НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р. Е. Алексеева

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА к курсовому проекту

Куликова Елена Александровна

(фамилия, имя, отчество)

Факультет ИРИТ

Кафедра ИСУ

Группа 18-ИСТ-4

День защиты « » 2020 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е. Алексеева

Кафедра Информатики и систем управления

Заведующий кафедрой

_____ Соколова Э.С.
(подпись) *(фамилия и. о.)*

_____
(дата)

Программа играющая в игру «щёлк»
(наименование темы проекта или работы)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

(вид документа-проект дипломный, курсовой, исследовательская работа или часть и т. п.)

КОНСУЛЬТАНТЫ:

1.По _____

_____ _____
(подпись) *(фамилия и. о.)*

_____
(дата)

2.По _____

_____ _____
(подпись) *(фамилия и. о.)*

_____
(дата)

3.По _____

_____ _____
(подпись) *(фамилия и. о.)*

_____
(дата)

РУКОВОДИТЕЛЬ:

_____ Капранов С.Н.
(подпись) *(фамилия и. о.)*

_____
(дата)

СТУДЕНТ

_____ Куликова Е.А.
(подпись) *(фамилия и. о.)*

_____ 18-ИСТ-4
(дата) *(группа или шифр)*

Проект защищен _____

Протокол № _____

С оценкой _____

РЕЦЕНЗЕНТ

_____ _____
(подпись) *(фамилия и. о.)*

_____
(дата)

2020 г.

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е. Алексеева

Кафедра: Информатики и систем управления

УТВЕРЖДАЮ
Зав. кафедрой

З А Д А Н И Е

на курсовое проектирование

Студент: Куликова Е.А. Группа: 18-ИСТ-4

Тема курсового проекта: Программа играющая в игру «Щёлк»

Исходные данные к проекту: Программа должна: оптимально рассчитывать наилучшие стратегии игры, при которых компьютер должен выиграть; давать возможность пользователю выбирать размеры игрового поля; давать возможность пользователю выбирать первенство хода; визуализировать процесс игры. Количественные характеристики: максимальные размеры поля 5 x 5; минимальные размеры поля 1 x 1; в игру может играть только один человек и один компьютер.

Содержание графического материала:

чертежи:
1. Изображения деревьев вариантов сыгранных партий
2. Дерево игры с альфа-бета отсечением
3. Демонстрация интерфейса
8. Скриншоты готовой программы

Содержание пояснительной записки:

1. Титульный лист
2. Задание на проектирование
3. Лист содержания
4. Перечень вопросов, подлежащих разработке
5. Разработка и анализ технического задания
6. Разработка программного обеспечения
7. Руководство пользователя
8. Документация
9. Список используемой литературы
10. Листинги

Руководитель: _____

"..." ..... 2020 г. Студент: Куликова Е.А.

Содержание

Введение

В настоящее время машинное обучение представляет собой актуальное и стремительно развивающееся направление в области информационных технологий, поскольку нынешнее аппаратное обеспечение обладает высокой эффективностью, необходимой для решения подобных задач. Цель машинного обучения — создание искусственного интеллекта, способного самостоятельно принимать решения, но на данном этапе развития технологии главное — автоматизировать процесс принятия решений человеком. В рамках данной курсовой работы исследован и реализован алгоритм, способный принимать оптимальные решения для выбора хода в игре «Щёлк», а также представлен метод оптимизации полученного решения. Результаты работы алгоритма предоставлены и проанализированы.

1 Разработка и анализ технического задания

1.1 Исследование предметной области

Правила игры – двое по очереди "откусывают" от прямоугольной доски. Игрок выбирает любое поле доски и снимает все фишки, которые находятся не выше и не левее избранного поля. Снявший последнею фишку – проигрывает.

«Щёлк» (от англ. Chomp) — математическая игра, заключающаяся в поедании двумя игроками плитки шоколада по определённым правилам. Современная геометрическая формулировка была придумана придумана Дэвидом Гейлом в 1974 году.

С точки зрения теории игр, «Щёлк» — беспристрастная детерминированная игра с совершенной информацией. Кроме того, в игре конечное число состояний, а потому из общих утверждений теории игр следует, что у одного из игроков должна быть выигрышная стратегия.

Заимствование стратегии позволяет показать, что выигрышная стратегия есть у первого игрока (кроме случая поля 1 × 1), однако доказательство неконструктивно. А именно, например, пусть у второго игрока существует выигрышная стратегия; докажем, что у первого игрока она также есть. Предположим, что первым ходом первый игрок съел только правую верхнюю дольку, и рассмотрим ход второго игрока, ведущий к выигрышной стратегии; тогда первый игрок может сам походить так своим первым ходом, тем самым «позаимствовав» стратегию второго игрока. Значит, у второго игрока не может быть выигрышной стратегии, а потому она есть у первого.

1.2 Техническое задание на курсовое проектирование

1.2.1 Назначение разработки

Программа предназначена для игры, основанной на определённых математических алгоритмах, которая помогает человеку в лёгкой форме избавится от потребности в обучении. Любая игра – есть обучение. Когда человек начинает играть в новую игру, он неловок, невнимателен и делает много ошибок, отчего человек, который по своей натуре не любит проигрывать, стремиться улучшать свои навыки, в результате чего появляется азарт, который в безобидной форме человеку принесёт приятные эмоции.

1.2.2 Требования к характеристикам

Требования к функциональным характеристикам:

1) программа должна оптимально рассчитывать наилучшие стратегии игры, при которых компьютер должен выиграть;

2) программа должна давать возможность пользователю выбирать размеры игрового поля;
3) программа должна давать возможность пользователю выбирать первенство хода;
4) программа должна визуализировать процесс игры.

Требования к количественным характеристикам:

1) максимальные размеры поля – 5 × 5;
2) минимальные размеры поля – 1 × 1;
3) в игру может играть только 1 человек и 1 компьютер.

1.2.3 Требования к техническим средствам

Требования к техническим средствам определяются установленной операционной системой. Программа не является ресурсоемкой и не предъявляет никаких особых требований к техническим средствам.

Минимальные системные требования:

- Операционная система: Windows 2000/XP/Vista/7/8/10 (32 или 64 bit).
- Процессор Pentium75 MHz.
- Оперативная память 16 МБ и выше.
- Жесткий диск от 90 МБ.
- Дисплей.
- Мышь, клавиатура.

