**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «Объектно-ориентированное программирование»

на тему: **«**Основы синтаксиса объектно-ориентированного языка программирования**»**

Выполнил: студент гр. ИТИ-21

Мусафиров А.Ю.

Принял: преподаватель

Башаримов Ю. С.

Гомель 2023

**Цель работы**: научится разрабатывать классы для решения задач; уметь переносить и сохранять классы внутри библиотек классов, подключать персональную библиотеку, создавать различные методы для решения задач.

**Задание:**

1. Необходимо разработать класс, представляющий собой плоскую фигуру, согласно варианту задания (Вариант 2: Окружность).
2. При именовании класса, полей и методов руководствоваться соглашением о наименовании кода.
3. Фигуру следует задавать координатами вершин.
4. Реализовать метод проверки возможности существования данной фигуры.
5. Реализовать методы вычисление длин сторон, площади и периметра.
6. Реализовать методы, проверяющие принадлежность точки, заданной своими координатами на плоскости, фигуре, её границе.
7. Класс должен быть размещён в библиотеке классов.
8. Весь код должен быть снабжён элементами документирования.
9. Сгенерировать *XML*-файл с документацией по проекту.
10. Создать консольное приложение с интерфейсом пользователя.
11. Подключить созданную библиотеку в консольное приложение.
12. Верифицировать разработанное приложение.
13. Сгенерировать исполняемый файл для последующего переноса на другие компьютеры.

**Ход работы**

На рисунке 1 изображен результат работы программы при условии, что радиус окружности равен или меньше 0.

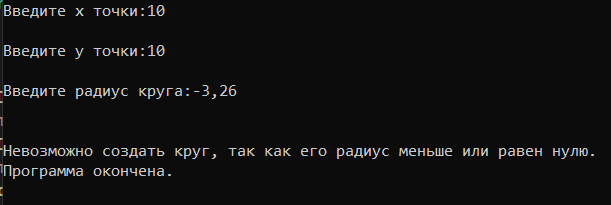


Рисунок 1 – Окно консоли при радиусе равном 0.

На рисунке 2 уже изображён результат работы программы, когда радиус положительный, однако точка, которая проверяется на нахождение внутри или на границе фигуры, не входит в окружность.

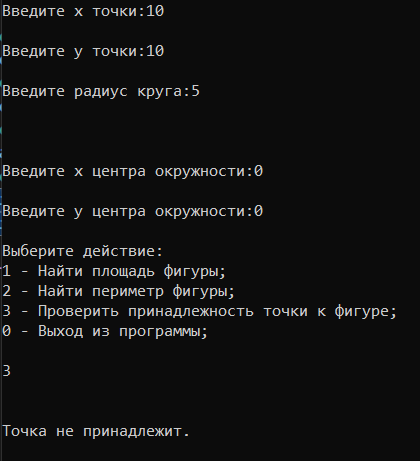


Рисунок 2 – Окно консоли при положительном радиусе и точке,  
которая не принадлежит фигуре.

На рисунке 3 изображён результат при точке, лежащей внутри или на границе окружности.

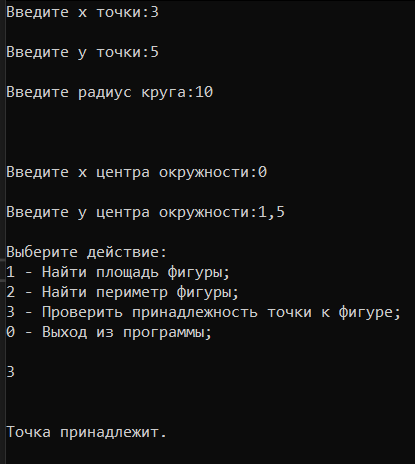


Рисунок 3 – Окно консоли при положительном радиусе и точке,  
которая принадлежит окружности.

На рисунке 4 изображён результат работы программы при выборе вычисления площади и периметра фигуры.

