**Лабораторная работа №5**

**Классы, объекты, наследование в С#**

**Цель работы**

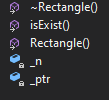
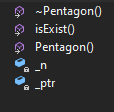
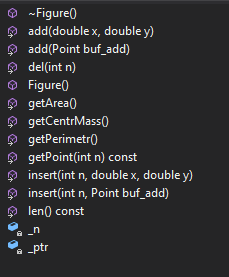
Ознакомление с основами программирования на С#.

**Формулировка задачи**

В качестве задания выбирается один из вариантов 2-й работы. Выбран вариант №5.

Создать класс Figure, объект которого задается координатами точек, c вычислением площади, центра тяжести и периметра. На его основе реализовать классы Rectangle (прямоугольник) и Pentagon (пятиугольник).

**Диаграммы классов**



**Перечень сигнатур методов**

Метод add (): добавление вершины.

Метод getPoint (): получение координат точек и линий.

Метод getArea (): получение площади.

Метод getPerimetr (): получение периметра.

Метод getCentrMass (): получение центра масс.

Метод len (): количества вершин.

Метод del (): удаление точки.

Метод isExist (): проверка на соответствию типу.

**Листинг lab5.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_oop

{

class lab5

{

static void Main(string[] args)

{

Consol View = new Consol();

View.init();

}

}

}

**Листинг Consol.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab5\_oop

{

class Consol

{

public void init()

{

int search=-1;

while (search != 0) {

Console.WriteLine("search: 1 - Figure 2 - Pentagon 3 - Rectangle\n");

string buf = Console.ReadLine();

search = int.Parse(buf);

switch (search)

{

case 1: viewFigure(); break;

case 2: viewPentagon(); break;

case 3: viewRectangle(); break;

default: break;

}

}

}

public int viewPentagon()

{

Pentagon Fig=new Pentagon();

double x, y;

int n = 5;

string s;

for (int i = 0; i < n; i++)

{

Console.WriteLine( "enter vertex to enter:");

x = double.Parse(Console.ReadLine());

y = double.Parse(Console.ReadLine());

Fig.add(x, y);

}

s = "Area=";

s += Fig.getArea().ToString();

s += ("\nPerimetr=");

s += Fig.getPerimetr().ToString();

s += (" \nCentrMass=");

Console.Write(s);

pprint(Fig.getCentrMass());

Console.WriteLine("\nPrint figure\n");

for (int i = 0; i < Fig.len(); i++)

pprint(Fig.getPoint(i));

Console.WriteLine( "\nenter vertex for delete\n");

n = int.Parse(Console.ReadLine()); ;

Fig.del(n);

for (int i = 0; i < Fig.len(); i++)

pprint(Fig.getPoint(i));

return 0;

}

public int viewRectangle()

{

Rectangle Fig = new Rectangle();

double x, y;

int n = 4;

string s;

for (int i = 0; i < n; i++)

{

Console.WriteLine( "enter vertex to enter:");

x = double.Parse(Console.ReadLine());

y = double.Parse(Console.ReadLine());

Fig.add(x, y);

}

s = "Area=";

s += Fig.getArea().ToString();

s += ("\nPerimetr=");

s += Fig.getPerimetr().ToString();

s += (" \nCentrMass=");

Console.Write(s);

pprint(Fig.getCentrMass());

Console.WriteLine( "\nPrint figure\n");

for (int i = 0; i < Fig.len(); i++)

pprint(Fig.getPoint(i));

Console.WriteLine("\nenter vertex for delete\n");

n = int.Parse(Console.ReadLine());

Fig.del(n);

for (int i = 0; i < Fig.len(); i++)

pprint(Fig.getPoint(i));

return 0;

}

public int viewFigure()

{

Figure Fig = new Figure();

double x, y;

int n;

string s;

Console.WriteLine("enter the number of vertexes:");

n = int.Parse(Console.ReadLine());

if (n > 0)

{

for (int i = 0; i < n; i++)

{

Console.WriteLine( "enter vertex to enter:");

x = double.Parse(Console.ReadLine());

y = double.Parse(Console.ReadLine());

Fig.add(x, y);

}

s = "Area=";

s += Fig.getArea().ToString();

s += ("\nPerimetr=");

s += Fig.getPerimetr().ToString();

s += (" \nCentrMass=");

Console.Write(s);

pprint(Fig.getCentrMass());

Console.WriteLine("\nPrint figure\n");

for (int i = 0; i < Fig.len(); i++)

pprint(Fig.getPoint(i));

Console.WriteLine("\nenter vertex for delete\n");

n = int.Parse(Console.ReadLine());

Fig.del(n);

for (int i = 0; i < Fig.len(); i++)

pprint(Fig.getPoint(i));

}

else

Console.WriteLine("there is no such figure");

return 0;

}

public void pprint(Point buf)

