Практическое задание 1

Описать класс *Triangle* (*Треугольник*) со скрытыми полями *a, b, c* – длины 3-х сторон. Определить в нем:

- конструктор, принимающий поля класса;
- свойства;
- метод, выводящий информацию об объекте;
- метод, выводящий тип треугольника по сторонам (разносторонний, равнобедренный, равносторонний).

Написать программу, использующую этот класс и методы.

```
using System;

// Класс Треугольник
class Triangle
{
    // Поля для хранения сторон треугольника
    private double a, b, c;

    // Конструктор для установки сторон треугольника
    public Triangle(double sideA, double sideB, double sideC)
    {
        a = sideA; // Устанавливаем значение первой стороны
        b = sideB; // Устанавливаем значение второй стороны
        c = sideC; // Устанавливаем значение третьей стороны
    }

    // Метод для вывода информации о треугольнике
    public void DisplayInfo()
    {
        Console.WriteLine($"Стороны треугольника: a = {a}, b = {b}, c = {c}"); // Выводим длины сторон треугольника
        Console.WriteLine($"Тип треугольника: {GetTriangleType()}"); // Выводим тип треугольника
    }

    // Метод для определения типа треугольника
    public string GetTriangleType()
    {
        if (a == b && b == c) // Проверяем, равны ли все стороны
            return "Равносторонний"; // Если равны, возвращаем "Равносторонний"
        if (a == b || b == c || a == c) // Проверяем, равны ли две стороны
            return "Равнобедренный"; // Если равны две стороны, возвращаем "Равнобедренный"
        return "Разносторонний"; // В остальных случаях возвращаем "Разносторонний"
    }
}

// Главная программа
class Program
{
    static void Main(string[] args)
    {
        // Создаем треугольник с заданными сторонами
        Triangle triangle = new Triangle(3, 4, 5); // Создаем объект треугольника с длинами сторон 3, 4 и 5
        triangle.DisplayInfo(); // Вызываем метод для отображения информации о треугольнике
    }
}
```

Практическое задание 2

Описать класс *Product* (*Товар*) со скрытыми полями: наименование, цена, количество. Определить в нем:

- конструктор, принимающий поля класса;
- свойства;
- метод, выводящий информацию об объекте;
- метод, возвращающий стоимость товара.

Написать программу, использующую этот класс и методы.

```
using System;

// Класс Товар
class Product
{
    // Поля для хранения информации о товаре
    private string name; // Наименование товара
    private double price; // Цена товара
```

```
    private int quantity; // Количество товара

    // Конструктор для инициализации полей
    public Product(string name, double price, int quantity)
    {
        this.name = name; // Устанавливаем наименование товара
        this.price = price; // Устанавливаем цену товара
        this.quantity = quantity; // Устанавливаем количество товара
    }

    // Свойства для доступа к полям
    public string Name
    {
        get { return name; } // Получение наименования
        set { name = value; } // Установка наименования
    }

    public double Price
    {
        get { return price; } // Получение цены
        set { price = value; } // Установка цены
    }

    public int Quantity
    {
        get { return quantity; } // Получение количества
        set { quantity = value; } // Установка количества
    }

    // Метод для вывода информации о товаре
    public void DisplayInfo()
    {
        Console.WriteLine($"Товар: {name}, Цена: {price}, Количество: {quantity}"); // Выводим информацию о товаре
    }

    // Метод для расчета общей стоимости товара
    public double GetTotalCost()
    {
        return price * quantity; // Умножаем цену на количество
    }
}

// Главная программа
class Program
{
    static void Main(string[] args)
    {
        // Создаем объект товара с заданными параметрами
        Product product = new Product("Ноутбук", 50000, 2); // Товар: Ноутбук, Цена: 50000, Количество: 2

        // Выводим информацию о товаре
        product.DisplayInfo(); // Метод выводит наименование, цену и количество

        // Выводим общую стоимость товара
        Console.WriteLine($"Общая стоимость: {product.GetTotalCost()} руб."); // Рассчитываем и выводим стоимость
    }
}
```

Практическое задание 3

Описать класс, соответствующий заданию. Создать массив из *N* объектов данного класса. При подсчете объектов использовать статические поля. При необходимости написать методы, реализующие операции с экземплярами класса. Написать программу решения задачи.

Ввести информацию по *N* школьникам (поля: Ф.И.О, пол, год рождения). Определить количество мальчиков и девочек. Вывести список каждых.

```
using System;

class Student
{
    // Статические поля для подсчета количества мальчиков и девочек
    public static int boysCount = 0;
    public static int girlsCount = 0;
```

```
    // Поля для хранения Ф.И.О, пола и года рождения
    public string FullName { get; set; }
    public string Gender { get; set; }
    public int YearOfBirth { get; set; }

    // Конструктор для создания объекта Student
    public Student(string fullName, string gender, int yearOfBirth)
    {
        FullName = fullName;
        Gender = gender;
        YearOfBirth = yearOfBirth;

        // Увеличиваем счетчики мальчиков и девочек в зависимости от пола
        if (gender.ToLower() == "мужской")
        {
            boysCount++;
        }
        else if (gender.ToLower() == "женский")
        {
            girlsCount++;
        }
    }

    // Метод для вывода информации о студенте
    public void PrintStudentInfo()
    {
        Console.WriteLine($"Ф.И.О.: {FullName}, Пол: {Gender}, Год рождения: {YearOfBirth}");
    }
}

class Program
{
    static void Main()
    {
        Console.Write("Введите количество школьников: ");
        int N = int.Parse(Console.ReadLine());  // Читаем количество школьников

        // Массив для хранения N объектов класса Student
        Student[] students = new Student[N];

        // Вводим данные о школьниках
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"\nШкольник #{i + 1}:");

            // Вводим Ф.И.О.
            Console.Write("Введите Ф.И.О. школьника: ");
            string fullName = Console.ReadLine();

            // Вводим пол
            Console.Write("Введите пол (мужской/женский): ");
            string gender = Console.ReadLine();

            // Вводим год рождения
            Console.Write("Введите год рождения: ");
            int yearOfBirth = int.Parse(Console.ReadLine());

            // Создаем объект и сохраняем его в массив
            students[i] = new Student(fullName, gender, yearOfBirth);
        }

        // Выводим количество мальчиков и девочек
        Console.WriteLine($"\nКоличество мальчиков: {Student.boysCount}");
        Console.WriteLine($"Количество девочек: {Student.girlsCount}");

        // Выводим списки мальчиков и девочек
        Console.WriteLine("\nСписок мальчиков:");
        foreach (var student in students)
        {
            if (student.Gender.ToLower() == "мужской")
            {
                student.PrintStudentInfo();
            }
        }
```

```
        Console.WriteLine("\nСписок девочек:");
        foreach (var student in students)
        {
            if (student.Gender.ToLower() == "женский")
            {
                student.PrintStudentInfo();
            }
        }
    }
}
```

Практическое задание 4

Описать класс, соответствующий заданию. Создать массив из *N* объектов данного класса. При подсчете объектов использовать статические поля. При необходимости написать методы, реализующие операции с экземплярами класса. Написать программу решения задачи.

Ввести информацию по *N* треугольникам (поля: три угла треугольника). Определить количество остроугольных, прямоугольных и тупоугольных треугольников. Вывести список каждых.

```
using System;

class Triangle
{
    // Статические поля для подсчета типов треугольников
    public static int acuteCount = 0;  // Количество остроугольных
    public static int rightCount = 0;  // Количество прямоугольных
    public static int obtuseCount = 0; // Количество тупоугольных

    // Поля для хранения углов треугольника
    public int Angle1 { get; set; }
    public int Angle2 { get; set; }
    public int Angle3 { get; set; }

    // Конструктор для создания объекта Triangle
    public Triangle(int angle1, int angle2, int angle3)
    {
        Angle1 = angle1;
        Angle2 = angle2;
        Angle3 = angle3;

        // Проверка на тип треугольника
        if (angle1 == 90 || angle2 == 90 || angle3 == 90)
        {
            rightCount++;  // Прямоугольный треугольник
        }
        else if (angle1 < 90 && angle2 < 90 && angle3 < 90)
        {
            acuteCount++;  // Остроугольный треугольник
        }
        else
        {
            obtuseCount++;  // Тупоугольный треугольник
        }
    }

    // Метод для вывода информации о треугольнике
    public void PrintTriangleInfo()
    {
        Console.WriteLine($"Углы: {Angle1}, {Angle2}, {Angle3}");
    }
}

class Program
{
    static void Main()
    {
        Console.Write("Введите количество треугольников: ");
        int N = int.Parse(Console.ReadLine());  // Читаем количество треугольников

        // Массив для хранения N объектов класса Triangle
        Triangle[] triangles = new Triangle[N];
```

```
        // Вводим данные о треугольниках
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"\nТреугольник #{i + 1}:");

            // Вводим углы треугольника
            Console.Write("Введите угол 1: ");
            int angle1 = int.Parse(Console.ReadLine());

            Console.Write("Введите угол 2: ");
            int angle2 = int.Parse(Console.ReadLine());

            Console.Write("Введите угол 3: ");
            int angle3 = int.Parse(Console.ReadLine());

            // Создаем объект и сохраняем его в массив
            triangles[i] = new Triangle(angle1, angle2, angle3);
        }

        // Выводим количество треугольников каждого типа
        Console.WriteLine($"\nКоличество остроугольных треугольников: {Triangle.acuteCount}");
        Console.WriteLine($"Количество прямоугольных треугольников: {Triangle.rightCount}");
        Console.WriteLine($"Количество тупоугольных треугольников: {Triangle.obtuseCount}");

        // Выводим списки треугольников каждого типа
        Console.WriteLine("\nСписок остроугольных треугольников:");
        foreach (var triangle in triangles)
        {
            if (triangle.Angle1 < 90 && triangle.Angle2 < 90 && triangle.Angle3 < 90)
            {
                triangle.PrintTriangleInfo();
            }
        }

        Console.WriteLine("\nСписок прямоугольных треугольников:");
        foreach (var triangle in triangles)
        {
            if (triangle.Angle1 == 90 || triangle.Angle2 == 90 || triangle.Angle3 == 90)
            {
                triangle.PrintTriangleInfo();
            }
        }

        Console.WriteLine("\nСписок тупоугольных треугольников:");
        foreach (var triangle in triangles)
        {
            if (triangle.Angle1 > 90 || triangle.Angle2 > 90 || triangle.Angle3 > 90)
            {
                triangle.PrintTriangleInfo();
            }
        }
    }
}
```

Практическое задание 5

Описать класс, соответствующий заданию, содержащий поля, свойства, конструктор с параметрами. При необходимости вложить в свойства дополнительную логику проверки значений и написать методы, реализующие операции с экземплярами класса. Создать массив из *N* объектов данного класса. Написать программу решения задачи.

Ввести информацию по *N* перевозкам (№ рейса, пункт назначения, вес). Вывести сведения о перевозке с минимальным весом. Найти суммарный объем всех перевозок.

```
using System;

class Transport
{
    // Поля для хранения данных
    public int FlightNumber;
    public string Destination;
    public double Weight;

    // Конструктор для инициализации данных
```

```
    public Transport(int flightNumber, string destination, double weight)
    {
        FlightNumber = flightNumber;
        Destination = destination;
        Weight = weight;
    }
}

class Program
{
    static void Main()
    {
        // Вводим количество перевозок
        Console.Write("Введите количество перевозок: ");
        int N = int.Parse(Console.ReadLine());  // Читаем количество перевозок

        Transport[] transports = new Transport[N];  // Создаем массив для хранения перевозок

        // Вводим данные для каждой перевозки
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"\nПеревозка #{i + 1}:");

            // Вводим номер рейса
            Console.Write("Введите номер рейса: ");
            int flightNumber = int.Parse(Console.ReadLine());

            // Вводим пункт назначения
            Console.Write("Введите пункт назначения: ");
            string destination = Console.ReadLine();

            // Вводим вес перевозки
            Console.Write("Введите вес (кг): ");
            double weight = double.Parse(Console.ReadLine());

            // Создаем новый объект перевозки и добавляем его в массив
            transports[i] = new Transport(flightNumber, destination, weight);
        }

        // Находим перевозку с минимальным весом
        Transport minWeightTransport = transports[0];  // Изначально минимальный вес у первой перевозки
        for (int i = 1; i < N; i++)
        {
            // Если вес текущей перевозки меньше, обновляем минимальный вес
            if (transports[i].Weight < minWeightTransport.Weight)
            {
                minWeightTransport = transports[i];
            }
        }

        // Выводим информацию о перевозке с минимальным весом
        Console.WriteLine($"\nПеревозка с минимальным весом:");
        Console.WriteLine($"№ рейса: {minWeightTransport.FlightNumber}, Пункт назначения: {minWeightTransport.Destination}, Вес: {minWeightTransport.Weight} кг");

        // Вычисляем суммарный вес всех перевозок
        double totalWeight = 0;  // Переменная для накопления общего веса
        for (int i = 0; i < N; i++)
        {
            totalWeight += transports[i].Weight;  // Прибавляем вес каждой перевозки
        }

        // Выводим суммарный вес всех перевозок
        Console.WriteLine($"\nСуммарный вес всех перевозок: {totalWeight} кг");
    }
}
```

Практическое задание 6

Составить программу с одним родительским классом и потомком. Базовый класс должен содержать поля/свойства, конструктор с параметрами, виртуальный метод вывода информации об объекте и указанный в задании метод. Производный класс содержит дополнения и изменения, переопределенный метод вывода информации о потомке. Создать объекты базового и производного классов. В программе должна выполняться проверка всех разработанных элементов класса.