1.2.4 Требование к информационной и программной совместимости

Программное обеспечение должно функционировать на операционных системах семейства MS Windows 2000/XP/Vista/7/10. Функционирование программы под операционной системой MS Windows 98/Me также возможно, но тестироваться в рамках данного проекта не будет.

Программа не предъявляет особых требований к интерфейсу пользователя.

1.3 Анализ технического задания

Написать программу играющую в игру «Щёлк».

Разработать удобный пользовательский интерфейс. Пользователь игровой программы должен иметь возможность пользоваться как клавиатурой, так и мышью.

Разработать структуры данных для оперирования такими понятиями как состояние игры, дерево игры, ход игрока и т.д.

Разработать оценочную функцию хода игрока.

Разработать алгоритм поиска лучшего хода для игрока (компьютера).

1.4 Выбор методов и средств решения задачи

Для реализации поставленной задачи потребуется выбрать язык программирования и среду разработки.

1.4.1 Выбор языка программирования

Язык Java относится к объектно-ориентированным языкам и принадлежит к механизму, в котором достигнут простой синтаксис. Его преобразование происходит компиляторами и интерпретаторами, отчего упрощенная разработка сделала его легким, чтобы писать, читать и обслуживать. При условии, что разработчик имеет базовые навыки, понимает, как работать с Фреймворками, пакетами, классами и объектами.

Достоинства языка Java:

- Использование в корпоративных приложениях, Java способен поддержать строительные блоки системы или различные библиотеки, с их помощью создают необходимые функции.
- Запуск приложений в «песочнице» с устранением распространённых, уязвимых объектов в соответствии с политикой безопасности.
- Повышенная производительность труда благодаря встроенному механизму, чтобы совместно использовать данные программы на нескольких компьютерах.

Недостатки языка Java:

- Простым пользователям версии Java изначально предоставлялись бесплатно. Но в 2019 году компанией Oracle объявлено, что теперь они начнут взымать плату за коммерческое использование языка. Начнется оценка, кто и с какой целью пользуется Java. Потребители в свою очередь будут вести поиск альтернативных решений.
- Низкая скорость и безопасность. Все языки с высоким уровнем страдают малой производительностью, этому способствуют различные функции – очистка памяти, настройки, блокировки.
- Многословность и сложность кода. Язык с длинными, трудными предложениями помогает при его изучении, но лишняя информация затрудняет чтение, поэтому в среде программистов Java считается слишком громоздким.

Язык C++ – компилируемый статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные

функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

Достоинства языка C++:

- Масштабируемость. На языке C++ разрабатывают программы для самых различных платформ и систем.
- Возможность работы на низком уровне с памятью, адресами, портами. Что, при неосторожном использовании, может легко превратиться в недостаток.
- Возможность создания обобщенных алгоритмов для разных типов данных, их специализация, и вычисления на этапе компиляции, используя шаблоны.

Недостатки языка C++:

- Наличие множества возможностей, нарушающих принципы безопасности приводит к тому, что в C++ программе может легко закрасться трудноуловимая ошибка. Вместо контроля со стороны компилятора разработчики вынуждены придерживаться весьма нетривиальных правил кодирования. По сути эти правила ограничивают C++ рамками некоего более безопасного подъязыка. Большинство проблем безопасности C++ унаследовано от C. Так визитной карточкой C++ стали уязвимости типа "переполнение буфера".
- Плохая поддержка модульности. Подключение интерфейса внешнего модуля через препроцессорную вставку заголовочного файла (#include) серьёзно замедляет компиляцию, при подключении большого количества модулей. Для устранения этого недостатка, многие компиляторы реализуют механизм прекомпиляции заголовочных файлов Precompiled Headers.
- Недостаток информации о типах данных во время компиляции (CTTI).
- Некоторые преобразования типов неинтуитивны. В частности, операция над беззнаковыми и знаковыми числами выдаёт беззнаковый результат.
- Препроцессор C++ (унаследованный от C) очень примитивен. Это приводит с одной стороны к тому, что с его помощью нельзя (или тяжело) осуществлять некоторые задачи метапрограммирования, а с другой, в следствии своей примитивности, он часто приводит к ошибкам и требует много действий по обходу потенциальных проблем. Некоторые языки программирования (например, Scheme и Nemerle) имеют намного более мощные и более безопасные системы метапрограммирования (также называемые макросами, но мало напоминающие макросы C/C++).

Язык C# активно развивается. Регулярно выходят новые версии C#, которые добавляют новые синтаксические конструкции в язык, а также увеличивают его быстродействие и надежность.

Достоинства языка C#:

- Поддержка Microsoft. В отличии от Java, которой не пошел на пользу переход в собственность Oracle, C# хорошо развивается благодаря усилиям Microsoft.
- В последнее время много совершенствуется. Так как C# был создан позже, чем Java и другие языки, то требовалось очень много доработать. Также это касается популяризации и бесплатности - было обещано открыть исходный код, а инструменты (Visual Studio, Xamarin) стали бесплатными для частных лиц и небольших компаний.
- Много синтаксического сахара. Синтаксический сахар - это такие конструкции, которые созданы для облегчения написания и понимания кода (особенно если это код другого программиста) и не играют роли при компиляции.
- Средний порог вхождения. Синтаксис похожий на C, C++ или Java облегчает переход для других программистов. Для новичков это также один из самых перспективных языков для изучения.
- Большое сообщество программистов.

Недостатки языка C#:

- Ориентированность, в основном, только на .NET (на Windows платформу).
- Бесплатность только для небольших компании, учащихся и программистов-одиночек. Для больших команд покупка лицензий обойдется недешево. Поэтому если есть своя фирма, то придется раскошелиться.
- Сохранили оператор go to.

Из всех выше перечисленных языков я остановилась на C# так как данный язык был хорошо изучен в предыдущем семестре, и с помощью него можно гораздо быстрее и проще реализовать поставленную задачу, также класс Windows Forms удобен в работе и позволяет реализовать один из важнейших пунктов в работе – это создание интерфейса. Перечисленные недостатки же не затрагивают мою работу.

