

**Московский государственный технический  
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»  
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3

Выполнил:  
Студент группы ИУ5-31Б  
Чоботов Лука

Проверил:  
Преподаватель каф. ИУ5  
Гапанюк Ю. Е.

Москва, 2025 г.

## Задание:

Разработать программу, реализующую работу с коллекциями.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создать объекты классов «Прямоугольник», «Квадрат», «Круг».
3. Для реализации возможности сортировки геометрических фигур для класса «Геометрическая фигура» добавить реализацию интерфейса `Comparable`. Сортировка производится по площади фигуры.
4. Создать коллекцию класса `ArrayList`. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.
5. Создать коллекцию класса `List<Figure>`. Сохранить объекты в коллекцию. Отсортировать коллекцию. Вывести в цикле содержимое коллекции.
6. Модифицировать класс разреженной матрицы (проект `SparseMatrix`) для работы с тремя измерениями –  $x, y, z$ . Вывод элементов в методе `ToString()` осуществлять в том виде, который Вы считаете наиболее удобным. Разработать пример использования разреженной матрицы для геометрических фигур.
7. Реализовать класс «`SimpleStack`» на основе односвязного списка. Класс `SimpleStack` наследуется от класса `SimpleList` (проект `SimpleListProject`). Необходимо добавить в класс методы:
  - `public void Push(T element)` – добавление в стек;
  - `public T Pop()` – чтение с удалением из стека.
8. Пример работы класса `SimpleStack` реализовать на основе геометрических фигур.

## Листинг программы:

### Program.cs:

```
using System.Collections;

public class Program
{
    public static void Main(string[] args)
    {
        var rect = new Rectangle(5, 10);
        var square = new Square(7);
        var circle = new Circle(4);

        var arrayList = new ArrayList { rect, square, circle };

        var list = new System.Collections.Generic.List<Figure> { rect, square, circle };
```

```
Console.WriteLine("Коллекция до сортировки:");
foreach (var fig in list)
{
    Console.WriteLine(fig);
}

list.Sort();

Console.WriteLine("\nКоллекция после сортировки по площади:");
foreach (var fig in list)
{
    Console.WriteLine(fig);
}

var matrix = new SparseMatrix<Figure>();

matrix[0, 0, 1] = new Circle(3);
matrix[1, 2, 3] = new Square(5);
matrix[10, 5, 0] = new Rectangle(2, 4);

Console.WriteLine(matrix);

var stack = new Stack<Figure>();

var fig1 = new Circle(2);
var fig2 = new Square(3);
var fig3 = new Rectangle(2, 5);

stack.Push(fig1);
Console.WriteLine($"Push: {fig1}");
stack.Push(fig2);
Console.WriteLine($"Push: {fig2}");
stack.Push(fig3);
Console.WriteLine($"Push: {fig3}");

Console.WriteLine("\nСодержимое стека:");
Console.WriteLine(stack);
```

```

        try
        {
            Console.WriteLine($"Pop: {stack.Pop()}");
            Console.WriteLine($"Pop: {stack.Pop()}");
            Console.WriteLine($"Pop: {stack.Pop()}");
            Console.WriteLine($"Pop: {stack.Pop()}");
        }
        catch (InvalidOperationException ex)
        {
            Console.WriteLine($"Ошибка: {ex.Message}");
        }
    }
}

```

## Figure.cs:

```

public interface IPrint
{
    void Print();
}

public abstract class Figure : IComparable<Figure>
{
    public abstract double Area();

    public int CompareTo(Figure? other)
    {
        if (other == null)
        {
            return 1;
        }
        return this.Area().CompareTo(other.Area());
    }
}

```

## Rectangle.cs:

```

public class Rectangle : Figure, IPrint
{
    public double Width { get; protected set; }
}

```

```

public double Height { get; protected set; }

public Rectangle(double width, double height)
{
    if (width <= 0 || height <= 0)
    {
        throw new ArgumentException("Ширина и высота должны быть положительными числами.");
    }
    Width = width;
    Height = height;
}

public override double Area()
{
    return Width * Height;
}

public override string ToString()
{
    return $"Прямоугольник (ширина: {Width}, высота: {Height}), площадь: {Area():F2}";
}

public void Print()
{
    Console.WriteLine(this.ToString());
}
}

```

## Square.cs:

```

public class Square : Rectangle, IPrint
{
    public Square(double side) : base(side, side) {}

    public override string ToString()
    {
        return $"Квадрат (сторона: {Width}), площадь: {Area():F2}";
    }
}

