using System;

using System.Collections.Generic;

using System.Linq;

namespace SnakeGame

{

class Program

{

static void Main(string[] args)

{

Game game = new Game(20, 20);

game.Start();

}

}

public class Game

{

private int width;

private int height;

private Snake snake;

private Food food;

private bool isGameOver;

public Game(int width, int height)

{

this.width = width;

this.height = height;

snake = new Snake(width / 2, height / 2);

food = new Food(width, height);

isGameOver = false;

}

public void Start()

{

while (!isGameOver)

{

Draw();

Input();

Update();

}

Console.WriteLine("Game Over!");

}

private void Draw()

{

Console.Clear();

for (int y = 0; y < height + 1; y++)

{

for (int x = 0; x < width + 1; x++)

{

if (x == 0 || x == width || y == 0 || y == height)

{

Console.Write("#"); // границы

}

else if (snake.Body.Any(p => p.X == x && p.Y == y))

{

Console.Write("s"); // змейка

}

else if (food.Position.X == x && food.Position.Y == y)

{

Console.Write("@"); // еда

}

else

{

Console.Write(" ");

}

}

Console.WriteLine();

}

}

private void Input()

{

if (Console.KeyAvailable)

{

ConsoleKeyInfo key = Console.ReadKey(true);

switch (key.Key)

{

case ConsoleKey.UpArrow:

snake.ChangeDirection(Direction.Up);

break;

case ConsoleKey.DownArrow:

snake.ChangeDirection(Direction.Down);

break;

case ConsoleKey.LeftArrow:

snake.ChangeDirection(Direction.Left);

break;

case ConsoleKey.RightArrow:

snake.ChangeDirection(Direction.Right);

break;

}

}

}

private void Update()

{

snake.Move();

// Проверка на столкновение с едой

if (snake.Head.X == food.Position.X && snake.Head.Y == food.Position.Y)

{

snake.Grow();

food.GenerateNewPosition(snake.Body);

}

// Проверка на столкновение со стенами

if (snake.Head.X <= 0 || snake.Head.X >= width || snake.Head.Y <= 0 || snake.Head.Y >= height)

{

isGameOver = true;

}

// Проверка на столкновение с хвостом

if (snake.Body.Skip(1).Any(p => p.X == snake.Head.X && p.Y == snake.Head.Y))

{

isGameOver = true;

}

}

}

public class Snake

{

public List<Point> Body { get; private set; }

public Point Head => Body.First();

private Direction direction;

public Snake(int startX, int startY)

{

Body = new List<Point> { new Point(startX, startY) };

direction = Direction.Right;

}

public void ChangeDirection(Direction newDirection)

{

if ((direction == Direction.Left && newDirection != Direction.Right) ||

(direction == Direction.Right && newDirection != Direction.Left) ||

(direction == Direction.Up && newDirection != Direction.Down) ||

(direction == Direction.Down && newDirection != Direction.Up))

{

direction = newDirection;

}

}

public void Move()

{

Point newHead = new Point(Head.X, Head.Y);

switch (direction)

{

case Direction.Up:

newHead.Y--;

break;

case Direction.Down:

newHead.Y++;

break;

case Direction.Left:

newHead.X--;

break;

case Direction.Right:

newHead.X++;

break;

}

Body.Insert(0, newHead);

}

public void Grow()

{

Body.Insert(0, new Point(Head.X, Head.Y)); // Не убираем последний элемент, чтобы изменить длину

}

}

public class Food

{

public Point Position { get; private set; }

private Random random = new Random();

public Food(int width, int height)

{

GenerateNewPosition(new List<Point>());

}

public void GenerateNewPosition(List<Point> snakeBody)

{

do

{

Position = new Point(random.Next(1, width), random.Next(1, height));

} while (snakeBody.Any(p => p.X == Position.X && p.Y == Position.Y));

}

}

public class Point

{

public int X { get; set; }

public int Y { get; set; }

public Point(int x, int y)

{

X = x;

Y = y;

}

}

public enum Direction

{

Up,

Down,

Left,

Right

}

}