Міністерство освіти і науки України
Національний технічний університет України „КПІ”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

до лабораторної роботи з C#

з дисципліни “Основи технологій програмування”

Виконав студент *ІП-61 Денисенко Марк Олександрович*

(№ групи, прізвище, ім’я, по батькові)

Прийняв *доц. Ліщук К. І.*

(посада, прізвище, ім’я, по батькові)

Київ 2018

Постановка задачи

Варіант 5

При виконанні комп’ютерого практикуму слід реалізувати наступні задачі: a) дозволяти користувачу визначати кількість вершин графа самостійно з консолі; b) дозволяти користувачу вводити довільні матриці (списки суміжності) різної розмірності самостійно з консолі; c) мати можливість генерації довільної матриці (списку суміжності) з консолі; d) виводи на екран результат

На плоскости расположено N точек. Имеется робот, который двигается следующим образом. Стартуя с некоторой начальной точки и имея некоторое начальное направление, робот движется до первой встреченной на его пути точки, изменяя в ней свое текущее направление на 90 градусов, т.е. поворачивая налево или направо. После этого он продолжает движение аналогично. Если робот достиг начальной точки, либо не может достичь новой точки (которую он еще не посещал), то он останавливается. Определить, может ли робот посетить все N точек, если:

- Определены начальные точка и направление робота.
- Определена начальная точка, а направление робота можно выбирать.
- Начальную точку и направление робота можно выбирать.

Координаты точек - целые числа, угол измеряется в радианах относительно оси OX

Робота програми

```
Denysenko Mark IP-61 (variant 5)
Input size for field!
Input N: 4
Input M: 4
Input points for generating: 3
        0       1       2       3
0       0       0       0       0
1       0       0       0       0
2       1       0       0       0
3       0       1       1       0
Choose way for starting program:
 1 - Generate start point and direction
 2 - Generate start point and choose direction manually
 3 - Manually choose point and direction
Your operation: 3
Input start position and direction for robot!
Input X: 0
Input Y: 1
Input direction (up - u, left - l, down - d, right - r): d
Robot postion (0,1), direction (Down)
        0       1       2       3
0       0       0       0       0
1       R       0       0       0
2       1       0       0       0
3       0       1       1       0
Way (only special achieved points):
Robot postion (0,2), direction (Down)
NO, robot can't go across all points!
```

```
Denysenko Mark IP-61 (variant 5)
Input size for field!
Input N: 3
Input M: 3
Input points for generating: 3
        0       1       2
0       0       1       0
1       0       1       1
2       0       0       0
Choose way for starting program:
 1 - Generate start point and direction
 2 - Generate start point and choose direction manually
 3 - Manually choose point and direction
Your operation: 3
Input start position and direction for robot!
Input X: 0
Input Y: 0
Input direction (up - u, left - l, down - d, right - r): r
Robot postion (0,0), direction (Right)
        0       1       2
0       R       1       0
1       0       1       1
2       0       0       0
Way (only special achieved points):
Robot postion (1,0), direction (Down)
Robot postion (1,1), direction (Right)
Robot postion (2,1), direction (Right)
YES, robot can go across all points!
```

Код програми:

