МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА 25

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

Доцент, канд. техн. наук		Е. М. Линский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

ИГРА В ХЕКСАГОН (ГЕКСАГОН)

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТ гр. № 2354

	подпись, дата	А.И. Машкова
		инициалы, фамилия

Санкт-Петербург 2024

Содержание

1 Постановка задачи

Задачей данной курсовой работы является разработка программы, которая реализует игру "Hexxagon" — стратегическую настольную игру, предназначенная для двух игроков или против компьютера. Игровой процесс представляет собой шестиугольное игровое поле, на котором каждый игрок имеет возможность управлять своими фишками, с целью захвата и удержания чужих фишек, чтобы в итоге одержать победу.

Игровое поле имеет размеры, варьирующиеся в зависимости от выбора игрока, и состоит из ячеек, каждая из которых может быть пустой или заполненной фишкой одного из игроков ('R' для красного и 'B' для синего). Игра состоит из следующих действий: перемещение, удвоение, захват. Игра завершается, когда один из игроков больше не может сделать ход, либо все фишки одного из игроков будут захвачены, либо при заполнение всего поля.

Данная задача была рассмотрена и реализована в соответствии с работой Ч. Уэзерелла [1].

Для демонстрации актуальности задачи разработки программы "Hexxagon" можно привести следующие примеры:

1.Игровое поле с начальным расположением фишек представлено на рисунке [1].

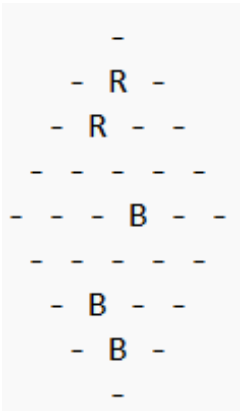

Рисунок 1 – Визуализация игрового поля 1

В реализации "-" - это представление пустых ячеек. Игрок, имеющий возможность сделать ход, проверяет возможности захвата фишек противника и, тем самым, создает собственное преимущество на поле.

2.После нескольких ходов состояние игрового поля представлено на рисунке [2].

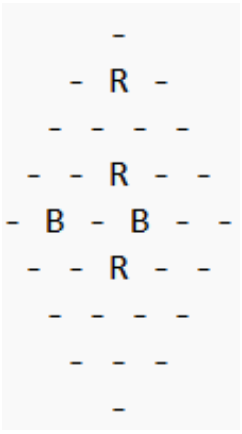

Рисунок 2 - Визуализация игрового поля 2

На данном поле красный игрок может захватить фишку синего игрока (B) с помощью одного удачного хода, продемонстрировав стратегическую глубину игры.

С помощью таких примеров видно, что задача разработки программы "Hexxagon" имеет практическое применение, позволяя игрокам анализировать и разрабатывать стратегии, взаимодействуя с игрой и друг с другом.

2 Алгоритм

Алгоритм игры "Hexxagon" основан на механике управления состоянием игрового поля и проверке допустимости ходов, а также на выполнении действий. Основная идея алгоритма заключается в том, чтобы обеспечить возможность игрокам выполнять свои ходы, проверяя при этом, являются ли эти ходы, разрешёнными согласно правилам игры [2].

Алгоритм включает в себя следующие ключевые компоненты:

- Структура данных: Игровое поле представлено в виде двумерного вектора, где каждая ячейка может содержать символ, обозначающий фишку игрока ('R' или 'B'), или быть пустой ('\0'). Размер поля определяется переменной, которая указывает на количество строк и столбцов.
- Инициализация: Алгоритм начинается с инициализации игрового поля, где он считывает поле из файла.
- Игровой цикл: В процессе игры алгоритм последовательно запрашивает ход текущего игрока, проверяет его допустимость и выполняет, если он допустим. Если ход недопустим, игроку предлагается повторить попытку.
- Проверка состояния игры: Алгоритм также проверяет, остались ли доступные ходы у противника, и завершает игру, если у одного из игроков не осталось доступных ходов.

2.1 Пошаговый пример для алгоритма

В данном подразделе рассмотрим выполнение алгоритма на примере с игровым поле с размером 3. Визуализация игрового поля представлена на рисунке [3].

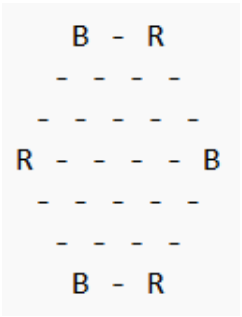

Рисунок 3 – Игровое поле размером 3

Шаги алгоритма.

