

using System;

class Counter

{

private int value;

private int minValue;

private int maxValue;

public Counter(int minValue, int maxValue)

{

this.minValue = minValue;

this.maxValue = maxValue;

this.value = minValue;

}

public void Increment()

{

if (value < maxValue)

{

value++;

}

}

public void Decrement()

{

if (value > minValue)

{

value--;

}

}

public int GetValue()

{

return value;

}

}

class Program

{

static void Main()

{

Counter counter = new Counter(0, 10);

// Тестируем методы

counter.Increment();

Console.WriteLine("Текущее значение: " + counter.GetValue()); // Должно быть 1

counter.Increment();

Console.WriteLine("Текущее значение: " + counter.GetValue()); // Должно быть 2

counter.Decrement();

Console.WriteLine("Текущее значение: " + counter.GetValue()); // Должно быть 1

// Проверка границ

for (int i = 0; i < 10; i++)

{

counter.Increment();

}

Console.WriteLine("Текущее значение: " + counter.GetValue()); // Должно быть 10

counter.Increment();

Console.WriteLine("Текущее значение: " + counter.GetValue()); // Должно остаться 10

for (int i = 0; i < 10; i++)

{

counter.Decrement();

}

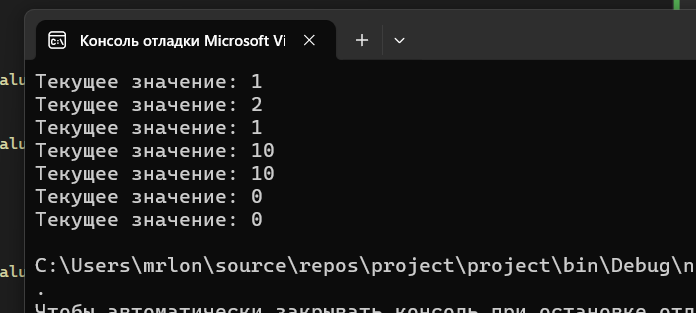
Console.WriteLine("Текущее значение: " + counter.GetValue()); // Должно быть 0

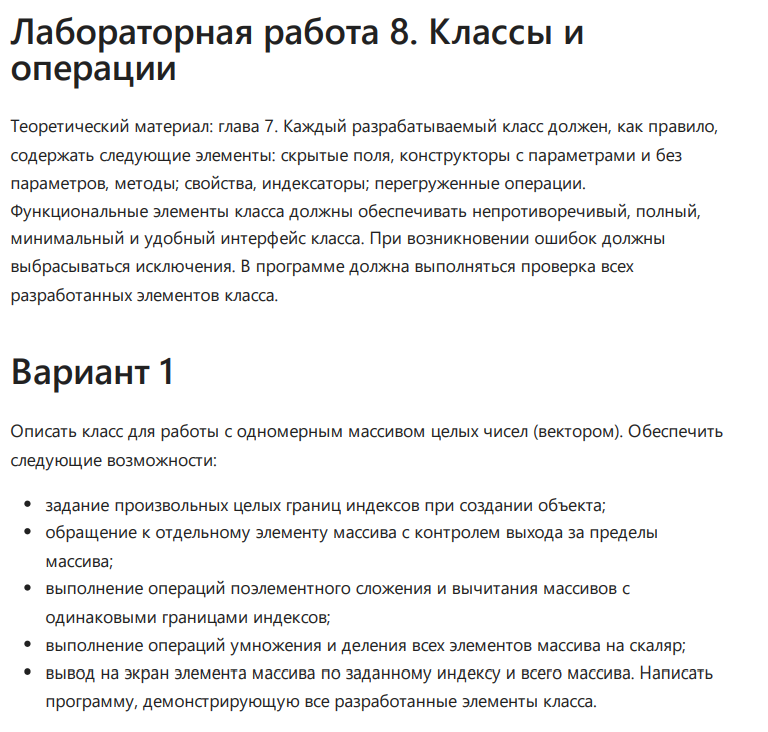
counter.Decrement();

Console.WriteLine("Текущее значение: " + counter.GetValue()); // Должно остаться 0

}

}





using System;

class Vector

{

private int[] elements;

private int minIndex;

private int maxIndex;

public Vector(int minIndex, int maxIndex)

{

if (minIndex > maxIndex)

{

throw new ArgumentException("Минимальный индекс не может быть больше максимального.");