1) Program.cs

```
using System;

namespace Robot_Program
{
    sealed class Program
    {
        static void Main(string[] args)
        {
            Console.Title = "Denysenko Mark IP-61 (variant 5)";

            Execute();

            Console.ReadKey();
        }

        static void Execute()
        {
            Console.WriteLine("Input size for field!");
            Console.Write("Input N: ");
            int n = Convert.ToInt32(Console.ReadLine());
            Console.Write("Input M: ");
            int m = Convert.ToInt32(Console.ReadLine());

            Field testField = new Field(n, m);

            Console.Write("Input points for generating: ");
            testField.GeneratePoints(Convert.ToInt32(Console.ReadLine()));
            testField.Output();

            Robot testRobot = MakeRobot(testField);
            testRobot.ShowInfo();
            testField.Output(testRobot);

            Console.WriteLine("Way (only special achieved points): ");
            if(testField.Check(testRobot))
            {
                Console.WriteLine("YES, robot can go across all points!");
            }
            else
            {
                Console.WriteLine("NO, robot can't go across all points!");
            }

        }

        static Point ReadPoint()
        {
            Console.Write("Input X: ");
            int x = Convert.ToInt32(Console.ReadLine());
            Console.Write("Input Y: ");
            int y = Convert.ToInt32(Console.ReadLine());

            return new Point(x, y);
        }

        static Direction ReadDirection()
        {
            Console.Write("Input direction (up - u, left - l, down - d, right - r): ");
            string dir = Console.ReadLine();

            switch(dir[0])
            {
```

```
                case 'U':
                case 'u':
                    return Direction.Up;
                case 'L':
                case 'l':
                    return Direction.Left;
                case 'D':
                case 'd':
                    return Direction.Down;
                case 'R':
                case 'r':
                    return Direction.Right;
            }

            return Direction.Default;
        }

        static Robot MakeRobot(Field f)
        {
            Console.WriteLine("Choose way for starting program:");
            Console.WriteLine(" 1 - Generate start point and direction");
            Console.WriteLine(" 2 - Generate start point and choose direction manually");
            Console.WriteLine(" 3 - Manually choose point and direction");

            Console.Write("Your operation: ");
            int operation = Convert.ToInt32(Console.ReadLine());

            switch(operation)
            {
                case 1:
                    Console.WriteLine("Robot generated automatically!");
                    return f.GenerateRobot();
                case 2:
                    Console.WriteLine("Point generated, input direction!");
                    return f.GenerateRobot(ReadDirection());
                case 3:
                    Console.WriteLine("Input start position and direction for robot!");
                    return new Robot(ReadPoint(), ReadDirection());
            }

            return f.GenerateRobot();
        }
    }
}
```

2) Direction.cs

```
namespace Robot_Program
{
    enum Direction
    {
        Right = 0,
        Down = 90,
        Left = 180,
        Up = 270,
        Default = Right
    }
}
```

3) Points.cs

```
namespace Robot_Program
{
    internal sealed class Point
    {
        public int x;
```

```
        public int y;

        public Point(int x, int y)
        {
            this.x = x;
            this.y = y;
        }

        public override int GetHashCode()
        {
            return x ^ y;
        }

        public override bool Equals(object obj)
        {
            if (obj == null || this.GetType() != obj.GetType())
                return false;

            Point p = (Point)obj;

            return (x == p.x) && (y == p.y);
        }

        public Point DeepCopy()
        {
            return new Point(this.x, this.y);
        }
    }
}
```

