**НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ** **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** **«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ** **“СИНЕРГИЯ”»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Синергия |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Програмист |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | Очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Отчет по лабораторной работе №5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **на тему** |  | Проектирование классов | | |
|  |  | (наименование темы) | | |
|  |  |  | | |
| **по дисциплине** | | |  | Разработка программных модулей |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | Осовицкий Б.Н. |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | ДКИП-205 прог |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев И.В. |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2025 г.**

**Лабораторная работа №5. «Проектирование классов»**

**Задание №1. Вариант 1.**

Описать класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, - и свойство, позволяющее получить его текущее состояние. При выходе за границы диапазона выбрасываются исключения. Написать программу, демонстрирующую все разработанные элементы класса.

**Листинг кода:**

using System;

namespace DecimalCounterDemo

{

// Класс, реализующий десятичный счётчик

public class DecimalCounter

{

private int \_minValue;

private int \_maxValue;

private int \_currentValue;

// Свойство для получения текущего значения

public int CurrentValue => \_currentValue;

// Конструктор по умолчанию (диапазон от 0 до 10, начальное значение 0)

public DecimalCounter()

{

\_minValue = 0;

\_maxValue = 10;

\_currentValue = 0;

}

// Конструктор с произвольными значениями

public DecimalCounter(int minValue, int maxValue, int startValue)

{

if (minValue > maxValue)

throw new ArgumentException("Минимальное значение не может быть больше максимального.");

if (startValue < minValue || startValue > maxValue)

throw new ArgumentOutOfRangeException(nameof(startValue), "Начальное значение должно быть в пределах диапазона.");

\_minValue = minValue;

\_maxValue = maxValue;

\_currentValue = startValue;

}

// Метод увеличения значения

public void Increment()

{

if (\_currentValue >= \_maxValue)

throw new InvalidOperationException("Превышен максимальный предел счётчика.");

\_currentValue++;

}

// Метод уменьшения значения

public void Decrement()

{

if (\_currentValue <= \_minValue)

throw new InvalidOperationException("Превышен минимальный предел счётчика.");

\_currentValue--;

}

}

// Демонстрационная программа

class Program

{

static void Main(string[] args)

{

try

{

Console.WriteLine("Демонстрация счётчика по умолчанию:");

DecimalCounter defaultCounter = new DecimalCounter();

Console.WriteLine($"Текущее значение: {defaultCounter.CurrentValue}");

defaultCounter.Increment();

Console.WriteLine($"После увеличения: {defaultCounter.CurrentValue}");

defaultCounter.Decrement();

Console.WriteLine($"После уменьшения: {defaultCounter.CurrentValue}");

Console.WriteLine("\nДемонстрация счётчика с произвольными значениями:");

DecimalCounter customCounter = new DecimalCounter(5, 15, 10);

Console.WriteLine($"Текущее значение: {customCounter.CurrentValue}");

for (int i = 0; i < 6; i++)

{

customCounter.Increment();

Console.WriteLine($"Увеличено до: {customCounter.CurrentValue}");

}

// Попытка выйти за пределы

customCounter.Increment(); // Это должно вызвать исключение

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Console.WriteLine("\nПрограмма завершена.");

}

}

}

Код на компиляторе:

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Задание №2. Вариант 1.**

Описать класс для работы с одномерным массивом целых чисел (вектором).

Обеспечить следующие возможности:

- задание произвольных целых границ индексов при создании объекта;

-обращение к отдельному элементу массива с контролем выхода за пределы массива;

- выполнение операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов;

- выполнение операций умножения и деления всех элементов массива на скаляр;

- вывод на экран элемента массива по заданному индексу и всего массива.

- Написать программу, демонстрирующую все разработанные элементы класса.

**Листинг кода:**

using System;

namespace VectorDemo

{

public class Vector

{

private int[] \_data;

private int \_lowerBound;

private int \_upperBound;

// Индексатор с контролем выхода за пределы

public int this[int index]

{

get

{

CheckIndex(index);

return \_data[index - \_lowerBound];

}

set

{

CheckIndex(index);

\_data[index - \_lowerBound] = value;

}

}

// Конструктор с произвольными границами индексов

public Vector(int lowerBound, int upperBound)

{

if (upperBound < lowerBound)

throw new ArgumentException("Верхняя граница должна быть больше или равна нижней.");

\_lowerBound = lowerBound;

\_upperBound = upperBound;

\_data = new int[upperBound - lowerBound + 1];

}

// Метод проверки выхода за границы

private void CheckIndex(int index)

{

if (index < \_lowerBound || index > \_upperBound)

throw new IndexOutOfRangeException("Индекс вне диапазона.");

}

// Операция поэлементного сложения

public static Vector operator +(Vector a, Vector b)

{

CheckBounds(a, b);

Vector result = new Vector(a.\_lowerBound, a.\_upperBound);

for (int i = a.\_lowerBound; i <= a.\_upperBound; i++)

{

result[i] = a[i] + b[i];

}

return result;

}

// Операция поэлементного вычитания

public static Vector operator -(Vector a, Vector b)

{

CheckBounds(a, b);

Vector result = new Vector(a.\_lowerBound, a.\_upperBound);

for (int i = a.\_lowerBound; i <= a.\_upperBound; i++)

{

result[i] = a[i] - b[i];

}

return result;

}

// Умножение на скаляр

public void MultiplyByScalar(int scalar)

{

for (int i = 0; i < \_data.Length; i++)

{

\_data[i] \*= scalar;

}

}

// Деление на скаляр

public void DivideByScalar(int scalar)

{

if (scalar == 0)

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

for (int i = 0; i < \_data.Length; i++)

{

\_data[i] /= scalar;

}

}

// Вывод одного элемента

public void PrintElement(int index)

{

Console.WriteLine($"Элемент [{index}] = {this[index]}");

}

// Вывод всего массива

public void PrintAll()

{

for (int i = \_lowerBound; i <= \_upperBound; i++)

{

Console.Write($"{this[i]} ");

}

Console.WriteLine();

}

// Проверка, что границы векторов совпадают

private static void CheckBounds(Vector a, Vector b)

{

if (a.\_lowerBound != b.\_lowerBound || a.\_upperBound != b.\_upperBound)

throw new InvalidOperationException("Границы векторов не совпадают.");

}

}

class Program

{

static void Main(string[] args)

{

try

{

// Создание двух векторов

Vector v1 = new Vector(-2, 2);

Vector v2 = new Vector(-2, 2);

// Заполнение значений

for (int i = -2; i <= 2; i++)

{

v1[i] = i;

v2[i] = i \* 2;

}

Console.WriteLine("Вектор v1:");

v1.PrintAll();

Console.WriteLine("Вектор v2:");

v2.PrintAll();

// Сложение

Vector sum = v1 + v2;

Console.WriteLine("Сумма v1 + v2:");

sum.PrintAll();

// Вычитание

Vector diff = v1 - v2;

Console.WriteLine("Разность v1 - v2:");

diff.PrintAll();

// Умножение на скаляр

Console.WriteLine("v1 после умножения на 3:");

v1.MultiplyByScalar(3);

v1.PrintAll();

// Деление на скаляр

Console.WriteLine("v1 после деления на 3:");

v1.DivideByScalar(3);

v1.PrintAll();

// Обращение к элементу

Console.WriteLine("Вывод одного элемента v1[0]:");

v1.PrintElement(0);

// Попытка выйти за границу

// Console.WriteLine(v1[5]); // Раскомментировать для теста исключения

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Console.WriteLine("Программа завершена.");

}

}

}

Код на компиляторе:

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, меню

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

**Задание №3. Вариант 1.**

Создать класс Point (точка). Нa еro основе создать классы ColoredPoint и Line (линия). На основе класса Line создать классы ColoredLine и PolyLine (многоугольник).

В классах описать следующие элементы:

- конструкторы с параметрами и конструкторы по умолчанию;

- свойства для установки и получения значений всех координат, а также для изменения цвета и - получения текущего цвета;

- для линий - методы изменения угла поворота линий относительно первой точки;

- для многоугольника - метод масштабирования.

**Листинг кода:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace GeometryDemo

{

public class Point

{

public double X { get; set; }

public double Y { get; set; }

public Point()

{

X = 0;

Y = 0;

}

public Point(double x, double y)

{

X = x;

Y = y;

}

}

public class ColoredPoint : Point

{

public string Color { get; set; }

public ColoredPoint() : base()

{

Color = "Black";

}

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

{

Color = color;

}

}

public class Line

{

public Point Start { get; set; }

public Point End { get; set; }

public Line()

{

Start = new Point(0, 0);

End = new Point(1, 0);

}

public Line(Point start, Point end)

{

Start = start;

End = end;

