1. **События**

Задание 1. Создать 3 метода, подсчитывающих периметр, площадь квадрата, и выводящий сторону треугольника. Создать делегат. В программе вызывать все три метода при помощи делегата.

Листинг программы:

namespace Task\_1

{

delegate void EventDelegate(double side);

class Program

{

public static void Main(string[] args)

{

try

{

EventDelegate del = new EventDelegate(GetPerimetr);

del += GetAreaSquare;

del += GetSideTriangle;

Console.Write("Введите значение: ");

del(double.Parse(Console.ReadLine()));

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

static void GetPerimetr(double sideA)

{

Console.WriteLine($"Периметр квадрата: {4 \* sideA}");

}

static void GetAreaSquare(double sizeA)

{

Console.WriteLine($"Площадь квадрата: {Math.Pow(sizeA, 2)}");

}

static void GetSideTriangle(double side)

{

Console.WriteLine($"Сторона треугольника: {side}");

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5,8 | 23,2 33,64 5,8 |

Анализ результатов:

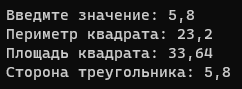


Рисунок 1.1 – Результат работы программы

Источник: собственная разработка

Задание 2. На основе своей программы создать метод, в котором делегат будет параметром.

Листинг программы:

namespace Task\_2

{

class Program

{

delegate void EventDelegate(double side);

public static void Main(string[] args)

{

try

{

EventDelegate del = new EventDelegate(GetPerimetr);

del += GetAreaSquare;

del += GetSideTriangle;

PrintDelegate(del);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

static void PrintDelegate(EventDelegate delegat)

{

try

{

Console.Write("Введите значение: ");

delegat(double.Parse(Console.ReadLine()));

}

catch (FormatException ex)

{

Console.WriteLine(ex.Message);

}

}

static void GetPerimetr(double sideA)

{

Console.WriteLine($"Периметр квадрата: {4 \* sideA}");

}

static void GetAreaSquare(double sizeA)

{

Console.WriteLine($"Площадь квадрата: {Math.Pow(sizeA, 2)}");

}

static void GetSideTriangle(double side)

{

Console.WriteLine($"Сторона треугольника: {side}");

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 6 | 24 36 6 |

Анализ результатов:

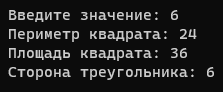


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка

Задание 3. Для класса MyInfo содержащего поле name (Ваше имя) создать событие выводящее оповещение в случае изменения значения поля name.

Листинг программы:

namespace Task\_3

{

class Program

{

public static void Main(string[] args)

{

try

{

Console.Write("Введите ФИО: ");

string name = Console.ReadLine();

if (name.Length < 10) throw new FormatException();

MyInfo mi = new MyInfo(name);

mi.Notify += (string notify) => { Console.WriteLine(notify); };

Console.Write("Выберите 1-сменить ФИО, 2-не менять ФИО: ");

switch (int.Parse(Console.ReadLine()))

{

case 1:

Console.Write("Введите ФИО: ");

mi.Name = Console.ReadLine();

break;

case 2:

mi.Name = name;

break;

default:

Console.WriteLine("Такого пункта нет");

break;

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

public class MyInfo

{

public event Action<string> Notify;

private string \_name;

public MyInfo(string name)

{

\_name = name;

}

public string Name

{

get { return \_name; }

set

{

if (!\_name.Equals(value))

{

Notify?.Invoke($"Значение поля {nameof(Name)} изменено.\n" +

$"Предыдущее значение = {\_name}, актуальное = {value}.");

}

\_name = value;

}

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Иванов Иван Иванович 1  Мики Вяч Никол | Иванов Иван Иванович 1  Мики Вяч Никол |

Анализ результатов:

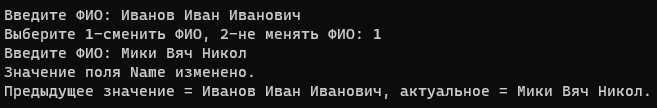


Рисунок 3.1 – Результат работы программы

Источник: собственная разработка

Задание 4. Создать класс, в нем делегат и соответствующее ему событие. Создать два класса-наблюдателя, в методах которых будет описана их реакция на событие. Добавить 2 обработчика к событию из первого класса и один из второго. Вывести результат на экран. Удалить один обработчик события и вывести результат на экран.

Листинг программы:

using static Task\_4.Program;

namespace Task\_4

{

class Program

{

public delegate void DelegateEvent(double sideA, double sideB, double sideC);

public static void Main(string[] args)

{

Console.Write("Введите сторону А: ");

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

Console.Write("Введите сторону В: ");

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

Console.Write("Введите сторону С: ");

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

ActionClass ac = new ActionClass(sideA, sideB, sideC);

ObservationOne observation1 = new ObservationOne();

ac.Action += observation1.SquareTriangle;

ac.Action += observation1.PerimetrTriangle;

ObservationTwo observation2 = new ObservationTwo();

ac.Action += observation2.ViewTriangle;

ac.Raise();

ac.Action -= observation2.ViewTriangle;

ac.Raise();

}

}

class ActionClass

{

public event DelegateEvent Action;

private double sideA;

private double sideB;

private double sideC;

public ActionClass(double sideA, double sideB, double sideC)

{

this.sideA = sideA;

this.sideB = sideB;

this.sideC = sideC;

}

public void Raise()

{

if (Action != null)

{

Action(sideA, sideB, sideC);

}

}

}

class ObservationOne

{

public void SquareTriangle(double sideA, double sideB, double sideC)

{

double semi = (sideA + sideB + sideC) / 2;

double result = Math.Sqrt(semi \* (semi - sideA) \* (semi - sideB) \* (semi - sideC));

Console.WriteLine($"Площадь треугольника: {result}");

}

public void PerimetrTriangle(double sideA, double sideB, double sideC)

{

Console.WriteLine($"Периметр треугольника: {sideA + sideB + sideC}");

}

}

class ObservationTwo

{

public void ViewTriangle(double sideA, double sideB, double sideC)

{

if (sideA == sideB && sideA == sideC)

{

Console.WriteLine($"Тип: Равносторонний");

}

else if (sideB == sideC)

{

Console.WriteLine($"Тип: Равнобедренный");

}

else

{

Console.WriteLine($"Тип: Разносторонний");

}

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 4 5 | 6 12 Разносторонний 6 12 |

Анализ результатов:

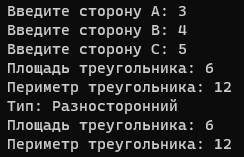


Рисунок 4.1 – Результат работы программы

Источник: собственная разработка