```

## Circle.cs:

```
public class Circle : Figure, IPrint
{
    public double Radius { get; }

    public Circle(double radius)
    {
        if (radius <= 0)
        {
            throw new ArgumentException("Радиус должен быть положительным числом.");
        }
        Radius = radius;
    }

    public override double Area()
    {
        return Math.PI * Radius * Radius;
    }

    public override string ToString()
    {
        return $"Круг (радиус: {Radius}), площадь: {Area():F2}";
    }

    public void Print()
    {
        Console.WriteLine(this.ToString());
    }
}
```

## SparseMatrix.cs:

```
public class SparseMatrix<T>
{
    private readonly Dictionary<int, Dictionary<int, Dictionary<int, T>>> _matrix = new();

    public T? this[int x, int y, int z]
    {
        get
        {
            {
```

```

        if (_matrix.TryGetValue(x, out var plane) &&
            plane.TryGetValue(y, out var row) &&
            row.TryGetValue(z, out var value))
        {
            return value;
        }

        return default;
    }

    set
    {
        if (value != null)
        {
            if (!_matrix.ContainsKey(x)) _matrix[x] = new();
            if (!_matrix[x].ContainsKey(y)) _matrix[x][y] = new();
            _matrix[x][y][z] = value;
        }
        else
        {
            if (_matrix.TryGetValue(x, out var plane) &&
                plane.TryGetValue(y, out var row))
            {
                row.Remove(z);
                if (row.Count == 0) plane.Remove(y);
                if (plane.Count == 0) _matrix.Remove(x);
            }
        }
    }
}

public override string ToString()
{
    var sb = new System.Text.StringBuilder();
    sb.AppendLine("\nСодержимое разреженной матрицы: ");

    if (_matrix.Count == 0)
    {
        return sb.AppendLine("Матрица пуста.").ToString();
    }

    foreach (var x in _matrix.Keys.OrderBy(k => k))

```

```

        {
            foreach (var y in _matrix[x].Keys.OrderBy(k => k))
            {
                foreach (var z in _matrix[x][y].Keys.OrderBy(k => k))
                {
                    sb.AppendLine($"x:{x}, y:{y}, z:{z}] = {_matrix[x][y][z]}");
                }
            }
        }
        return sb.ToString();
    }
}

```

## Stack.cs:

```

public class Node<T>
{
    public T Data { get; set; }
    public Node<T>? Next { get; set; }

    public Node(T data)
    {
        Data = data;
    }
}

public class List<T>
{
    protected Node<T>? head;

    public void AddFirst(T item)
    {
        var newNode = new Node<T>(item) { Next = head };
        head = newNode;
    }

    public T RemoveFirst()
    {
        if (head == null)
        {

```

```

        throw new ArgumentException("Список пуст.");
    }

    T data = head.Data;
    head = head.Next;
    return data;
}

public override string ToString()
{
    var sb = new System.Text.StringBuilder();
    var current = head;
    while (current != null)
    {
        if (current.Data != null) sb.AppendLine(current.Data.ToString());
        current = current.Next;
    }
    return sb.ToString();
}
}

public class Stack<T> : List<T>
{
    public void Push(T element)
    {
        AddFirst(element);
    }

    public T Pop()
    {
        if (head == null)
        {
            throw new ArgumentException("Стек пуст.");
        }
        return RemoveFirst();
    }
}

```

## Результат выполнения:

```
lukachobotov@iMac-Luka C#Proj % dotnet run
Коллекция до сортировки:
Прямоугольник (ширина: 5, высота: 10), площадь: 50,00
Квадрат (сторона: 7), площадь: 49,00
Круг (радиус: 4), площадь: 50,27

Коллекция после сортировки по площади:
Квадрат (сторона: 7), площадь: 49,00
Прямоугольник (ширина: 5, высота: 10), площадь: 50,00
Круг (радиус: 4), площадь: 50,27

Содержимое разреженной матрицы:
[x:0, y:0, z:1] = Круг (радиус: 3), площадь: 28,27
[x:1, y:2, z:3] = Квадрат (сторона: 5), площадь: 25,00
[x:10, y:5, z:0] = Прямоугольник (ширина: 2, высота: 4), площадь: 8,00

Push: Круг (радиус: 2), площадь: 12,57
Push: Квадрат (сторона: 3), площадь: 9,00
Push: Прямоугольник (ширина: 2, высота: 5), площадь: 10,00

Содержимое стека:
Прямоугольник (ширина: 2, высота: 5), площадь: 10,00
Квадрат (сторона: 3), площадь: 9,00
Круг (радиус: 2), площадь: 12,57

Pop: Прямоугольник (ширина: 2, высота: 5), площадь: 10,00
Pop: Квадрат (сторона: 3), площадь: 9,00
Pop: Круг (радиус: 2), площадь: 12,57
Unhandled exception. System.ArgumentException: Стек пуст.
```