

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-34Б
Бромберг Е.А.

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.

Москва, 2024 г.

Описание задания

Написать игру 2048 на языке программирования C#.

При запуске игры создаётся игровое поле в виде квадрата 4×4 . В двух клеточках игрового поля написаны 2 числа (либо 2, либо 4), остальные клеточки заполнены нулями.

Игрок использует клавиши со стрелками, чтобы «сдвигать» клеточки в четырёх направлениях: вверх, вниз, влево и вправо. Когда две клеточки с одинаковыми числами сталкиваются, они сливаются в одну, а их значение удваивается.

Цель игры: игрок должен стараться сделать так, чтобы в каждой клеточке было как можно большее число (в частности, число 2048).

Проигрыш в игре – это невозможность сделать следующий ход.

Текст программы с комментариями

Grid.cs

```
using System;
using System.Collections.Generic;

namespace Program {
    class Grid {
        private int row;
        private int col;
        private int[,] grids;
        private bool isFirst;
        public bool IsGameOver;
        private int rdNumber;

        public Grid(int row, int col) {
            this.row = row;
            this.col = col;
            this.grids = new int[row, col];
            InitGrids();
            IsGameOver = false;
            isFirst = true;
        }

        private void InitGrids() {
            for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++) {
                    grids[i, j] = 0;
                }
            }
        }

        public void PrintGrid() {
            Console.WriteLine("-----");
            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < col; j++)
                {
                    Console.Write(grids[i, j] + "\t");
                }
                Console.WriteLine();
            }
            Console.WriteLine("-----");
        }

        public void RandomGrid() {
            int count = 1;

            // В начале игры нужно сгенерировать 2 числа
            if (isFirst) {
                count = 2;
            }
        }
    }
}
```

```

        isFirst = false;
    }

    Random rd = new Random();

    for (int i = 0; i < count; ) {
        int num = rd.Next(1, 11);
        if (num < 9) {
            rdNumber = 2;
        }
        else {
            rdNumber = 4;
        }
        int gridsRow = rd.Next(0, row);
        int gridsCol = rd.Next(0, col);
        if (grids[gridsRow, gridsCol] == 0) {
            grids[gridsRow, gridsCol] = rdNumber;
        }
        else {
            continue;
        }
        i++;
    }
}

public bool MoveDirection(Direction dir) {
    bool isMove = false;
    switch (dir) {
        case Direction.Up:
            isMove = MoveUp();
            break;
        case Direction.Down:
            isMove = MoveDown();
            break;
        case Direction.Left:
            isMove = MoveLeft();
            break;
        case Direction.Right:
            isMove = MoveRight();
            break;
    }
    CheckIsGameOver();
    return isMove;
}

// Игра заканчивается, если все клетки заполнены не нулями и
// если в соседних (сверху, снизу, слева и справа) клетках нет
одинаковых чисел
private void CheckIsGameOver() {
    // Проверяем, все ли клетки заполнены не нулями

```

```

    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (grids[i, j] == 0) {
                IsGameOver = false;
                return;
            }
        }
    }

    // Проверяем, есть ли в соседних клеточках одинаковые числа
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (i - 1 >= 0 && grids[i, j] == grids[i - 1, j]) {
                IsGameOver = false;
                return;
            }
            if (i + 1 < row && grids[i, j] == grids[i + 1, j]) {
                IsGameOver = false;
                return;
            }
            if (j - 1 >= 0 && grids[i, j] == grids[i, j - 1]) {
                IsGameOver = false;
                return;
            }
            if (j + 1 < col && grids[i, j] == grids[i, j + 1]) {
                IsGameOver = false;
                return;
            }
        }
    }
    IsGameOver = true;
}

private bool MoveDown() {
    bool isMoveGrids = false;
    for (int j = 0; j < col; j++) {
        // Сдвигаем все клеточки столбца вниз
        for (int i = row - 1; i >= 0; i--) {
            // Если в текущей клеточке 0, сдвигаем верхнюю клеточку вниз
            if (grids[i, j] == 0) {
                for (int k = i - 1; k >= 0; k--) {
                    if (grids[k, j] != 0) {
                        isMoveGrids = true;
                        grids[i, j] = grids[k, j];
                        grids[k, j] = 0;
                        break;
                    }
                }
            }
        }
    }
}

