# **Московский государственный технический университет имени Н. Э. Баумана.**

Факультет “Информатика и системы управления”

Кафедра ИУ5 ”Системы обработки информации и управления”

Курс “Парадигмы и конструкции языков программирования”

Отчет по лабораторным работам №2-3.

Выполнила:

Студент группы ИУ5-31Б

Савельева Д.А

Подпись и дата:

Проверил:

Преподаватель кафедры ИУ5

Нардид А.Н.

Подпись и дата:

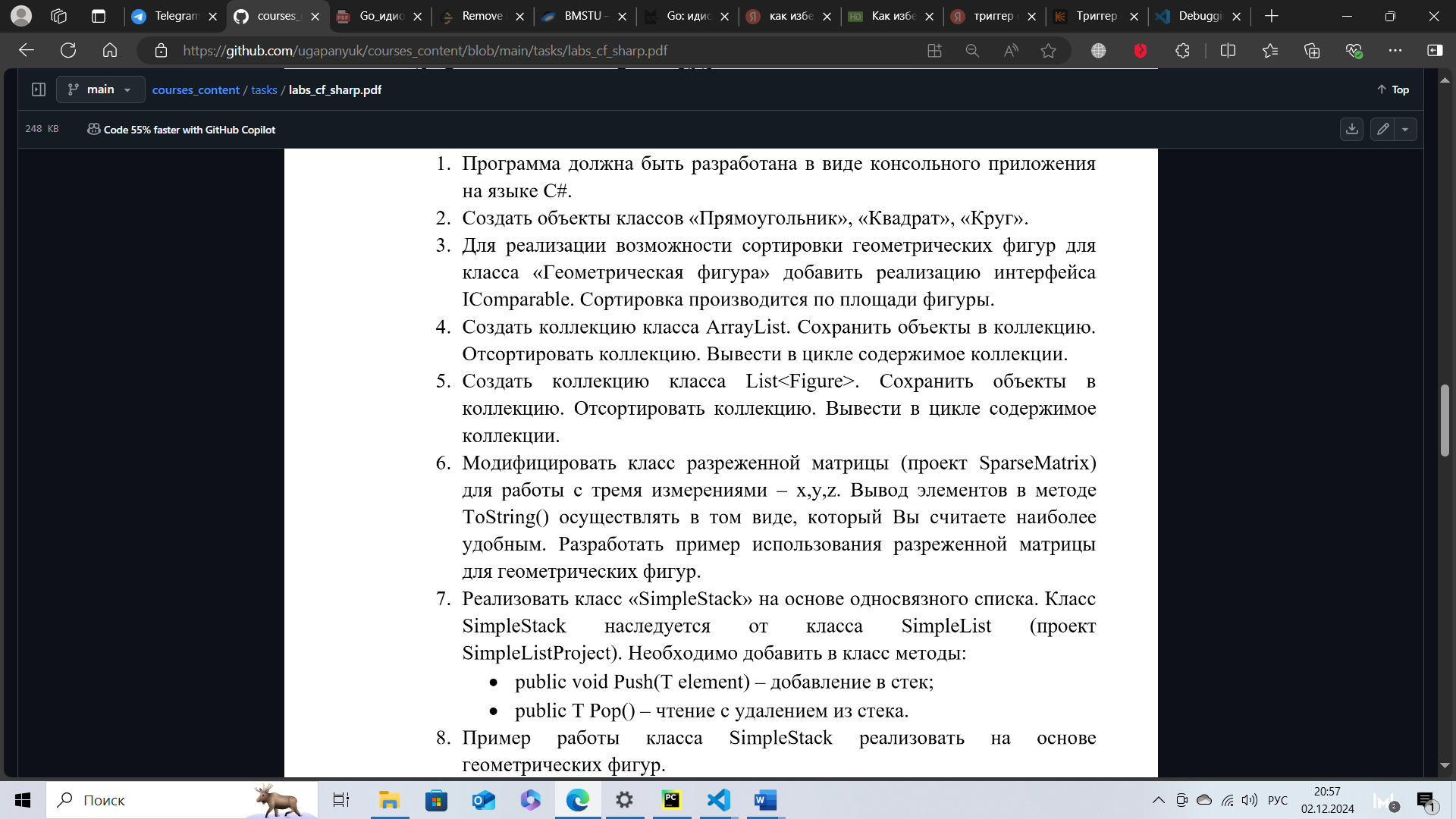
Москва,2024 г.

Постановка задачи.

Часть 1.

### 

Часть 2.