{

string s = "";

s += buf.X;

s += " ";

s += buf.Y;

s += "\n";

Console.Write(s);

}

}

}

**Листинг Figure.cs**

using System;

//System.Array;

//System.Array;

/// <summary>

/// Summary description for Class1

/// </summary>

namespace lab5\_oop

{

public class Figure

{

protected int \_n = 0;

protected Point[] \_ptr;

public Figure() { \_n = 0; \_ptr = new Point[\_n]; }

//public ~Figure() {}

public void add(double x, double y)

{

Point buf;

buf.X = x;

buf.Y = y;

\_n++;

Array.Resize<Point>(ref \_ptr, \_n);

\_ptr[\_n-1] = buf;

}

public void add(Point buf\_add)

{

\_n++;

Array.Resize<Point>(ref \_ptr, \_n);

\_ptr[\_n-1] = buf\_add;

}

/\*

void insert(int n, double x, double y);

void insert(int n, Point buf\_add);

\*/

public double getPerimetr()

{

double sum = 0;

if (\_n > 0)

{

if (\_n > 0)

{

for (int i = 0; i < \_n - 1; i++)

{

sum += Math.Sqrt((\_ptr[i + 1].X - \_ptr[i].X) \* (\_ptr[i + 1].X - \_ptr[i].X) + (\_ptr[i + 1].Y - \_ptr[i].Y) \* (\_ptr[i + 1].Y - \_ptr[i].Y));

}

sum += Math.Sqrt((\_ptr[0].X - \_ptr[\_n - 1].X) \* (\_ptr[0].X - \_ptr[\_n - 1].X) + (\_ptr[0].Y - \_ptr[\_n - 1].Y) \* (\_ptr[0].Y - \_ptr[\_n - 1].Y));

}

}

return sum;

}

public double getArea()

{

double sum1 = 0, sum2 = 0;

if (\_n > 0)

{

sum1 += \_ptr[\_n - 1].X \* \_ptr[0].Y;

sum2 += \_ptr[\_n - 1].Y \* \_ptr[0].X;

for (int i = 0; i < \_n - 1; i++)

{

sum1 += \_ptr[i].X \* \_ptr[i + 1].Y;

sum2 += \_ptr[i].Y \* \_ptr[i + 1].X;

}

sum1 -= sum2;

if (sum1 < 0)

sum1 \*= -1;

}

return sum1 / 2;

}

public bool del(int n)

{

if (n < 0)

n += \_n;

if (n >= 0 && n < \_n)

{

int i = n;

for (i = n; i < \_n - 1; i++)

{

\_ptr[i] = \_ptr[i + 1];

}

\_n--;

Array.Resize<Point>(ref \_ptr, \_n);

return true;

}

return false;

}

public Point getCentrMass()

{

double sum1 = 0, sum2 = 0;

Point f;

for (int i = 0; i < \_n; i++)

{

sum1 += \_ptr[i].X;

sum2 += \_ptr[i].Y;

}

sum1 /= \_n;

sum2 /= \_n;

f.X = sum1;

f.Y = sum2;

return f;

}

public Point getPoint(int n)

{

if (n < 0)

{

n = \_n + n;

}

if (n >= \_n || n < 0)

{

while (n < 0)

n += \_n;

while (n >= \_n)

n -= \_n;

}

return \_ptr[n];

}

public int len()

{

return \_n;

}

}

}

**Листинг Pentagon.cs**

using System;

namespace lab5\_oop

{

class Pentagon : Figure

{

public Pentagon()

{

\_n = 0;

\_ptr = new Point[\_n];

}

public bool isExist()

{

if (\_n == 5)

{

for (int i = 0; i < \_n; i++)

for (int j = i + 1; j < \_n; j++)

if (\_ptr[j].X == \_ptr[i].X && \_ptr[j].Y == \_ptr[i].Y)

return false;

return true;

}

return false;

}

}

}

**Листинг Rectangle.cs**

using System;

namespace lab5\_oop

{

class Rectangle : Figure

{

public Rectangle()

{

\_n = 0;

\_ptr = new Point[\_n];

}

public bool isExist()

{

if (\_n == 4)

{

if (\_ptr[0].X == \_ptr[1].X && \_ptr[0].Y == \_ptr[3].Y && \_ptr[2].Y == \_ptr[1].Y && \_ptr[2].X == \_ptr[3].X)

return true;

}

return false;

}

}

}

**Листинг Point.cs**

using System;

namespace lab5\_oop

{

public struct Point

{

public double X;

public double Y;

}

}

**Заключение**

В процессе выполнения лабораторной работы №5 мы ознакомились с отношением наследования и иерархией классов на языке С#.

Авторы: Преподаватель: Тарасов С.А.

Шутенко О.Р. Оценка\_\_\_\_

Подпись \_\_\_\_\_\_\_ Дата: \_\_.\_\_.2020