1.4.2 Выбор среды разработки

Visual Studio – это самая востребованная среда разработки на C#.

Достоинства среды:

- Официальная. Так как, и язык, и среда разработки созданы в Microsoft, логично предположить, что ничего более функционального вы не найдете во всем Интернете. В некоторых случаях без Visual Studio не обойтись – например, при использовании технологий UWP и WPF.
- Бесплатная. Версии «Community edition» для рядового пользователя будет достаточно. Тем более, теперь можно подключать плагины (в отличие от старой версии Express).
- Поддерживает платформы .NET. Visual Studio имеет широкие возможности по разработке приложений под Windows, в том числе в .NET-сегменте.

Недостатки среды:

- Баги при переходах с триал-версии. При переходе на платную версию могут теряться настройки и нарушаться работа корпоративного сервера.
- Сложность. Самостоятельно освоить Visual Studio новичку будет непросто – слишком много доступных функций, спрятанных в подразделах меню.

Project Rider – среда от JetBrains для работы с платформой .NET.

Достоинства среды:

- ReSharper. Это плагин, изначально разработанный для повышения производительности Visual Studio. Теперь на его основе выпущена IDE.
- Поддержка полного цикла. Фирменная черта продуктов JetBrains, воплощенная и в Project Rider. С ним можно организовать весь цикл создания ПО: от идеи до поддержки.
- Функциональность. Project Rider позволяет подключить MSBuild и XBuild, работать с CLI-проектами и организовать отладку приложений .NET and Mono. Множество опций для быстрого создания кода улучшает производительность.
- Multiple runtime. Поддержка нескольких запущенных программ.
- Кроссплатформенность. Project Rider работает с Windows, Linux и MacOS.

Недостатки среды:

- Молодость. Часть функциональности еще в разработке, не все стартовые баги исправлены.
- Стоимость. Самая дешевая версия Project Rider обойдется в 139 долларов за первый год использования. Но есть триал-версия и специальные предложения для студентов и непрофильных организаций.

Eclipse – одна из самых популярных мультиязычных сред. Ориентирована преимущественно на разработку Java-приложений, но полезна и для кодов на C#.

Достоинства среды:

- Множество плагинов. У Eclipse едва ли не самое большое число надстроек – «на все случаи жизни».
- Активное сообщество. Помогает быстрее освоить среду разработки, выпускает новые плагины.
- Отличные компилятор и отладчик. Первый работает на порядок быстрее, чем у конкурентов, второй – показывает потоки, пересечения, позволяет гибко управлять ходом отладки.
- Кастомизация. Благодаря плагинам и настройкам можно полностью персонализировать Eclipse.
- Бесплатность. Это open-source проект, абсолютно бесплатный.
- Высокая функциональность. Благодаря разработчикам-официалам и членам сообщества с помощью Eclipse можно провести любой C#-продукт по полному циклу разработки.

Недостатки среды:

- Сложность. Как и любой функциональный продукт, Eclipse может показаться новичку слишком сложным.
- Нет гарантий надежности. Так как плагины создаются сообществом, за их качество отвечает только разработчик. Кроме того, сами создатели Eclipse с каждой новой версией плодят баги, не успевая порой исправлять старые.

Visual Studio Code – кроссплатформенный редактор кода, который при помощи плагинов можно «подтянуть» к статусу IDE.

Достоинства среды:

- Кроссплатформенность. Работает на MacOS, Ubuntu и Windows. Пока недоступен на Android и iOS.
- Бесплатность. Простой open-source редактор и плагины — платить не надо.
- Легковесность. Потребуется совсем мало ресурсов, чтобы приступить к работе с минималистичным VSC.

Недостатки среды:

- Низкая функциональность. Несмотря на поддержку .NET-платформы, VCS неудобен для сложных проектов.
- Сомнительная надежность. Многие надстройки имеют низкое качество сборки и не всегда выполняют даже основные функции.

Code::Blocks – среда разработки, известная простой и удобством в настройке и использовании.

Достоинства среды:

- Бесплатность. Полноценный open-source проект.
- Простота. В отличие от Visual Studio, среда Code::Blocks понятна новичку, знающему один из поддерживаемых языков.
- Кроссплатформенность. IDE запускается на любой десктопной ОС.
- Выбор компилятора. Code::Blocks ограничена в функциональности, но эта возможность — несомненный плюс.
- Легковесность.

Недостатки среды:

- Недостаточная функциональность. Для создания комплексных приложений Code::Blocks категорически не подходит.
- Нестабильность. Приходится сталкиваться с нелепыми ошибками в отладке и некорректной работой всей среды.

Из всех выше перечисленных сред разработок я выбрала Visual Studio, так как я уже работала с ней, и она является официальной, ведь она была разработана Microsoft. Также эта среда изучается в ходе данного курса. Visual Studio является рекомендованной средой производителя Windows, поэтому её компилятор, лучше других IDE, взаимодействует с системными библиотеками Windows.

2 Разработка программного обеспечения

2.1 Разработка алгоритмов

Оценка игрового поля производится для «крайнего случая» рекурсии, т.е. когда осталась только «отравленная» фишка, с помощью оценочной функции, и для узлов дерева игры по стратегии minimax. Для игрока-max, если он победил, то оценка 1, если проиграл, то -∞. Для игрока-min при победе оценка -1, при проигрыше +∞.

