

Міністерство освіти і науки України
Національний технічний університет України
Київський Політехнічний Інститут імені Ігоря Сікорського
Кафедра ОТ

Лабораторна робота № 4
з дисципліни "Основи Web-програмування"

Виконав:
студент 2-го курсу
Групи: ІП-64 ФІОТ
Гордієнко Нікіта
Заліковка: №6404

Київ-2018

Постановка задачі до комп'ютерного практикуму № 4

При виконанні комп'ютерного 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

ВЛАСНИЙ ВАРІАНТ ЗАВДАННЯ:

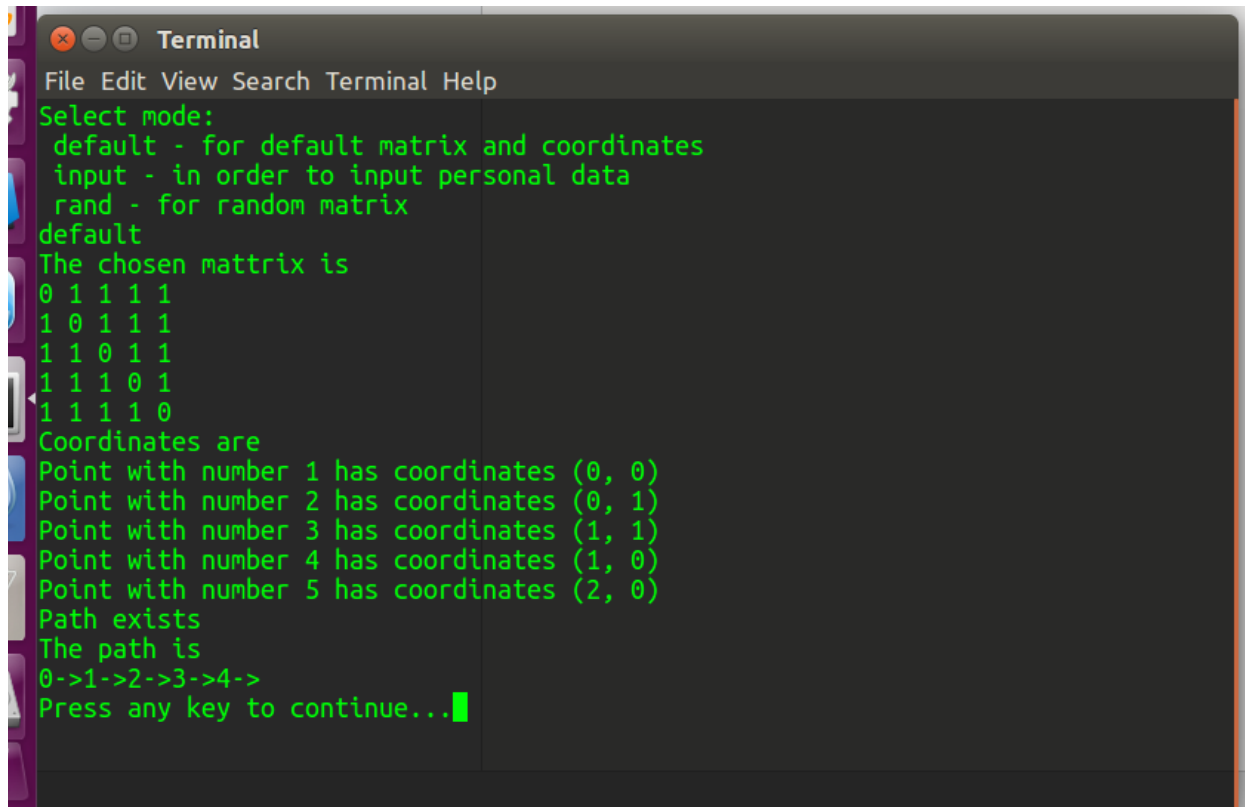
Варіант 5

На плоскости расположено N точек. Имеется робот, который движется 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 начальная 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

СКРИНШОТИ:

A screenshot of a macOS Terminal window. The window has a title bar with standard macOS window controls (red, yellow, green buttons) and the title "Terminal". Below the title bar is a menu bar with the options "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal displays the following text in green on a dark background:

```
Select mode:  
  default - for default matrix and coordinates  
  input - in order to input personal data  
  rand - for random matrix  
default  
The chosen matrix is  
0 1 1 1 1  
1 0 1 1 1  
1 1 0 1 1  
1 1 1 0 1  
1 1 1 1 0  
Coordinates are  
Point with number 1 has coordinates (0, 0)  
Point with number 2 has coordinates (0, 1)  
Point with number 3 has coordinates (1, 1)  
Point with number 4 has coordinates (1, 0)  
Point with number 5 has coordinates (2, 0)  
Path exists  
The path is  
0->1->2->3->4->  
Press any key to continue...
```