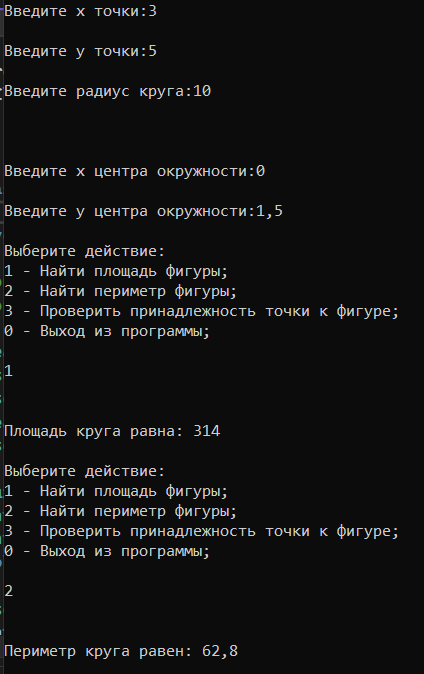


Рисунок 4 – результат работы программы при выборе вычисления площади и периметра фигуры.

На рисунке 5 изображён результат работы метода по проверке корректности вводимых данных с клавиатуры.

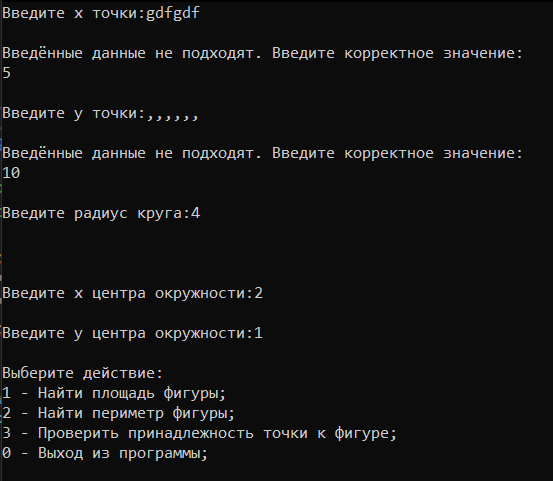


Рисунок 5 – Окно консоли при положительном радиусе и точке,  
которая принадлежит окружности.

В результате изменений начальных данных получили значения, представленные в таблице 1.

Таблица 1 – Полученные значения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Координаты  центра окружности | Радиус | Координаты  точки | Площадь  окружности | Длина  окружности | Принадлежность  точки |
| (0;0) | -3,26 | (0;0) | - | - | - |
| (0;0) | 5 | (10;10) | 78,5 | 31,4 | нет |
| (4;4) | 6,3 | (-4;-3) | 124,6266 | 39,564 | нет |
| (0;0) | 1 | (0;0) | 3,14 | 6,28 | да |
| (11;11) | 2 | (10;11) | 12.56 | 12.56 | да |
| (0.1;2.5) | 4.5 | (0.3;2.2) | 63.585 | 28.26 | да |

В приложении А, Б и В представлен код программы и код файлов библиотеки.

Для нахождения площади окружности использовалась стандартная формула, которая представлена ниже(1.1):

|  |  |
| --- | --- |
| *S=πr2* | (1.1) |

где *S* – площадь окружности;

*r* – радиус окружности.

Дальше чтобы найти длину окружности использовалась вот такая формула(1.2)

|  |  |
| --- | --- |
| *L=2πr* | (1.2) |

где *L* – длина окружности;

*r* – радиус окружности.

Так как были известны координаты точки и центра окружности, можно было найти расстояние от центра окружности до точки по теореме Пифагора и сравнить это расстояние с радиусом. Если расстояние больше, чем радиус, точка находится вне окружности. Если расстояние меньше или равно, то точка принадлежит.

Во время выполнения данной работы были сделаны следующие вещи:

– созданы классы для описания фигуры, в частности окружности.;

– создана и подключена к основной программе библиотека;

– разработаны методы для выполнения поставленных задач;

– реализовано консольное меню пользователя;

**Вывод:** в результате выполнения лабораторной работы были изучены основы синтаксиса объектно-ориентированного языка программирования, созданы классы для реализации фигуры, а также получены навыки создания и использования пользовательской библиотеки.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

***Program.cs*:**

using System;

using System.Threading;

using LibraryOfClass;

namespace Lab\_1

{

internal class Program

{

static void Main(string[] args)

{

Thread.Sleep(1000);

Console.WriteLine("Приветствую!");

Console.WriteLine("\nЛабораторная работа №1");

Thread.Sleep(2000);

Console.Clear();

Figure = new Figure(); // Создание переменной типа Figure.

Point = new Point(); // Создание переменной типа Point.