1 шаг:

Игрок 'R' пытается удвоить фишку на позиции (1, 3) на соседнюю пустую ячейку (1, 2), захватывая фишку противника на (1, 1). Алгоритм проверяет возможно ли такое действие. Удвоение фишки осуществляется только на соседние ячейки от выбранной позиции, игрок правильно ввел все координаты, а также смог захватить фишку другого игрока. Алгоритм представляет новое состояние поля на рисунке [4].

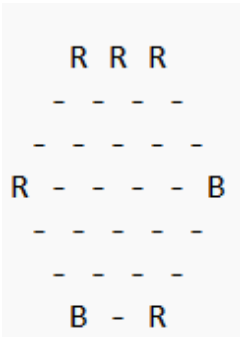

Рисунок 4 – Состояние игрового поля после 1 шага

2 шаг:

Алгоритм проверяет остались ли фишки и доступные ходы у игроков. Игра продолжается, потому что игроки имеют фишки и доступные ходы. Игрок 'B' выбирает перемещение фишки с позиции (7, 1) на (5, 3). Алгоритм проводит проверку на возможность такого действия. Перемещение фишки выполняется только через одну ячейку от выбранной позиции, игрок правильно ввел все координаты. Алгоритм представляет новое состояние поля на рисунке [5].

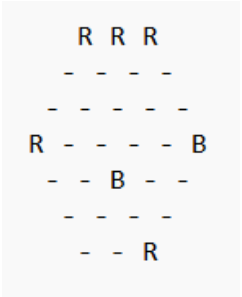

Рисунок 5 – Состояние игрового поля после 2 шага

3 шаг:

Алгоритм проверяет остались ли фишки и доступные ходы у игроков. Игра продолжается, потому что игроки имеют фишки и доступные ходы. Игрок 'R' пытается удвоить фишку на позиции (7, 3) на соседнюю пустую ячейку (6, 3), захватывая фишку противника на (5, 3). Алгоритм аналогично

проверяет возможность удвоения. Игрок правильно ввел все координаты, а также смог захватить фишку другого игрока. Алгоритм представляет новое состояние поля на рисунке [6].

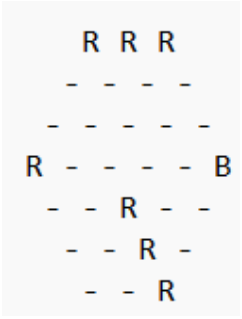

Рисунок 6 – Состояние игрового поля после 3 шага

4 шаг:

Алгоритм проверяет остались ли фишки и доступные ходы у игроков. Игра продолжается, потому что игроки имеют фишки и доступные ходы. Игрок 'B' пытается удвоить фишку на позиции (4, 6) на соседнюю пустую ячейку (4, 5). Алгоритм аналогично проверяет возможность удвоения. Игрок правильно ввел все координаты. Алгоритм представляет новое состояние поля на рисунке [7].

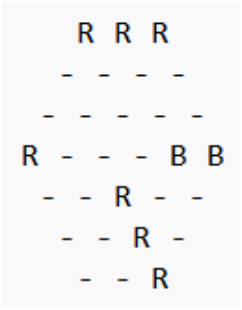

Рисунок 7 - Состояние игрового поля после 4 шага

5 шаг:

Игрок 'R' выбирает перемещение фишки с позиции (5, 3) на (5, 5), захватывая фишки противника на (4, 5), на (4, 6). Алгоритм аналогично проверяет возможность перемещения. Игрок правильно ввел все координаты, а также смог захватить фишки другого игрока. Алгоритм представляет новое состояние поля на рисунке [8].

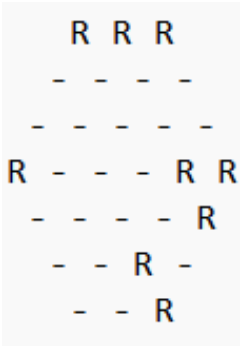

Рисунок 8 - Состояние игрового поля после 5 шага

6 шаг:

Алгоритм проверяет остались ли фишки и доступные ходы у игроков, проанализировав поле, алгоритм завершает игру, потому что игрок 'B' не владеет доступными фишками. Далее, алгоритм выводит результат игры и называет цвет победителя, в данной игре победил игрок 'R'.