Код Програми:

```
using System;
namespace Laba1
{
    class Laba1
    {
        static bool calcAngle(int x1, int y1, int x2, int y2, int x3, int y3)
        {
            double angle = Math.Atan2(y2 - y1, x2 - x1);
            double degrees = angle * (180 / Math.PI);
            double angle1 = Math.Atan2(y3 - y2, x3 - x2);
            double degrees1 = angle1 * (180 / Math.PI);
            int sub = Convert.ToInt32(degrees - degrees1);
            if (sub == 0 || sub == 180 || sub == -180)
            {
                return false;
            }
            else if (sub % 90 == 0)
                return true;
            else
                return false;
        }

        static void Main(string[] args)
        {
            int[] c; // номер хода, на котором посещается вершина
            int[] path; // номера посещаемых вершин

            int v0 = 0; // начальная вершина
            int[] y, x;
            int[,] matrix;
            int size;
            int chosen_path;
            Console.WriteLine("Select mode:\n default - for default matrix and
coordinates\n input - in order to input personal data \n rand - for random matrix");
            String mode = Console.ReadLine();
            switch (mode)
            {
                case "rand":
                    Console.WriteLine("Enter the size");
                    size = GetSize();
                    if (size == -1)
                    {
                        Console.WriteLine("Size is incorrect");
                        return;
                    }
                    c = new int[size];
```

```
path = new int[size];
x = GenerateRandomCoordinates(size, 10);
y = GenerateRandomCoordinates(size, 10);
chosen_path = GenerateRandomPath(size);
```

```
matrix = GenerateRandomMatrix(size);
```

```
break;
```

```
case "input":
```

```
    x = readCoordinates("x");
    y = readCoordinates("y");
    size = x.GetLength(0);
    if (x.GetLength(0)==y.GetLength(0))
    {
        matrix = readMatrix(size);
        c = new int[size];
        path = new int[size];
        chosen_path = GetChosenPath();
        if (chosen_path == -1)
        {
            Console.WriteLine("Path is incorrect");
            return;
        }
    }
    else{
        Console.WriteLine("Size of coordinates is incorrect");
        return;
    }
    break;
```

```
case "default":
```

```
    size = 5;
    x = new int[] { 0, 0, 1, 1, 2 };
    y = new int[] { 0, 1, 1, 0, 0 };
    c = new int[5];
    path = new int[5];
    chosen_path = 1;
    matrix = new int[5, 5]
    {
        {0, 1, 1, 1, 1},
        {1, 0, 1, 1, 1},
        {1, 1, 0, 1, 1},
        {1, 1, 1, 0, 1},
        {1, 1, 1, 1, 0}
    };
    break;
```

```
default:
```

```
    return;
```

```

    }
    for (int i = 0; i < path.GetLength(0); i++)
    {
        path[i] = -1;
    }

    Console.WriteLine("The chosen matrix is");
    PrintMatrix(matrix);
    Console.WriteLine("Coordinates are");
    PrintCoordinates(x, y);
    if(!checkIfMatrixCorrect(matrix))
        return;

    for (int j = 0; j < size; j++) c[j] = -1;
    path[0] = v0;
    c[v0] = 0;
    if (gamilton(1, size, path, matrix, c, v0, x, y, chosen_path) == 1 ||
    path[path.GetLength(0)-1]!=-1) {
        Console.WriteLine("Path exists");
        Console.WriteLine("The path is");
        for (int i = 0; i < path.GetLength(0); i++)
            Console.Write(path[i] + "->");
    }
    else { Console.WriteLine("No path\n"); }

}

static void PrintCoordinates (int [] x, int [] y)
{
    for (int i = 0; i < x.GetLength(0); i++)
    {
        Console.Write($"Point with number {i + 1} has coordinates ({x[i]}, {y[i]})\n");
    }
}

static int[] getNumOfWays(int[,] a)
{
    int[] res = new int[a.GetLength(0)];
    for (int i = 0; i < a.GetLength(0); i++)
    {
        res[i] = 0;
        for (int j = 0; j < a.GetLength(1); j++)
        {
            if (i != j && a[i, j] == 1)
            {
                res[i] += 1;
            }
        }
    }
}