Базовый класс: *Стол* (поля: название, площадь S в м^2).
Метод: Стоимость C=50+100*S.
Потомок: *Письменный стол* (поля: используемый материал, стоимость отделки).
Изменения в потомке: Изменить стоимость с учетом отделки.

```
using System;

class Table
{
    // Поля базового класса
    public string Name { get; set; }  // Название стола
    public double Area { get; set; }  // Площадь стола в м2

    // Конструктор с параметрами для инициализации
    public Table(string name, double area)
    {
        Name = name;
        Area = area;
    }

    // Виртуальный метод для расчета стоимости
    public virtual double CalculateCost()
    {
        return 50 + 100 * Area;  // Формула стоимости: C = 50 + 100 * S
    }

    // Виртуальный метод для вывода информации о столе
    public virtual void DisplayInfo()
    {
        Console.WriteLine($"Стол: {Name}");
        Console.WriteLine($"Площадь: {Area} м²");
        Console.WriteLine($"Стоимость: {CalculateCost()} руб.");
    }
}

class WritingTable : Table
{
    // Дополнительные поля производного класса
    public string Material { get; set; }  // Материал, из которого сделан стол
    public double FinishingCost { get; set; }  // Стоимость отделки

    // Конструктор с параметрами, включая параметры базового класса
    public WritingTable(string name, double area, string material, double finishingCost)
        : base(name, area)  // Вызов конструктора базового класса
    {
        Material = material;
        FinishingCost = finishingCost;
    }

    // Переопределенный метод для расчета стоимости
    public override double CalculateCost()
    {
        // Стоимость стола с учетом отделки
        return base.CalculateCost() + FinishingCost;
    }

    // Переопределенный метод для вывода информации о письменном столе
    public override void DisplayInfo()
    {
        base.DisplayInfo();  // Выводим информацию из базового класса
        Console.WriteLine($"Материал: {Material}");
        Console.WriteLine($"Стоимость отделки: {FinishingCost} руб.");
        Console.WriteLine($"Итоговая стоимость (с учетом отделки): {CalculateCost()} руб.");
    }
}

class Program
{
    static void Main()
    {
        // Создание объекта базового класса
        Table table = new Table("Обычный стол", 4.5);
        Console.WriteLine("Информация о базовом столе:");
        table.DisplayInfo();
```

```
        Console.WriteLine();

        // Создание объекта производного класса
        WritingTable writingTable = new WritingTable("Письменный стол", 6.0, "Дерево", 1500);
        Console.WriteLine("Информация о письменном столе:");
        writingTable.DisplayInfo();
    }
}
```

Практическое задание 7

Составить программу с одним родительским классом и потомком. Базовый класс должен содержать поля/свойства, конструктор с параметрами, виртуальный метод вывода информации об объекте и указанный в задании метод. Производный класс содержит дополнения и изменения, переопределенный метод вывода информации о потомке. Создать объекты базового и производного классов. В программе должна выполняться проверка всех разработанных элементов класса.

Базовый класс: *Автобус* (поля: марка, количество мест, стоимость билета).

Метод: Общая стоимость всех мест.

Потомок: *Туристический автобус* (поле: стоимость экскурсии).

Изменения в потомке: Общая стоимость всех мест с учетом увеличения цены билета за экскурсию.

```
using System;

class Bus
{
    // Поля базового класса
    public string Brand { get; set; }  // Марка автобуса
    public int Seats { get; set; }  // Количество мест
    public double TicketPrice { get; set; }  // Стоимость билета

    // Конструктор с параметрами для инициализации
    public Bus(string brand, int seats, double ticketPrice)
    {
        Brand = brand;
        Seats = seats;
        TicketPrice = ticketPrice;
    }

    // Метод для расчета общей стоимости всех мест
    public virtual double CalculateTotalCost()
    {
        return Seats * TicketPrice;  // Общая стоимость = количество мест * стоимость билета
    }

    // Метод для вывода информации о автобусе
    public virtual void DisplayInfo()
    {
        Console.WriteLine($"Марка автобуса: {Brand}");
        Console.WriteLine($"Количество мест: {Seats}");
        Console.WriteLine($"Стоимость билета: {TicketPrice} руб.");
        Console.WriteLine($"Общая стоимость всех мест: {CalculateTotalCost()} руб.");
    }
}

class TouristBus : Bus
{
    // Дополнительное поле для производного класса
    public double ExcursionPrice { get; set; }  // Стоимость экскурсии

    // Конструктор с параметрами для инициализации, включая параметры базового класса
    public TouristBus(string brand, int seats, double ticketPrice, double excursionPrice)
        : base(brand, seats, ticketPrice)  // Вызов конструктора базового класса
    {
        ExcursionPrice = excursionPrice;
    }

    // Переопределенный метод для расчета общей стоимости с учетом экскурсии
    public override double CalculateTotalCost()
    {
        // Увеличиваем стоимость билета с учетом экскурсии
        double newTicketPrice = TicketPrice + ExcursionPrice;
        return Seats * newTicketPrice;  // Общая стоимость = количество мест * (стоимость билета + стоимость экскурсии)
```

```
    }

    // Переопределенный метод для вывода информации о туристическом автобусе
    public override void DisplayInfo()
    {
        base.DisplayInfo();  // Выводим информацию из базового класса
        Console.WriteLine($"Стоимость экскурсии: {ExcursionPrice} руб.");
        Console.WriteLine($"Общая стоимость всех мест с экскурсионной платой: {CalculateTotalCost()} руб.");
    }
}

class Program
{
    static void Main()
    {
        // Создание объекта базового класса
        Bus bus = new Bus("Mercedes", 40, 150);  // Марка, количество мест, стоимость билета
        Console.WriteLine("Информация о базовом автобусе:");
        bus.DisplayInfo();
        Console.WriteLine();

        // Создание объекта производного класса
        TouristBus touristBus = new TouristBus("Volvo", 50, 200, 100);  // Марка, количество мест, стоимость билета, стоимость экскурсии
        Console.WriteLine("Информация о туристическом автобусе:");
        touristBus.DisplayInfo();
    }
}
```

Практическое задание 8

Составить программу с абстрактным родительским классом и двумя потомками. Потомки должны содержать переопределенные методы (метод, указанный в таблице, и метод с описанием объекта). Создать объекты производных классов. В программе должна выполняться проверка всех разработанных элементов класса.

Абстрактный родительский класс	Потомки	Переопределенный метод
Населенный пункт (поле: название)	Село (поля: количество домов h, среднее число жителей в доме, площадь села) Город (поля: количество жителей h, площадь города)	Плотность населения

```
using System;

// Абстрактный класс "Населённый пункт"
abstract class Settlement
{
    // Поле для хранения названия населённого пункта
    public string Name { get; set; }

    // Конструктор для инициализации названия
    public Settlement(string name)
    {
        Name = name;
    }

    // Абстрактный метод для вычисления плотности населения (реализация будет зависеть от потомка)
    public abstract double PopulationDensity();

    // Абстрактный метод для вывода описания объекта
    public abstract void Describe();
}

// Класс "Село" — потомок абстрактного класса Settlement
class Village : Settlement
{
    // Поля для села
    public int NumberOfHouses { get; set; }  // Количество домов
    public double AveragePeoplePerHouse { get; set; }  // Среднее число жителей в доме
    public double Area { get; set; }  // Площадь села

    // Конструктор с параметрами для инициализации
    public Village(string name, int numberOfHouses, double averagePeoplePerHouse, double area)
        : base(name)  // Вызов конструктора родительского класса
```

```
    {
        NumberOfHouses = numberOfHouses;
        AveragePeoplePerHouse = averagePeoplePerHouse;
        Area = area;
    }

    // Переопределённый метод для вычисления плотности населения в селе
    public override double PopulationDensity()
    {
        double totalPopulation = NumberOfHouses * AveragePeoplePerHouse;
        return totalPopulation / Area;  // Плотность населения = общее количество людей / площадь
    }

    // Переопределённый метод для вывода описания объекта
    public override void Describe()
    {
        Console.WriteLine($"Село {Name}:");
        Console.WriteLine($"Количество домов: {NumberOfHouses}");
        Console.WriteLine($"Среднее количество жителей в доме: {AveragePeoplePerHouse}");
        Console.WriteLine($"Площадь села: {Area} кв. км");
    }
}

// Класс "Город" — потомок абстрактного класса Settlement
class City : Settlement
{
    // Поля для города
    public int NumberOfResidents { get; set; }  // Количество жителей
    public double Area { get; set; }  // Площадь города

    // Конструктор с параметрами для инициализации
    public City(string name, int numberOfResidents, double area)
        : base(name)  // Вызов конструктора родительского класса
    {
        NumberOfResidents = numberOfResidents;
        Area = area;
    }

    // Переопределённый метод для вычисления плотности населения в городе
    public override double PopulationDensity()
    {
        return NumberOfResidents / Area;  // Плотность населения = количество жителей / площадь
    }

    // Переопределённый метод для вывода описания объекта
    public override void Describe()
    {
        Console.WriteLine($"Город {Name}:");
        Console.WriteLine($"Количество жителей: {NumberOfResidents}");
        Console.WriteLine($"Площадь города: {Area} кв. км");
    }
}

class Program
{
    static void Main()
    {
        // Создание объектов для каждого типа населённого пункта
        Village village = new Village("Greenfield", 120, 4.5, 50.0);
        City city = new City("Metropolis", 2000000, 300.0);

        // Вывод информации о селе
        village.Describe();
        Console.WriteLine($"Плотность населения: {village.PopulationDensity()} человек/кв. км");
        Console.WriteLine();

        // Вывод информации о городе
        city.Describe();
        Console.WriteLine($"Плотность населения: {city.PopulationDensity()} человек/кв. км");
    }
}
```