}

// Метод поворота линии относительно начальной точки

public void Rotate(double angleDegrees)

{

double angleRadians = angleDegrees \* Math.PI / 180.0;

double dx = End.X - Start.X;

double dy = End.Y - Start.Y;

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

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

End = new Point(Start.X + newX, Start.Y + newY);

}

}

public class ColoredLine : Line

{

public string Color { get; set; }

public ColoredLine() : base()

{

Color = "Black";

}

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

{

Color = color;

}

}

public class PolyLine

{

public List<Point> Points { get; set; }

public PolyLine()

{

Points = new List<Point>();

}

public PolyLine(List<Point> points)

{

Points = points;

}

// Масштабирование относительно первой точки

public void Scale(double factor)

{

if (Points.Count == 0) return;

Point origin = Points[0];

for (int i = 1; i < Points.Count; i++)

{

double dx = Points[i].X - origin.X;

double dy = Points[i].Y - origin.Y;

Points[i].X = origin.X + dx \* factor;

Points[i].Y = origin.Y + dy \* factor;

}

}

}

class Program

{

static void Main(string[] args)

{

// Точка и цветная точка

Point p1 = new Point(1, 2);

ColoredPoint cp1 = new ColoredPoint(3, 4, "Red");

Console.WriteLine($"Точка p1: ({p1.X}, {p1.Y})");

Console.WriteLine($"Цветная точка cp1: ({cp1.X}, {cp1.Y}), цвет: {cp1.Color}");

// Линия и поворот

Line line = new Line(new Point(0, 0), new Point(1, 0));

Console.WriteLine($"Линия до поворота: Start=({line.Start.X}, {line.Start.Y}), End=({line.End.X}, {line.End.Y})");

line.Rotate(90);

Console.WriteLine($"Линия после поворота на 90°: Start=({line.Start.X}, {line.Start.Y}), End=({line.End.X}, {line.End.Y})");

// Цветная линия

ColoredLine cl = new ColoredLine(new Point(0, 0), new Point(1, 1), "Blue");

Console.WriteLine($"Цветная линия: Start=({cl.Start.X}, {cl.Start.Y}), End=({cl.End.X}, {cl.End.Y}), цвет: {cl.Color}");

// Многоугольник и масштабирование

PolyLine poly = new PolyLine(new List<Point>

{

new Point(0, 0),

new Point(1, 0),

new Point(1, 1),

new Point(0, 1)

});

Console.WriteLine("Многоугольник до масштабирования:");

foreach (var pt in poly.Points)

{

Console.WriteLine($"({pt.X}, {pt.Y})");

}

poly.Scale(2);

Console.WriteLine("Многоугольник после масштабирования (×2):");

foreach (var pt in poly.Points)

{

Console.WriteLine($"({pt.X}, {pt.Y})");

}

Console.WriteLine("Программа завершена.");

}

}

}

Код на компиляторе:

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Задание №4. Вариант 1.**

Описать структуру с именем STUDENT, содержащую следующие поля:

фамилия и инициалы;

номер групны;

усневаемость (массив из пяти элементов).

Написать программу, выполняющую следующие действия: ввод с клавиатуры дапных в массив, состоящий из десяти структур типа STUDENT (записи должны быть упорядочены по возрастанию номера групны); вывод на экран фамилий и номеров групи для всех студентов, включенных в массив, если средний балл студента больше 4,0 (если таких студентов нет, вывести соответствующее сообщение).

**Листинг кода:**

using System;

using System.Linq;

struct STUDENT

{

public string LastNameInitials; // Фамилия и инициалы

public int GroupNumber; // Номер группы

public int[] Performance; // Успеваемость (массив из 5 оценок)

// Метод для вычисления среднего балла

public double GetAverageScore()

{

return Performance.Average();

}

}

class Program

{

static void Main()

{

const int studentCount = 10;

STUDENT[] students = new STUDENT[studentCount];

Console.WriteLine($"Введите данные о {studentCount} студентах:");

// Ввод данных о студентах

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

{

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

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

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

Console.Write("Номер группы: ");

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

students[i].Performance = new int[5];

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

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

{

Console.Write($"Оценка {j + 1}: ");

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

}

}

// Упорядочивание студентов по номеру группы

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

// Вывод студентов с средним баллом > 4.0

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

bool found = false;

foreach (var student in students)

{

if (student.GetAverageScore() > 4.0)

{

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

found = true;

}

}

if (!found)

{

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

}

}

}

Код на компиляторе:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.