4) Field.cs

```
using System;

namespace Robot_Program
{
    /// <summary>
    /// Field has matrix with references on Point, that should be achieve !
    /// If reference has NULL, it`s only road
    /// </summary>
    class Field
    {
        private Point[][] matrix;
        private int valueOfPoints;
        private Random generator;
        private readonly int N;
        private readonly int M;

        public Field(int n, int m)
        {
            matrix = new Point[n][];
            for (int i = 0; i < n; i++)
            {
               matrix[i] = new Point[m];
            }
            N = n;
            M = m;

            generator = new Random();
        }

             /// <summary>
        /// Main method of program. Calculate possibility to achieve all generated points
        /// </summary>
        /// <param name="rob">Robot with start point and direction, which should achieve all
points</param>
```

```
        /// <returns>return true if it possible or false in all another conditions</returns>
        public bool Check(Robot rob)
        {
            int pointCounter = 0;
            int x = rob.Position.x;
            int y = rob.Position.y;

            if (matrix[y][x] != null)
            {
                pointCounter++;
                matrix[y][x] = null;
            }

            while(pointCounter != valueOfPoints)
            {
                if (GoToPoint(rob))
                {
                    pointCounter++;
                    x = rob.Position.x;
                    y = rob.Position.y;
                    matrix[y][x] = null;
                    ChangeDirection(rob);
                    rob.ShowInfo();
                }
                else
                {
                    return false;
                }
            }

            return true;
        }

        public bool ChangeDirection(Robot r)                    // return false if robot can`t
find direction with special points
        {
            Robot RightRobot = r.DeepCopy();
            RightRobot.TurnRight();
            if (GoToPoint(RightRobot))
            {
                r.Direction = RightRobot.Direction;
                return true;
            }

            Robot LeftRobot = r.DeepCopy();
            LeftRobot.TurnLeft();
            if (GoToPoint(LeftRobot))
            {
                r.Direction = LeftRobot.Direction;
                return true;
            }

            return false;
        }

        private bool GoToPoint(Robot rob)
        {
            while (true)
            {
                rob.MakeStep();
                int x = rob.Position.x;
                int y = rob.Position.y;

                if (y >= N || x >= M || y < 0 || x < 0)
                    return false;
```

```
            if (matrix[y][x] != null)
            {
                return true;
            }
        }
    }

    public Robot GenerateRobot(Direction d = Direction.Default)
    {
        int y = generator.Next(matrix.Length);
        int x = generator.Next(matrix[y].Length);

        return new Robot(new Point(x, y), d);
    }

    public void GeneratePoints(int value)
    {
        valueOfPoints = value;

        for(int i = 0; i < value; i++)
        {
            int y = generator.Next(matrix.Length);
            int x = generator.Next(matrix[y].Length);

            if (matrix[y][x] == null)
            {
                matrix[y][x] = new Point(x, y);
            }
            else
            {
                i--;
            }
        }
    }

    public void Output()
    {
        for(int i = 0; i < matrix.Length; i++)
        {
            if(i == 0)                                          // write OX
            {
                Console.Write("\t");

                for (int j = 0; j < matrix[i].Length; j++)
                    Console.Write(j + "\t");

                Console.WriteLine();
            }
            for(int j = 0; j < matrix[i].Length; j++)
            {
                if (j == 0)                                     // write OY
                    Console.Write(i + "\t");

                if(matrix[i][j] != null)
                {
                    Console.Write("1\t");
                }
                else
                {
                    Console.Write("0\t");
                }
            }
            Console.WriteLine();
        }
```

```
        }

        public void Output(Robot r)
        {
            for (int i = 0; i < matrix.Length; i++)
            {
                if (i == 0)                                          // write OX
                {
                    Console.Write("\t");

                    for (int j = 0; j < matrix[i].Length; j++)
                        Console.Write(j + "\t");

                    Console.WriteLine();
                }
                for (int j = 0; j < matrix[i].Length; j++)
                {
                    if (j == 0)                                     // write OY
                        Console.Write(i + "\t");

                    if(r.Position.x == j && r.Position.y == i)
                    {
                        Console.Write("R\t");
                        continue;
                    }

                    if (matrix[i][j] != null)
                    {
                        Console.Write("1\t");
                    }
                    else
                    {
                        Console.Write("0\t");
                    }
                }
                Console.WriteLine();
            }
        }
    }
}
```

5) Robot.cs

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

namespace Robot_Program
{
    class Robot
    {
        Point startPosition;
        Point currentPosition;
        Direction direction;
        //List<Point> way;

        public Point Position
        {
            get
            {
                return currentPosition;
            }
        }

        public Direction Direction
        {
            get
```

```
    {
        return direction;
    }
    set
    {
        direction = value;
    }
}

public Robot():
    this(new Point(0, 0), Direction.Default) { }

public Robot(Point p, Direction d = Direction.Default)
{
    startPosition = p;
    currentPosition = startPosition;
    //way.Add(currentPosition);
    direction = d;
}

public bool MakeStep()
{
    switch(direction)
    {
        case Direction.Right:
            currentPosition.x++;
            break;
        case Direction.Down:
            currentPosition.y++;
            break;
        case Direction.Left:
            currentPosition.x--;
            break;
        case Direction.Up:
            currentPosition.y--;
            break;
    }

    return true;
}

public void TurnRight()                     // + Pi/2
{
    direction += 90;

    if((int)direction == 360)
    {
        direction = Direction.Right;
    }
}

public void TurnLeft()                      // - Pi/2
{
    switch(direction)
    {
        case Direction.Right:
            direction = Direction.Up;
            break;
        case Direction.Down:
            direction = Direction.Right;
            break;
        case Direction.Left:
            direction = Direction.Down;
            break;
        case Direction.Up:
```

```
                    direction = Direction.Left;
                    break;
                }
            }

            public void ShowInfo()
            {
                Console.WriteLine($"Robot postion ({currentPosition.x},{currentPosition.y}),
direction ({direction.ToString()})");
            }

            public Robot DeepCopy()
            {
                Robot tmp = new Robot(startPosition.DeepCopy(), direction);
                tmp.currentPosition = this.currentPosition.DeepCopy();

                return tmp;
            }
        }
}
```