Практическое задание 9

Описать интерфейс ISolid для геометрических тел. Интерфейс должен содержать методы: Volume и SurfaceArea, возвращающие объём и площадь поверхности соответственно.
Описать классы Cube (куб) и Cylinder (цилиндр), реализующие этот интерфейс. Параметры тел должны задаваться при создании экземпляра.
Написать метод, принимающий тело и выводящий на экран её название, параметры, объём и площадь поверхности.
Написать программу, использующую этот метод.

```
using System;

// Интерфейс ISolid с методами Volume и SurfaceArea
public interface ISolid
{
    // Метод для вычисления объёма
    double Volume();

    // Метод для вычисления площади поверхности
    double SurfaceArea();
}

// Класс Cube (куб), реализующий интерфейс ISolid
public class Cube : ISolid
{
    // Сторона куба
    public double Side { get; set; }

    // Конструктор для инициализации параметров куба
    public Cube(double side)
    {
        Side = side;
    }

    // Реализация метода Volume для куба
    public double Volume()
    {
        return Math.Pow(Side, 3); // Объём куба = сторона в кубе
    }

    // Реализация метода SurfaceArea для куба
    public double SurfaceArea()
    {
        return 6 * Math.Pow(Side, 2); // Площадь поверхности куба = 6 * сторона^2
    }
}

// Класс Cylinder (цилиндр), реализующий интерфейс ISolid
public class Cylinder : ISolid
{
    // Радиус основания цилиндра
    public double Radius { get; set; }

    // Высота цилиндра
    public double Height { get; set; }

    // Конструктор для инициализации параметров цилиндра
    public Cylinder(double radius, double height)
    {
        Radius = radius;
        Height = height;
    }

    // Реализация метода Volume для цилиндра
    public double Volume()
    {
        return Math.PI * Math.Pow(Radius, 2) * Height; // Объём цилиндра = π * r^2 * h
    }

    // Реализация метода SurfaceArea для цилиндра
    public double SurfaceArea()
    {
        return 2 * Math.PI * Radius * (Radius + Height); // Площадь поверхности цилиндра = 2πr(h + r)
    }
}
```

```
// Класс с методом, который выводит информацию о теле
public class SolidPrinter
{
    // Метод, принимающий тело, выводящий его параметры, объём и площадь поверхности
    public static void PrintSolidInfo(ISolid solid)
    {
        if (solid is Cube cube)
        {
            // Если объект — это куб
            Console.WriteLine("Это Куб:");
            Console.WriteLine($"Сторона: {cube.Side} см");
            Console.WriteLine($"Объём: {cube.Volume()} куб. см");
            Console.WriteLine($"Площадь поверхности: {cube.SurfaceArea()} кв. см");
        }
        else if (solid is Cylinder cylinder)
        {
            // Если объект — это цилиндр
            Console.WriteLine("Это Цилиндр:");
            Console.WriteLine($"Радиус основания: {cylinder.Radius} см");
            Console.WriteLine($"Высота: {cylinder.Height} см");
            Console.WriteLine($"Объём: {cylinder.Volume()} куб. см");
            Console.WriteLine($"Площадь поверхности: {cylinder.SurfaceArea()} кв. см");
        }
    }
}

class Program
{
    static void Main()
    {
        // Создаём экземпляры классов Cube и Cylinder
        Cube cube = new Cube(5);  // Куб с длиной стороны 5 см
        Cylinder cylinder = new Cylinder(3, 7);  // Цилиндр с радиусом 3 см и высотой 7 см

        // Выводим информацию о кубе
        SolidPrinter.PrintSolidInfo(cube);
        Console.WriteLine();

        // Выводим информацию о цилиндре
        SolidPrinter.PrintSolidInfo(cylinder);
    }
}
```

Практическое задание 10

Описать класс, соответствующий заданию, содержащий поля, свойства, конструктор с параметрами. При необходимости вложить в свойства дополнительную логику проверки значений и написать методы, реализующие операции с экземплярами класса. Создать массив из *N* объектов данного класса. Написать программу решения задачи.

Ввести информацию по *N* спортсменам (Ф.И.О., рост, вес). Вывести сведения о спортсменах, чей вес превышает 70 кг. Определить их количество.

```
using System;

public class Athlete
{
    // Поля
    public string FullName { get; set; }
    public double Height { get; set; }
    public double Weight { get; set; }

    // Конструктор с параметрами
    public Athlete(string fullName, double height, double weight)
    {
        FullName = fullName;
        Height = height;
        Weight = weight;
    }

    // Метод для вывода информации о спортсмене
    public void PrintInfo()
    {
```

```
        Console.WriteLine($"Ф.И.О: {FullName}, Рост: {Height} см, Вес: {Weight} кг");
    }
}

class Program
{
    static void Main()
    {
        // Ввод количества спортсменов
        Console.Write("Введите количество спортсменов: ");
        int N = int.Parse(Console.ReadLine());

        // Создание массива спортсменов
        Athlete[] athletes = new Athlete[N];

        // Ввод данных о спортсменах
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"\nВведите данные для спортсмена {i + 1}:");
            Console.Write("Ф.И.О: ");
            string fullName = Console.ReadLine();
            Console.Write("Рост (в см): ");
            double height = double.Parse(Console.ReadLine());
            Console.Write("Вес (в кг): ");
            double weight = double.Parse(Console.ReadLine());

            // Создание нового объекта Athlete и добавление в массив
            athletes[i] = new Athlete(fullName, height, weight);
        }

        // Переменная для подсчета спортсменов с весом больше 70 кг
        int countAbove70 = 0;

        // Вывод спортсменов, чей вес больше 70 кг
        Console.WriteLine("\nСведения о спортсменах с весом более 70 кг:");
        foreach (var athlete in athletes)
        {
            if (athlete.Weight > 70)
            {
                athlete.PrintInfo(); // Выводим информацию о спортсмене
                countAbove70++; // Увеличиваем счетчик
            }
        }

        // Выводим количество спортсменов с весом больше 70 кг
        Console.WriteLine($"\nКоличество спортсменов с весом более 70 кг: {countAbove70}");
    }
}
```

Практическое задание 11

Используя статический метод, составить программу для решения задачи.
Определить статический метод double Average (int m, int n), который вычисляет и возвращает среднее арифметическое всех целых от *m* до *n* включительно. Используя этот метод, вычислить среднее арифметическое всех чисел от *a* до *b* (*a*, *b* вводятся с клавиатуры).

```
using System;

class Program
{
    // Статический метод для вычисления среднего арифметического
    public static double Average(int m, int n)
    {
        // Если m больше n, меняем их местами для корректной работы
        if (m > n)
        {
            int temp = m;
            m = n;
            n = temp;
        }

        // Вычисляем сумму всех чисел от m до n включительно
        int sum = 0;
        int count = 0; // Для подсчета количества чисел
```

```
        for (int i = m; i <= n; i++)
        {
            sum += i;  // Добавляем текущее число к сумме
            count++;    // Увеличиваем счётчик чисел
        }

        // Возвращаем среднее арифметическое
        return (double)sum / count;
    }

    static void Main()
    {
        // Вводим значения a и b с клавиатуры
        Console.Write("Введите число a: ");
        int a = int.Parse(Console.ReadLine());

        Console.Write("Введите число b: ");
        int b = int.Parse(Console.ReadLine());

        // Вызов статического метода Average для вычисления среднего арифметического
        double result = Average(a, b);

        // Выводим результат
        Console.WriteLine($"Среднее арифметическое чисел от {a} до {b} = {result}");
    }
}
```

Практическое задание 12

Описать класс, соответствующий заданию. Создать массив из *N* объектов данного класса. При подсчете объектов использовать статические поля. При необходимости написать методы, реализующие операции с экземплярами класса. Написать программу решения задачи.

Ввести информацию по *N* треугольникам (поля: стороны треугольника). Определить количество равносторонних, равнобедренных и разносторонних треугольников. Вывести список каждых.

```
using System;
using System.Collections.Generic;

class Triangle
{
    // Стороны треугольника
    public double A { get; set; }
    public double B { get; set; }
    public double C { get; set; }

    // Статическое поле для подсчёта объектов
    public static int CountEquilateral = 0;  // Количество равносторонних треугольников
    public static int CountIsosceles = 0;    // Количество равнобедренных треугольников
    public static int CountScalene = 0;      // Количество разносторонних треугольников

    // Конструктор с параметрами
    public Triangle(double a, double b, double c)
    {
        A = a;
        B = b;
        C = c;

        // Определяем тип треугольника
        if (a == b && b == c)
        {
            CountEquilateral++; // Равносторонний
        }
        else if (a == b || b == c || a == c)
        {
            CountIsosceles++;   // Равнобедренный
        }
        else
        {
            CountScalene++;     // Разносторонний
        }
    }
```

```
    // Метод для вывода информации о треугольнике
    public void PrintInfo()
    {
        Console.WriteLine($"Треугольник: {A}, {B}, {C}");
        if (A == B && B == C)
            Console.WriteLine("Тип: Равносторонний");
        else if (A == B || B == C || A == C)
            Console.WriteLine("Тип: Равнобедренный");
        else
            Console.WriteLine("Тип: Разносторонний");
    }
}

class Program
{
    static void Main()
    {
        // Вводим количество треугольников
        Console.Write("Введите количество треугольников: ");
        int N = int.Parse(Console.ReadLine());

        // Создаём массив для хранения треугольников
        List<Triangle> triangles = new List<Triangle>();

        // Вводим данные о треугольниках
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"Введите данные для треугольника {i + 1}:");
            Console.Write("Сторона A: ");
            double a = double.Parse(Console.ReadLine());
            Console.Write("Сторона B: ");
            double b = double.Parse(Console.ReadLine());
            Console.Write("Сторона C: ");
            double c = double.Parse(Console.ReadLine());

            // Создаём треугольник и добавляем в список
            triangles.Add(new Triangle(a, b, c));
        }

        // Выводим количество каждого типа треугольников
        Console.WriteLine($"\nКоличество равносторонних треугольников: {Triangle.CountEquilateral}");
        Console.WriteLine($"Количество равнобедренных треугольников: {Triangle.CountIsosceles}");
        Console.WriteLine($"Количество разносторонних треугольников: {Triangle.CountScalene}");

        // Выводим информацию о каждом треугольнике
        Console.WriteLine("\nСведения о треугольниках:");
        foreach (var triangle in triangles)
        {
            triangle.PrintInfo();
            Console.WriteLine();
        }
    }
}
```

Практическое задание 13

Описать класс *Parallelogram* (*Параллелограмм*) со скрытыми полями *a, b* – стороны параллелограмма и α – угол между сторонами. Определить в нем:

- конструктор, принимающий поля класса;
- свойства;
- метод, выводящий информацию об объекте;
- метод, выводящий вид четырехугольника (квадрат, прямоугольник, ромб, параллелограмм).

Написать программу, использующую этот класс и методы.

```
using System;

class Parallelogram
{
    // Скрытые поля
    private double a;  // Сторона a
    private double b;  // Сторона b
    private double angle; // Угол между сторонами a и b (в градусах)
```

```
    // Конструктор класса
    public Parallelogram(double a, double b, double angle)
    {
        this.a = a;
        this.b = b;
        this.angle = angle;
    }

    // Свойства для доступа к полям
    public double A
    {
        get { return a; }
        set { a = value; }
    }

    public double B
    {
        get { return b; }
        set { b = value; }
    }

    public double Angle
    {
        get { return angle; }
        set { angle = value; }
    }

    // Метод для вывода информации о параллелограмме
    public void DisplayInfo()
    {
        Console.WriteLine($"Параллелограмм: Сторона a = {a}, Сторона b = {b}, Угол между сторонами = {angle}°");
    }

    // Метод для определения типа четырёхугольника
    public void DetermineType()
    {
        if (a == b && angle == 90)
        {
            Console.WriteLine("Это квадрат.");
        }
        else if (a == b)
        {
            Console.WriteLine("Это ромб.");
        }
        else if (angle == 90)
        {
            Console.WriteLine("Это прямоугольник.");
        }
        else
        {
            Console.WriteLine("Это параллелограмм.");
        }
    }
}

class Program
{
    static void Main()
    {
        // Ввод данных о параллелограмме
        Console.Write("Введите длину стороны a: ");
        double a = double.Parse(Console.ReadLine());

        Console.Write("Введите длину стороны b: ");
        double b = double.Parse(Console.ReadLine());

        Console.Write("Введите угол между сторонами a и b (в градусах): ");
        double angle = double.Parse(Console.ReadLine());

        // Создаём объект параллелограмма
        Parallelogram parallelogram = new Parallelogram(a, b, angle);

        // Выводим информацию о параллелограмме
        parallelogram.DisplayInfo();
```

```
        // Определяем и выводим тип четырёхугольника
        parallelogram.DetermineType();
    }
}
```

Практическое задание 14

Описать класс *Product* (*Товар*) со скрытыми полями: наименование, цена, количество. Определить в нем:

- свойства, соответствующие полям; в случае необходимости вложите в свойства дополнительную логику проверки значений.
- конструктор, принимающий наименование, цену и количество товара;

Создать массив объектов типа **Product**. Ввести информацию по *N* видам товаров. Отсортировать массив по:

1) наименованию товара;
2) цене товара.

