Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Звіт

про виконання лабораторної роботи № 2

з курсу "Алгоритмізація та програмування"

«Трикутник і точка»

Виконав:

ст. гр. ФЕІ-11

Стасів Петро

Перевірив:

доц. Хвищун І.О.

Львів-2021

Звіт

Мета: у середовищі Delphi написати програму в якій можливо задати довільний трикутник на прямокутній системі координат і точку, для якої буде здійснено три перевірки:

1) Чи точка знаходиться всередині трикутника.
2) Чи точка знаходиться на одній з сторін трикутника.
3) Чи точка знаходиться на одній з вершин трикутника.

Виконання лабораторної роботи:

1) Описуємо класи які будуть використані при реалізації програми

```
1. type
2.  Vector2D = class
3.  public
4.    x : double;
5.    y : double;
6.
7.    constructor Create(_x : double; _y : double);
8.
9.    function Sub(v : Vector2D) : Vector2D;
10.
11.      function Length : real;
12.    end;
13.
14.    constructor Vector2D.Create(_x: double; _y: double);
15.    begin
16.      x := _x;
17.      y := _y;
18.    end;
19.
20.    function Vector2D.Sub(v: Vector2D): Vector2D;
21.    begin
22.      var r := Vector2D.Create(0, 0);
23.
24.      r.x := x - v.x;
25.      r.y := y - v.y;
26.
27.      Sub := r;
28.    end;
29.
30.    function Vector2D.Length: real;
31.    begin
```

```
32.      Length := Sqrt(x * x + y * y);
33.    end;
34.
35.
36.   type
37.    Triangle = class
38.    public
39.      Point1 : Vector2D;
40.      Point2 : Vector2D;
41.      Point3 : Vector2D;
42.
43.      constructor Create(p1 : Vector2D; p2 : Vector2D; p3 :
   Vector2D);
44.
45.      function GetArea() : double;
46.      function ComputeBarycentricCoords(p : Vector2D) :
   Vector2D;
47.    end;
48.
49.    constructor Triangle.Create(p1: Vector2D; p2: Vector2D;
   p3: Vector2D);
50.    begin
51.      Point1 := p1;
52.      Point2 := p2;
53.      Point3 := p3;
54.    end;
55.
56.    //Рахує площу трикутника за формулою Герона
57.    function Triangle.GetArea: double;
58.    begin
59.      var side1 := Point2.Sub(Point1).Length();
60.      var side2 := Point3.Sub(Point1).Length();
61.      var side3 := Point3.Sub(Point2).Length();
62.
63.      var p := 0.5 * (side1 + side2 + side3);
64.
65.      GetArea := Sqrt(p * (p - side1) * (p - side2) * (p -
   side3));
66.    end;
67.
68.    //Рахує барицентричні координати трикутника за формулою,
   яка припускає, що координати
69.    //пропорційні до площ трикутників утворенних з точкою p
70.    //Повертає тільки 2 координати тому, що зможемо відновити
   третю: w = 1 - (u + v)
71.    function Triangle.ComputeBarycentricCoords(p : Vector2D):
   Vector2D;
72.    begin
```

```
73.      var t1 := Triangle.Create(Point3, Point1, p);
74.      var t2 := Triangle.Create(Point1, Point2, p);
75.
76.      var u := t1.GetArea() / GetArea();
77.      var v := t2.GetArea() / GetArea();
78.
79.      ComputeBarycentricCoords := Vector2D.Create(u, v);
80.    end;
```

2) Перевірка розташування точки відносно трикутника

```
1.       //point1, point2, point3 і userPoint - точки задані
   користувачем
2.       var userTrig := Triangle.Create(point1, point2, point3);
3.       var barycentricP :=
   userTrig.ComputeBarycentricCoords(userPoint);
4.
5.       //Перевіряємо чи точка в межах трикутника
6.       if(barycentricP.x >= 0.0) and (barycentricP.y >= 0.0)
7.          and ((barycentricP.x + barycentricP.y) <= 1.0)  then
8.          begin
9.
10.            //Відновлюємо третю координату
11.            var thirdCoord := 1.0 - (barycentricP.x +
   barycentricP.y);
12.
13.            //Так як ця перевірка можлива тільки, якщо точка в
   трикутнику
14.            //отже сума її координат має дорівнювати 1.0
15.            //Тому для перевірки того чи точка знаходиться на
   координаті трикутника
16.            //ми можемо використати просто перевірку нижче
17.            if(barycentricP.x = 1.0) or (barycentricP.y = 1.0)
18.               or (thirdCoord = 1.0) then
19.                Writeln('Point on the triangle vertex!')
20.            else if(barycentricP.x = 0.0) or (barycentricP.y =
   0.0) //Перша умова з попередньої превірки застосовується і тут
21.                    or (thirdCoord = 0.0) then
22.                Writeln('Point on the triangle side!')
23.            else
24.                Writeln('Point inside triangle!');
25.            end
26.         else
27.            Writeln('Point lies outside of the triangle!');
28.    end;
```