```

```

    }
    return res;
}

static int gamilton(int k, int n, int[] path, int[,] a, int[] c, int v0, int[] x, int[] y, int
chosen_path = 0)
{
    int v, q1 = 0;
    if (chosen_path == 0)
    {
        for (v = 0; v < n && q1 != 1; v++)
        {
            if (k < 2)
            {
                if (a[v, path[k - 1]] == 1 || a[path[k - 1], v] == 1)
                {
                    if (k == n) q1 = 1;

                    else if (c[v] == -1)
                    {
                        c[v] = k; path[k] = v;
                        q1 = gamilton(k + 1, n, path, a, c, v0, x, y);
                        if (q1 != 1) c[v] = -1;
                    }
                    else continue;
                }
            }
            else
            {
                if ((a[v, path[k - 1]] == 1 || a[path[k - 1], v] == 1) && calcAngle(x[path[k - 2]],
y[path[k - 2]], x[path[k - 1]], y[path[k - 1]], x[v], y[v]) )
                {
                    if (k == n) q1 = 1;
                    else if (c[v] == -1)
                    {
                        c[v] = k; path[k] = v;
                        q1 = gamilton(k + 1, n, path, a, c, v0, x, y);
                        if (q1 != 1) c[v] = -1;
                    }
                    else continue;
                }
            }
        }
    }
    return q1;
}
else{
    int z = 0;
    for (v = 0; v < n && q1 != 1; v++)
    {
        if (a[v, path[k - 1]] == 1 || a[path[k - 1], v] == 1)
        {

```



```

        z++;
        if (z == chosen_path)
        {
            if (k == n) q1 = 1;
            else if (c[v] == -1)
            {
                c[v] = k; path[k] = v;
                q1 = gamilton(k + 1, n, path, a, c, v0, x, y);
                if (q1 != 1) c[v] = -1;
            }
            else continue;
        }
    }
}
return q1;
}
}

```

```

static bool checkIfMatrixCorrect(int[,] a) {
    for (int i = 0; i < a.GetLength(0); i++)
    {
        for (int j = 0; j < a.GetLength(1); j++)
        {
            if (a[i, j] != a[j, i]){
                return false;
            }
        }
    }
    return true;
}

```

```

static int[] readCoordinates(string x)
{
    Console.WriteLine(String.Format("Enter the {0}: ", x));
    String foo = Console.ReadLine();
    string[] tokens = foo.Split(',');
    int size = tokens.GetLength(0);
    Console.WriteLine(size);
    int[] numsX = new int[size];
    int i = 0;
    foreach (string s in tokens)
    {
        numsX[i] = Convert.ToInt32(s);
        i++;
    }
}

```

```

        foreach (int z in numsX)
            Console.WriteLine(z);
        return numsX;
    }

```

```

static int[,] readMatrix(int size)
{

```

```

    if (size < 0)
        return null;
    int[,] nums = new int[size, size];
    for (int j = 0; j < size; j++)
    {
        string foo = Console.ReadLine();
        string[] tokens = foo.Split(',');
        if (tokens.GetLength(0) > size)
            return null;

```

```

        int i = 0;
        foreach (string s in tokens)
        {
            nums[j, i] = Convert.ToInt32(s);
            i++;
        }

```

```

        for (; i < size; i++)
        {
            nums[j, i] = 0;
        }
    }
    return nums;
}
static void PrintMatrix(int[,] matrix)
{
    for (int i = 0; i < matrix.GetLength(0); i++)
    {
        for (int j = 0; j < matrix.GetLength(0); j++)
        {
            Console.Write(matrix[i, j]);
            Console.Write(" ");
        }
        Console.WriteLine("\n");
    }
}

```

```

static int[,] GenerateRandomMatrix(int size)

```

```

{
    int[,] matrices = new int[size, size];
    Random rnd = new Random();
    for (int i = 0; i < size; i++)
    {
        for (int j = i; j < size; j++)
        {
            if (i == j)
            {
                matrices[i, j] = 0;
            }
            else
            {
                if (rnd.Next(-3, 10) > 0)
                {
                    matrices[i, j] = 1;
                }
                else
                {
                    matrices[i, j] = 0;
                    matrices[j, i] = matrices[i, j];
                }
            }
        }
    }
    return matrices;
}

static int GenerateRandomPath(int size)
{
    if (size < 0)
        return -1;
    Random rnd = new Random();
    return rnd.Next(0, size);
}

static int[] GenerateRandomCoordinates(int size, int range)
{
    int[] nums = new int[size];
    Random rnd = new Random();
    for (int i = 0; i < size; i++)
    {
        nums[i] = rnd.Next(-range, range);
    }
    return nums;
}

static int GetSize()
{
    try
    {

```

```

        return Convert.ToInt32(Console.ReadLine());
    }
    catch(Exception)
    {
        Console.WriteLine("String is not in correct format");
        return -1;
    }
}

static int GetChosenPath()
{
    try
    {
        return Convert.ToInt32(Console.ReadLine());
    }
    catch (Exception)
    {
        Console.WriteLine("String is not in correct format");
        return -1;
    }
}

}
}

```