```

```

        for (int i = row - 1; i >= 0; i--) {
            // Если верхней клеточки нет или в верхней клеточке 0,
            прерываем цикл, т.к. больше передвигать нечего
            if (i == 0 || grids[i - 1, j] == 0) {
                break;
            }

            if (grids[i, j] == grids[i - 1, j]) {
                isMoveGrids = true;
                grids[i - 1, j] = 0;
                grids[i, j] *= 2;

                // Сдвигаем вниз выше расположенные числа
                for (int k = i - 2; k >= 0; k--) {
                    if (grids[k, j] == 0) {
                        break;
                    }
                    grids[k + 1, j] = grids[k, j];
                    grids[k, j] = 0;
                }
            }
        }
    }
    return isMoveGrids;
}

private bool MoveUp() {
    bool isMoveGrids = false;
    for (int j = 0; j < col; j++) {
        // Сдвигаем все клеточки столбца вверх
        for (int i = 0; i < row; i++) {
            // Если в текущей клеточке 0, сдвигаем нижнюю клеточку вверх
            if (grids[i, j] == 0) {
                for (int k = i + 1; k < row; k++) {
                    if (grids[k, j] != 0) {
                        isMoveGrids = true;
                        grids[i, j] = grids[k, j];
                        grids[k, j] = 0;
                        break;
                    }
                }
            }
        }
    }

    for (int i = 0; i < row; i++) {
        // Если нижней клеточки нет или в нижней клеточке 0,
        прерываем цикл, т.к. больше передвигать нечего
        if (i == row - 1 || grids[i + 1, j] == 0) {
            break;
        }
    }
}

```

```

    }

    if (grids[i, j] == grids[i + 1, j]) {
        isMoveGrids = true;
        grids[i + 1, j] = 0;
        grids[i, j] *= 2;

        // Сдвигаем вверх ниже расположенные числа
        for (int k = i + 2; k < row; k++) {
            if (grids[k, j] == 0) {
                break;
            }
            grids[k - 1, j] = grids[k, j];
            grids[k, j] = 0;
        }
    }
}

return isMoveGrids;
}

private bool MoveLeft() {
    bool isMoveGrids = false;
    for (int i = 0; i < row; i++) {
        // Сдвигаем все клеточки строки влево
        for (int j = 0; j < col; j++) {
            // Если в текущей клеточке 0, сдвигаем правую клеточку влево
            if (grids[i, j] == 0) {
                for (int k = j + 1; k < col; k++) {
                    if (grids[i, k] != 0) {
                        isMoveGrids = true;
                        grids[i, j] = grids[i, k];
                        grids[i, k] = 0;
                        break;
                    }
                }
            }
        }
    }

    for (int j = 0; j < col; j++) {
        // Если правой клеточки нет или в правой клеточке 0,
        прерываем цикл, т.к. больше передвигать нечего
        if (j == col - 1 || grids[i, j + 1] == 0) {
            break;
        }
        if (grids[i, j] == grids[i, j + 1]) {
            isMoveGrids = true;
            grids[i, j + 1] = 0;
            grids[i, j] *= 2;
        }
    }
}