### Текст программы:

using System;

using System.Collections;

using System.Collections.Generic;

class Program

{

    static void Main(string[] args)

    {

        Figure rect = new Rectangle() { A = 6, B = 2 };

        Figure sq = new Square(2);

        Figure c = new Circle { R = 3 };

        List<Figure> list1 = new List<Figure> { rect, sq, c };

        ArrayList list2 = new ArrayList { rect, sq, c };

        list2.Sort(new FigureComparer()); // Сортировка используя IComparer

        list1.Sort();

        Console.WriteLine("Вывод Массивов");

        Console.WriteLine("\nОтсортированные фигуры 1 массива:");

        foreach (Figure figure in list1)

        {

            figure.Print();

        }

        Console.WriteLine("\nОтсортированные фигуры 2 массива:");

        foreach (Figure figure in list2)

        {

            figure.Print();

        }

        Cube cube = new Cube(6);

        cube.PrintVer();

        SparseMatrix sparseMatrix = new SparseMatrix();

        sparseMatrix.SetValue(2, 4, 1, 1);

        sparseMatrix.SetValue(2, 1, 1, 1);

        Console.WriteLine("Разреженная матрица:");

        Console.WriteLine(sparseMatrix.ToString());

        SimpleStack<Figure> stack = new SimpleStack<Figure>();

        stack.Push(rect);

        stack.Push(sq);

        stack.Push(c);

        stack.Pop();

    }

}

public interface IPrint

{

    void Print();

}

public class FigureComparer : IComparer

{

    public int Compare(object x, object y)

    {

        Figure figure1 = x as Figure;

        Figure figure2 = y as Figure;

        if (figure1 == null || figure2 == null)

            throw new ArgumentException();

        return figure1.Area.CompareTo(figure2.Area);

    }

}

public abstract class Figure : IPrint, IComparable<Figure>

{

    public double Area { get; set; }

    public virtual string ToString()

    {

        return "Фигура не задана";

    }

    public virtual void Print()

    {

        Console.WriteLine(this.ToString());

    }

    public int CompareTo(Figure? other)

    {

        if (other == null)

            return 1;

        return Area.CompareTo(other.Area);

    }

}

class Rectangle : Figure

{

    public double A { get; set; }

    public double B { get; set; }

    public override string ToString()

    {

        string t1 = A.ToString();

        string t2 = B.ToString();

        string t3 = (A \* B).ToString();

        Area = A \* B;

        string res = "Параметры: " + t1 + ", " + t2 + "\nПлощадь: " + t3;

        return res;

    }

    public override void Print()

    {

        Console.WriteLine(this.ToString());

    }

}

class Square : Rectangle

{

    public Square(double a)

    {

        A = a;

        B = a;

    }

}

class Circle : Figure

{

    public double R { get; set; }

    public override string ToString()

    {

        string t1 = R.ToString();

        string t2 = (Math.PI \* R \* R).ToString();

        Area = Math.PI \* R \* R;

        string res = "Параметры: " + t1 + "\nПлощадь: " + t2;

        return res;

    }

    public override void Print()

    {

        Console.WriteLine(this.ToString());

    }

}

class SparseMatrix

{

    private Dictionary<(int x, int y, int z), double> matrix;

    public SparseMatrix()

    {

        matrix = new Dictionary<(int, int, int), double>();

    }

    public void SetValue(int x, int y, int z, double value)

    {

        if (value != 0)

        {

            matrix[(x, y, z)] = value;

        }

        else

        {

            matrix.Remove((x, y, z));

        }

    }

    public double GetValue(int x, int y, int z)

    {

        matrix.TryGetValue((x, y, z), out double value);

        return value;

    }

    public override string ToString()

    {

        if (matrix.Count == 0)

            return "Разреженная матрица пуста.";

        var result = "\n";

        foreach (var item in matrix)

        {

            result += $"Координаты: ({item.Key.x}, {item.Key.y}, {item.Key.z}), Значение: {item.Value}\n";

        }

        return result;

    }

}

public class Cube

{

    public int Size { get; set; }

    private SparseMatrix ver;

    public Cube(int size)

    {

        Size = size;

        ver = new SparseMatrix();

        InitVer();

    }

    private void InitVer()

    {

        ver.SetValue(0, 0, 0, 1.0);

        ver.SetValue(Size, 0, 0, 1.0);

        ver.SetValue(Size, Size, 0, 1.0);

        ver.SetValue(0, Size, 0, 1.0);

        ver.SetValue(0, 0, Size, 1.0);

        ver.SetValue(Size, 0, Size, 1.0);

        ver.SetValue(Size, Size, Size, 1.0);

        ver.SetValue(0, Size, Size, 1.0);

    }

    public void PrintVer()

    {

        Console.WriteLine("\nВершины куба:");

        Console.WriteLine(ver.ToString());

    }

}

public class SimpleStack<T>

{

    private LinkedList<T> list = new LinkedList<T>();

    public T Pop()

    {

        if (list.Count == 0)

        {

            throw new InvalidOperationException("пустой стек"); //проверка на пустой стек

        }

        T element = list.First.Value;

        list.RemoveFirst();

        return element;

    }

    public void Push(T element)

    {

        list.AddFirst(element);

    }

}

Результат:

