

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний
інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт
з лабораторної роботи № 4 з дисципліни
«Основи програмування 2. Модульне програмування»
«Успадкування та поліморфізм»

Варіант 27

Виконав студент	<u>ІП-15, Пономаренко Маргарита Альбертівна</u> (шифр, прізвище, ім'я, по батькові)
Перевірила	Муха Ірина Павлівна (прізвище, ім'я, по батькові)

Лабораторна робота 4

Успадкування та поліморфізм

Індивідуальне завдання

Варіант 27

27. Створити клас `TTriangle`, який містить координати вершин і методи обчислення його площі та периметру. На основі цього класу створити класи-нащадки, які представляють рівносторонні, прямокутні та рівнобедрені трикутники. Створити певну кількість трикутників кожного виду, щоб їх сумарна кількість дорівнювала n . Для рівносторонніх та прямокутних трикутників обчислити суму їх площ, а для рівнобедрених – суму всіх периметрів.

Код

C++, Isosceles.cpp :

```
Isosceles.cpp  x Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h  TTriangle.cpp  TTriangle.h
ConsoleApplication1  (Глобальная область)
1  #include <sstream>
2  #include "Isosceles.h"
3
4  Isosceles::Isosceles(Coordinates point1, Coordinates point2, int angle):
5  {
6      TTriangle(point1, point2, point1.distanceTo(point2), angle)
7  {
8      this->angle = angle;
9      angle1 = 180 - (2 * angle); //обчислення кута при вершині трикутника
10     side2 = round(point1.distanceTo(point2) * 100) / 100; //обчислення бічної сторони
11     side3 = side2;
12     side1 = side2 * sqrt(2 * (1 - cos((angle1 * 3.14) / 180))); //обчислення сторони основи
13     side1 = round(side1 * 100) / 100;
14 }
15 double Isosceles::CalculateArea()
16 {
17     return 0.0;
18 }
19
20 double Isosceles::CalculatePerimeter() {
21     return 2 * side2 + side1;
22 }
23
```

Основи програмування 2. Модульне програмування

```
24 string Isosceles::info() //виведення даних про трикутник
25 {
26     stringstream output;
27     output << endl << "Дані про рівнобедренний трикутник:" << endl;
28     int i = 1;
29     for (auto point : coordinates)
30     {
31         output << "x" << i << " = " << point.x << "; y" << i << " = " << point.y << endl;
32         i++;
33     }
34     output << "side1 = " << side1 << "; side2 = side3 = " << side2 << endl;
35     output << "angle = " << angle << "; angle1 = " << angle1 << endl << "Периметр трикутника = " << CalculatePerimeter() << endl;
36     return output.str();
37 }
```

Isosceles.h :

```
Isosceles.cpp  Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h
ConsoleApplication1  (Глобальная область)

1  #pragma once
2  #include "TTriangle.h"
3  class Isosceles : //клас-нащадок "Рівнобедренні трикутники"
4  {
5  public:
6  private:
7      double angle1;
8
9  public:
10     Isosceles(Coordinates point1, Coordinates point2, int angle);
11
12     TriangleType getType() override {
13         return TriangleType::ISOSCELES;
14     }
15
16     double CalculatePerimeter() override;
17     double CalculateArea() override;
18     string info() override;
19 };
20
```

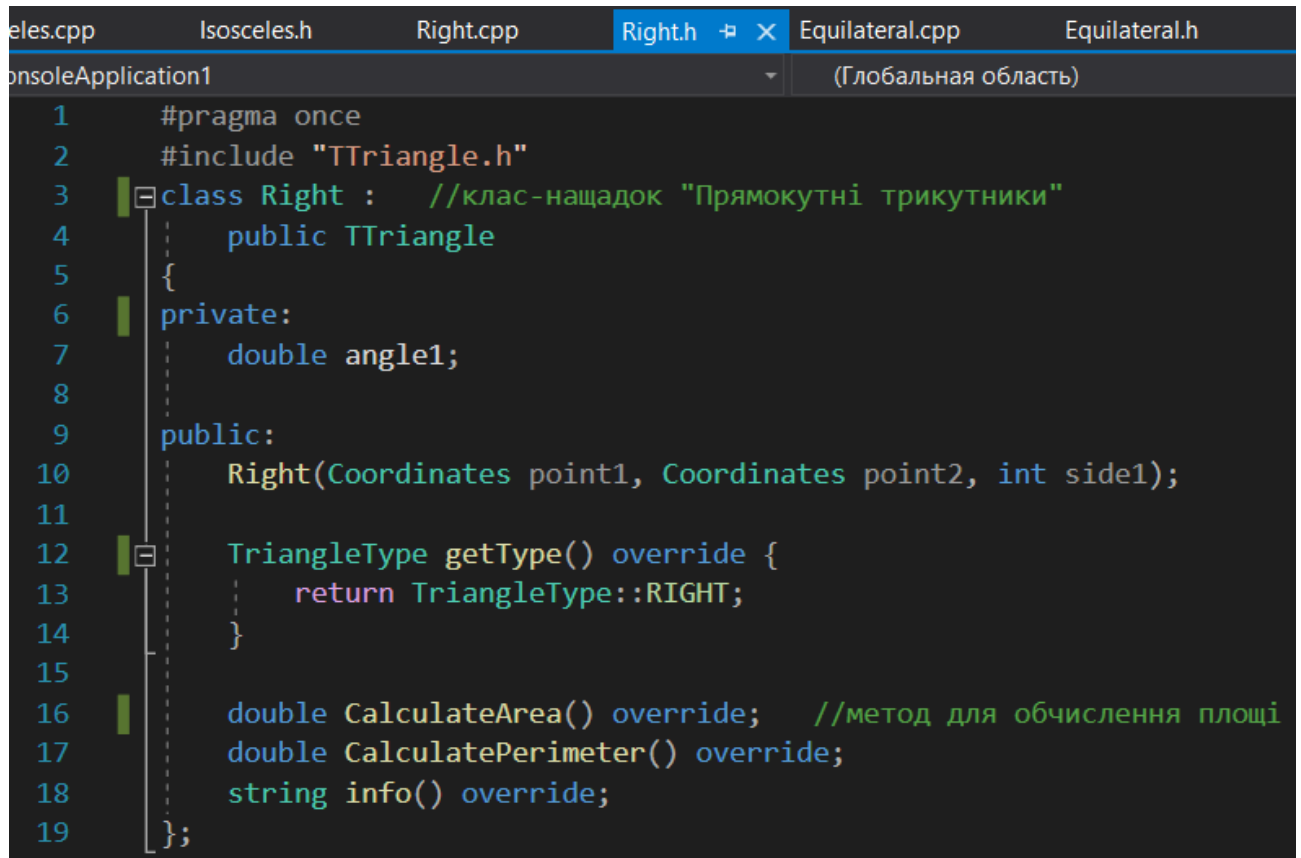
Right.cpp :

Основи програмування 2. Модульне програмування

```
Isosceles.cpp  Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h  TTriangle.cpp  TTriangle.h  OP-Lab5(2 term).cpp
ConsoleApplication1  (Глобальная область)
1  #include <sstream>
2  #include "Right.h"
3
4  Right::Right(Coordinates point1, Coordinates point2, int side1):
5      TTriangle(point1, point2, side1, 90)
6  {
7      this->side1 = side1;
8      angle = 90;
9      side2 = round(point1.distanceTo(point2) * 100) / 100;  //розрахунок другого катета
10
11      angle1 = side1 / side2;
12      angle1 = round(atan(angle1) * 180 / 3.14 * 100) / 100;  //розрахунок кута між гіпотенузою і катетом в основі
13
14  }
15
16  double Right::CalculateArea() {
17      return side1 * side2 / 2;
18  }
19
20  double Right::CalculatePerimeter()
21  {
22      return 0.0;
23  }
24
```

```
25  string Right::info()  //виведення даних про трикутник
26  {
27      stringstream output;
28      output << endl << "Дані про прямокутний трикутник:" << endl;
29      int i = 1;
30      for (auto point : coordinates)
31      {
32          output << "x" << i << " = " << point.x << "; y" << i << " = " << point.y << endl;
33          i++;
34      }
35      output << "side1 = " << side1 << "; side2 = " << side2 << endl;
36      output << "angle = " << angle << "; angle1 = " << angle1 << endl << "Площа трикутника = " << CalculateArea() << endl;
37      return output.str();
38  }
```

