Московский государственный технический университет им. Н.Э. Баумана

Факультет Радиотехнический
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по Домашнему заданию “Проектирование игры Tetris на языке C# (консольный вариант)”

Выполнил:
студент группы РТ5-31Б:
Каландаров Алим
Шамильевич
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич
Подпись и дата:

Москва, 2025 г.

Текст программы

```
using System;
using System.Threading;
using System.Diagnostics;

namespace Tetris
{
    class Program
    {
        // Размеры поля
        const int mapSizeX = 10;
        const int mapSizeY = 20;

        // Игровое поле (фон): '-' = пусто, иначе буква фигуры
        static char[,] bg = new char[mapSizeY, mapSizeX];

        // Последняя нажатая клавиша (читается отдельным потоком)
        static ConsoleKeyInfo input;

        // Текущая фигура
        static int currentX = 4;
        static int currentY = 0;
        static int currentRot = 0;
        static int currentIndex = 0;
        static char currentChar = 'O';

        // "Мешок" фигур
        static int[] bag = Array.Empty<int>();
        static int bagIndex = 0;

        // Таймер падения
        static int maxTime = 20;
        static int timer = 0;

        // Символы фигур (0..6)
        readonly static string characters = "OILJSZT";

        // Координаты блоков фигур: [тип, поворот, блок(0..3), {x,y}]
        readonly static int[,,,] positions = {
            {   // O
                { {0,0},{1,0},{0,1},{1,1} },
                { {0,0},{1,0},{0,1},{1,1} },
                { {0,0},{1,0},{0,1},{1,1} },
                { {0,0},{1,0},{0,1},{1,1} }
            },
            {   // I
                { {2,0},{2,1},{2,2},{2,3} },
                { {0,2},{1,2},{2,2},{3,2} },
                { {1,0},{1,1},{1,2},{1,3} },
```

```
            { {0,1},{1,1},{2,1},{3,1} }
        },
        {   // L
            { {1,0},{1,1},{1,2},{2,2} },
            { {1,2},{1,1},{2,1},{3,1} },
            { {1,1},{2,1},{2,2},{2,3} },
            { {2,1},{2,2},{1,2},{0,2} }
        },
        {   // J
            { {2,0},{2,1},{2,2},{1,2} },
            { {1,1},{1,2},{2,2},{3,2} },
            { {2,1},{1,1},{1,2},{1,3} },
            { {0,1},{1,1},{2,1},{2,2} }
        },
        {   // S
            { {2,1},{1,1},{1,2},{0,2} },
            { {1,0},{1,1},{2,1},{2,2} },
            { {2,1},{1,1},{1,2},{0,2} },
            { {1,0},{1,1},{2,1},{2,2} }
        },
        {   // Z
            { {0,1},{1,1},{1,2},{2,2} },
            { {1,0},{1,1},{0,1},{0,2} },
            { {0,1},{1,1},{1,2},{2,2} },
            { {1,0},{1,1},{0,1},{0,2} }
        },
        {   // T
            { {0,1},{1,1},{1,0},{2,1} },
            { {1,0},{1,1},{2,1},{1,2} },
            { {0,1},{1,1},{1,2},{2,1} },
            { {1,0},{1,1},{0,1},{1,2} }
        }
    };

    static void Main()
    {
        Console.CursorVisible = false;
        Console.Title = "Tetris (Console C#) - Simple";

        // Инициализация поля
        for (int y = 0; y < mapSizeY; y++)
            for (int x = 0; x < mapSizeX; x++)
                bg[y, x] = '-';

        // Поток ввода
        Thread inputThread = new Thread(Input);
        inputThread.IsBackground = true;
        inputThread.Start();

        // Первый мешок + первая фигура
        bag = GenerateBag();
```

```
            bagIndex = 0;
            NewBlock();

            // Главный цикл
            while (true)
            {
                // Автопадение
                if (timer >= maxTime)
                {
                    if (!Collision(currentIndex, bg, currentX, currentY + 1,
currentRot))
                    {
                        currentY++;
                    }