}

this.minIndex = minIndex;

this.maxIndex = maxIndex;

elements = new int[maxIndex - minIndex + 1];

}

public int this[int index]

{

get

{

if (index < minIndex || index > maxIndex)

{

throw new IndexOutOfRangeException("Индекс выходит за пределы массива.");

}

return elements[index - minIndex];

}

set

{

if (index < minIndex || index > maxIndex)

{

throw new IndexOutOfRangeException("Индекс выходит за пределы массива.");

}

elements[index - minIndex] = value;

}

}

public static Vector operator +(Vector v1, Vector v2)

{

if (v1.minIndex != v2.minIndex || v1.maxIndex != v2.maxIndex)

{

throw new ArgumentException("Массивы должны иметь одинаковые границы индексов.");

}

Vector result = new Vector(v1.minIndex, v1.maxIndex);

for (int i = v1.minIndex; i <= v1.maxIndex; i++)

{

result[i] = v1[i] + v2[i];

}

return result;

}

public static Vector operator -(Vector v1, Vector v2)

{

if (v1.minIndex != v2.minIndex || v1.maxIndex != v2.maxIndex)

{

throw new ArgumentException("Массивы должны иметь одинаковые границы индексов.");

}

Vector result = new Vector(v1.minIndex, v1.maxIndex);

for (int i = v1.minIndex; i <= v1.maxIndex; i++)

{

result[i] = v1[i] - v2[i];

}

return result;

}

public static Vector operator \*(Vector v, int scalar)

{

Vector result = new Vector(v.minIndex, v.maxIndex);

for (int i = v.minIndex; i <= v.maxIndex; i++)

{

result[i] = v[i] \* scalar;

}

return result;

}

public static Vector operator /(Vector v, int scalar)

{

if (scalar == 0)

{

throw new DivideByZeroException("Деление на ноль.");

}

Vector result = new Vector(v.minIndex, v.maxIndex);

for (int i = v.minIndex; i <= v.maxIndex; i++)

{

result[i] = v[i] / scalar;

}

return result;

}

public void Print()

{

for (int i = minIndex; i <= maxIndex; i++)

{

Console.WriteLine($"Элемент {i}: {this[i]}");

}

}

}

class Program

{

static void Main()

{

Vector v1 = new Vector(1, 5);

v1[1] = 10;

v1[2] = 20;

v1[3] = 30;

v1[4] = 40;

v1[5] = 50;

Vector v2 = new Vector(1, 5);

v2[1] = 5;

v2[2] = 10;

v2[3] = 15;

v2[4] = 20;

v2[5] = 25;

Vector sum = v1 + v2;

Vector diff = v1 - v2;

Vector mul = v1 \* 2;

Vector div = v1 / 2;

Console.WriteLine("Сумма:");

sum.Print();

Console.WriteLine("Разность:");

diff.Print();

Console.WriteLine("Умножение на скаляр:");

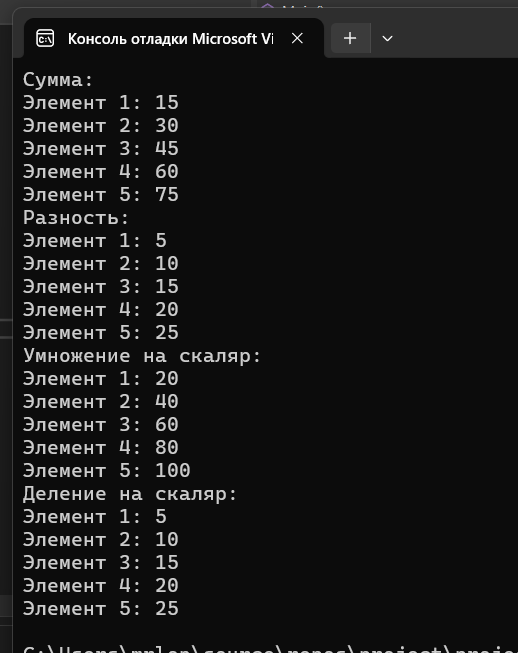
mul.Print();

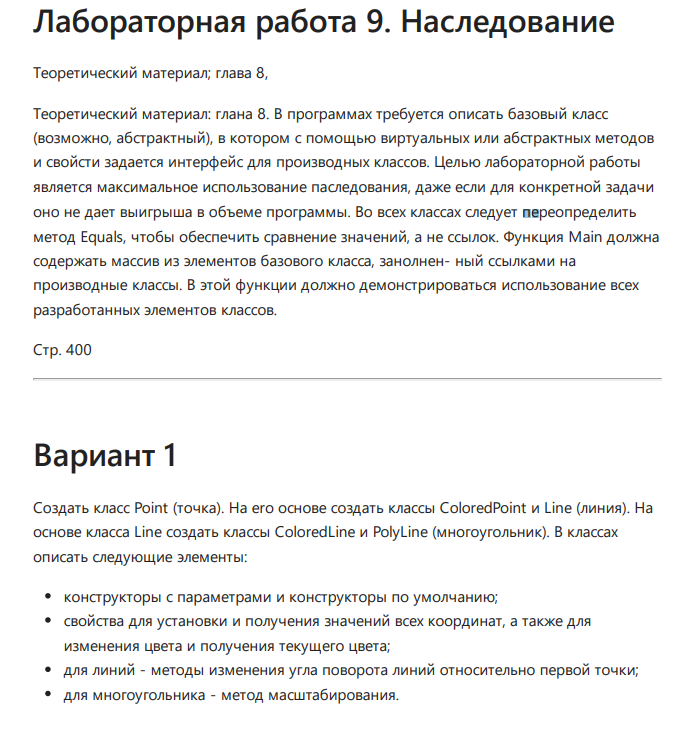
Console.WriteLine("Деление на скаляр:");

div.Print();

}

}





using System;

// Базовый класс Point

public class Point

{

protected double x;

protected double y;

public Point(double x, double y)

{

this.x = x;

this.y = y;

}

public Point() : this(0, 0) { }

public double X { get => x; set => x = value; }

public double Y { get => y; set => y = value; }

public override bool Equals(object obj)

{

if (obj == null || GetType() != obj.GetType())

return false;

Point other = (Point)obj;

return x == other.x && y == other.y;

}

public override int GetHashCode()

{

return HashCode.Combine(x, y);

}

}

// Класс ColoredPoint

public class ColoredPoint : Point

{

private string color;

public ColoredPoint(double x, double y, string color) : base(x, y)

{

this.color = color;

}

public ColoredPoint() : this(0, 0, "Black") { }

public string Color { get => color; set => color = value; }

public override bool Equals(object obj)

{

if (!base.Equals(obj))

return false;

ColoredPoint other = (ColoredPoint)obj;

return color == other.color;

}

public override int GetHashCode()

{

return HashCode.Combine(base.GetHashCode(), color);

}

}

// Класс Line

public class Line

{

protected Point start;

protected Point end;

public Line(Point start, Point end)

{

this.start = start;

this.end = end;

}

public Line() : this(new Point(), new Point()) { }

public Point Start { get => start; set => start = value; }

public Point End { get => end; set => end = value; }

public void Rotate(double angle)

{

double dx = end.X - start.X;

double dy = end.Y - start.Y;

double newX = dx \* Math.Cos(angle) - dy \* Math.Sin(angle);

double newY = dx \* Math.Sin(angle) + dy \* Math.Cos(angle);

end.X = start.X + newX;

end.Y = start.Y + newY;

}

public override bool Equals(object obj)

{

if (obj == null || GetType() != obj.GetType())

return false;

Line other = (Line)obj;

return start.Equals(other.start) && end.Equals(other.end);

}

public override int GetHashCode()

{

return HashCode.Combine(start, end);

}

}

// Класс ColoredLine

public class ColoredLine : Line

{

private string color;

public ColoredLine(Point start, Point end, string color) : base(start, end)

{

this.color = color;

}

public ColoredLine() : this(new Point(), new Point(), "Black") { }

public string Color { get => color; set => color = value; }

public override bool Equals(object obj)

{

if (!base.Equals(obj))

return false;

ColoredLine other = (ColoredLine)obj;

return color == other.color;

}

public override int GetHashCode()

{

return HashCode.Combine(base.GetHashCode(), color);

}

}

// Класс PolyLine

public class PolyLine : Line

{

private double scaleFactor;

public PolyLine(Point start, Point end, double scaleFactor) : base(start, end)

{

this.scaleFactor = scaleFactor;

}

public PolyLine() : this(new Point(), new Point(), 1.0) { }

public double ScaleFactor { get => scaleFactor; set => scaleFactor = value; }

public void Scale()