Right.h :



```
1  #pragma once
2  #include "TTriangle.h"
3  class Right : //клас-нащадок "Прямокутні трикутники"
4  {
5  public:
6  private:
7      double angle1;
8
9  public:
10     Right(Coordinates point1, Coordinates point2, int side1);
11
12     TriangleType getType() override {
13         return TriangleType::RIGHT;
14     }
15
16     double CalculateArea() override; //метод для обчислення площі
17     double CalculatePerimeter() override;
18     string info() override;
19 };
```

Equilateral.cpp :

```

Isosceles.cpp  Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h
ConsoleApplication1  (Глобальная область)

1  #include <sstream>
2  #include "Equilateral.h"
3
4  Equilateral::Equilateral(Coordinates point1, Coordinates point2):
5      TTriangle(point1, point2, point1.distanceTo(point2), 60)
6  {
7      side3 = round(point1.distanceTo(point2) * 100) / 100;
8      side1 = side2 = side3;
9      angle = 60;
10 }
11
12 double Equilateral::CalculatePerimeter()
13 {
14     return 0.0;
15 }
16
17 double Equilateral::CalculateArea() {
18     return round((pow(side3, 2) * sqrt(3)) / 4 * 100) / 100;
19 }
20

```

```

21 string Equilateral::info() //виведення даних про трикутник
22 {
23     stringstream output;
24     output << endl << "Дані про рівносторонній трикутник:" << endl;
25     int i = 1;
26     for (auto point:coordinates)
27     {
28         output << "x" << i << " = " << point.x << "; y" << i << " = " << point.y << endl;
29         i++;
30     }
31     output << "side1 = side2 = side3 = " << side3 << endl << "angle = " << angle << endl << "Площа трикутника = " << CalculateArea() << endl;
32     return output.str();
33 }

```

Equilateral.h :

Основи програмування 2. Модульне програмування

```
Isosceles.cpp  Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h  ConsoleApplication1
(Глобальная область)

1  #pragma once
2  #include "TTriangle.h"
3  class Equilateral : //клас-нащадок "Рівносторонні трикутники"
4  {
5  public:
6
7      Equilateral(Coordinates point1, Coordinates point2);
8
9
10     TriangleType getType() override {
11         return TriangleType::EQUILATERAL;
12     }
13
14     double CalculateArea() override; //метод для обчислення площі
15     double CalculatePerimeter() override;
16     string info() override;
17 };
```

TTriangle.cpp :

```
Isosceles.cpp  Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h  TTriangle.cpp  TTriangle.h  OP-Lab5(2 term).cpp
(Глобальная область)

1  #include <sstream>
2  #include "TTriangle.h"
3
4  TTriangle::TTriangle(Coordinates point1, Coordinates point2, int side1, int angle)
5  {
6      coordinates.push_back(point1);
7      coordinates.push_back(point2);
8      double x3 = 0.0;
9      double y3 = 0.0;
10     double side3 = point1.distanceTo(point2);
11     double side2 = sqrt(pow(side1, 2) + pow(side3, 2) - 2 * side1 * side3 * cos((angle * 3.14) / 180));
12
13     double k = (pow(side3, 2) + pow(side2, 2) - pow(side1, 2)) / (2 * side3); //обчислення координат третьої вершини
14     double h = sqrt(pow(side2, 2) - pow(k, 2));
15     if (angle <= 90) {
16         x3 = round((point1.x + (k / side3) * (point2.x - point1.x) - (h / side3) * (point2.y - point1.y)) * 100) / 100;
17         y3 = round((point1.y + (k / side3) * (point2.y - point1.y) + (h / side3) * (point2.x - point1.x)) * 100) / 100;
18     }
19     else {
20         x3 = round((point1.x + (k / side3) * (point2.x - point1.x) + (h / side3) * (point2.y - point1.y)) * 100) / 100;
21         y3 = round((point1.y + (k / side3) * (point2.y - point1.y) - (h / side3) * (point2.x - point1.x)) * 100) / 100;
22     }
23     Coordinates point3(x3, y3);
24     coordinates.push_back(point3);
25 }
26
```