                    else
                    {
                        FixCurrentToBackground();
                        ClearFullLines();
                        NewBlock();
                    }
                    timer = 0;
                }
                timer++;

                // Управление
                InputHandler();
                input = new ConsoleKeyInfo();

                // Рендер и печать
                var view = RenderView();
                Print(view);

                Thread.Sleep(20);
            }
        }

        static void InputHandler()
        {
            switch (input.Key)
            {
                case ConsoleKey.A:
                case ConsoleKey.LeftArrow:
                    if (!Collision(currentIndex, bg, currentX - 1, currentY,
currentRot))
                        currentX--;
                    break;

                case ConsoleKey.D:
                case ConsoleKey.RightArrow:
                    if (!Collision(currentIndex, bg, currentX + 1, currentY,
currentRot))
```

```
                    currentX++;
                break;

            case ConsoleKey.W:
            case ConsoleKey.UpArrow:
                int newRot = (currentRot + 1) % 4;
                if (!Collision(currentIndex, bg, currentX, currentY, newRot))
                    currentRot = newRot;
                break;

            case ConsoleKey.S:
            case ConsoleKey.DownArrow:
                timer = maxTime; // ускорить падение
                break;

            case ConsoleKey.Spacebar:
                // Hard drop: опускаем до упора и фиксируем
                while (!Collision(currentIndex, bg, currentX, currentY + 1,
currentRot))
                    currentY++;

                FixCurrentToBackground();
                ClearFullLines();
                NewBlock();
                timer = 0;
                break;

            case ConsoleKey.R:
                Restart();
                break;

            case ConsoleKey.Escape:
                Environment.Exit(0);
                break;

            default:
                break;
        }
    }

    static bool Collision(int index, char[,] field, int x, int y, int rot)
    {
        for (int i = 0; i < 4; i++)
        {
            int blockX = positions[index, rot, i, 0] + x;
            int blockY = positions[index, rot, i, 1] + y;

            // границы
            if (blockY >= mapSizeY || blockX < 0 || blockX >= mapSizeX)
                return true;
```

```
            // столкновение с фоном
            if (field[blockY, blockX] != '-')
                return true;
        }
        return false;
    }

    static void FixCurrentToBackground()
    {
        for (int i = 0; i < 4; i++)
        {
            int bx = positions[currentIndex, currentRot, i, 0] + currentX;
            int by = positions[currentIndex, currentRot, i, 1] + currentY;
            bg[by, bx] = currentChar;
        }
    }

    static void ClearFullLines()
    {
        while (true)
        {
            int lineY = FindFullLine(bg);
            if (lineY == -1) break;
            ClearLine(lineY);
        }
    }

    static int FindFullLine(char[,] field)
    {
        for (int y = 0; y < mapSizeY; y++)
        {
            bool full = true;
            for (int x = 0; x < mapSizeX; x++)
            {
                if (field[y, x] == '-')
                {
                    full = false;
                    break;
                }
            }
            if (full) return y;
        }
        return -1;
    }

    static void ClearLine(int lineY)
    {
        // очистить линию
        for (int x = 0; x < mapSizeX; x++)
            bg[lineY, x] = '-';
```