Простой вариант

```
using System;

class Product
{
    // Поля класса
    public string Name;
    public double Price;
    public int Quantity;

    // Конструктор для инициализации объекта
    public Product(string name, double price, int quantity)
    {
        Name = name;
        Price = price;
        Quantity = quantity;
    }

    // Метод для вывода информации о товаре
    public void DisplayInfo()
    {
        Console.WriteLine($"Наименование: {Name}, Цена: {Price} руб., Количество: {Quantity} шт.");
    }
}

class Program
{
    static void Main()
    {
        Console.Write("Введите количество товаров: ");
        int n = int.Parse(Console.ReadLine()); // Количество товаров

        Product[] products = new Product[n]; // Массив товаров

        // Ввод данных о товарах
        for (int i = 0; i < n; i++)
        {
            Console.WriteLine($"Товар {i + 1}:");
            Console.Write("Наименование: ");
            string name = Console.ReadLine();

            Console.Write("Цена: ");
            double price = double.Parse(Console.ReadLine());

            Console.Write("Количество: ");
            int quantity = int.Parse(Console.ReadLine());

            // Создание и добавление товара в массив
            products[i] = new Product(name, price, quantity);
        }

        // Сортировка по наименованию
        Array.Sort(products, (p1, p2) => p1.Name.CompareTo(p2.Name));

        // Вывод отсортированных товаров по наименованию
```

```
        Console.WriteLine("\nТовары, отсортированные по наименованию:");
        foreach (var product in products)
        {
            product.DisplayInfo();
        }

        // Сортировка по цене
        Array.Sort(products, (p1, p2) => p1.Price.CompareTo(p2.Price));

        // Вывод отсортированных товаров по цене
        Console.WriteLine("\nТовары, отсортированные по цене:");
        foreach (var product in products)
        {
            product.DisplayInfo();
        }
    }
}
```

Сложный вариант со свойствами

```
using System;

class Product
{
    // Свойства класса
    private string _name;
    private double _price;
    private int _quantity;

    // Свойство для наименования товара
    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }

    // Свойство для цены товара с проверкой
    public double Price
    {
        get { return _price; }
        set
        {
            if (value < 0) // Проверка на отрицательную цену
                throw new ArgumentException("Цена не может быть отрицательной.");
            _price = value;
        }
    }

    // Свойство для количества товара с проверкой
    public int Quantity
    {
        get { return _quantity; }
        set
        {
            if (value < 0) // Проверка на отрицательное количество
                throw new ArgumentException("Количество не может быть отрицательным.");
            _quantity = value;
        }
    }

    // Конструктор для инициализации товара
    public Product(string name, double price, int quantity)
    {
        Name = name;
        Price = price;
        Quantity = quantity;
    }

    // Метод для вывода информации о товаре
    public void DisplayInfo()
    {
        Console.WriteLine($"Наименование: {Name}, Цена: {Price} руб., Количество: {Quantity} шт.");
    }
}
```

```
class Program
{
    static void Main()
    {
        Console.Write("Введите количество товаров: ");
        int n = int.Parse(Console.ReadLine()); // Количество товаров

        Product[] products = new Product[n]; // Массив товаров

        // Ввод данных о товарах
        for (int i = 0; i < n; i++)
        {
            Console.WriteLine($"Товар {i + 1}:");
            Console.Write("Наименование: ");
            string name = Console.ReadLine();

            Console.Write("Цена: ");
            double price = double.Parse(Console.ReadLine());

            Console.Write("Количество: ");
            int quantity = int.Parse(Console.ReadLine());

            try
            {
                // Создание и добавление товара в массив с использованием конструктора
                products[i] = new Product(name, price, quantity);
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine($"Ошибка: {ex.Message}");
                i--; // Повторить ввод текущего товара
            }
        }

        // Сортировка по наименованию
        Array.Sort(products, (p1, p2) => p1.Name.CompareTo(p2.Name));

        // Вывод отсортированных товаров по наименованию
        Console.WriteLine("\nТовары, отсортированные по наименованию:");
        foreach (var product in products)
        {
            product.DisplayInfo();
        }

        // Сортировка по цене
        Array.Sort(products, (p1, p2) => p1.Price.CompareTo(p2.Price));

        // Вывод отсортированных товаров по цене
        Console.WriteLine("\nТовары, отсортированные по цене:");
        foreach (var product in products)
        {
            product.DisplayInfo();
        }
    }
}
```

Практическое задание 15

Описать класс, соответствующий заданию. Создать коллекцию (List) из объектов данного класса. Вывести:

- информацию обо всех объектах;
- информацию об объектах согласно условию задачи.

Создать еще 2 списка, добавив в них объекты согласно условию задачи. Вывести информацию об объектах каждого списка. В случае отсутствия информации об объектах, вывести соответствующее сообщение.

Поля класса	Задача
Наименование Дата производства Срок годности Цена	Вывести наименование товара и конечную дату применения (годен до:). Создать 2 списка: • список товаров, у которых срок годности уже истек • список тех, у которых срок годности заканчивается в течение ближайших 30 суток

```
using System;
using System.Collections.Generic;

class Product
{
    // Поля класса для хранения данных о товаре
    public string Name { get; set; }              // Наименование товара
    public DateTime ManufactureDate { get; set; }  // Дата производства товара
    public DateTime ExpiryDate { get; set; }      // Срок годности товара
    public double Price { get; set; }             // Цена товара

    // Конструктор класса, который инициализирует поля товара
    public Product(string name, DateTime manufactureDate, DateTime expiryDate, double price)
    {
        Name = name;                          // Инициализация наименования товара
        ManufactureDate = manufactureDate;        // Инициализация даты производства
        ExpiryDate = expiryDate;                  // Инициализация срока годности
        Price = price;                        // Инициализация цены товара
    }

    // Метод для вывода информации о товаре: наименование и дата истечения срока годности
    public void DisplayInfo()
    {
        // Выводим наименование товара и дату, до которой он годен
        Console.WriteLine($"Наименование: {Name}, Годен до: {ExpiryDate.ToShortDateString()}");
    }

    // Метод для проверки, истек ли срок годности товара
    public bool IsExpired()
    {
        // Если текущая дата больше срока годности, то товар просрочен
        return DateTime.Now > ExpiryDate;
    }

    // Метод для проверки, заканчивается ли срок годности товара в ближайшие 30 дней
    public bool ExpiringSoon()
    {
        // Если срок годности товара заканчивается в пределах 30 дней от текущей даты
        return ExpiryDate > DateTime.Now && ExpiryDate <= DateTime.Now.AddDays(30);
    }
}

class Program
{
    static void Main()
    {
        // Создаем список товаров
        List<Product> products = new List<Product>
        {
            // Добавляем товары в список с разными датами производства и сроками годности
            new Product("Молоко", new DateTime(2023, 10, 1), new DateTime(2023, 12, 31), 50),
            new Product("Хлеб", new DateTime(2023, 11, 1), new DateTime(2024, 1, 15), 25),
            new Product("Масло", new DateTime(2023, 7, 1), new DateTime(2023, 12, 1), 120),
            new Product("Йогурт", new DateTime(2023, 9, 15), new DateTime(2024, 1, 5), 60)
        };

        // Списки для товаров с истекшим сроком годности и сроком годности в пределах 30 дней
        List<Product> expiredProducts = new List<Product>();
        List<Product> expiringSoonProducts = new List<Product>();

        // Разделяем товары на два списка
        foreach (var product in products)
        {
            if (product.IsExpired())  // Если срок годности товара истек
            {
                expiredProducts.Add(product);  // Добавляем в список истекших товаров
            }
            else if (product.ExpiringSoon())  // Если срок годности заканчивается в ближайшие 30 дней
            {
                expiringSoonProducts.Add(product);  // Добавляем в список товаров, срок годности которых заканчивается
            }
        }

        // Вывод информации обо всех товарах
        Console.WriteLine("Информация обо всех товарах:");
```

```
        foreach (var product in products)
        {
            product.DisplayInfo();  // Выводим информацию о каждом товаре
        }

        // Вывод информации о товарах с истекшим сроком годности
        Console.WriteLine("\nТовары с истекшим сроком годности:");
        if (expiredProducts.Count == 0)
        {
            Console.WriteLine("Нет товаров с истекшим сроком годности.");  // Если список пуст, выводим сообщение
        }
        else
        {
            foreach (var product in expiredProducts)
            {
                product.DisplayInfo();  // Выводим информацию о просроченных товарах
            }
        }

        // Вывод информации о товарах, срок годности которых заканчивается в ближайшие 30 дней
        Console.WriteLine("\nТовары, срок годности которых заканчивается в ближайшие 30 дней:");
        if (expiringSoonProducts.Count == 0)
        {
            Console.WriteLine("Нет товаров, срок годности которых заканчивается в ближайшие 30 дней.");  // Если список пуст, выводим сообщение
        }
        else
        {
            foreach (var product in expiringSoonProducts)
            {
                product.DisplayInfo();  // Выводим информацию о товарах, срок годности которых заканчивается в ближайшие 30 дней
            }
        }
    }
}
```

Практическое задание 16

Описать класс, соответствующий заданию. Создать коллекцию (List) из объектов данного класса. Вывести информацию обо всех объектах. Вывести информацию об объектах согласно заданию. При решении использовать класс Dictionary<K, V>.

Поля класса	Задача
Название фильма Дата и время начала сеанса Продолжительность сеанса Жанр	Вывести информацию о количестве фильмов каждого жанра. Определить, есть ли фильмы определенного жанра (жанр вводится с клавиатуры).

```
using System;
using System.Collections.Generic;

class Movie
{
    // Поля класса для хранения данных о фильме
    public string Title { get; set; }             // Название фильма
    public DateTime StartTime { get; set; }       // Дата и время начала сеанса
    public TimeSpan Duration { get; set; }        // Продолжительность сеанса
    public string Genre { get; set; }             // Жанр фильма

    // Конструктор класса
    public Movie(string title, DateTime startTime, TimeSpan duration, string genre)
    {
        Title = title;                      // Инициализация названия фильма
        StartTime = startTime;                  // Инициализация даты и времени начала
        Duration = duration;                    // Инициализация продолжительности сеанса
        Genre = genre;                      // Инициализация жанра
    }

    // Метод для вывода информации о фильме
    public void DisplayInfo()
    {
        Console.WriteLine($"Название: {Title}, Начало: {StartTime}, Продолжительность: {Duration}, Жанр: {Genre}");
    }
}

class Program
{
```

```
    static void Main()
    {
        // Создаем коллекцию для хранения фильмов с жанрами
        Dictionary<string, List<Movie>> moviesByGenre = new Dictionary<string, List<Movie>>();

        // Добавляем фильмы в коллекцию
        moviesByGenre.Add("Action", new List<Movie>
        {
            new Movie("Die Hard", new DateTime(2024, 12, 27, 18, 30, 0), new TimeSpan(2, 10, 0), "Action"),
            new Movie("Mad Max", new DateTime(2024, 12, 27, 20, 45, 0), new TimeSpan(2, 0, 0), "Action")
        });

        moviesByGenre.Add("Comedy", new List<Movie>
        {
            new Movie("The Hangover", new DateTime(2024, 12, 27, 16, 0, 0), new TimeSpan(1, 50, 0), "Comedy"),
            new Movie("Superbad", new DateTime(2024, 12, 27, 18, 30, 0), new TimeSpan(1, 50, 0), "Comedy")
        });

        moviesByGenre.Add("Drama", new List<Movie>
        {
            new Movie("The Shawshank Redemption", new DateTime(2024, 12, 27, 19, 0, 0), new TimeSpan(2, 20, 0), "Drama"),
            new Movie("Forrest Gump", new DateTime(2024, 12, 27, 21, 30, 0), new TimeSpan(2, 20, 0), "Drama")
        });

        // Выводим информацию о всех фильмах
        Console.WriteLine("Информация обо всех фильмах:");
        foreach (var genre in moviesByGenre)
        {
            foreach (var movie in genre.Value)
            {
                movie.DisplayInfo();
            }
        }

        // Подсчет количества фильмов каждого жанра
        Console.WriteLine("\nКоличество фильмов по жанрам:");
        foreach (var genre in moviesByGenre)
        {
            Console.WriteLine($"{genre.Key}: {genre.Value.Count} фильмов");
        }

        // Ввод жанра с клавиатуры
        Console.WriteLine("\nВведите жанр для поиска фильмов:");
        string genreInput = Console.ReadLine();

        // Проверяем, есть ли фильмы этого жанра
        if (moviesByGenre.ContainsKey(genreInput))
        {
            Console.WriteLine($"\nФильмы жанра {genreInput}:");
            foreach (var movie in moviesByGenre[genreInput])
            {
                movie.DisplayInfo();
            }
        }
        else
        {
            Console.WriteLine($"Фильмы жанра {genreInput} не найдены.");
        }
    }
}
```

Практическое задание 17

Описать класс, соответствующий заданию. Создать коллекцию (List) из объектов данного класса. Вывести:

- информацию обо всех объектах;
- информацию об объектах согласно условию задачи.

Создать еще 2 списка, добавив в них объекты согласно условию задачи. Вывести информацию об объектах каждого списка.

В случае отсутствия информации об объектах, вывести соответствующее сообщение.