TTriangle.h :

Основи програмування 2. Модульне програмування

```
es.cpp  Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h  TTriangle.cpp  TTriangle.h
soleApplication1  (Глобальная область)
1  #pragma once
2  #include <iostream>
3  #include <vector>
4  #include <cmath>
5
6  using namespace std;
7
8  enum class TriangleType
9  {
10     EQUILATERAL,
11     RIGHT,
12     ISOSCELES,
13     TTRIANGLE,
14 };
15
16 struct Coordinates {
17     double x = 0;
18     double y = 0;
19     double distanceTo(Coordinates other) { //знаходження відстані між точками
20         return sqrt(pow(x - other.x, 2) + pow(y - other.y, 2));
21     }
22     Coordinates(double x, double y) : x(x), y(y) {};
23 };
24
```

```
25 class TTriangle
26 {
27 protected:
28     vector<Coordinates> coordinates; //vector для зберігання координат вершин
29     double side1;
30     double side2;
31     double side3;
32     double angle;
33
34 public:
35     TTriangle(Coordinates point1, Coordinates point2, int side1, int angle);
36
37     virtual TriangleType getType() {
38         return TriangleType::TTRIANGLE;
39     }
40
41     virtual double CalculateArea() = 0; //чиста віртуальна функція, що не має визначення та вказує на абстрактність класу TTriangle
42     virtual double CalculatePerimeter() = 0;
43     vector<Coordinates> const& GetCoordinates() { return coordinates; }
44
45     virtual string info() = 0; //віртуальна функція, що виводить проміжні дані
46
47     virtual ~TTriangle() {}
48 };

```

OP-Lab5(2 term).cpp :