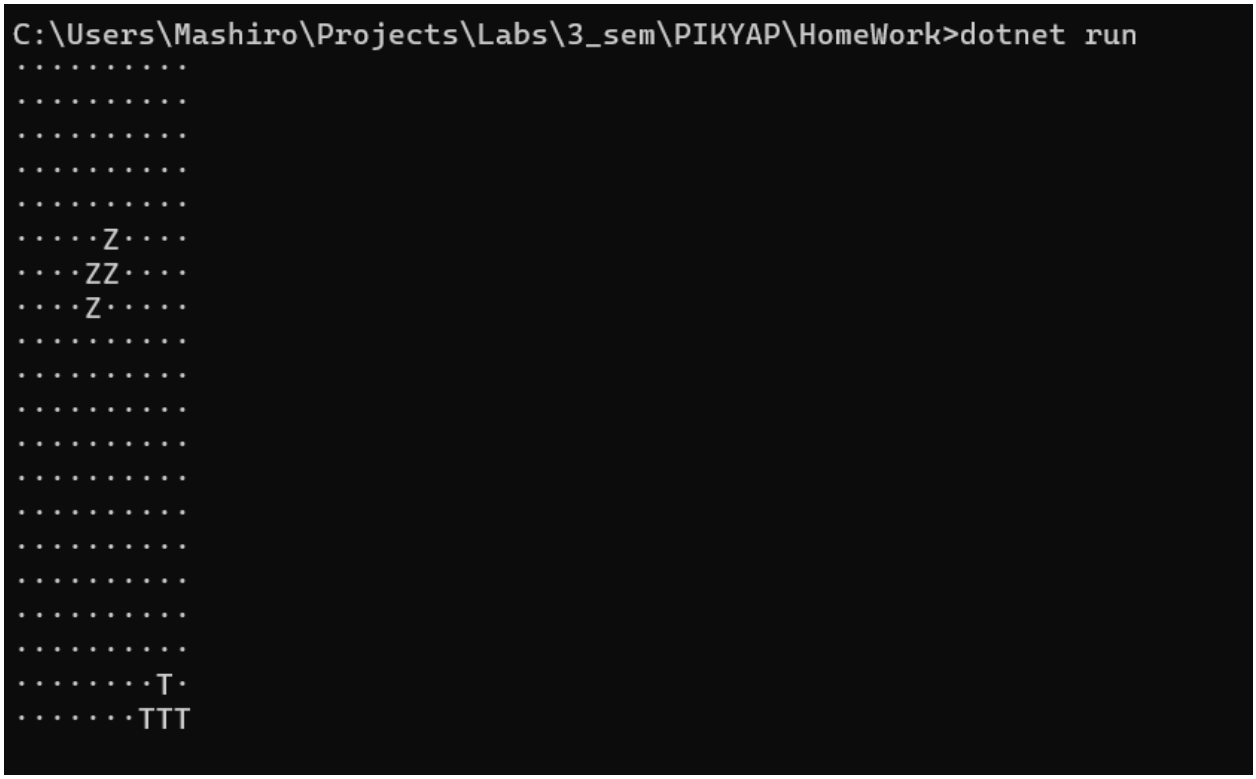
```
            // опустить всё выше вниз
            for (int y = lineY - 1; y >= 0; y--)
            {
                for (int x = 0; x < mapSizeX; x++)
                {
                    if (bg[y, x] != '-')
                    {
                        bg[y + 1, x] = bg[y, x];
                        bg[y, x] = '-';
                    }
                }
            }
        }

        static void NewBlock()
        {
            if (bagIndex >= 7)
            {
                bag = GenerateBag();
                bagIndex = 0;
            }

            currentY = 0;
            currentX = 4;
            currentRot = 0;
            currentIndex = bag[bagIndex];
            currentChar = characters[currentIndex];
            bagIndex++;

            if (Collision(currentIndex, bg, currentX, currentY, currentRot))
                GameOver();
        }

        static void GameOver()
        {
            Console.SetCursorPosition(0, mapSizeY + 1);
            Console.WriteLine("GAME OVER!");
            Console.WriteLine("Press R to restart or ESC to exit.");

            // ждём R/ESC в лоб (вне основного цикла)
            while (true)
            {
                var k = Console.ReadKey(true).Key;
                if (k == ConsoleKey.R) Restart();
                if (k == ConsoleKey.Escape) Environment.Exit(0);
            }
        }

        static int[] GenerateBag()
        {
            int[] arr = { 0, 1, 2, 3, 4, 5, 6 };
```