Поля класса	Задача
Марка автомобиля Производитель Тип двигателя	Вывести сведения об автомобилях марки «Toyota», зарегистрированных до 01.01.2015 г. Создать 2 списка:

Год выпуска Дата регистрации	• список автомобилей, выпущенных за последние 10 лет • список автомобилей с регистрацией в 2018-2021 г.г.

```
using System;
using System.Collections.Generic;
using System.Linq;

class Car
{
    // Поля класса для хранения данных об автомобиле
    public string Make { get; set; }          // Марка автомобиля
    public string Manufacturer { get; set; }  // Производитель
    public string EngineType { get; set; }    // Тип двигателя
    public int Year { get; set; }             // Год выпуска
    public DateTime RegistrationDate { get; set; } // Дата регистрации

    // Конструктор класса
    public Car(string make, string manufacturer, string engineType, int year, DateTime registrationDate)
    {
        Make = make;
        Manufacturer = manufacturer;
        EngineType = engineType;
        Year = year;
        RegistrationDate = registrationDate;
    }

    // Метод для вывода информации о автомобиле
    public void DisplayInfo()
    {
        Console.WriteLine($"Марка: {Make}, Производитель: {Manufacturer}, " +
                          $"Тип двигателя: {EngineType}, Год выпуска: {Year}, " +
                          $"Дата регистрации: {RegistrationDate.ToShortDateString()}");
    }
}

class Program
{
    static void Main()
    {
        // Список всех автомобилей
        List<Car> cars = new List<Car>
        {
            new Car("Toyota", "Toyota", "Бензиновый", 2013, new DateTime(2014, 6, 15)),
            new Car("Toyota", "Toyota", "Дизельный", 2016, new DateTime(2016, 8, 20)),
            new Car("Ford", "Ford", "Бензиновый", 2018, new DateTime(2018, 3, 1)),
            new Car("Toyota", "Toyota", "Гибридный", 2020, new DateTime(2020, 11, 10)),
            new Car("Honda", "Honda", "Бензиновый", 2012, new DateTime(2013, 12, 5)),
            new Car("Nissan", "Nissan", "Электрический", 2021, new DateTime(2021, 2, 20))
        };

        // 1. Выводим все автомобили
        Console.WriteLine("Информация обо всех автомобилях:");
        foreach (var car in cars)
        {
            car.DisplayInfo();
        }

        // 2. Список автомобилей марки Toyota, зарегистрированных до 01.01.2015
        Console.WriteLine("\nИнформация о автомобилях марки Toyota, зарегистрированных до 01.01.2015:");
        var toyotaBefore2015 = cars.Where(car => car.Make == "Toyota" && car.RegistrationDate < new DateTime(2015, 1, 1)).ToList();
        if (toyotaBefore2015.Any())
        {
            foreach (var car in toyotaBefore2015)
            {
                car.DisplayInfo();
            }
        }
        else
        {
            Console.WriteLine("Нет автомобилей марки Toyota, зарегистрированных до 01.01.2015.");
        }

        // 3. Список автомобилей, выпущенных за последние 10 лет
        Console.WriteLine("\nИнформация о автомобилях, выпущенных за последние 10 лет:");
        var recentCars = cars.Where(car => car.Year >= DateTime.Now.Year - 10).ToList();
```

```
        if (recentCars.Any())
        {
            foreach (var car in recentCars)
            {
                car.DisplayInfo();
            }
        }
        else
        {
            Console.WriteLine("Нет автомобилей, выпущенных за последние 10 лет.");
        }

        // 4. Список автомобилей с регистрацией в 2018-2021 гг.
        Console.WriteLine("\nИнформация о автомобилях с регистрацией в 2018-2021 гг.:");
        var carsRegistered2018_2021 = cars.Where(car => car.RegistrationDate.Year >= 2018 && car.RegistrationDate.Year <= 2021).ToList();
        if (carsRegistered2018_2021.Any())
        {
            foreach (var car in carsRegistered2018_2021)
            {
                car.DisplayInfo();
            }
        }
        else
        {
            Console.WriteLine("Нет автомобилей с регистрацией в 2018-2021 гг.");
        }
    }
}
```

Практическое задание 18

Описать класс, соответствующий заданию. Создать коллекцию из объектов данного класса. Вывести информацию обо всех объектах. Отсортировать список объектов по возрастанию по различным полям. Вывести отсортированные списки в табличном виде.

Поля класса	Задача
Салон мебели: № заказа; дата заказа; ФИО заказчика; адрес заказчика; срок выполнения в днях; стоимость заказа.	Отсортировать список объектов по полям: • ФИО заказчика; • срок выполнения в днях; • стоимость заказа.

```
using System;
using System.Collections.Generic;
using System.Linq;

class FurnitureOrder
{
    // Поля класса
    public int OrderNumber { get; set; }         // Номер заказа
    public DateTime OrderDate { get; set; }      // Дата заказа
    public string CustomerName { get; set; }     // ФИО заказчика
    public string CustomerAddress { get; set; }  // Адрес заказчика
    public int ExecutionTime { get; set; }       // Срок выполнения в днях
    public decimal OrderCost { get; set; }       // Стоимость заказа

    // Конструктор класса
    public FurnitureOrder(int orderNumber, DateTime orderDate, string customerName,
                          string customerAddress, int executionTime, decimal orderCost)
    {
        OrderNumber = orderNumber;
        OrderDate = orderDate;
        CustomerName = customerName;
        CustomerAddress = customerAddress;
        ExecutionTime = executionTime;
        OrderCost = orderCost;
    }

    // Метод для вывода информации о заказе
    public void DisplayInfo()
    {
        Console.WriteLine($"№ заказа: {OrderNumber}, Дата заказа: {OrderDate.ToShortDateString()}, " +
                          $"ФИО заказчика: {CustomerName}, Адрес заказчика: {CustomerAddress}, " +
                          $"Срок выполнения: {ExecutionTime} дней, Стоимость заказа: {OrderCost:C}");
    }
}
```

```
class Program
{
   static void Main()
   {
      // Создаем список заказов
      List<FurnitureOrder> orders = new List<FurnitureOrder>
      {
         new FurnitureOrder(1, new DateTime(2023, 5, 10), "Иванов Иван Иванович", "ул. Ленина, 10", 15, 25000m),
         new FurnitureOrder(2, new DateTime(2023, 6, 15), "Петров Петр Петрович", "ул. Пушкина, 5", 30, 45000m),
         new FurnitureOrder(3, new DateTime(2023, 7, 25), "Сидоров Сидор Сидорович", "ул. Гагарина, 12", 10, 20000m),
         new FurnitureOrder(4, new DateTime(2023, 5, 20), "Алексеева Алёна Дмитриевна", "ул. Толстого, 8", 20, 35000m),
         new FurnitureOrder(5, new DateTime(2023, 8, 1), "Николаев Николай Михайлович", "ул. Тимирязева, 3", 25, 40000m)
      };

      // 1. Выводим информацию обо всех объектах
      Console.WriteLine("Информация о всех заказах:");
      foreach (var order in orders)
      {
         order.DisplayInfo();
      }

      // 2. Сортировка по ФИО заказчика
      var sortedByCustomerName = orders.OrderBy(order => order.CustomerName).ToList();
      Console.WriteLine("\nЗаказы, отсортированные по ФИО заказчика:");
      foreach (var order in sortedByCustomerName)
      {
         order.DisplayInfo();
      }

      // 3. Сортировка по сроку выполнения в днях
      var sortedByExecutionTime = orders.OrderBy(order => order.ExecutionTime).ToList();
      Console.WriteLine("\nЗаказы, отсортированные по сроку выполнения (в днях):");
      foreach (var order in sortedByExecutionTime)
      {
         order.DisplayInfo();
      }

      // 4. Сортировка по стоимости заказа
      var sortedByOrderCost = orders.OrderBy(order => order.OrderCost).ToList();
      Console.WriteLine("\nЗаказы, отсортированные по стоимости заказа:");
      foreach (var order in sortedByOrderCost)
      {
         order.DisplayInfo();
      }
   }
}
```

Практическое задание 19

Описать класс, соответствующий заданию. Создать коллекцию (List) из объектов данного класса. Вывести информацию обо всех объектах.

Используя технология Linq, создать еще 2 списка, добавив в них объекты согласно условию задачи. Вывести информацию об объектах каждого списка.

В случае отсутствия информации об объектах, вывести соответствующее сообщение.

Поля класса	Задача
Фамилия Имя Отчество Должность Пол Дата приема на работу	Создать 2 списка: • список сотрудников, чей стаж меньше 10 лет • список сотрудников, чей стаж не меньше 10 лет

```
using System;
using System.Collections.Generic;
using System.Linq;

class Employee
{
   // Поля класса для хранения информации о сотруднике
   public string LastName { get; set; }   // Фамилия
   public string FirstName { get; set; }   // Имя
   public string Patronymic { get; set; }  // Отчество
```

```
    public string Position { get; set; }    // Должность
    public string Gender { get; set; }      // Пол
    public DateTime HireDate { get; set; }  // Дата приема на работу

    // Конструктор класса для инициализации данных сотрудника
    public Employee(string lastName, string firstName, string patronymic,
                    string position, string gender, DateTime hireDate)
    {
        LastName = lastName;
        FirstName = firstName;
        Patronymic = patronymic;
        Position = position;
        Gender = gender;
        HireDate = hireDate;
    }

    // Метод для вывода информации о сотруднике
    public void DisplayInfo()
    {
        Console.WriteLine($"Фамилия: {LastName}, Имя: {FirstName}, Отчество: {Patronymic}, " +
                          $"Должность: {Position}, Пол: {Gender}, Дата приема на работу: {HireDate.ToShortDateString()}");
    }

    // Метод для вычисления стажа сотрудника в годах
    public int GetYearsOfExperience()
    {
        return DateTime.Now.Year - HireDate.Year;  // Разница между текущим годом и годом приема на работу
    }
}

class Program
{
    static void Main()
    {
        // Создаем список сотрудников
        List<Employee> employees = new List<Employee>
        {
            new Employee("Иванов", "Иван", "Иванович", "Менеджер", "Мужской", new DateTime(2010, 6, 15)),
            new Employee("Петрова", "Мария", "Александровна", "Разработчик", "Женский", new DateTime(2015, 7, 10)),
            new Employee("Сидоров", "Сидор", "Сидорович", "Аналитик", "Мужской", new DateTime(2005, 3, 20)),
            new Employee("Алексеева", "Алёна", "Дмитриевна", "Дизайнер", "Женский", new DateTime(2012, 12, 5)),
            new Employee("Кузнецов", "Николай", "Анатольевич", "Техник", "Мужской", new DateTime(2020, 9, 15))
        };

        // 1. Выводим информацию обо всех сотрудниках
        Console.WriteLine("Информация о всех сотрудниках:");
        foreach (var employee in employees)
        {
            employee.DisplayInfo();  // Вызываем метод для вывода информации о каждом сотруднике
        }

        // 2. Создаем два списка с использованием LINQ:
        // Сотрудники с стажем меньше 10 лет
        var employeesWithLessThan10Years = employees.Where(emp => emp.GetYearsOfExperience() < 10).ToList();

        // Сотрудники с стажем не меньше 10 лет
        var employeesWith10OrMoreYears = employees.Where(emp => emp.GetYearsOfExperience() >= 10).ToList();

        // 3. Выводим информацию о сотрудниках с стажем меньше 10 лет
        Console.WriteLine("\nСотрудники с стажем меньше 10 лет:");
        if (employeesWithLessThan10Years.Any())  // Проверяем, есть ли сотрудники в этом списке
        {
            foreach (var employee in employeesWithLessThan10Years)
            {
                employee.DisplayInfo();  // Выводим информацию о каждом сотруднике в списке
            }
        }
        else
        {
            Console.WriteLine("Нет сотрудников с таким стажем.");  // Если список пуст, выводим сообщение
        }

        // 4. Выводим информацию о сотрудниках с стажем не меньше 10 лет
        Console.WriteLine("\nСотрудники с стажем не меньше 10 лет:");
        if (employeesWith10OrMoreYears.Any())  // Проверяем, есть ли сотрудники в этом списке
```

```
        {
            foreach (var employee in employeesWith10OrMoreYears)
            {
                employee.DisplayInfo();  // Выводим информацию о каждом сотруднике в списке
            }
        }
        else
        {
            Console.WriteLine("Нет сотрудников с таким стажем.");  // Если список пуст, выводим сообщение
        }
    }
}
```

Практическое задание 20

Описать класс **Bus** (*Автобус*) с полями: марка, количество мест, стоимость билета. Определить в нем:

- конструктор, принимающий поля класса;
- метод, выводящий информацию об объекте;
- метод, возвращающий стоимость всех мест.