Основи програмування 2. Модульне програмування

```
Isosceles.cpp  Isosceles.h  Right.cpp  Right.h  Equilateral.cpp  Equilateral.h  TTriangle.cpp  TTriangle.h  OP-Lab5(2 term).cpp x
ConsoleApplication1  (Глобальная область)
1  //Створити клас TTriangle, який містить координати вершин і методи обчислення його площі та периметру.
2  //На основі цього класу створити класи-нащадки, які представляють рівносторонні, прямокутні та рівнобедрені трикутники.
3  //Створити певну кількість трикутників кожного виду, щоб їх сумарна кількість дорівнювала n.
4  //Для рівносторонніх та прямокутних трикутників обчислити суму їх площ, а для рівнобедрених - суму всіх периметрів.
5
6  #include <iostream>
7  #include <vector>
8  #include <string>
9  #include "Equilateral.h"
10 #include "Right.h"
11 #include "Isosceles.h"
12
13 using namespace std;
14
15 void createEquilateral(vector<TTriangle*> &triangles, int count); //функції для створення трикутників кожного виду
16 void createRight(vector<TTriangle*> &triangles, int count);
17 void createIsosceles(vector<TTriangle*> &triangles, int count);
18 // ...
19
20
21 int input(string message);
22
```

```
23 int main()
24 {
25     setlocale(LC_CTYPE, "ukr");
26     int n;
27     cout << "Input number of triangles: ";
28     cin >> n;
29     vector<TTriangle*> triangles;
30
31     for (size_t i = 0; i < 4; i++)
32     {
33         switch (i) {
34             case 0:
35                 createEquilateral(triangles, n / 3); //створення трикутників певних видів
36                 break;
37             case 1:
38                 createRight(triangles, n / 3);
39                 break;
40             case 2:
41                 createIsosceles(triangles, n / 3 + n % 3);
42                 break;
43             /* ... */
44         }
45     }
46 }
47
48
```

```
49     double areaSyma = 0;
50     double perimeterSyma = 0;
51     for (auto triangle:triangles)
52     {
53         switch (triangle->getType()) {
54             case TriangleType::EQUILATERAL:
55             case TriangleType::RIGHT :
56                 areaSyma += triangle->CalculateArea(); //підрахунок суми площ
57                 cout << triangle->info(); //виведення даних про трикутник
58                 break;
59             case TriangleType::ISOSCELES:
60                 perimeterSyma += triangle->CalculatePerimeter(); //підрахунок суми периметрів
61                 cout << triangle->info();
62                 break;
63             default:
64                 cout << (int)(triangle->getType());
65         }
66     }
67     cout << endl << "Сума площ = " << areaSyma << endl;
68     cout << "Сума периметрів = " << perimeterSyma << endl;
69 }
70
```

```

71 void createEquilateral(vector<TTriangle*> &triangles, int count) {
72     for (size_t i = 0; i < count; i++)
73     {
74         cout << "Створення рівностороннього трикутника..." << endl;
75         int x1 = input("x1 = ");
76         int y1 = input("y1 = ");
77         int x2 = input("x2 = ");
78         int y2 = input("y2 = ");
79         Coordinates c1(x1, y1);
80         Coordinates c2(x2, y2);
81         triangles.push_back(new Equilateral(c1, c2));
82     }
83 }
84
85 void createRight(vector<TTriangle*>& triangles, int count) {
86     for (size_t i = 0; i < count; i++)
87     {
88         cout << "Створення прямокутного трикутника..." << endl;
89         int x1 = input("x1 = ");
90         int y1 = input("y1 = ");
91         int x2 = input("x2 = ");
92         int y2 = input("y2 = ");
93         int side = input("side = ");
94         Coordinates c1(x1, y1);
95         Coordinates c2(x2, y2);
96         triangles.push_back(new Right(c1, c2, side));
97     }
98 }

```

```
99
100 void createIsosceles(vector<TTriangle*>& triangles, int count) {
101     for (size_t i = 0; i < count; i++)
102     {
103         cout << "Створення рівнобедренного трикутника..." << endl;
104         int x1 = input("x1 = ");
105         int y1 = input("y1 = ");
106         int x2 = input("x2 = ");
107         int y2 = input("y2 = ");
108         int angle = input("angle = ");
109         Coordinates c1(x1, y1);
110         Coordinates c2(x2, y2);
111         triangles.push_back(new Isosceles(c1, c2, angle));
112     }
113 }
114
115 // ...
130
131 int input(string message) //функція для отримання даних від користувача
132 {
133     cout << message<<": ";
134     string data;
135     cin >> data;
136     return stoi(data);
137 }
```

Результат виконання

Основи програмування 2. Модульне програмування

```
Выбрать Консоль отладки Microsoft Visual Studio
Input number of triangles: 3
Створення рівностороннього трикутника...
x1 = : 0
y1 = : -4
x2 = : 2
y2 = : 3
Створення прямокутного трикутника...
x1 = : 1
y1 = : 4
x2 = : -2
y2 = : -6
side = : 7
Створення рівнобедренного трикутника...
x1 = : 1
y1 = : 3
x2 = : -2
y2 = : 4
angle = : 30

Дані про рівносторонній трикутник:
x1 = 0; y1 = -4
x2 = 2; y2 = 3
x3 = -4.79; y3 = 1.3
side1 = side2 = side3 = 7.28
angle = 60
Площа трикутника = 22.95

Дані про прямокутний трикутник:
x1 = 1; y1 = 4
x2 = -2; y2 = -6
x3 = 4.71; y3 = -8.01
side1 = 7; side2 = 10.44
angle = 90; angle1 = 33.86
Площа трикутника = 36.54
```

```
Дані про рівнобедренний трикутник:
x1 = 1; y1 = 3
x2 = -2; y2 = 4
x3 = -0.01; y3 = 1.76
side1 = 5.47; side2 = side3 = 3.16
angle = 30; angle1 = 120
Периметр трикутника = 11.79

Сума площ = 59.49
Сума периметрів = 11.79
```