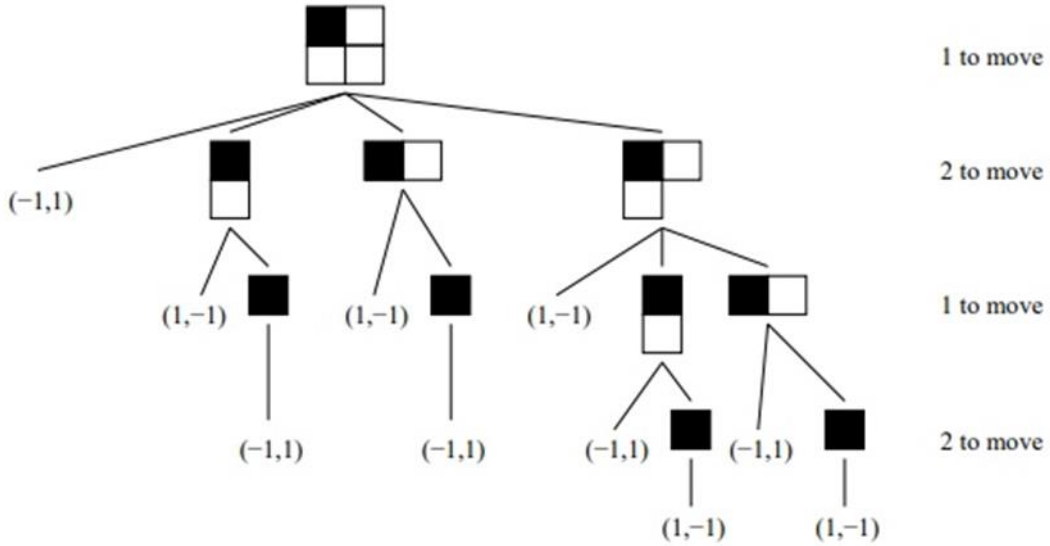

Рисунок 1 – Дерево решений для поля 2 × 2

Альфа-бета отсечение. Основная его идея заключается в том, чтобы после «прохода» по одной из ветвей дерева решений «отсекать» ветви, заведомо не имеющие оптимального решения, относительно коэффициентов альфа и бета, полученных на первом проходе. Альфа — это минимально возможная оценка, которую может получить максимизирующий игрок, инициализируется значением минус бесконечности и наоборот для беты. Если в одном из узлов ветви значения альфа больше либо равно бета, найден ход, который гарантирует победу максимизирующему игроку. Если значения этих параметров в текущем узле не улучшаются, ветвь не следует рассматривать, поскольку все ее «потомки», как и она сама, не содержат оптимального решения.

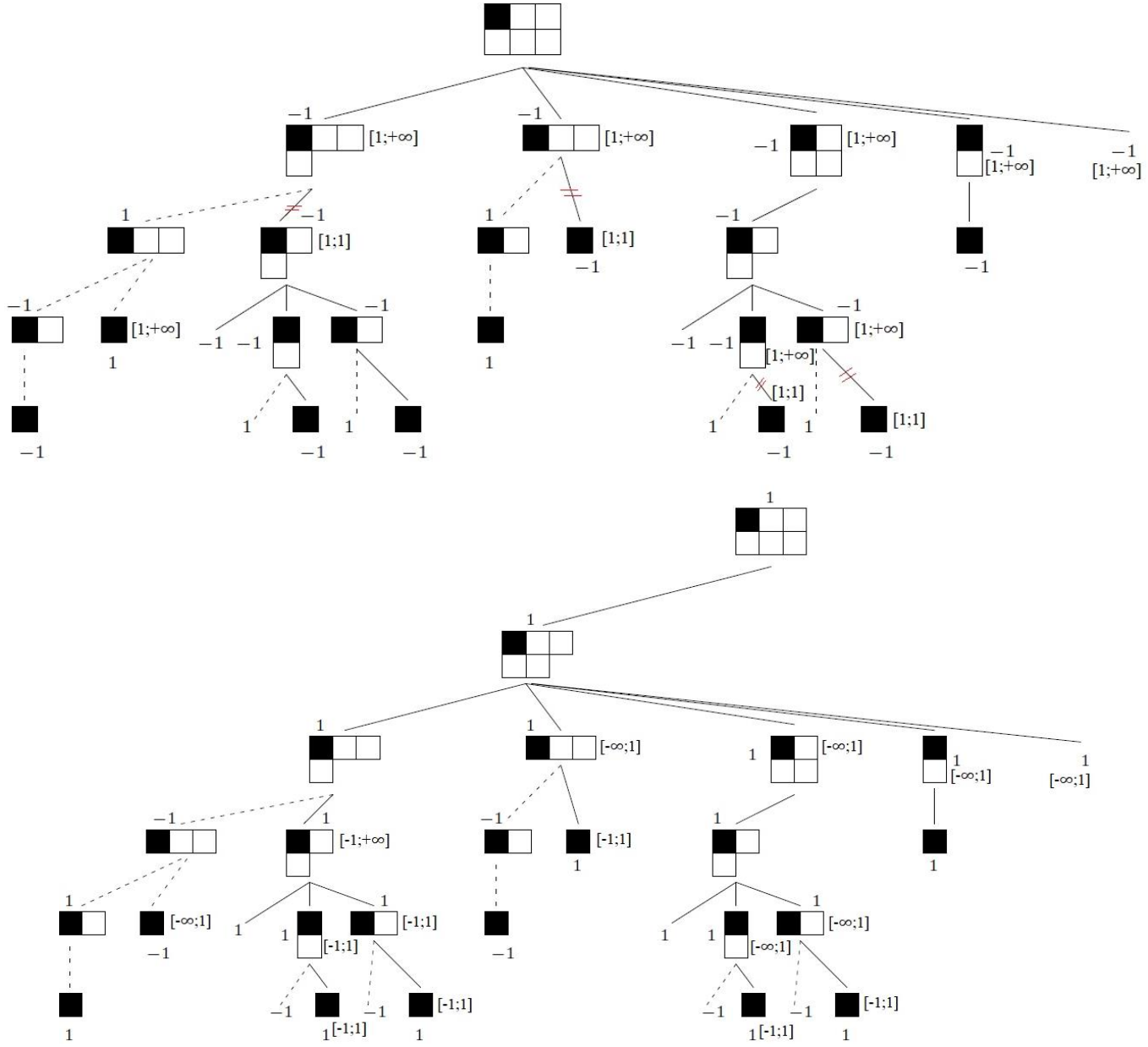

Рисунок 2 – Дерево решений для поля 3 × 2 с альф-бета отсечением

2.2 Разработка функций

В программе реализовано 7 основных функций:

1) available_moves – вызов функции возвращает доступные ходы для конкретного игрового поля;
2) has_won – вызов данной функции возвращает true если победил игрок, иначе false;
3) game_is_over – данная функция возвращает true если один из игроков победил;
4) evaluate_board – данная функция возвращает оценку при достижении "крайнего случая";

5) deepcopy – создает копию доски;
6) select_space – вызов функции означает выполнение хода игроком на конкретной игровой доске;
7) minimax – производит оценку игрового поля для «крайнего случая» рекурсии, т.е. когда осталась только «отравленная» фишка, с помощью оценочной функции, и для узлов дерева игры по стратегии minimax.