Тестування:

```
Enter first triangle vertex:
1 1
Enter second triangle vertex:
5 5
Enter third triangle vertex:
7 1
Enter point:
1 1
Point on the triangle vertex!
Enter point:
5 1
Point on the triangle side!
Enter point:
5 2
Point inside triangle!
Enter point:
10 1
Point lies outside of the triangle!
```

```
Enter first triangle vertex:
-2 0
Enter second triangle vertex:
-3 3
Enter third triangle vertex:
-4 0
Enter point:
-3 0
Point on the triangle side!
Enter point:
-3 3
Point on the triangle vertex!
Enter point:
-3 2
Point inside triangle!
Enter point:
5 5
Point lies outside of the triangle!
```

```
Enter first triangle vertex:
3.5 0
Enter second triangle vertex:
3.5 -5
Enter third triangle vertex:
7.2 -6
Enter point:
3.5 0
Point on the triangle vertex!
Enter point:
5 -5
Point inside triangle!
Enter point:
3.5 -2
Point on the triangle side!
Enter point:
10 -1
Point lies outside of the triangle!
```

Текст програми:

```
1. uses
2.  System.SysUtils;
3.
4. type
5.  Vector2D = class
6.  public
7.    x : double;
8.    y : double;
9.
10.     constructor Create(_x : double; _y : double);
11.
12.     function Sub(v : Vector2D) : Vector2D;
13.
14.     function Length : real;
15.   end;
16.
17.   constructor Vector2D.Create(_x: double; _y: double);
18.   begin
19.     x := _x;
20.     y := _y;
21.   end;
22.
23.   function Vector2D.Sub(v: Vector2D): Vector2D;
24.   begin
25.     var r := Vector2D.Create(0, 0);
26.
27.     r.x := x - v.x;
28.     r.y := y - v.y;
29.
30.     Sub := r;
31.   end;
32.
33.   function Vector2D.Length: real;
34.   begin
35.     Length := Sqrt(x * x + y * y);
36.   end;
37.
38.
39.  type
40.   Triangle = class
41.   public
42.     Point1 : Vector2D;
43.     Point2 : Vector2D;
44.     Point3 : Vector2D;
45.
46.     constructor Create(p1 : Vector2D; p2 : Vector2D; p3 :
  Vector2D);
```

```
47.
48.       function GetArea() : double;
49.       function ComputeBarycentricCoords(p : Vector2D) :
  Vector2D;
50.     end;
51.
52.     constructor Triangle.Create(p1: Vector2D; p2: Vector2D;
  p3: Vector2D);
53.     begin
54.       Point1 := p1;
55.       Point2 := p2;
56.       Point3 := p3;
57.     end;
58.
59.     //Рахує площу трикутника за формулою Герона
60.     function Triangle.GetArea: double;
61.     begin
62.       var side1 := Point2.Sub(Point1).Length();
63.       var side2 := Point3.Sub(Point1).Length();
64.       var side3 := Point3.Sub(Point2).Length();
65.
66.       var p := 0.5 * (side1 + side2 + side3);
67.
68.       GetArea := Sqrt(p * (p - side1) * (p - side2) * (p -
  side3));
69.     end;
70.
71.     //Рахує барицентричні координати трикутника за формулою,
  яка припускає, що координати
72.     //пропорційні до площ трикутників утворенних з точкою p
73.     //Повертає тільки 2 координати тому, що зможемо відновити
  третю: w = 1 - (u + v)
74.     function Triangle.ComputeBarycentricCoords(p : Vector2D):
  Vector2D;
75.     begin
76.       var t1 := Triangle.Create(Point3, Point1, p);
77.       var t2 := Triangle.Create(Point1, Point2, p);
78.
79.       var u := t1.GetArea() / GetArea();
80.       var v := t2.GetArea() / GetArea();
81.
82.       ComputeBarycentricCoords := Vector2D.Create(u, v);
83.     end;
84.
85.   begin
86.     var point1 := Vector2D.Create(0.0, 0.0);
87.     var point2 := Vector2D.Create(0.0, 0.0);
88.     var point3 := Vector2D.Create(0.0, 0.0);
```

```
89.
90.     Writeln('Enter first triangle vertex:');
91.     Readln(point1.x, point1.y);
92.
93.     Writeln('Enter second triangle vertex:');
94.     Readln(point2.x, point2.y);
95.
96.     Writeln('Enter third triangle vertex:');
97.     Readln(point3.x, point3.y);
98.
99.     while True do
100.    begin
101.        var userPoint := Vector2D.Create(0.0, 0.0);
102.
103.        Writeln('Enter point:');
104.        Readln(userPoint.x, userPoint.y);
105.
106.
107.        var userTrig := Triangle.Create(point1, point2,
    point3);
108.        var barycentricP :=
    userTrig.ComputeBarycentricCoords(userPoint);
109.
110.        //Перевіряємо чи точка в межах трикутника
111.        if(barycentricP.x >= 0.0) and (barycentricP.y >=
    0.0)
112.            and ((barycentricP.x + barycentricP.y) <= 1.0)
    then
113.            begin
114.
115.            //Відновлюємо третю координату
116.            var thirdCoord := 1.0 - (barycentricP.x +
    barycentricP.y);
117.
118.            //Так як ця перевірка можлива тільки, якщо точка в
    трикутнику
119.            //отже сума її координат має дорівнювати 1.0
120.            //Тому для перевірки того чи точка знаходиться на
    координаті трикутника
121.            //ми можемо використати просто перевірку нижче
122.            if(barycentricP.x = 1.0) or (barycentricP.y =
    1.0)
123.                or (thirdCoord = 1.0) then
124.                 Writeln('Point on the triangle vertex!')
125.            else if(barycentricP.x = 0.0) or (barycentricP.y =
    0.0) //Перша умова з попередньої превірки застосовується і тут
126.                    or (thirdCoord = 0.0) then
127.                Writeln('Point on the triangle side!')
```

```
128.            else
129.                Writeln('Point inside triangle!');
130.            end
131.        else
132.            Writeln('Point lies outside of the triangle!');
133.    end;
134. end.
```

Додаткове завдання, програма на C++:

```
1. #include <iostream>
2. #include <cmath>
3.
4. struct Vector2D
5. {
6.       float x;
7.       float y;
8.
9.       inline Vector2D(const float _x = 0.0f, const float _y =
   0.0f)
10.           : x(_x), y(_y) {}
11.
12.      inline Vector2D operator - (const Vector2D& v) const
13.      {
14.           return { x - v.x, y - v.y };
15.      }
16.
17.      inline float Length() const
18.      {
19.           return sqrt(x * x + y * y);
20.      }
21.   };
22.
23.   struct Triangle
24.   {
25.      Vector2D Point1;
26.      Vector2D Point2;
27.      Vector2D Point3;
28.
29.      inline Triangle(const Vector2D& p1, const Vector2D& p2,
   const Vector2D& p3)
30.           : Point1(p1), Point2(p2), Point3(p3) {}
31.
32.      inline float GetArea() const
33.      {
34.           float side1 = (Point2 - Point1).Length();
35.           float side2 = (Point3 - Point1).Length();
36.           float side3 = (Point3 - Point2).Length();
37.
```

```
38.           float p = (side1 + side2 + side3) * 0.5f;
39.
40.           return sqrt(p * (p - side1) * (p - side2) * (p -
   side3));
41.      }
42.
43.      inline Vector2D ComputeBarycentricCoordinates(const
   Vector2D& p)
44.      {
45.           Triangle t1(Point3, Point1, p);
46.           Triangle t2(Point1, Point2, p);
47.
48.           float u = t1.GetArea() / GetArea();
49.           float v = t2.GetArea() / GetArea();
50.
51.           return { u, v };
52.      }
53.   };
54.
55.   int main()
56.   {
57.      float x, y;
58.
59.      std::cout << "Triangle 1 point coordinates: ";
60.      std::cin >> x >> y;
61.
62.      Vector2D p1(x, y);
63.
64.
65.      std::cout << "\nTriangle 2 point coordinates: ";
66.      std::cin >> x >> y;
67.
68.      Vector2D p2(x, y);
69.
70.
71.      std::cout << "\nTriangle 3 point coordinates: ";
72.      std::cin >> x >> y;
73.
74.      Vector2D p3(x, y);
75.
76.      while (1)
77.      {
78.           Triangle userTrig(p1, p2, p3);
79.
80.
81.           std::cout << "\nTest point coordinates: ";
82.           std::cin >> x >> y;
83.
```

```
84.           Vector2D testP(x, y);
85.
86.
87.           auto barycentricP =
   userTrig.ComputeBarycentricCoordinates(testP);
88.
89.           if (barycentricP.x >= 0.0f && barycentricP.y >=
   0.0f &&
90.                 (barycentricP.x + barycentricP.y) <= 1.0f)
91.           {
92.                 float thirdCoord = 1.0f - (barycentricP.x +
   barycentricP.y);
93.
94.                 if(barycentricP.x == 1.0f || barycentricP.y
   == 1.0f || thirdCoord == 1.0f)
95.                       std::cout << "\nPoint on the triangle
   vertex!\n";
96.                 else if(barycentricP.x == 0.0f ||
   barycentricP.y == 0.0f || thirdCoord == 0.0f)
97.                       std::cout << "\nPoint on the triangle
   side!\n";
98.                 else
99.                       std::cout << "\nPoint inside
   triangle!\n";
100.          }
101.          else
102.                std::cout << "\nPoint lies outside of the
   triangle!\n";
103.
104.          std::cout << "\n" << barycentricP.x << " " <<
   barycentricP.y << std::endl;
105.    }
106.
107.    return 0;
108. }
```

Код лабораторної: https://github.com/ptrstasiv/Lab2.git

Висновок: при виконанні даної лабораторної роботи ми ознайомилися з аспектами розробки математичних алгоритмів в середовищі Delphi. Реалізувавши програму яка визначає положення точки відносно трикутника, дані яких вказані користувачем, на прямокутній системі координат.