Python, Isosceles.py:

Основи програмування 2. Модульне програмування

```
Isosceles.py × Right.py × Equilateral.py × TTriangle.py × Main.py × Triangles.py ×
1 from TTriangle import *
2
3
4 class Isosceles(TTriangle):
5     angle1 = 0.0
6
7     def __init__(self, point1, point2, angle):
8         super().__init__(point1, point2, point1.distance_to(point2), angle)
9         self.angle = angle
10        self.angle1 = 180 - (2 * angle)
11        self.side2 = round(point1.distance_to(point2), 2)
12        self.side3 = self.side2
13        self.side1 = 2 * self.side2 * math.cos((angle * 3.14) / 180)
14        self.side1 = round(self.side1, 2)
15
16    def calculate_perimeter(self):
17        return 2 * self.side2 + self.side1
18
19    def info(self):
20        output = ''
21        output += 'Дані про рівнобедренний трикутник:\n'
22        i = 1
23        for point in self.coordinates_list:
24            output += 'x' + str(i) + ' = ' + str(point.x) + '; y' + str(i) + ' = ' + str(point.y) + '\n'
25            i += 1
26        output += 'side1 = ' + str(self.side1) + '; side2 = side3 = ' + str(self.side2) + '\n'
27        output += 'angle = ' + str(self.angle) + '; angle1 = ' + str(self.angle1) + '\nПериметр трикутника = '
28        output += str(Isosceles.calculate_perimeter(self)) + '\n'
29        return output
30
```

Right.py:

Основи програмування 2. Модульне програмування

```
Isosceles.py x Right.py x Equilateral.py x TTriangle.py x Main.py x Triangles.py x
1 from TTriangle import *
2
3
4 class Right(TTriangle):
5     angle1 = 0.0
6
7     def __init__(self, point1, point2, side1):
8         super().__init__(point1, point2, side1, 90)
9         self.side1 = side1
10        self.angle = 90
11
12        self.side2 = round(point1.distance_to(point2), 2)
13        self.angle1 = round((math.atan(side1 / self.side2) * 180 / 3.14), 2)
14
15    def calculate_area(self):
16        return round((self.side1 * self.side2 / 2), 2)
17
18    def info(self):
19        output = ''
20        output += 'Дані про прямокутний трикутник:\n'
21        i = 1
22        for point in self.coordinates_list:
23            output += 'x' + str(i) + ' = ' + str(point.x) + '; y' + str(i) + ' = ' + str(point.y) + '\n'
24            i += 1
25        output += 'side1 = ' + str(self.side1) + '; side2 = ' + str(self.side2) + '\n'
26        output += 'angle = ' + str(self.angle) + '; angle1 = ' + str(self.angle1) + '\nПлоща трикутника = '
27        output += str(Right.calculate_area(self)) + '\n'
28        return output
```

Equilateral.py:

```
Isosceles.py x Right.py x Equilateral.py x TTriangle.py x Main.py x Triangles.py x
1 from TTriangle import *
2
3
4 class Equilateral(TTriangle):
5
6     def __init__(self, point1, point2):
7         super().__init__(point1, point2, point1.distance_to(point2), 60)
8         self.side3 = round(point1.distance_to(point2), 2)
9         self.side1 = self.side2 = self.side3
10        self.angle = 60
11
12    def calculate_area(self):
13        return round((self.side3 ** 2 * math.sqrt(3) / 4), 2)
14
15    def info(self):
16        output = ''
17        output += '\nДані про рівносторонній трикутник:\n'
18        i = 1
19        for point in self.coordinates_list:
20            output += 'x' + str(i) + ' = ' + str(point.x) + '; y' + str(i) + ' = ' + str(point.y) + '\n'
21            i += 1
22        output += 'side1 = side2 = side3 = ' + str(self.side3) + '\nangle = ' + str(self.angle) + '\nПлоща трикутника = '
23        output += str(Equilateral.calculate_area(self)) + '\n'
24        return output
25
```