Листинги функций приведены в приложении.

2.3 Программная реализация

Сначала включается рабочее окно программы, после чего пользователю предоставляется возможность выбрать размеры игрового поля и первенство между компьютером и человеком в начальном ходе. После всех сделанных пользователем настроек необходимо нажать на кнопку «Начать игру». Далее запускается игра, состоящая из множества квадратов, при нажатии на квадраты зелёного цвета уничтожается (становится белым) часть поля по правилам игры, при нажатии на красный квадрат заканчивается игра и выскакивает окно, оповещающее о проигрыше, если же игрок приходит к результату когда его последний ход происходит на зелёном квадрате и на поле остаётся лишь красный квадрат, то появляется окно с оповещением, что пользователь победил.

2.3.1 Структура программы

Структура программы состоит из созданных функций и их взаимосвязей, выраженных передаваемыми между ними данными.

2.3.2 Соглашение по именованию переменных и функций

В ходе работы над курсовым проектом были выработаны следующие соглашения по именованию переменных и написанию кода.

1) все переменные в программе имеют осмысленное название;
2) имена всех пользовательских классов начинаются с заглавной буквы, имена функций классов начинаются с маленькой буквы;
3) текст программы снабжен комментариями, важные блоки функций отделены друг от друга с помощью комментариев;
4) текст программы структурирован, вложенные операторы смещены вправо относительно заголовков.

3 Руководство пользователя

Программа предназначена для удовлетворения человеком потребностей в умственной деятельности. Данная программа может быть использована пользователем в качестве развлекательной успокаивающей игры.

3.1 Описание интерфейса

Программа представляет собой приложение Window Forms.

При запуске программы отображается окно меню, в котором можно выбрать размеры поля в виде двух выпадающих списков, и также выбрать первенство в начальном ходе в виде радиобаттона с названиями выбова. Также имеется кнопка «Начать игру».

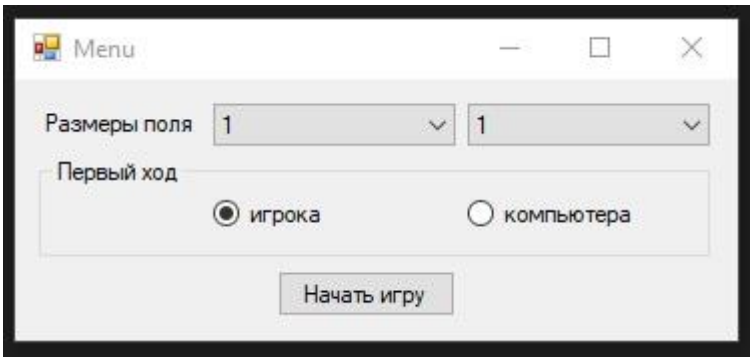

Рисунок 3 – Начальное меню

При нажатии на кнопку «Начать игру» закрывается форма с меню, и открывается форма с игровым полем, которое имеет размеры, выбранные пользователем (ниже показано начальное поле при игре где пользователь ходит первым).

Рисунок 4 – Начальный вид игрового поля

В ходе игры поле по правилам изменяется и отсечённое поле меняет цвет на белый. Ниже представлен вариант сыгранной партии, где первый ход компьютера.