{

start.X \*= scaleFactor;

start.Y \*= scaleFactor;

end.X \*= scaleFactor;

end.Y \*= scaleFactor;

}

public override bool Equals(object obj)

{

if (!base.Equals(obj))

return false;

PolyLine other = (PolyLine)obj;

return scaleFactor == other.scaleFactor;

}

public override int GetHashCode()

{

return HashCode.Combine(base.GetHashCode(), scaleFactor);

}

}

class Program

{

static void Main()

{

Point p1 = new Point(1, 2);

Point p2 = new Point(3, 4);

ColoredPoint cp1 = new ColoredPoint(5, 6, "Red");

ColoredPoint cp2 = new ColoredPoint(7, 8, "Blue");

Line l1 = new Line(p1, p2);

ColoredLine cl1 = new ColoredLine(p1, p2, "Green");

PolyLine pl1 = new PolyLine(p1, p2, 2.0);

Point[] points = { p1, p2, cp1, cp2 };

Line[] lines = { l1, cl1, pl1 };

foreach (Point point in points)

{

Console.WriteLine($"Точка: ({point.X}, {point.Y})");

}

foreach (Line line in lines)

{

Console.WriteLine($"Линия: ({line.Start.X}, {line.Start.Y}) -> ({line.End.X}, {line.End.Y})");

}

l1.Rotate(Math.PI / 4);

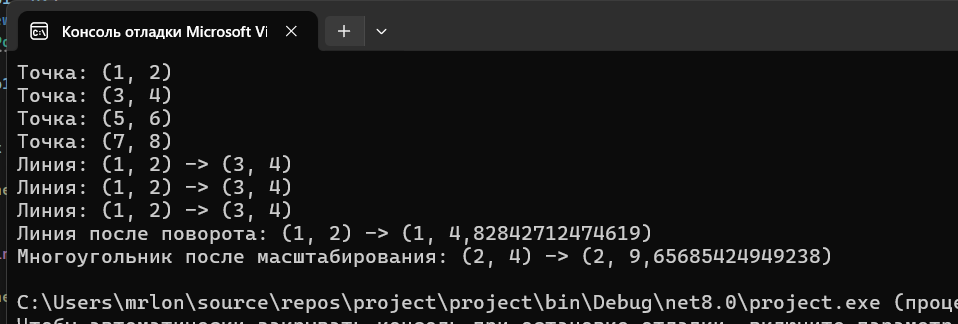
Console.WriteLine($"Линия после поворота: ({l1.Start.X}, {l1.Start.Y}) -> ({l1.End.X}, {l1.End.Y})");

pl1.Scale();

Console.WriteLine($"Многоугольник после масштабирования: ({pl1.Start.X}, {pl1.Start.Y}) -> ({pl1.End.X}, {pl1.End.Y})");

}

}



using System;

// Структура STUDENT

struct STUDENT

{

public string фамилия;

public string инициалы;

public int номерГруппы;

public float[] успеваемость;

public STUDENT(string фамилия, string инициалы, int номерГруппы, float[] успеваемость)

{

this.фамилия = фамилия;

this.инициалы = инициалы;

this.номерГруппы = номерГруппы;

this.успеваемость = успеваемость;

}

}

class Program

{

static void Main()

{

// Массив из десяти структур STUDENT

STUDENT[] students = new STUDENT[10];

// Ввод данных с клавиатуры

for (int i = 0; i < 10; i++)

{

Console.Write("Введите фамилию студента: ");

string фамилия = Console.ReadLine();

Console.Write("Введите инициалы студента: ");

string инициалы = Console.ReadLine();

Console.Write("Введите номер группы студента: ");

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

Console.Write("Введите успеваемость (5 оценок через пробел): ");

string[] успеваемостьStr = Console.ReadLine().Split(' ');

float[] успеваемость = new float[5];

for (int j = 0; j < 5; j++)

{

успеваемость[j] = float.Parse(успеваемостьStr[j]);

}

students[i] = new STUDENT(фамилия, инициалы, номерГруппы, успеваемость);

}

// Сортировка массива по возрастанию номера группы

Array.Sort(students, (x, y) => x.номерГруппы.CompareTo(y.номерГруппы));

// Вычисление среднего балла для каждого студента

float[] средниеБаллы = new float[10];

for (int i = 0; i < 10; i++)