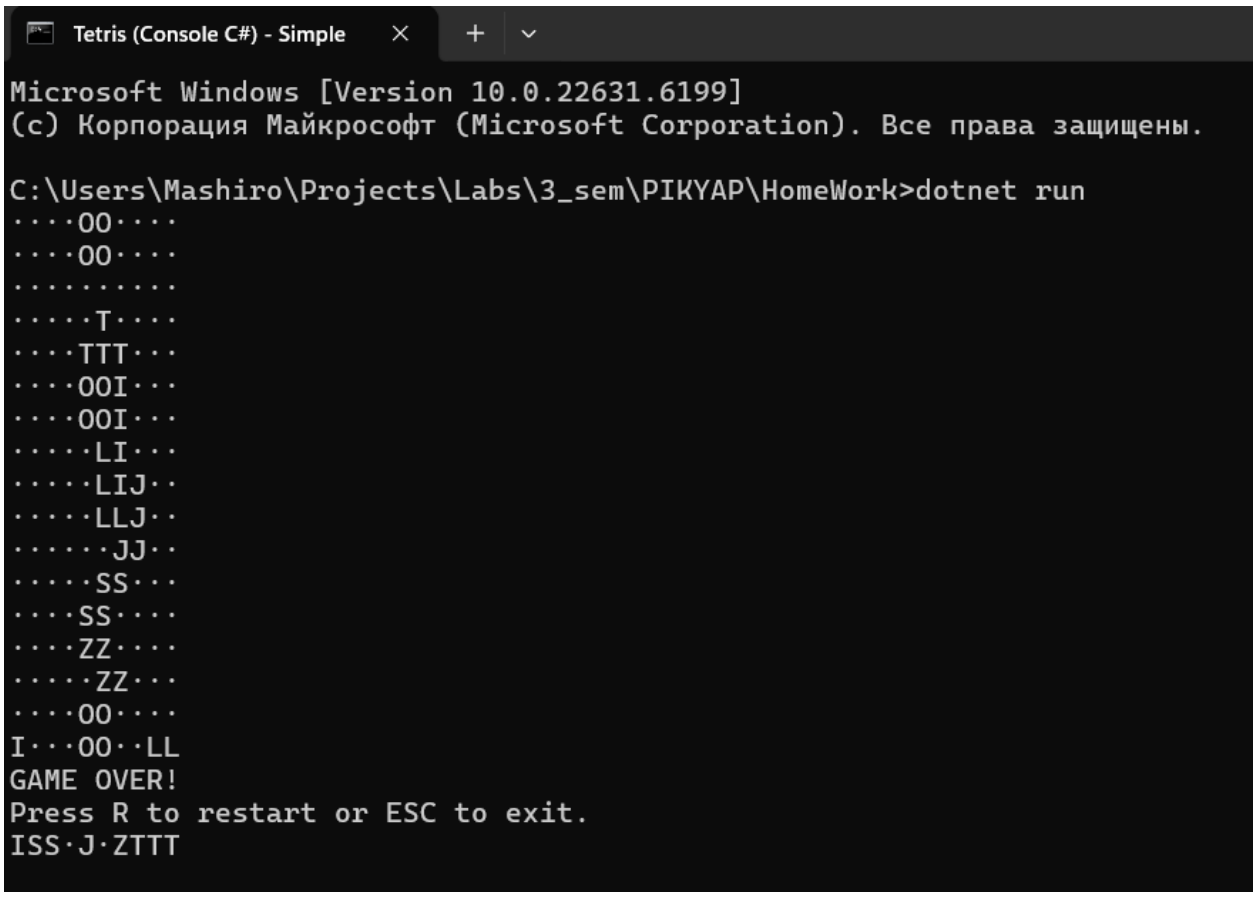
```
        Random rand = new Random();
        for (int i = arr.Length - 1; i > 0; i--)
        {
            int j = rand.Next(i + 1);
            (arr[i], arr[j]) = (arr[j], arr[i]);
        }
        return arr;
    }

    static char[,] RenderView()
    {
        char[,] view = new char[mapSizeY, mapSizeX];

        // копия фона
        for (int y = 0; y < mapSizeY; y++)
            for (int x = 0; x < mapSizeX; x++)
                view[y, x] = bg[y, x];

        // наложить текущую фигуру
        for (int i = 0; i < 4; i++)
        {
            int bx = positions[currentIndex, currentRot, i, 0] + currentX;
            int by = positions[currentIndex, currentRot, i, 1] + currentY;

            // На всякий случай защита (при гейм-овере может быть на границе)
            if (by >= 0 && by < mapSizeY && bx >= 0 && bx < mapSizeX)
                view[by, bx] = currentChar;
        }

        return view;
    }

    static void Print(char[,] view)
    {
        for (int y = 0; y < mapSizeY; y++)
        {
            for (int x = 0; x < mapSizeX; x++)
            {
                char c = view[y, x];
                // Пустоту показываем точкой для читаемости
                Console.Write(c == '-' ? '·' : c);
            }
            Console.WriteLine();
        }

        // перерисовка кадра поверх предыдущего
        Console.SetCursorPosition(0, Console.CursorTop - mapSizeY);
    }

    static void Restart()
    {
```

```
            Process.Start(Process.GetCurrentProcess().MainModule!.FileName!);
            Environment.Exit(0);
        }

        static void Input()
        {
            while (true)
                input = Console.ReadKey(true);
        }
    }
}
```

Анализ результатов

Выходные данные:




Tetris (Console C#) - Simple
Microsoft Windows [Version 10.0.22631.6199]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\Mashiro\Projects\Labs\3_sem\PIKYAP\HomeWork>dotnet run
····OO····
····OO····
··········
·····T····
····TTT···
····OOI···
····OOI···
·····LI···
·····LIJ··
·····LLJ··
·····JJ··
·····SS···
····SS····
····ZZ····
·····ZZ···
····OO····
I···OO··LL
GAME OVER!
Press R to restart or ESC to exit.
ISS·J·ZTTT