```

```

        // Сдвигаем влево расположенные справа числа
        for (int k = j + 2; k < col; k++) {
            if (grids[k, j] == 0) {
                break;
            }
            grids[i, k - 1] = grids[i, k];
            grids[i, k] = 0;
        }
    }
}

return isMoveGrids;
}

private bool MoveRight() {
    bool isMoveGrids = false;
    for (int i = 0; i < row; i++) {
        // Сдвигаем все клеточки строки вправо
        for (int j = col - 1; j >= 0; j--) {
            // Если в текущей клеточке 0, сдвигаем левую клеточку вправо
            if (grids[i, j] == 0) {
                for (int k = j - 1; k >= 0; k--) {
                    if (grids[i, k] != 0) {
                        isMoveGrids = true;
                        grids[i, j] = grids[i, k];
                        grids[i, k] = 0;
                        break;
                    }
                }
            }
        }

        for (int j = col - 1; j >= 0; j--) {
            // Если левой клеточки нет или в левой клеточке 0, прерываем
            цикл, т.к. больше передвигать нечего
            if (j == 0 || grids[i, j - 1] == 0) {
                break;
            }
            if (grids[i, j] == grids[i, j - 1]) {
                isMoveGrids = true;
                grids[i, j - 1] = 0;
                grids[i, j] *= 2;

                // Сдвигаем вправо расположенные слева числа
                for (int k = j - 2; k >= 0; k--) {
                    if (grids[k, j] == 0) {
                        break;
                    }
                }
                grids[i, k + 1] = grids[i, k];
                grids[i, k] = 0;
            }
        }
    }
}

```



```

    }
    }
    }
    }
    return isMoveGrids;
}
}

enum Direction {
    Up,
    Down,
    Left,
    Right
}
}

```

Program.cs

```

using System;

namespace Program {
    class Program {
        private static Grid grid = new Grid(4, 4);
        static void Main() {
            StartGame();
        }

        public static void StartGame() {
            grid.RandomGrid();
            grid.PrintGrid();
            Console.WriteLine("Press the up, down, left or right arrows");

            while (grid.IsGameOver == false) {
                if (Play() == false) {
                    if (grid.IsGameOver) {
                        Console.WriteLine("Game Over");
                        break;
                    }
                    continue;
                }

                grid.RandomGrid();
                Console.Clear();
                grid.PrintGrid();
            }
        }

        public static bool Play() {
            bool isCanMove = false;
            while (true) {

```

```

        bool isRightKey = false;
        ConsoleKeyInfo info = Console.ReadKey();
        switch (info.Key) {
            case ConsoleKey.UpArrow:
                isRightKey = true;
                isCanMove = grid.MoveDirection(Direction.Up);
                if (isCanMove == false) {
                    Console.WriteLine("Can't move up!");
                }
                break;
            case ConsoleKey.DownArrow:
                isRightKey = true;
                isCanMove = grid.MoveDirection(Direction.Down);
                if (isCanMove == false) {
                    Console.WriteLine("Can't move down!");
                }
                break;
            case ConsoleKey.LeftArrow:
                isRightKey = true;
                isCanMove = grid.MoveDirection(Direction.Left);
                if (isCanMove == false) {
                    Console.WriteLine("Can't move left!");
                }
                break;
            case ConsoleKey.RightArrow:
                isRightKey = true;
                isCanMove = grid.MoveDirection(Direction.Right);
                if (isCanMove == false) {
                    Console.WriteLine("Can't move right!");
                }
                break;
        }
        if (isRightKey) {
            break;
        }
    }
    return isCanMove;
}
}
}

```

Экранные формы с примерами выполнения программы

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
-----
0      0      0      4
0      0      0      0
0      0      0      0
2      0      0      0
-----
Press the up, down, left or right arrows
█
onsoleApp1 (Game2048)  Projects: ConsoleApp1  Debug Any CPU
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
-----
16     8      4      4
4      0      0      2
0      0      0      0
0      0      0      0
-----
Can't move up!
█
onsoleApp1 (Game2048)  Projects: ConsoleApp1  Debug Any CPU
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
-----
32     4      8      0
4      0      0      2
2      0      0      0
4      0      0      0
-----
█
onsoleApp1 (Game2048)  Projects: ConsoleApp1  Debug Any CPU
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
-----
16     2     32      2
4      16    256     8
8      128   32      4
16     4      8      2
-----
Can't move right!
Game Over
PS C:\Users\user\Desktop\HW\Game2048> █
onsoleApp1 (Game2048)  Projects: ConsoleApp1  Debug Any CPU
```