Написать программу, использующую этот класс и методы. Информацию об объекте ввести с клавиатуры. Предусмотреть обработку исключительных ситуаций.

```
using System;

class Bus
{
    // Поля класса
    public string Brand { get; set; }      // Марка автобуса
    public int Seats { get; set; }         // Количество мест
    public double TicketPrice { get; set; } // Стоимость билета

    // Конструктор, принимающий параметры
    public Bus(string brand, int seats, double ticketPrice)
    {
        Brand = brand;
        Seats = seats;
        TicketPrice = ticketPrice;
    }

    // Метод для вывода информации об автобусе
    public void DisplayInfo()
    {
        Console.WriteLine($"Марка автобуса: {Brand}");
        Console.WriteLine($"Количество мест: {Seats}");
        Console.WriteLine($"Стоимость билета: {TicketPrice} рублей");
    }

    // Метод для вычисления стоимости всех мест
    public double GetTotalPrice()
    {
        return Seats * TicketPrice;  // Стоимость всех мест = количество мест * стоимость билета
    }
}

class Program
{
    static void Main()
    {
        try
        {
            // Ввод данных с клавиатуры
            Console.WriteLine("Введите марку автобуса:");
            string brand = Console.ReadLine();
            if (string.IsNullOrEmpty(brand))
            {
                throw new ArgumentException("Марка автобуса не может быть пустой.");
            }

            Console.WriteLine("Введите количество мест:");
            int seats = int.Parse(Console.ReadLine());
            if (seats <= 0)
            {
                throw new ArgumentException("Количество мест должно быть положительным числом.");
            }
```

```
        Console.WriteLine("Введите стоимость билета:");
        double ticketPrice = double.Parse(Console.ReadLine());
        if (ticketPrice <= 0)
        {
            throw new ArgumentException("Стоимость билета должна быть положительным числом.");
        }

        // Создание объекта Bus с введенными данными
        Bus bus = new Bus(brand, seats, ticketPrice);

        // Вывод информации об автобусе
        Console.WriteLine("\nИнформация об автобусе:");
        bus.DisplayInfo();

        // Вывод стоимости всех мест
        Console.WriteLine($"\nСтоимость всех мест: {bus.GetTotalPrice()} рублей");
    }
    catch (FormatException ex)
    {
        Console.WriteLine("Ошибка ввода данных. Пожалуйста, введите правильные значения.");
    }
    catch (ArgumentException ex)
    {
        Console.WriteLine(ex.Message);  // Выводим сообщение об ошибке из исключения
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Произошла непредвиденная ошибка: {ex.Message}");
    }
  }
}
```

Практическое задание 21

Описать класс **Triangle** (*Треугольник*) с полями *a, b, c* – длины 3-х сторон. Определить в нем:

- конструктор, принимающий поля класса;
- метод, выводящий информацию об объекте;
- метод, выводящий тип треугольника по сторонам (разносторонний, равнобедренный, равносторонний).

Написать программу, использующую этот класс и методы. Информацию об объекте ввести с клавиатуры. Предусмотреть обработку исключительных ситуаций.

```
using System;

class Triangle
{
  // Поля класса для сторон треугольника
  public double A { get; set; }
  public double B { get; set; }
  public double C { get; set; }

  // Конструктор, который инициализирует поля класса
  public Triangle(double a, double b, double c)
  {
    A = a;
    B = b;
    C = c;
  }

  // Метод для вывода информации о треугольнике
  public void DisplayInfo()
  {
    Console.WriteLine($"Стороны треугольника: A = {A}, B = {B}, C = {C}");
  }

  // Метод для определения типа треугольника по сторонам
  public string GetTriangleType()
  {
    if (A <= 0 || B <= 0 || C <= 0)
    {
      return "Стороны треугольника должны быть положительными числами.";
    }

    // Проверка на существование треугольника (сумма длин двух сторон должна быть больше длины третьей)
```

```
        if (A + B <= C || A + C <= B || B + C <= A)
        {
            return "Не существует треугольника с такими сторонами.";
        }

        // Определение типа треугольника по сторонам
        if (A == B && B == C)
        {
            return "Равносторонний треугольник";
        }
        else if (A == B || B == C || A == C)
        {
            return "Равнобедренный треугольник";
        }
        else
        {
            return "Разносторонний треугольник";
        }
    }
}

class Program
{
    static void Main()
    {
        try
        {
            // Ввод данных с клавиатуры
            Console.WriteLine("Введите длину стороны A:");
            double a = double.Parse(Console.ReadLine());

            Console.WriteLine("Введите длину стороны B:");
            double b = double.Parse(Console.ReadLine());

            Console.WriteLine("Введите длину стороны C:");
            double c = double.Parse(Console.ReadLine());

            // Создание объекта Triangle
            Triangle triangle = new Triangle(a, b, c);

            // Вывод информации о треугольнике
            triangle.DisplayInfo();

            // Вывод типа треугольника
            Console.WriteLine("Тип треугольника: " + triangle.GetTriangleType());
        }
        catch (FormatException)
        {
            // Обработка ошибок ввода (неправильный формат чисел)
            Console.WriteLine("Ошибка: введены некорректные данные. Пожалуйста, введите числа.");
        }
    }
}
```

Практическое задание 22

Описать класс **Magazine** (*Журнал*) с полями: тираж, цена 1 экземпляра. Определить в нем:

- конструктор, принимающий поля класса;
- метод, выводящий информацию об объекте;
- метод, возвращающий стоимость тиража.

Написать программу, использующую этот класс и методы. Информацию об объекте ввести с клавиатуры. Предусмотреть обработку исключительных ситуаций.

```
using System;

class Magazine
{
    // Поля класса для хранения тиража и цены одного экземпляра
    public int Circulation { get; set; }  // Тираж журнала (целое число)
    public double PricePerCopy { get; set; }  // Цена одного экземпляра (вещественное число)

    // Конструктор класса, который принимает два параметра: тираж и цену одного экземпляра
    public Magazine(int circulation, double pricePerCopy)
```

```
    {
        Circulation = circulation;  // Инициализация поля тиража значением из параметра
        PricePerCopy = pricePerCopy;  // Инициализация поля цены одного экземпляра значением из параметра
    }

    // Метод для вывода информации о журнале
    public void DisplayInfo()
    {
        Console.WriteLine($"Тираж: {Circulation} экземпляров");  // Выводим информацию о тираже
        Console.WriteLine($"Цена одного экземпляра: {PricePerCopy} руб.");  // Выводим информацию о цене одного экземпляра
    }

    // Метод для вычисления стоимости всего тиража
    public double CalculateTotalCost()
    {
        return Circulation * PricePerCopy;  // Возвращаем стоимость всего тиража (тираж * цена одного экземпляра)
    }
}

class Program
{
    static void Main()
    {
        try
        {
            // Ввод данных с клавиатуры: тираж
            Console.WriteLine("Введите тираж журнала:");
            int circulation = int.Parse(Console.ReadLine());  // Читаем ввод и преобразуем в целое число

            // Ввод данных с клавиатуры: цена одного экземпляра
            Console.WriteLine("Введите цену одного экземпляра журнала:");
            double pricePerCopy = double.Parse(Console.ReadLine());  // Читаем ввод и преобразуем в вещественное число

            // Проверка введенных данных на корректность
            if (circulation <= 0 || pricePerCopy <= 0)  // Если тираж или цена меньше или равны нулю
            {
                Console.WriteLine("Ошибка: тираж и цена должны быть положительными числами.");  // Выводим сообщение об ошибке
                return;  // Завершаем программу
            }

            // Создаем объект журнала с заданными параметрами
            Magazine magazine = new Magazine(circulation, pricePerCopy);

            // Выводим информацию о журнале с помощью метода DisplayInfo()
            magazine.DisplayInfo();

            // Выводим стоимость всего тиража, используя метод CalculateTotalCost()
            double totalCost = magazine.CalculateTotalCost();
            Console.WriteLine($"Стоимость всего тиража: {totalCost} руб.");  // Выводим рассчитанную стоимость
        }
        catch (FormatException)
        {
            // Обработка ошибки формата ввода (если пользователь ввел некорректные данные, например текст)
            Console.WriteLine("Ошибка: введены некорректные данные. Пожалуйста, введите числа.");  // Выводим сообщение об ошибке
        }
    }
}
```

Практическое задание 23

Описать класс **Newspaper** (*Газета*) с полями: тираж, количество листов, стоимость 1 листа. Определить в нем:

- конструктор, принимающий поля класса;
- метод, выводящий информацию об объекте;
- метод, возвращающий стоимость тиража.

Написать программу, использующую этот класс и методы. Информацию об объекте ввести с клавиатуры. Предусмотреть обработку исключительных ситуаций.

```
using System;

class Newspaper
{
    // Поля класса для хранения тиража, количества листов и стоимости одного листа
    public int Circulation { get; set; }  // Тираж газеты (целое число)
```

```
    public int NumberOfPages { get; set; }  // Количество листов (целое число)
    public double PricePerPage { get; set; }  // Стоимость одного листа (вещественное число)

    // Конструктор класса, который принимает три параметра: тираж, количество листов и стоимость одного листа
    public Newspaper(int circulation, int numberOfPages, double pricePerPage)
    {
        Circulation = circulation;  // Инициализация поля тиража значением из параметра
        NumberOfPages = numberOfPages;  // Инициализация поля количества листов значением из параметра
        PricePerPage = pricePerPage;  // Инициализация поля стоимости одного листа значением из параметра
    }

    // Метод для вывода информации о газете
    public void DisplayInfo()
    {
        Console.WriteLine($"Тираж газеты: {Circulation} экземпляров");  // Выводим информацию о тираже
        Console.WriteLine($"Количество листов в газете: {NumberOfPages}");  // Выводим количество листов
        Console.WriteLine($"Стоимость одного листа: {PricePerPage} руб.");  // Выводим стоимость одного листа
    }

    // Метод для вычисления стоимости всего тиража
    public double CalculateTotalCost()
    {
        return Circulation * NumberOfPages * PricePerPage;  // Стоимость тиража = тираж * количество листов * стоимость одного листа
    }
}

class Program
{
    static void Main()
    {
        try
        {
            // Ввод данных с клавиатуры: тираж
            Console.WriteLine("Введите тираж газеты:");
            int circulation = int.Parse(Console.ReadLine());  // Читаем ввод и преобразуем в целое число

            // Ввод данных с клавиатуры: количество листов
            Console.WriteLine("Введите количество листов в газете:");
            int numberOfPages = int.Parse(Console.ReadLine());  // Читаем ввод и преобразуем в целое число

            // Ввод данных с клавиатуры: стоимость одного листа
            Console.WriteLine("Введите стоимость одного листа в рублях:");
            double pricePerPage = double.Parse(Console.ReadLine());  // Читаем ввод и преобразуем в вещественное число

            // Проверка введенных данных на корректность
            if (circulation <= 0 || numberOfPages <= 0 || pricePerPage <= 0)  // Если тираж, количество листов или цена меньше или равны нулю
            {
                Console.WriteLine("Ошибка: все значения должны быть положительными числами.");  // Выводим сообщение об ошибке
                return;  // Завершаем программу
            }

            // Создаем объект газеты с заданными параметрами
            Newspaper newspaper = new Newspaper(circulation, numberOfPages, pricePerPage);

            // Выводим информацию о газете с помощью метода DisplayInfo()
            newspaper.DisplayInfo();

            // Выводим стоимость всего тиража, используя метод CalculateTotalCost()
            double totalCost = newspaper.CalculateTotalCost();
            Console.WriteLine($"Стоимость всего тиража: {totalCost} руб.");  // Выводим рассчитанную стоимость
        }
        catch (FormatException)
        {
            // Обработка ошибки формата ввода (если пользователь ввел некорректные данные, например текст)
            Console.WriteLine("Ошибка: введены некорректные данные. Пожалуйста, введите числа.");  // Выводим сообщение об ошибке
        }
    }
}
```

Практическое задание 24

Описать класс **Student** (*Студент*) с полями: фамилия, имя, 3 оценки. Определить в нем:

- конструктор, принимающий поля класса;
- метод, выводящий информацию об объекте;

- метод, возвращающий среднее арифметическое значение оценок.