Point center = new Point();

double radius;

Console.Write("Введите x точки:");

point.x = Program.ReadDouble();

Console.Write("\nВведите y точки:");

point.y = Program.ReadDouble();

Console.Write("\nВведите радиус круга:");

radius = Program.ReadDouble();

Console.WriteLine("\n");

figure = new Figure(radius, center);

if (figure.IsExist()) // Проверка на существование фигуры.

{

Console.Write("\nВведите x центра окружности:");

center.x = Program.ReadDouble();

Console.Write("\nВведите y центра окружности:");

center.y = Program.ReadDouble();

bool isRun = figure.IsExist();

while (isRun) // Консольное меню.

{

Console.WriteLine("\nВыберите действие:");

Console.WriteLine("1 - Найти площадь фигуры;");

Console.WriteLine("2 - Найти периметр фигуры;");

Console.WriteLine("3 - Проверить принадлежность точки к фигуре;");

Console.WriteLine("0 - Выход из программы;\n");

switch (Program.ReadDouble())

{

case 1:

Console.WriteLine("\n");

figure.Square(); // Метод вычисления площади фигуры.

break;

case 2:

Console.WriteLine("\n");

figure.Perimeter(); // Методы вычисления периметра фигуры.

break;

case 3:

Console.WriteLine("\n");

figure.IsContain(point, center); // Метод определения нахождения точки внутри и на границе фигуры.

break;

case 0:

isRun = false; // Выход из программы.

break;

default:

Console.WriteLine("Некорректный выбор функции!");

break;

}

}

}

else Console.WriteLine("Невозможно создать круг, так как его радиус меньше или равен нулю.");

Console.WriteLine("Программа окончена.");

Console.ReadKey();

}

/// <summary>

/// Метод для проверки вводимых данных на корректность.

/// </summary>

/// <returns></returns>

public static double ReadDouble()

{

string numeral = Console.ReadLine();

double value;

while (!Double.TryParse(numeral, out value))

{

Console.WriteLine("\nВведённые данные не подходят. Введите корректное значение: ");

numeral = Console.ReadLine();

}

return value;

}

}

}

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст первого файла библиотеки**

***Figure.cs*:**

using System;

namespace LibraryOfClass

{

/// <summary>

/// Класс, служащий для создания объекта типа окружность.

/// </summary>

public class Figure

{

/// <summary>

/// Радиус круга.

/// </summary>

public double radius;

/// <summary>

/// Переменная для хранения координат центра окружности.

/// </summary>

public Point center = new Point();

/// <summary>

/// Конструктор для создания пустой окружности.

/// </summary>

public Figure()

{

this.radius = 0;

this.center.x = 0;

this.center.y = 0;

}

/// <summary>

/// Конструктор для создания окружности с известными радиусом и координатами.

/// </summary>

public Figure(double radius, Point center)

{

this.radius = radius;

this.center = center;

}

/// <summary>

/// Метод для нахождения площади круга.

/// </summary>

public void Square()

{

double square = 0;

square = 3.14 \* radius \* radius;

Console.WriteLine($"Площадь круга равна: {square}");

}

/// <summary>

/// Метод для нахождения длины окружности круга.

/// </summary>

public void Perimeter()

{

double perimeter = 2 \* 3.14 \* radius;

Console.WriteLine($"Периметр круга равен: {perimeter}");

}

/// <summary>

/// Метод для проверки существования круга.

/// </summary>

public bool IsExist()

{

bool isExist = false;

if (radius > 0)

{

isExist = true;

}

return isExist;

}

/// <summary>

/// Метод для проверки принадлежности точки к оркужности.

/// </summary>

public void IsContain(Point A, Point B)

{

if (radius >= Math.Sqrt((A.x - B.x) \* (A.x - B.x) + (A.y - B.y) \* (A.y - B.y)))

Console.WriteLine("Точка принадлежит.\n");

else Console.WriteLine("Точка не принадлежит.\n");

}

}

}

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Текст второго файла библиотеки**

***Point.cs*:**

namespace LibraryOfClass

{

/// <summary>

/// Класс, служащий для отображения точки. Внутри класса используются две переменные для хранения координат точки.

/// </summary>

public class Point

{

/// <summary>

/// X координата точки.

/// </summary>

public double x;

/// <summary>

/// Y координата точки.

/// </summary>

public double y;

}

}