TTriangle.py:

```
Isosceles.py × Right.py × Equilateral.py × TTriangle.py × Main.py × Triangles.py ×
1      import math
2
3
4      def equilateral(triangle):
5          area = triangle.calculate_area()
6          print(triangle.info())
7          return area
8
9
10     def right(triangle):
11         area = triangle.calculate_area()
12         print(triangle.info())
13         return area
14
15
16     def isosceles(triangle):
17         perimeter = triangle.calculate_perimeter()
18         print(triangle.info())
19         return perimeter
20
21
22     # словник для визначення виду трикутника
23     TriangleType = {
24         'Equilateral': equilateral,
25         'Right': right,
26         'Isosceles': isosceles
27     }
28
29
```


Основи програмування 2. Модульне програмування

```
Isosceles.py x Right.py x Equilateral.py x TTriangle.py x Main.py x Triangles.py x
28
29
30 class Coordinates:
31
32     def __init__(self, x, y):
33         self.x = x
34         self.y = y
35
36     def distance_to(self, point):
37         return math.sqrt((point.x - self.x) ** 2 + (point.y - self.y) ** 2)
38
39
40 class TTriangle:
41
42     def __init__(self, point1, point2, side1, angle):
43         self.angle = angle
44         self.side1 = side1
45         self.coordinates_list = []
46         self.coordinates_list.append(point1)
47         self.coordinates_list.append(point2)
48         self.side3 = point1.distance_to(point2)
49         self.side2 = math.sqrt(side1 ** 2 + self.side3 ** 2 - 2 * side1 * self.side3 * math.cos((angle * 3.14) / 180))
50
```

```
51 # обчислення координат третьої вершини
52 k = (self.side3 ** 2 + self.side2 ** 2 - side1 ** 2) / (2 * self.side3)
53 h = math.sqrt(self.side2 ** 2 - k ** 2)
54 if angle <= 90:
55     x3 = round((point1.x + (k / self.side3) * (point2.x - point1.x) - (h / self.side3) * (point2.y - point1.y)), 2)
56     y3 = round((point1.y + (k / self.side3) * (point2.y - point1.y) + (h / self.side3) * (point2.x - point1.x)), 2)
57 else:
58     x3 = round((point1.x + (k / self.side3) * (point2.x - point1.x) + (h / self.side3) * (point2.y - point1.y)), 2)
59     y3 = round((point1.y + (k / self.side3) * (point2.y - point1.y) - (h / self.side3) * (point2.x - point1.x)), 2)
60 point3 = Coordinates(x3, y3)
61 self.coordinates_list.append(point3)
62
63 def calculate_area(self):
64     return 0
65
66 def calculate_perimeter(self):
67     return 0
68
69 def info(self):
70     return 0
```

Main.py:

Основи програмування 2. Модульне програмування

```
Isosceles.py x Right.py x Equilateral.py x TTriangle.py x Main.py x
1  from Triangles import *
2
3
4  def create_equilateral(triangle_vector, count):
5      j = 0
6      while j < count:
7          print('Створення рівностороннього трикутника...')
8          x1 = int(input('x1 = '))
9          y1 = int(input('y1 = '))
10         x2 = int(input('x2 = '))
11         y2 = int(input('y2 = '))
12         c1 = Coordinates(x1, y1)
13         c2 = Coordinates(x2, y2)
14         new_object = Equilateral(c1, c2)
15         triangle_vector.append(new_object)
16         j += 1
17
18
```

```
19 def create_right(triangle_vector, count):
20     j = 0
21     while j < count:
22         print('Створення прямокутного трикутника...')
23         x1 = int(input('x1 = '))
24         y1 = int(input('y1 = '))
25         x2 = int(input('x2 = '))
26         y2 = int(input('y2 = '))
27         side = int(input('side = '))
28         c1 = Coordinates(x1, y1)
29         c2 = Coordinates(x2, y2)
30         new_object = Right(c1, c2, side)
31         triangle_vector.append(new_object)
32         j += 1
33
34
35 def create_isosceles(triangle_vector, count):
36     j = 0
37     while j < count:
38         print('Створення рівнобедренного трикутника...')
39         x1 = int(input('x1 = '))
40         y1 = int(input('y1 = '))
41         x2 = int(input('x2 = '))
42         y2 = int(input('y2 = '))
43         angle = int(input('angle = '))
44         c1 = Coordinates(x1, y1)
45         c2 = Coordinates(x2, y2)
46         new_object = Isosceles(c1, c2, angle)
47         triangle_vector.append(new_object)
48         j += 1
```

```
49
50
51     n = int(input('Введіть кількість трикутників: '))
52     triangles = []
53
54     for i in range(0, 3):
55         match i:
56             case 0:
57                 create_equilateral(triangles, n // 3)
58             case 1:
59                 create_right(triangles, n // 3)
60             case 2:
61                 create_isosceles(triangles, n // 3 + n % 3)
62
63     area_soma = 0.0
64     perimeter_soma = 0.0
65
66     # підрахунок суми площ та периметрів
67     for triangl in triangles:
68         result = TriangleType[type(triangl).__name__](triangl)
69         match type(triangl).__name__:
70             case 'Equilateral':
71                 area_soma += result
72             case 'Right':
73                 area_soma += result
74             case 'Isosceles':
75                 perimeter_soma += result
76
```

```
76
77     area_soma = round(area_soma, 2)
78     perimeter_soma = round(perimeter_soma, 2)
79     print('Сума площ =', area_soma)
80     print('Сума периметрів =', perimeter_soma)
81
```

Triangles.py:

Основи програмування 2. Модульне програмування

```
Isosceles.py × Right.py × Equilateral.py × TTriangle.py × Main.py × Triangles.py ×
1 from TTriangle import *
2 from Equilateral import *
3 from Right import *
4 from Isosceles import *
5 # для підключення всіх потрібних файлів
6
```

Результат виконання

Main	
Введіть кількість трикутників: 4	Дані про рівносторонній трикутник:
Створення рівностороннього трикутника...	x1 = 0; y1 = -4
x1 = 0	x2 = 2; y2 = 3
y1 = -4	x3 = -5.06; y3 = 1.23
x2 = 2	side1 = side2 = side3 = 7.28
y2 = 3	angle = 60
Створення прямокутного трикутника...	Площа трикутника = 22.95
x1 = 1	Дані про прямокутний трикутник:
y1 = 4	x1 = 1; y1 = 4
x2 = -2	x2 = -2; y2 = -6
y2 = -6	x3 = 4.71; y3 = -8.01
side = 7	side1 = 7; side2 = 10.44
Створення рівнобедренного трикутника...	angle = 90; angle1 = 33.86
x1 = 1	Площа трикутника = 36.54
y1 = 3	Дані про рівнобедренний трикутник:
x2 = -2	x1 = 1; y1 = 3
y2 = 4	x2 = -2; y2 = 4
angle = 30	x3 = 0.1; y3 = 1.63
Створення рівнобедренного трикутника...	side1 = 5.47; side2 = side3 = 3.16
x1 = 2	angle = 30; angle1 = 120
y1 = -2	Периметр трикутника = 11.79
x2 = 5	
y2 = 1	
angle = 20	

```
Дані про рівнобедренний трикутник:  
x1 = 2; y1 = -2  
x2 = 5; y2 = 1  
x3 = 1.16; y3 = -0.79  
side1 = 7.97; side2 = side3 = 4.24  
angle = 20; angle1 = 140  
Периметр трикутника = 16.45  
  
Сума площ = 59.49  
Сума периметрів = 28.24  
  
Process finished with exit code 0
```