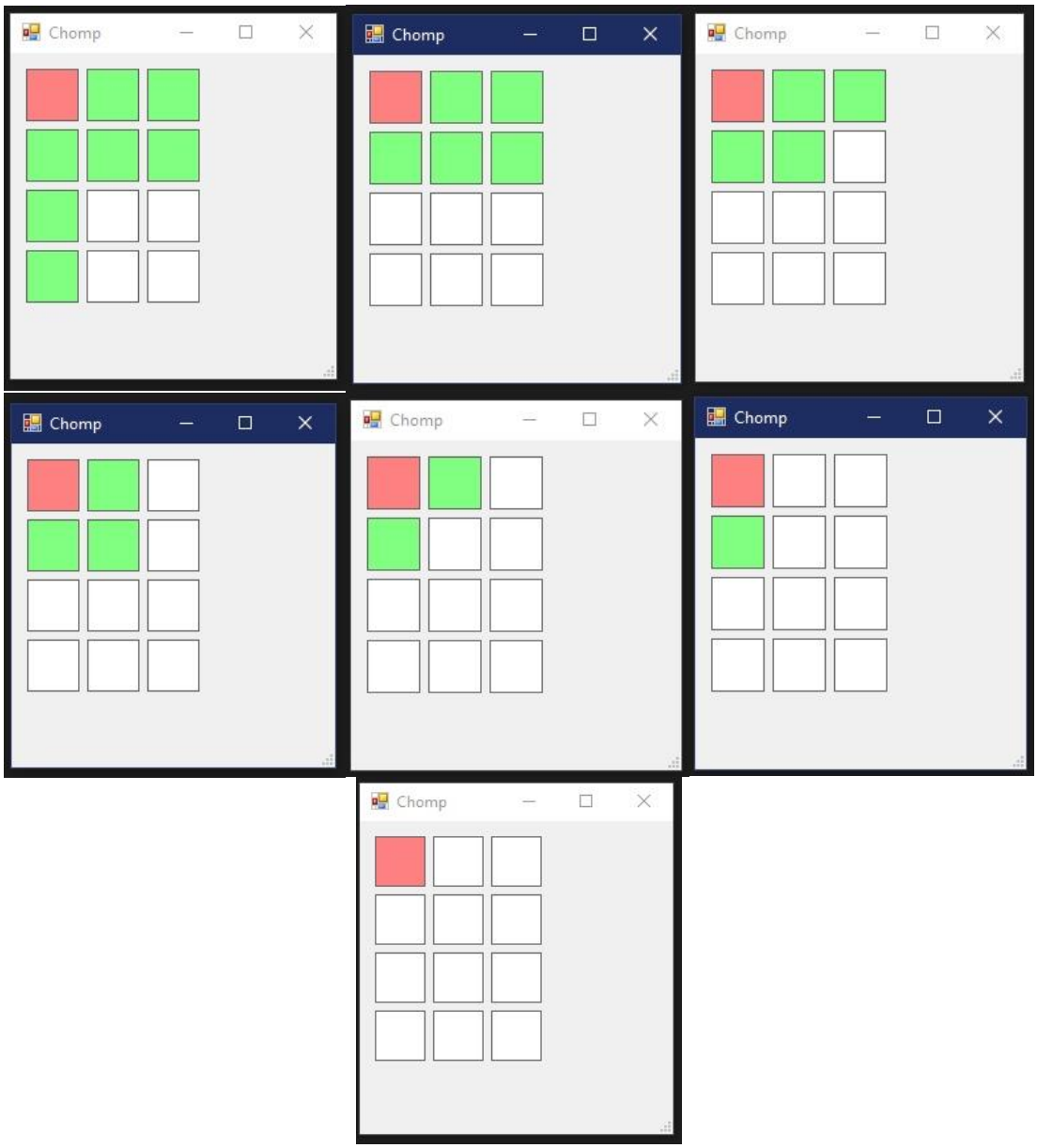

Рисунок 5 – Пример сыгранной партии с проигрышем человека

При окончании игры появляется окно, которое оповещает пользователя о победе или проигрыше.

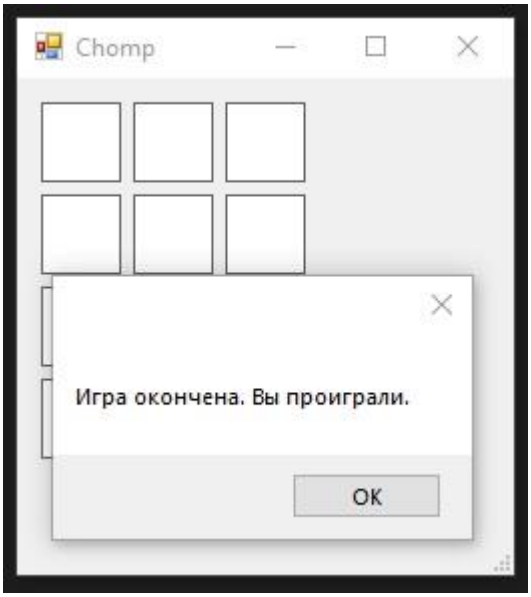

Рисунок 6 – Окно результата игры

При нажатии на кнопку «ок» в окне с выводом результата сыгранной партии, игра полностью закрывается.

4 Документация

4.1 Правила игры

Правила игры – двое по очереди (человек и компьютер) "откусывают" от прямоугольной доски. Игрок выбирает любое поле доски и снимает все фишки, которые находятся не выше и не левее избранного поля. Снявший последнею фишку – проигрывает.

4.2 Инструкция пользователя

При запуске игры следует поочерёдно открыть рядом с фразой «Размеры поля» выпадающие списки, путём нажатия мыши на прямоугольную область с цифрой и стрелкой вниз. При появлении списка необходимо выбрать пункт от одного до пяти с соответствующими значениями. В результате данных действий будет сгенерировано игровое поле.

В разделе «Первый ход» необходимо выбрать из двух вариантов – «игрока» или «компьютера» один, путём нажатия мыши на соответствующий кружок, расположенный рядом с пунктами. В результате данных действий будет осуществляться право на первый ход в игре.

Для начала игры необходимо нажать на кнопку «Начать игру». В результате данного действия закроется окно с изначальным меню и появится игровое поле.

В зависимости от выбранных размеров на поле будут расположены белые (недоступные для взаимодействия) и зелёные квадраты, а также один красный. При доступном ходе игрока, следует нажимать на зелёные и красный квадраты (на красный квадрат рекомендуется нажимать в крайних безвыходных положениях, так как данное действие ведёт к проигрышу). Поле при нажатии на зелёные квадраты будет отсекаться от игрового пространства по правилам игры (всё правее и ниже выбранного зелёного квадрата включительно будут становиться белыми и переставать быть доступными для взаимодействия) до тех пор, пока не останется красный квадрат. В результате данных действий будет реализована игровая партия человека и компьютера.

После нажатия игрока на красный квадрат появляется окно с результатом сыгранной игры для человека. После ознакомления с результатами необходимо нажать на кнопку «ок». В результате данных действий игрок-человек узнаёт свой результат в игре и полностью закрывает приложение.

4.3 Инструкция программиста

Используемые классы, структуры и функции:

List<T> – Представляет строго типизированный список объектов, доступных по индексу. Поддерживает методы для поиска по списку, выполнения сортировки и других операций со списками.

- List<T>.Add(T) – Добавляет объект в конец очереди List<T>.

Control – Определяет базовый класс для элементов управления, являющихся компонентами с визуальным представлением.

- Control.BackColor – Возвращает или задает цвет фона для элемента управления.
- Control.Enabled – Возвращает или задает значение, указывающее, может ли элемент управления отвечать на действия пользователя.
- Control.TabIndex – Возвращает или задает последовательность перехода по клавише TAB между элементами управления внутри контейнера.
- Control.Location – Возвращает или задает координаты левого верхнего угла элемента управления относительно левого верхнего угла его контейнера.
- Control.Size – Возвращает или задает высоту и ширину элемента управления.
- Control.TabStop – Возвращает или задает значение, указывающее, может ли пользователь перевести фокус на данный элемент управления при помощи клавиши TAB.
- Control.Click – Происходит при щелчке элемента управления.
- Control.Controls – Возвращает коллекцию элементов управления, содержащихся в элементе управления.
- Control.Hide – Скрывает элемент управления от пользователя.

Button – Представляет элемент управления "кнопка Windows".

MessageBox – Отображает окно сообщения.

- MessageBox.Show – Отображает окно сообщения.

Form – Представляет окно или диалоговое окно, которое составляет пользовательский интерфейс приложения.

- Form.ClientSize – Возвращает или задает размер клиентской области формы.
- Form.Close – Закрывает форму.
- Form.Load – Происходит до первоначального отображения формы.
- Form.ShowDialog – Отображает форму как модальное диалоговое окно.

EventHandler – Представляет метод, обрабатывающий событие, не имеющее данных.

Thread – Создает и контролирует поток, устанавливает его приоритет и получает его статус.

- Thread.Sleep – Приостанавливает текущий поток на указанное время.

Color – Представляет цвета в терминах каналов альфа, красного, зеленого и синего (ARGB).

- Color.FromArgb – Создает структуру Color из указанных 8-разрядных значений компонентов ARGB (альфа, красный, зеленый и синий).

Size – Сохраняет упорядоченную пару целых чисел, указывающих Height и Width.

Int32 – Представляет 32-разрядное целое число со знаком.

- Int32.MinValue – Представляет наименьшее возможное значение Int32. Это поле является постоянным.
- Int32.MaxValue – Представляет максимально возможное значение Int32. Это поле является постоянным.

ComboBox – Представляет элемент управления со списком Windows.

- ComboBox.SelectedIndex – Получает или задает индекс, указывающий текущий выбранный элемент.

RadioButton – Позволяет пользователю выбрать единственный вариант из группы доступных, когда используется вместе с другими элементами управления RadioButton.

- RadioButton.Checked – Возвращает или задает значение, указывающее, выбран ли данный элемент управления.

Требования к техническим средствам определяются установленной операционной системой. Программа не является ресурсоемкой и не предъявляет никаких особых требований к техническим средствам.

Минимальные системные требования:

- Операционная система: Windows 2000/XP/Vista/7/8/10 (32 или 64 bit).
- Процессор Pentium75 MHz.
- Оперативная память 16 МБ и выше.
- Жесткий диск от 90 МБ.
- Дисплей.
- Мышь, клавиатура.

Программное обеспечение должно функционировать на операционных системах семейства MS Windows 2000/XP/Vista/7/10. Функционирование программы под операционной системой MS Windows 98/Me также возможно, но тестироваться в рамках данного проекта не будет.

Программа не предъявляет особых требований к интерфейсу пользователя.

4.4 Инструкция администратора

Программа предназначена для игры, основанной на определённых математических алгоритмах, которая помогает человеку в лёгкой форме избавится от потребности в обучении. Любая игра – есть обучение. Когда человек начинает играть в новую игру, он неловок, невнимателен и делает

много ошибок, отчего человек, который по своей натуре не любит проигрывать, стремиться улучшать свои навыки, в результате чего появляется азарт, который в безобидной форме человеку принесёт приятные эмоции.

Обязанности администратора – поставлять exe файл приложения.

Заключение

В ходе курсовой работы была разработана программа играющая в игру «Щёлк». Для неё был разработан удобный пользовательский интерфейс. Пользователь игровой программы имеет возможность пользоваться как клавиатурой, так и мышью. Была разработана структура данных для оперирования такими понятиями как состояние игры, дерево игры, ход игрока и т.д. Разработана оценочная функция хода игрока. И также разработан алгоритм поиска лучшего хода для игрока (компьютера).

К достоинствам системы можно отнести простоту работы с приложением. Игра находит гарантировано оптимальное решение.

К недостаткам – отсутствие режима игры для двух людей, слишком простой внешний вид, нет выбора уровней сложности, игра с компьютером имеет самую высокую сложность.

Данный проект может дальше развиваться, можно создать более приятный дизайн, разработать настройку уровней сложности, добавить таймер на ход игрока или создать систему бонусов, которая бы поощряла победы игрока.

В целом задание, поставленное на курсовую работу, выполнено в полном объеме. Проект соответствует техническому заданию.

Список используемой литературы

1. Гарднер М. Математические новеллы. Пер. с англ. Ю. А. Данилова. Под ред. Я. А. Смородинского. М., «Мир», 1974. 456 с. с илл.; стр. 407—412
2. Губина Г. Г. Компьютерный английский. Ч. I. Computer English. Part I. Учебное пособие. — С. 385.
3. Герберт Шилдт. Полный справочник по C# = C#: The Complete Reference. — М.: Издательский дом «Вильямс», 2004. — С. 26—27. — 752 с. — ISBN 5-8459-0563-X.
4. Кей С. Хорстманн. Java SE 9. Базовый курс = Core Java SE 9 for the Impatient. — М.: «Вильямс», 2018. — 576 с. — ISBN 978-5-6040043-0-2, 978-0-13-469472-6.
5. Кормен Т. М. и др. Часть VI. Алгоритмы для работы с графами // Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. — М.: Вильямс, 2006. — С. 1296. — ISBN 0-07-013151-1.
6. Bjarne Stroustrup's FAQ (англ.). Bjarne Stroustrup (Ocober 1, 2017). — «The name C++ (pronounced "see plus plus")».
7. Winning ways for your mathematical plays, Volume 3 (2nd edn), by E. R. Berlekamp, J. H. Conway and R. K. Guy. Pp. 275. 2018. ISBN 9780429945618. CRC Press, 2018. Стр. 39
8. Fred Schuh: Spel van delers. Nieuw Tijdschrift voor Wiskunde 39 (1952), S. 299–304
9. David Gale: A curious Nim-type game. Amer. Math. Monthly 81 (1974), S. 876–879
10. Elwyn R. Berlekamp et al.: Gewinnen — Strategien für mathematische Spiele, Band 3. Vieweg, Braunschweig/Wiesbaden 1986, ISBN 3-528-08533-9, S. 172f
11. p. 482 in: Games of No Chance (R. J. Nowakowski, ed.), Cambridge University Press, 1998.