{

средниеБаллы[i] = 0;

for (int j = 0; j < 5; j++)

{

средниеБаллы[i] += students[i].успеваемость[j];

}

средниеБаллы[i] /= 5;

}

// Вывод фамилий и номеров групп для студентов с средним баллом больше 4.0

bool found = false;

for (int i = 0; i < 10; i++)

{

if (средниеБаллы[i] > 4.0f)

{

Console.WriteLine($"Фамилия: {students[i].фамилия}, Номер группы: {students[i].номерГруппы}");

found = true;

}

}

// Если таких студентов нет, вывести соответствующее сообщение

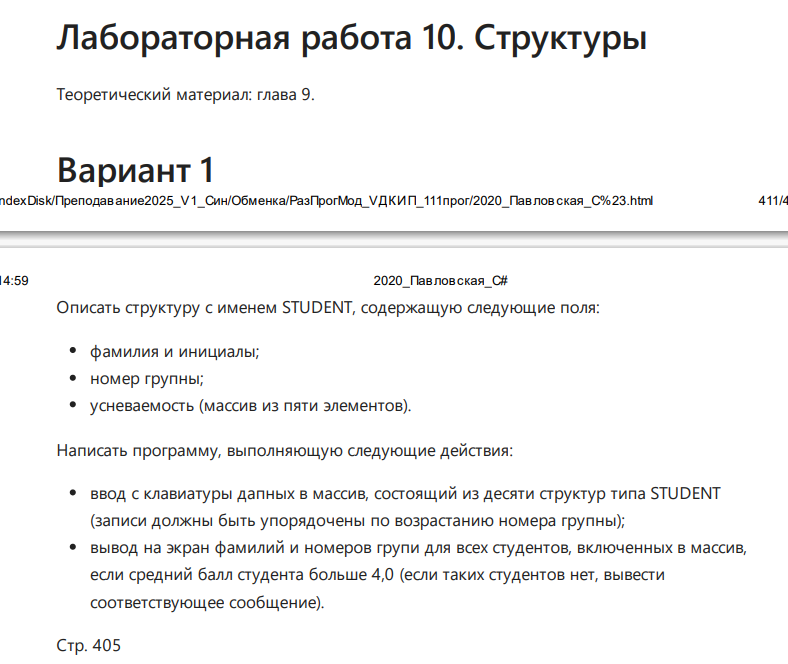
if (!found)

{

Console.WriteLine("Студентов с средним баллом больше 4.0 нет.");

}

}

}

using System;

using System.Linq;

namespace Lab10\_Structures

{

// Определение структуры STUDENT

struct STUDENT

{

public string FullName; // Фамилия и инициалы

public int GroupNumber; // Номер группы

public int[] Grades; // Успеваемость (массив из 5 элементов)

// Метод для вычисления среднего балла

public double GetAverageGrade()

{

return Grades.Average();

}

}

class Program

{

static void Main(string[] args)

{

const int studentCount = 10;

const int gradeCount = 5;

STUDENT[] students = new STUDENT[studentCount];

Console.WriteLine("Введите данные о 10 студентах:");

// Ввод данных о студентах

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

{

Console.WriteLine($"\nСтудент {i + 1}:");

Console.Write("Фамилия и инициалы: ");

students[i].FullName = Console.ReadLine();

Console.Write("Номер группы: ");

students[i].GroupNumber = int.Parse(Console.ReadLine());

students[i].Grades = new int[gradeCount];

Console.WriteLine("Введите 5 оценок:");

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

{

Console.Write($"Оценка {j + 1}: ");

students[i].Grades[j] = int.Parse(Console.ReadLine());

}

}

// Сортировка студентов по номеру группы (по возрастанию)

students = students.OrderBy(s => s.GroupNumber).ToArray();

// Вывод студентов с средним баллом > 4.0

Console.WriteLine("\nСтуденты со средним баллом > 4.0:");

bool found = false;

foreach (var student in students)

{

if (student.GetAverageGrade() > 4.0)

{

Console.WriteLine($"Фамилия: {student.FullName}, Группа: {student.GroupNumber}, Средний балл: {student.GetAverageGrade():F2}");

found = true;

}

}

if (!found)

{

Console.WriteLine("Студентов со средним баллом > 4.0 не найдено.");

}

}

}

}