **Проверка ничьей:** Проверяет, заполнены все клетки на поле символами (крестиками и ноликами).
* **Отображение результата:** Вывод сообщения о победе компьютера или о ничьей.

**1.2. Условия, необходимые для использования приложения**

* **Операционная система:** Windows, macOS, Linux.
* **Язык программирования:** Python.
* **Инструментальная среда:** Любая среда разработки Python (например, PyCharm, Visual Studio Code).
* **Библиотеки:**
  + tkinter (для создания графического интерфейса).

**2. Характеристики программы**

**2.1. Характеристики приложения**

* **Количество строк кода:** Примерно 133 строк кода.
* **Количество структур данных:**
  + **Игровое поле:** Двумерный массив 3x3 для хранения состояния игры.
  + **Текущий игрок:** Переменная для хранения текущего игрока или компьютера (крестик или нолик).
  + **Результат игры:** Переменная для хранения результата (выигрыш, ничья).
* **Количество алгоритмов:**
  + **Алгоритм проверки на выигрыш:** Проверка всех строк, столбцов и диагоналей на наличие выигрышной комбинации (трех символов крестиков или ноликов на игровом поле).
  + **Алгоритм проверки на ничью:** Проверка, заполнены ли все клетки на поле.
  + **Алгоритм хода компьютера:** использован алгоритм minimax, для определения хода компьютера.
  + **Алгоритм отрисовки игрового поля:** Отрисовка игрового поля и символов на экране.
  + **Алгоритмы обработки событий:** Обработка действий пользователя, таких как нажатие кнопок или выбор клетки на поле.
* **Используемые библиотеки:** tkinter (для создания графического интерфейса).
* **Работа приложения:**
  + Игровое поле инициализируется и отображается на экране.
  + Игрок и компьютер делают ходы по очереди, выбирая клетки на поле.
  + После каждого хода проверяется наличие выигрыша или ничьей.
  + В случае победы или ничьей выводится соответствующее сообщение.
  + Игроку предлагается начать новую игру.
* **Внешний вид приложения:** Приложение имеет простой графический интерфейс с полем размером 3 на 3 и кнопки для новой игры.
* **Средства контроля корректности ввода/вывода:**
  + **Проверка выбора клетки:** Проверка, является ли выбранная клетка пустой.
  + **Проверка границ поля:** Проверка, находится ли выбранная клетка в пределах игрового поля (3x3).
  + **Проверка окончания игры:** Проверка, не завершена ли уже игра (выигрыш или ничья).
  + **Обработка ошибок:** Вывод соответствующих сообщений в случае некорректного ввода.

**2.2. Особенности реализации приложения**

* **Структуры данных:**
  + **Двумерный массив:** Используется для представления игрового поля, где каждый элемент массива соответствует клетке на поле и хранит информацию о его состоянии (пусто, крестик, нолик).
  + **Переменные:** Используются для хранения текущего игрока или компьютера и результата игры.
* **Обоснование выбора:**
  + **Двумерный массив:** Удобен для хранения состояния игрового поля, так как обеспечивает доступ к клеткам по индексу.
  + **Переменные:** Необходимы для хранения данных, которые меняются в течение игры.
  + **Список или стек:** Удобны для реализации функций отмены хода, просмотра истории игры, повтора игры или анализа хода.
* **Заимствованные решения:**
  + **Алгоритм хода компьютера:**
    - **Minimax:**

1. Проверка на победу:   
   Алгоритм сначала проверяет, есть ли возможность для компьютера выиграть на следующем ходу. Он перебирает все свободные клетки и ставит в них свой символ "O". Если после этого вызывается функция `self.check\_win()`, которая возвращает `True`, то компьютер делает этот ход и выигрывает. Если нет возможности выиграть, то алгоритм проверяет, есть ли возможность заблокировать победу противника (игрока).
2. Проверка на блокирование:

Аналогично предыдущему пункту, компьютер перебирает все свободные клетки, ставит в них символ противника "X" и проверяет, выигрывает ли противник. Если да, то компьютер ставит "O" в эту клетку, чтобы заблокировать ход противника.

1. Случайный ход:

Если нет ни возможности выиграть, ни заблокировать победу противника, то компьютер делает случайный ход. Это не идеальный алгоритм, но он гарантирует, что компьютер не проиграет, если у игрока нет возможности выиграть..

* + **Графический интерфейс:**
    - Если используются элементы графического интерфейса из сторонних библиотек, то необходимо указать источник этих элементов (библиотека, документация).

**3. Обращение к программе**

* **Методы:**
  + **new\_game():** Инициализация нового игрового поля, выбор режима игры, очистка предыдущих результатов.
  + **make\_move (x, y):** Размещение символа игрока в выбранную клетку двумерного массива.
  + **computer\_move ():** Выбор клетки для размещения символа компьютера.
  + **check\_win ():** Проверка игрового поля на наличие выигрышной комбинации (в ряд построенных 3 символа из крестиков или ноликов)
  + **check\_draw ():** Проверка, заполнены ли все клетки на поле без выигрышной комбинации.
  + **show\_result (message):** Вывод сообщения о победе компьютера или ничьей.

**4. Сообщения**

По результатам выполнения кода могут быть следующие результаты:

1. Победа компьютера, если собраны 3 в ряд символа.
2. Ничья, если игровое поле занято, но не собраны 3 в ряд символа.

**Листинг программы**