Написать программу, использующую этот класс и методы. Информацию об объекте ввести с клавиатуры. Предусмотреть обработку исключительных ситуаций.

```
using System;

class Student
{
    // Поля класса для фамилии, имени и оценок
    public string LastName { get; set; }  // Фамилия студента
    public string FirstName { get; set; }  // Имя студента
    public double Grade1 { get; set; }  // Первая оценка
    public double Grade2 { get; set; }  // Вторая оценка
    public double Grade3 { get; set; }  // Третья оценка

    // Конструктор класса, принимающий фамилию, имя и три оценки
    public Student(string lastName, string firstName, double grade1, double grade2, double grade3)
    {
        LastName = lastName;  // Инициализация фамилии
        FirstName = firstName;  // Инициализация имени
        Grade1 = grade1;  // Инициализация первой оценки
        Grade2 = grade2;  // Инициализация второй оценки
        Grade3 = grade3;  // Инициализация третьей оценки
    }

    // Метод для вывода информации о студенте
    public void DisplayInfo()
    {
        Console.WriteLine($"Фамилия: {LastName}");
        Console.WriteLine($"Имя: {FirstName}");
        Console.WriteLine($"Оценки: {Grade1}, {Grade2}, {Grade3}");
    }

    // Метод для вычисления среднего арифметического оценок
    public double CalculateAverage()
    {
        return (Grade1 + Grade2 + Grade3) / 3;  // Возвращаем среднее значение оценок
    }
}

class Program
{
    static void Main()
    {
        try
        {
            // Ввод данных с клавиатуры: фамилия студента
            Console.WriteLine("Введите фамилию студента:");
            string lastName = Console.ReadLine();

            // Ввод данных с клавиатуры: имя студента
            Console.WriteLine("Введите имя студента:");
            string firstName = Console.ReadLine();

            // Ввод данных с клавиатуры: первая оценка
            Console.WriteLine("Введите первую оценку:");
            double grade1 = double.Parse(Console.ReadLine());  // Чтение и преобразование в число с плавающей точкой

            // Ввод данных с клавиатуры: вторая оценка
            Console.WriteLine("Введите вторую оценку:");
            double grade2 = double.Parse(Console.ReadLine());

            // Ввод данных с клавиатуры: третья оценка
            Console.WriteLine("Введите третью оценку:");
            double grade3 = double.Parse(Console.ReadLine());

            // Проверка введенных данных на корректность
            if (grade1 < 0 || grade2 < 0 || grade3 < 0)  // Если хоть одна оценка отрицательна
            {
                Console.WriteLine("Ошибка: оценки не могут быть отрицательными.");
                return;
            }

            // Создаем объект студента с заданными параметрами
```

```
        Student student = new Student(lastName, firstName, grade1, grade2, grade3);

        // Выводим информацию о студенте
        student.DisplayInfo();

        // Вычисляем и выводим среднее арифметическое оценок
        double average = student.CalculateAverage();
        Console.WriteLine($"Среднее арифметическое оценок: {average:F2}");  // Выводим среднее значение с точностью до двух знаков
    }
    catch (FormatException)
    {
        // Обработка ошибки формата ввода (если пользователь ввел некорректные данные, например текст вместо чисел)
        Console.WriteLine("Ошибка: введены некорректные данные. Пожалуйста, введите числа.");
    }
  }
}
```

Практическое задание 25

Описать структуру, соответствующую заданию. Создать массив из *N* объектов данной структуры. Ввод данных осуществить с клавиатуры. Сведения об объектах вывести в табличном виде. В случае отсутствия информации об объектах, вывести соответствующее сообщение. Написать программу решения задачи.

Поля структуры	Задача
Наименование Изготовитель Количество Цена Год_выпуска	Определить общую стоимость всех товаров, выпущенных в текущем году, и вывести сведения об этих товарах.

```
using System;  // Подключение пространства имен для работы с консолью и стандартными операциями

class Program
{
  // Определение структуры "Товар"
  struct Product
  {
    // Поля структуры для хранения данных о товаре
    public string Name;  // Наименование товара
    public string Manufacturer;  // Изготовитель товара
    public int Quantity;  // Количество товара
    public decimal Price;  // Цена одного экземпляра товара
    public int YearOfManufacture;  // Год выпуска товара

    // Метод для вывода информации о товаре
    public void DisplayInfo()
    {
      // Используем форматированный вывод для создания табличного вида
      Console.WriteLine($"{Name,-20} {Manufacturer,-15} {Quantity,-10} {Price,-8:C} {YearOfManufacture,-10}");
    }

    // Метод для вычисления стоимости товара
    public decimal GetTotalPrice()
    {
      return Quantity * Price;  // Возвращаем стоимость товара (количество * цена)
    }
  }

  static void Main()
  {
    // Запрос на ввод количества товаров
    Console.WriteLine("Введите количество товаров:");
    int N = int.Parse(Console.ReadLine());  // Преобразуем введенное значение в целое число

    // Массив для хранения товаров
    Product[] products = new Product[N];

    // Ввод данных для каждого товара
    for (int i = 0; i < N; i++)
    {
      // Ввод информации о каждом товаре
      Console.WriteLine($"\nВведите данные для товара #{i + 1}:");

      // Ввод наименования товара
```

```
        Console.Write("Наименование товара: ");
        products[i].Name = Console.ReadLine();  // Записываем введенное значение в поле Name

        // Ввод изготовителя товара
        Console.Write("Изготовитель товара: ");
        products[i].Manufacturer = Console.ReadLine();  // Записываем введенное значение в поле Manufacturer

        // Ввод количества товара
        Console.Write("Количество товара: ");
        products[i].Quantity = int.Parse(Console.ReadLine());  // Преобразуем введенное значение в число и записываем в Quantity

        // Ввод цены товара
        Console.Write("Цена товара: ");
        products[i].Price = decimal.Parse(Console.ReadLine());  // Преобразуем введенное значение в десятичное число и записываем в Price

        // Ввод года выпуска товара
        Console.Write("Год выпуска товара: ");
        products[i].YearOfManufacture = int.Parse(Console.ReadLine());  // Преобразуем введенное значение в целое число и записываем в
YearOfManufacture
    }

    // Вывод всех товаров в табличном виде
    Console.WriteLine("\nСведения о всех товарах:");
    // Печатаем заголовок таблицы
    Console.WriteLine($"{ "Наименование",-20} { "Изготовитель",-15} { "Количество",-10} { "Цена",-8} { "Год выпуска",-10}");

    // Перебираем все товары и выводим их информацию
    foreach (var product in products)
    {
        product.DisplayInfo();  // Для каждого товара вызываем метод DisplayInfo, чтобы вывести его данные
    }

    // Вычисление общей стоимости товаров, выпущенных в текущем году
    int currentYear = DateTime.Now.Year;  // Получаем текущий год с помощью стандартной библиотеки
    decimal totalPrice = 0;  // Переменная для хранения общей стоимости товаров
    bool found = false;  // Флаг для проверки, были ли товары текущего года

    Console.WriteLine($"\nТовары, выпущенные в {currentYear} году:");

    // Перебираем товары и ищем те, которые выпущены в текущем году
    foreach (var product in products)
    {
        if (product.YearOfManufacture == currentYear)  // Если год выпуска товара совпадает с текущим годом
        {
            product.DisplayInfo();  // Выводим информацию о товаре
            totalPrice += product.GetTotalPrice();  // Добавляем стоимость товара к общей стоимости
            found = true;  // Устанавливаем флаг, что хотя бы один товар был найден
        }
    }

    // Если не было найдено товаров, выпущенных в текущем году
    if (!found)
    {
        Console.WriteLine("Нет товаров, выпущенных в текущем году.");
    }
    else
    {
        // Выводим общую стоимость товаров
        Console.WriteLine($"\nОбщая стоимость товаров, выпущенных в {currentYear} году: {totalPrice:C}");
    }
  }
}
```

Практическое задание 26

Описать структуру, соответствующую заданию. Создать массив из *N* объектов данной структуры. Ввод данных осуществить с клавиатуры. Сведения об объектах вывести в табличном виде. В случае отсутствия информации об объектах, вывести соответствующее сообщение. Написать программу решения задачи.

Поля структуры	Задача
Автор Жанр Название Тираж Цена	Вывести данные о книгах, тирах которых не превышает 10000 экземпляров. Подсчитать общую стоимость этих книг.

```
using System;  // Подключение стандартного пространства имен для работы с консолью

class Program
{
    // Определение структуры Book (Книга)
    struct Book
    {
        public string Author;  // Автор книги
        public string Genre;  // Жанр книги
        public string Title;  // Название книги
        public int Circulation;  // Тираж книги
        public decimal Price;  // Цена книги

        // Метод для вывода информации о книге
        public void DisplayInfo()
        {
            Console.WriteLine($"{Author,-20} {Genre,-15} {Title,-20} {Circulation,-10} {Price,-8:C}");
        }
    }

    static void Main()
    {
        // Запрос на ввод количества книг
        Console.WriteLine("Введите количество книг:");
        int N = int.Parse(Console.ReadLine());  // Вводим количество книг

        // Создаем массив объектов структуры Book
        Book[] books = new Book[N];

        // Ввод данных для каждой книги
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"\nВведите данные для книги #{i + 1}:");

            // Ввод данных о книге
            Console.Write("Автор: ");
            books[i].Author = Console.ReadLine();  // Вводим имя автора

            Console.Write("Жанр: ");
            books[i].Genre = Console.ReadLine();  // Вводим жанр книги

            Console.Write("Название книги: ");
            books[i].Title = Console.ReadLine();  // Вводим название книги

            Console.Write("Тираж: ");
            books[i].Circulation = int.Parse(Console.ReadLine());  // Вводим тираж книги

            Console.Write("Цена: ");
            books[i].Price = decimal.Parse(Console.ReadLine());  // Вводим цену книги
        }

        // Вывод всех книг в табличном виде
        Console.WriteLine("\nСведения о книгах с тиражом не более 10000 экземпляров:");
        // Печатаем заголовок таблицы
        Console.WriteLine($"{ "Автор",-20} { "Жанр",-15} { "Название",-20} { "Тираж",-10} { "Цена",-8}");

        decimal totalCost = 0;  // Переменная для подсчета общей стоимости книг с тиражом <= 10000
        bool foundBooks = false;  // Флаг для проверки наличия книг с тиражом <= 10000

        // Перебираем все книги и проверяем их тираж
        foreach (var book in books)
        {
            if (book.Circulation <= 10000)  // Если тираж книги не более 10000
            {
                book.DisplayInfo();  // Выводим информацию о книге
                totalCost += book.Price;  // Добавляем стоимость книги в общую стоимость
                foundBooks = true;  // Отмечаем, что книга с подходящим тиражом найдена
            }
        }

        // Если книги с тиражом <= 10000 не были найдены
        if (!foundBooks)
        {
            Console.WriteLine("Нет книг с тиражом менее или равным 10000 экземпляров.");
```

```
        }
        else
        {
            // Выводим общую стоимость книг с тиражом <= 10000
            Console.WriteLine($"\nОбщая стоимость книг с тиражом не более 10000 экземпляров: {totalCost:C}");
        }
    }
}
```

Практическое задание 27

Описать структуру, соответствующую заданию. Создать массив из *N* объектов данной структуры. Ввод данных осуществить с клавиатуры. Сведения об объектах вывести в табличном виде. В случае отсутствия информации об объектах, вывести соответствующее сообщение. Написать программу решения задачи.