Приложение

Файл Chomp.cs

```
using System.Collections.Generic;

namespace ChompWF
{
    class Chomp
    {
        int n, m;

        public Chomp(int _n, int _m)
        {
            n = _n;
            m = _m;
        }

        // Вызов функции вернет доступные ходы для конкретного игрового поля
        private List<List<int>> available_moves(List<List<bool>> board)
        {
            List<List<int>> moves = new List<List<int>>();
            for (int i = 0; i < n; i++)
                for (int j = 0; j < m; j++)
                    if (board[i][j] && (i != 0 || j != 0))
                        moves.Add(new List<int>() { i, j });
            return moves;
        }

        // Вызов данной функции вернет true если победил игрок, иначе false
        bool has_won(List<List<bool>> board, bool is_maximizing)
        {
            bool sum = false;
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    if (i == 0 && j == 0)
                        continue;
                    sum = sum || board[i][j];
                }
            }
            if (!sum && board[0][0] && !is_maximizing)
                return true;
            return false;
        }

        // Данная функция возвращает true если один из игроков победил
        bool game_is_over(List<List<bool>> board, bool is_maximizing)
        {
            return has_won(board, is_maximizing);
        }

        // Данная функция возвращает оценку при достижении "крайнего случая"
        int evaluate_board(List<List<bool>> board, bool is_maximizing)
        {
            if (has_won(board, is_maximizing))
                return 1;
            return -1;
        }
```

```
// Создает копию доски
List<List<bool>> deepcopy(List<List<bool>> board)
{
    List<List<bool>> res = new List<List<bool>>(n);
    for (int i = 0; i < n; i++)
    {
        res.Add(new List<bool>(m));
        for (int j = 0; j < m; j++)
            res[i].Add(board[i][j]);
    }
    return res;
}

// Вызов функции означает выполнение хода игроком на конкретной игровой доске
bool select_space(List<List<bool>> board, List<int> move)
{
    if (move[0] >= n || move[1] >= m)
        return false;
    if (board[move[0]][move[1]])
    {
        for (int i = move[0]; i < n; i++)
            for (int j = move[1]; j < m; j++)
                board[i][j] = false;
        return true;
    }
    return false;
}

public List<int> minimax(List<List<bool>> input_board, int alpha, int beta, bool
is_maximizing)
{
    // Крайний случай рекурсии - игра окончена
    if (game_is_over(input_board, is_maximizing))
        return new List<int>(3) { evaluate_board(input_board, is_maximizing), -1,
        -1 };
    // Инициализируем значения best_value и best_move
    List<int> best_move = new List<int>(2) { 0, 0 };
    int best_value;
    // Случай, когда ход максимизирующего игрока
    if (is_maximizing)
        best_value = alpha;
    // Случай, когда ход минимизирующего компьютера
    else
        best_value = beta;
    /* Пройдём циклом по всем возможным ходам, для того чтобы выбрать наилучший
    путем рекурсивных вызовов функции minimax с копией игровой доски.
    Как только рекурсия достигнет "крайнего" случая, она вернет значения из [1, -
    1]
    для функции, которая ее вызвала, до тех пор, пока самая "верхняя" функция в
    стеке
    вызовов (minimax с текущей игровой доской) не получит свое значение */
    foreach (var move in available_moves(input_board))
    {
        List<List<bool>> new_board = deepcopy(input_board);
        select_space(new_board, move);
        int hypothetical_value;
        if (is_maximizing)
        {
            hypothetical_value = minimax(new_board, best_value, beta,
            !is_maximizing)[0];
            if (hypothetical_value > best_value)
```

```
                    {
                        best_value = hypothetical_value;
                        best_move = move;
                    }
                    if (best_value >= beta)
                        return new List<int>() { best_value, best_move[0], best_move[1]
                        };
                }
                else
                {
                    hypothetical_value = minimax(new_board, alpha, best_value,
                    !is_maximizing)[0];
                    if (hypothetical_value < best_value)
                    {
                        best_value = hypothetical_value;
                        best_move = move;
                    }
                    if (best_value <= alpha)
                        return new List<int>() { best_value, best_move[0], best_move[1]
                        };
                }
            }
            return new List<int>() { best_value, best_move[0], best_move[1] };
        }
    }
}
```

Файл Main.cs

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;

namespace ChompWF
{
    public partial class Main : Form
    {
        List<List<Button>> labels;
        List<List<bool>> board;
        Chomp chomp;
        int n, m;
        bool player;

        bool PlayerMove(int iter1, int iter2, bool p)
        {
            for (int i = iter1; i < n; i++)
            {
                for (int j = iter2; j < m; j++)
                {
                    labels[i][j].BackColor = Color.FromArgb(255, 255, 255);
                    labels[i][j].Enabled = false;
                    board[i][j] = false;
                }
            }
            if (iter1 == 0 && iter2 == 0)
            {
                if (p)
                    MessageBox.Show("Игра окончена. Вы проиграли.");
                else
                    MessageBox.Show("Игра окончена. Вы победили.");
```

```
                return true;
            }
            return false;
        }

        public Main(int _n, int _m, bool p)
        {
            InitializeComponent();
            n = _n; m = _m;
            player = p;
            ClientSize = new Size(248, 248);
            labels = new List<List<Button>>(n);
            board = new List<List<bool>>(n);
            chomp = new Chomp(n, m);
            EventHandler handler = (s, ev) =>
            {
                Button me = (Button)s;
                if (PlayerMove(me.TabIndex / 10, me.TabIndex % 10, true))
                {
                    Close();
                    return;
                }
                Thread.Sleep(1500);
                List<int> res = new List<int>(3);
                res = chomp.minimax(board, int.MinValue, int.MaxValue, true);
                if (PlayerMove(res[1], res[2], false))
                    Close();
            };
            for (int i = 0; i < n; i++)
            {
                labels.Add(new List<Button>(m));
                board.Add(new List<bool>(m));
                for (int j = 0; j < m; j++)
                {
                    board[i].Add(true);
                    labels[i].Add(new Button
                    {
                        BackColor = Color.FromArgb(128, 255, 128),
                        Location = new Point(12 + 46 * j, 12 + 46 * i),
                        Size = new Size(40, 40),
                        TabIndex = i * 10 + j,
                        TabStop = true
                    });
                    labels[i][j].Click += handler;
                    Controls.Add(labels[i][j]);
                }
            }
            labels[0][0].BackColor = Color.FromArgb(255, 128, 128);
            if (!player)
            {
                List<int> res = new List<int>(3);
                res = chomp.minimax(board, int.MinValue, int.MaxValue, true);
                if (PlayerMove(res[1], res[2], false))
                {
                    Load += (se, ee) => Close();
                    return;
                }
            }
        }
    }
}
```

Файл Menu.cs

```
using System;
using System.Windows.Forms;

namespace ChompWF
{
    public partial class Menu : Form
    {
        public Menu()
        {
            InitializeComponent();
            comboBox1.SelectedIndex = 0;
            comboBox2.SelectedIndex = 0;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Main main = new Main(comboBox1.SelectedIndex + 1, comboBox2.SelectedIndex +
            1, radioButton1.Checked);
            Hide();
            if (main.ShowDialog() == DialogResult.Cancel)
                Close();
        }
    }
}
```