**3.Процедуры и функции – методы класса**

Задание №1. Разработать метод F, который нечетное число заменяет на 0, а четное уменьшает в два раза. Продемонстрировать работу этого метода.

Листинг программы:

static double F(double number)

{ if (number % 2 ==0)

{ return number = 0; }

else { return number / 2; } }

static void Main()

{ Console.Write("Enter number: ");

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

Console.WriteLine(F(number)); }

Таблица 3.1 – Входные и выходные данных задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Число: 5 | 2.5 |

Анализ результатов:



Рисунок 3.1 – Результат работы программы из задания №1

Задание №2. Построить таблицу значений функции y

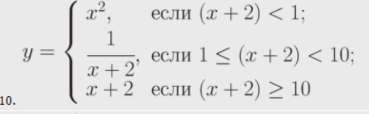


Рисунок 3.2 – Результат работы программы из задания №2

Листинг программы:

static double F(double x)

{ double y;

if (x + 2 < 1)

{ return y = Math.Pow(x, 2);}

else if (x + 2 >= 1 && x + 2 < 10)

{ return y = 1 / ( x + 2); }

else

{ return y = x + 2; }}

static void Main()

{ Console.Write("a = ");

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

Console.Write("b = ");

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

Console.Write("h = ");

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

for (double i = a; i <= b; i += h)

Console.WriteLine("F({0:F2}) = {1:F4}", i, F(i));}

Таблица 3.2 – Входные и выходные данных задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| а = 5  b = 8  h = 2 | F(5) = 0,1429  F(7) = 0,111 |

Анализ результатов:

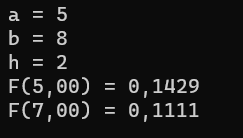


Рисунок 3.2 – Результат работы программы из задания №2

Задание №3. Перегрузить метод F из предыдущего раздела так, чтобы его сигнатура соответствовала виду static void(double x, out double y).Продемонстрировать работу перегруженных методов.

Листинг программы:

static double F(double x)

{ double y;

if (x + 2 < 1)

{ return y = Math.Pow(x, 2); }

else if (x + 2 >= 1 && x + 2 < 10)

{ return y = 1 / (x + 2); }

else

{ return y = x + 2;}}

static void F(double x, out double y )

{ y = 0;

if ((x + 2) < 1) y = Math.Pow(x, 2);

else if ((x + 2) < 10) y = 1 / (x + 2);

else y = x + 2;}

Console.Write("a = ");

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

Console.Write("b = ");

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

Console.Write("h = ");

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

Console.WriteLine("Первая перегрузка");

for (double i = a; i <= b; i += h)

Console.WriteLine("F({0:F2}) = {1:F4}", i, F(i));

Console.WriteLine("Вторая перегрузка");

for (double i = a; i <= b; i += h)

{ F(i, out double y);

Console.WriteLine("F({0:F2}) = {1:F4}", i, y); }}

Таблица 3.3 – Входные и выходные данных задание №3

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| A = 4  B = 7  H = 1 | Первая перегрузка  F(4) = 0,1667  F(5) = 0,1429  Вторая перегрузка  F(4) = 0,1667  F(5) = 0,1429 |

Анализ результатов:

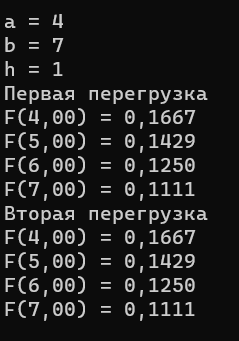


Рисунок 3.3 – Результат работы программы из задания №3

Задание №4. Создайте проект, в котором опишите класс для решения задачи Вашего варианта. Разрабатываемый класс должен содержать следующие элементы: скрытые и открытые поля, конструкторы, перегруженные операции. В программе должна выполняться проверка всех разработанных элементов класса. Описать класс для работы с двумерными массивами чисел. Реализовать возможность выполнения для согласованных массивов комбинированных операций присваивания (+=, -=).

Листинг программы:

int N = 3;

Random rand = new Random();

Matrix first = new Matrix(N);

Matrix second = new Matrix(N);

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

for (int j = 0; j < N; j++)

{ first[i, j] = rand.Next(1, 5);

second[i, j] = rand.Next(1, 5); }

Console.WriteLine("First matrix :\n\n");

first.PrintMatrix();

Console.WriteLine("\n\nSecond matrix:\n\n");

second.PrintMatrix();

Console.WriteLine("\n\nMatrix sum :\n\n");

(first + second).PrintMatrix();

Console.WriteLine("\n\nMatrix difference:\n\n");

(first - second).PrintMatrix();

Console.ReadKey();}

public class Matrix

{ private double[,] matrix;

int Row = 0, Col = 0;

public Matrix(int row, int col)

{ matrix = new double[row, col];

Row = row; Col = col; }

public Matrix(int N)

{ matrix = new double[N, N];

Row = Col = N; }

public double this[int i, int j]

{ get { return matrix[i, j]; }

set { matrix[i, j] = value; } }

public static Matrix operator +(Matrix first, Matrix second)