2.2 Псевдокод

```
void Game::start() {
    gameBoard.loadGameState(inputFilename);
    selectGameMode();

    Пока (true) {
        gameBoard.printBoard();
       вывести "Ход игрока: " + (если текущий игрок 'R' тогда "R" иначе "B")l;

        Если (!playerTurn()) {
            gameBoard.saveGame(outputFilename);
            break;
        }

        Если (!gameBoard.hasChips(currentPlayer == 'R' ? 'B' : 'R')) {
            вывести "Игрок " + (если текущий игрок 'R' тогда "R" иначе "B") + " выиграл! Противник
            break;
        }

        Если (!gameBoard.hasAvailableMoves(currentPlayer) || !gameBoard.hasChips(currentPlayer)) {
           вывести "Игрок " + (если текущий игрок 'R' тогда "R" иначе "B") + " не может сделать хо
            вывести "Игра окончена! Победили " + (если текущий игрок 'R' тогда "синие!" иначе "кра
            break;
        }
        Если (gameBoard.isFull()) {
            int redChips = gameBoard.countChips('R');
            int blueChips = gameBoard.countChips('B');
            Если (redChips > blueChips) {
                вывести "Игра окончена! Победили красные!";
            }
            Иначе если (blueChips > redChips) {
               вывести "Игра окончена! Победили синие!";
            }
            Иначе {
                вывести "Игра окончена! Ничья!";
            }
            break;
        }

        Если (gameMode == 2) {
            gameBoard.computerMove('B');

            Если (!gameBoard.hasChips('B')) {
                 вывести "Игрок R выиграл! Компьютер не имеет фишек!";
                break;
            }

            Если (!gameBoard.hasAvailableMoves('B')) {
               вывести "Компьютер не может сделать ход!";
               вывести "Игра окончена! Победили красные!";
                break;
            }
        }

        switchPlayers();
    }
}
```

Анализ сложности алгоритма

- Инициализация: $O(n^2)$, так как мы инициализируем двумерный массив, где n — размерность поля.
- Проверка допустимости хода: $O(1)$ для проверки конкретного хода, но $O(n)$ в худшем случае, если нужно проверить все возможные ходы.
- Выполнение хода: $O(1)$ для перемещения фишки, но $O(n)$ в случае, если необходимо проверить соседние ячейки.

В целом, общая временная сложность алгоритма можно оценить как $O(n^2)$, учитывая, что наибольшее время потребует печать и инициализация игрового поля.

3 Инструкция пользователя

Для запуска программы "Hexxagon" необходимо использовать параметры командной строки. Программа принимает два аргумента:

- Входной файл — файл с состоянием игры, который будет загружен при старте программы.
- Выходной файл — файл, в который будет сохранено состояние игры по завершении или в случае выхода.

Входной файл должен содержать информацию о состоянии игрового поля и инициализации игры. Формат входного файла выглядит следующим образом:

- Первая строка: Целое число, указывающее размер игрового поля.
- Следующие строки: Определяют содержимое игрового поля с помощью символов:

'R' — фишка красного игрока
'B' — фишка синего игрока
'-' — пустая ячейка

Выходной файл сохраняет текущее состояние игрового поля после завершения игры или при выходе из программы. Формат файла идентичен формату входного файла:

- Первая строка: Целое число, указывающее размер игрового поля.
- Следующие строки: Определяют текущее состояние игрового поля.

Таким образом, выходной файл позволяет сохранить и восстановить состояние игры для дальнейшего анализа или продолжения игры.

4 Текстовые примеры

1. Входные данные:

start.txt:

```
5
   B - - - R
  - - - - - -
 - - - - - - -
- - - - - - - -
R - - - - - - - B
- - - - - - - -
 - - - - - - -
  - - - - - -
   B - - - R
```

Выходные данные:
output.txt:

```
5
   R R R R B
  R R - R R R
 - R R - R R R
- R R - - - R R
R - R R R R R R R
- - - - - - - -
 - - - - - - -
  R R - - - -
   R R R - -
```

Игрок B не может сделать ход!
Игра окончена! Победили красные!

2. Входные данные:

start.txt:

```
3
   R - B
  - - - -
 B - - - R
  - - - -
   R - B
```

Выходные данные:
output.txt:

```
3
R R R
- - - -
R R B R R
R B R R
B R B
```

Игрок B не может сделать ход!
Игра окончена! Победили красные!

3. Входные данные:

start.txt:

```
4
  R - - B
 - - - - -
- - - - - -
B - - - - - R
- - - - - -
 - - - - -
```

R - - B
Выходные данные:
output.txt:
4
- - R R
- - - - -
- - - - - -
- - - - R R R
- - - - - -
- - - - -
- - R R
Игрок R выиграл! Противник не имеет фишек!

5 Список литературы

[1] Уэзерелл, Ч. Этюды для программистов / Ч. Уэзерелл, пер. с англ. Ю. Баяковского. – М.: Мир 1982. —288с

[2] Гексагон: Игра. - URL: https://hexxagon.com (дата обращения : 10.12.2024)