{ Matrix mat = new Matrix(first.Row, first.Col);

for (int i = 0; i < first.Row; i++)

for (int j = 0; j < first.Col; j++)

mat[i, j] = first[i, j] + second[i, j];

return mat; }

public static Matrix operator -(Matrix first, Matrix second)

{ Matrix mat = new Matrix(first.Row, first.Col);

for (int i = 0; i < first.Row; i++)

for (int j = 0; j < first.Col; j++)

mat[i, j] = first[i, j] - second[i, j];

return mat; }

public void PrintMatrix()

{ for (int i = 0; i < this.Row; i++)

{ for (int j = 0; j < this.Col; j++)

Console.Write("{0} ", this[i, j]);

Console.Write("\n"); }

}

Таблица 3.4 – Входные и выходные данных задание №4

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | First matrix:  1 2 1  1 1 4  2 1 2  Second matrix:  2 4 2  2 1 3  2 2 3  Matrix sum:  3 6 3  3 2 7  4 3 5  Matrix difference:  -1 -2 -1  -1 0 1  0 -1 -1 |

Анализ результатов:

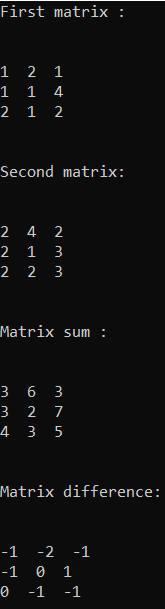


Рисунок 3.4 - Результат работы программы из задания №4

Задание №5. Описать класс «автостоянка» для хранения сведений об автомобилях. Для каждого автомобиля записываются госномер, цвет, фамилия

владельца и признак присутствия на стоянке. Обеспечить возможность поиска автомобиля по разным критериям, вывода списка присутствующих и отсутствующих на стоянке автомобилей, доступа к имеющимся сведениям по номеру места. Написать программу, демонстрирующую все разработанные элементы класса.

Листинг программы:

class Parking

{ Car[] cars;

public Parking()

{ int length = 0;

cars = new Car[length]; }

public void Print()

{ for (int i = 0; i < cars.Length; i++)

{ Console.WriteLine("Фамиля владельца: " + cars[i].OwnerName);

Console.WriteLine("Госномер: " + cars[i].NumberPlate);

Console.WriteLine("Цвет: " + cars[i].Color);

Console.WriteLine("Присутствие на стоянке: " + cars[i].Presence); } }

public void FindByLastName(string ownerName)

{ for (int i = 0; i < cars.Length; i++)

{ if (cars[i].OwnerName == ownerName)

{ Console.WriteLine($"Машина на стоянке" + "\nНомер авто - " + cars[i].NumberPlate + "\nЦвет авто - " + cars[i].Color + "\nФамилия владельца - " + cars[i].OwnerName);

} else

{ Console.WriteLine("Машина на стоянке отсутствует"); } } }

public void FindByNumber(int numberPlate)

{ for (int i = 0; i < cars.Length; i++)

{ if (cars[i].NumberPlate == numberPlate)

{ Console.WriteLine("Машина на стоянке " + "\nНомер авто - " + cars[i].NumberPlate + "\nЦвет авто - " + cars[i].Color + "\nФамилия владельца - " + cars[i].OwnerName); }

else

{ Console.WriteLine("Машина на стоянке отсутствует"); } }}

public void FindByColor(string color)

{ for (int i = 0; i < cars.Length; i++)

{ if (cars[i].Color == color)

{ Console.WriteLine("Машина на стоянке " + "\nНомер авто - " + cars[i].NumberPlate + "\nЦвет авто - " + cars[i].Color + "\nФамилия владельца - " + cars[i].OwnerName); }

else

{ Console.WriteLine("Машина на стоянке отсутствует"); }}}

public IReadOnlyList<Car> FindBy(Predicate<Car> filter)

{ var findCars = new List<Car>();

foreach (var car in cars)

{ if (filter(car))

{ findCars.Add(car); }}

return findCars; }

public void Add(string n, int a, string y, string u)

{ Array.Resize(ref cars, cars.Length + 1);

cars[cars.Length - 1] = new Car(a, y, n, u); } }

Таблица 3.1 – Входные и выходные данных задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Фамилия владельца – Дерюга  Госномер – 2342  Цвет – Зеленый  Присутствие на стоянке - Да |

Анализ результатов:

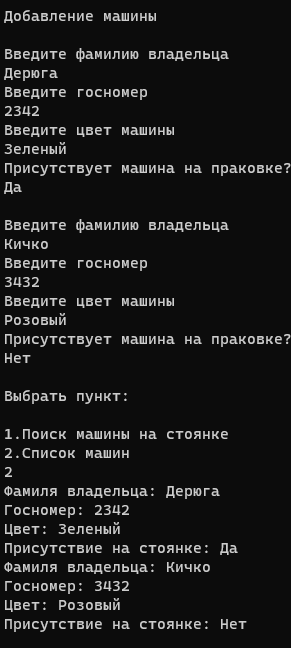


Рисунок 3.1 – Результат работы программы из задания №5