Поля структуры	Задача
Фамилия Возраст Количество игр Количество заброшенных шайб	Определить средний возраст хоккеистов. Вывести сведения о хоккеистах, возраст которых больше 25 лет.

```
using System;

class Program
{
    // Определение структуры HockeyPlayer (Хоккеист)
    struct HockeyPlayer
    {
        public string LastName;       // Фамилия хоккеиста
        public int Age;               // Возраст хоккеиста
        public int GamesPlayed;       // Количество игр
        public int GoalsScored;       // Количество заброшенных шайб

        // Метод для вывода информации о хоккеисте
        public void DisplayInfo()
        {
            Console.WriteLine($"{LastName,-15} {Age,-5} {GamesPlayed,-10} {GoalsScored,-10}");
        }
    }

    static void Main()
    {
        // Ввод количества хоккеистов
        Console.WriteLine("Введите количество хоккеистов:");
        int N = int.Parse(Console.ReadLine());  // Вводим количество хоккеистов

        if (N <= 0)
        {
            Console.WriteLine("Неверное количество хоккеистов.");
            return;
        }

        // Создание массива объектов структуры HockeyPlayer
        HockeyPlayer[] players = new HockeyPlayer[N];

        // Ввод данных о хоккеистах
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"\nВведите данные для хоккеиста #{i + 1}:");

            Console.Write("Фамилия: ");
            players[i].LastName = Console.ReadLine();  // Ввод фамилии хоккеиста

            Console.Write("Возраст: ");
            players[i].Age = int.Parse(Console.ReadLine());  // Ввод возраста хоккеиста

            Console.Write("Количество игр: ");
            players[i].GamesPlayed = int.Parse(Console.ReadLine());  // Ввод количества игр

            Console.Write("Количество заброшенных шайб: ");
            players[i].GoalsScored = int.Parse(Console.ReadLine());  // Ввод количества заброшенных шайб
        }

        // Расчет среднего возраста хоккеистов
        double totalAge = 0;  // Переменная для подсчета общей суммы возрастов хоккеистов
        int countAbove25 = 0;  // Переменная для подсчета хоккеистов старше 25 лет
```

```
    // Вывод заголовка таблицы
    Console.WriteLine("\nСведения о хоккеистах:");
    Console.WriteLine($"{ "Фамилия",-15} { "Возраст",-5} { "Игры",-10} { "Шайбы",-10}");

    // Перебор всех хоккеистов и вывод данных о них
    foreach (var player in players)
    {
        totalAge += player.Age;  // Суммируем возраст хоккеистов

        if (player.Age > 25)  // Если возраст хоккеиста больше 25
        {
            player.DisplayInfo();  // Выводим информацию о хоккеисте
            countAbove25++;  // Увеличиваем счетчик хоккеистов старше 25 лет
        }
    }

    // Если есть хоккеисты старше 25 лет
    if (countAbove25 == 0)
    {
        Console.WriteLine("\nНет хоккеистов старше 25 лет.");
    }

    // Расчет среднего возраста хоккеистов
    double averageAge = totalAge / N;
    Console.WriteLine($"\nСредний возраст хоккеистов: {averageAge:F2} лет");
  }
}
```

Практическое задание 28

Описать структуру, соответствующую заданию. Создать массив из *N* объектов данной структуры. Ввод данных осуществить с клавиатуры. Сведения об объектах вывести в табличном виде. В случае отсутствия информации об объектах, вывести соответствующее сообщение. Написать программу решения задачи.

Поля структуры	Задача
Фамилия Имя Отчество Должность Дата приема на работу	Определить средний стаж работы и вывести сведения о сотрудниках, стаж которых более 30 лет.

```
using System;

class Program
{
  // Структура для сотрудника
  struct Employee
  {
    public string LastName;      // Фамилия
    public string FirstName;     // Имя
    public string MiddleName;    // Отчество
    public string Position;      // Должность
    public DateTime DateOfHiring; // Дата приема на работу

    // Метод для вывода информации о сотруднике
    public void DisplayInfo()
    {
      Console.WriteLine($"{LastName,-15} {FirstName,-10} {MiddleName,-10} {Position,-20} {DateOfHiring.ToShortDateString()}");
    }

    // Метод для вычисления стажа работы
    public int GetWorkExperience()
    {
      return DateTime.Now.Year - DateOfHiring.Year;
    }
  }

  static void Main()
  {
    // Ввод количества сотрудников
    Console.WriteLine("Введите количество сотрудников:");
    int N = int.Parse(Console.ReadLine());

    if (N <= 0)
```

```
    {
        Console.WriteLine("Неверное количество сотрудников.");
        return;
    }

    // Массив сотрудников
    Employee[] employees = new Employee[N];

    // Ввод данных сотрудников
    for (int i = 0; i < N; i++)
    {
        Console.WriteLine($"\nВведите данные для сотрудника #{i + 1}:");

        Console.Write("Фамилия: ");
        employees[i].LastName = Console.ReadLine();

        Console.Write("Имя: ");
        employees[i].FirstName = Console.ReadLine();

        Console.Write("Отчество: ");
        employees[i].MiddleName = Console.ReadLine();

        Console.Write("Должность: ");
        employees[i].Position = Console.ReadLine();

        Console.Write("Дата приема на работу (формат: dd.mm.yyyy): ");
        employees[i].DateOfHiring = DateTime.ParseExact(Console.ReadLine(), "dd.MM.yyyy", null);
    }

    // Подсчет среднего стажа
    double totalExperience = 0;
    int countAbove30 = 0;  // Счетчик сотрудников с опытом больше 30 лет

    // Вывод заголовков таблицы
    Console.WriteLine("\nСведения о сотрудниках:");
    Console.WriteLine($"{ "Фамилия",-15} { "Имя",-10} { "Отчество",-10} { "Должность",-20} { "Дата приема",-15} { "Стаж(лет)"}");

    // Перебор сотрудников
    foreach (var employee in employees)
    {
        int experience = employee.GetWorkExperience();  // Получаем стаж сотрудника
        totalExperience += experience;

        // Если стаж сотрудника больше 30 лет, выводим его информацию
        if (experience > 30)
        {
            employee.DisplayInfo();
            countAbove30++;  // Увеличиваем счетчик сотрудников старше 30 лет
        }
    }

    // Если нет сотрудников с стажем более 30 лет
    if (countAbove30 == 0)
    {
        Console.WriteLine("\nНет сотрудников с опытом более 30 лет.");
    }

    // Подсчет среднего стажа
    double averageExperience = totalExperience / N;
    Console.WriteLine($"\nСредний стаж работы сотрудников: {averageExperience:F2} лет");
  }
}
```

Практическое задание 29

Описать структуру, соответствующую заданию. Создать массив из *N* объектов данной структуры. Ввод данных осуществить с клавиатуры. Сведения об объектах вывести в табличном виде. В случае отсутствия информации об объектах, вывести соответствующее сообщение. Написать программу решения задачи.

Поля структуры	Задача
Фамилия Имя Отчество Должность Зарплата	Вывести сведения о сотрудниках, у которых зарплата выше средней и возраст менее 30-ти лет.

Дата рождения	

```
using System;

class Program
{
    // Структура для сотрудника
    struct Employee
    {
        public string LastName;      // Фамилия
        public string FirstName;     // Имя
        public string MiddleName;    // Отчество
        public string Position;      // Должность
        public decimal Salary;       // Зарплата
        public DateTime BirthDate;   // Дата рождения

        // Метод для вывода информации о сотруднике
        public void DisplayInfo()
        {
            Console.WriteLine($"{LastName,-15} {FirstName,-10} {MiddleName,-10} {Position,-20} {Salary,-10:C} {BirthDate.ToShortDateString()}");
        }

        // Метод для вычисления возраста сотрудника
        public int GetAge()
        {
            int age = DateTime.Now.Year - BirthDate.Year;
            if (DateTime.Now.DayOfYear < BirthDate.DayOfYear) age--;
            return age;
        }
    }

    static void Main()
    {
        // Ввод количества сотрудников
        Console.WriteLine("Введите количество сотрудников:");
        int N = int.Parse(Console.ReadLine());

        if (N <= 0)
        {
            Console.WriteLine("Неверное количество сотрудников.");
            return;
        }

        // Массив сотрудников
        Employee[] employees = new Employee[N];

        // Ввод данных сотрудников
        for (int i = 0; i < N; i++)
        {
            Console.WriteLine($"\nВведите данные для сотрудника #{i + 1}:");

            Console.Write("Фамилия: ");
            employees[i].LastName = Console.ReadLine();

            Console.Write("Имя: ");
            employees[i].FirstName = Console.ReadLine();

            Console.Write("Отчество: ");
            employees[i].MiddleName = Console.ReadLine();

            Console.Write("Должность: ");
            employees[i].Position = Console.ReadLine();

            Console.Write("Зарплата: ");
            employees[i].Salary = decimal.Parse(Console.ReadLine());

            Console.Write("Дата рождения (формат: dd.mm.yyyy): ");
            employees[i].BirthDate = DateTime.ParseExact(Console.ReadLine(), "dd.MM.yyyy", null);
        }

        // Подсчет средней зарплаты
        decimal totalSalary = 0;
        foreach (var employee in employees)
        {
            totalSalary += employee.Salary;
```

```
        }
        decimal averageSalary = totalSalary / N;

        // Вывод заголовков таблицы
        Console.WriteLine("\nСведения о сотрудниках с зарплатой выше средней и возрастом меньше 30 лет:");
        Console.WriteLine($"{ "Фамилия",-15} { "Имя",-10} { "Отчество",-10} { "Должность",-20} { "Зарплата",-10} { "Дата рождения"}");

        // Перебор сотрудников
        bool found = false;  // Флаг для проверки наличия сотрудников, подходящих под условия
        foreach (var employee in employees)
        {
            if (employee.Salary > averageSalary && employee.GetAge() < 30)
            {
                employee.DisplayInfo();
                found = true;
            }
        }

        // Если нет сотрудников, которые удовлетворяют условиям
        if (!found)
        {
            Console.WriteLine("\nНет сотрудников с зарплатой выше средней и возрастом менее 30 лет.");
        }
    }
}
```

Практическое задание 30

Написать программу, исходными значениями которой будут назначенная дата и время посещения доктора, а результатом работы информационное сообщение о том, сколько на данный момент времени осталось часов до визита к доктору, и в которой части дня предстоит визит (в первой половине дня до 12:00 или во второй).

```
using System;

class Program
{
    static void Main()
    {
        // Ввод даты и времени визита к доктору
        Console.WriteLine("Введите дату и время визита к доктору (формат: dd.MM.yyyy HH:mm):");
        string inputDate = Console.ReadLine();

        // Преобразуем введенную строку в объект DateTime
        DateTime visitDate;
        if (!DateTime.TryParseExact(inputDate, "dd.MM.yyyy HH:mm", null, System.Globalization.DateTimeStyles.None, out visitDate))
        {
            Console.WriteLine("Некорректный формат даты и времени.");
            return;
        }

        // Получаем текущее время
        DateTime currentDateTime = DateTime.Now;

        // Проверяем, что визит еще не наступил
        if (visitDate < currentDateTime)
        {
            Console.WriteLine("Визит уже прошел.");
            return;
        }

        // Рассчитываем оставшееся время до визита
        TimeSpan timeLeft = visitDate - currentDateTime;

        // Определяем, в какой части дня предстоит визит
        string partOfDay = visitDate.Hour < 12 ? "в первой половине дня" : "во второй половине дня";

        // Выводим результат
        Console.WriteLine($"До визита осталось {timeLeft.Hours} часов и {timeLeft.Minutes} минут.");
        Console.WriteLine($"Визит будет {partOfDay}.");
    }
}
```

Практическое задание 31

Описать класс, соответствующий заданию. Создать коллекцию из объектов данного класса. Вывести информацию обо всех объектах. Отсортировать список объектов по возрастанию по различным полям. Вывести отсортированные списки в табличном виде.

Поля класса	Задача
Сотрудники предприятия: – табельный номер; – ФИО сотрудника; – дата рождения; – пол (м/ж); – дата поступления на работу; – должность; – оклад.	Отсортировать список объектов по полям: • табельный номер; • ФИО сотрудника; • оклад.

```
using System;
using System.Collections.Generic;
using System.Linq;

class Employee
{
    public int EmployeeID { get; set; }        // Табельный номер
    public string FullName { get; set; }       // ФИО сотрудника
    public DateTime BirthDate { get; set; }    // Дата рождения
    public string Gender { get; set; }         // Пол (м/ж)
    public DateTime HireDate { get; set; }     // Дата поступления на работу
    public string Position { get; set; }       // Должность
    public decimal Salary { get; set; }        // Оклад

    // Конструктор для создания сотрудника
    public Employee(int employeeID, string fullName, DateTime birthDate, string gender, DateTime hireDate, string position, decimal salary)
    {
        EmployeeID = employeeID;
        FullName = fullName;
        BirthDate = birthDate;
        Gender = gender;
        HireDate = hireDate;
        Position = position;
        Salary = salary;
    }

    // Метод для вывода информации о сотруднике
    public void DisplayInfo()
    {
        Console.WriteLine($"{EmployeeID,-15} {FullName,-30} {BirthDate.ToString("dd.MM.yyyy"),-12} {Gender,-4}
{HireDate.ToString("dd.MM.yyyy"),-12} {Position,-20} {Salary,10:C2}");
    }
}

class Program
{
    static void Main()
    {
        // Создание списка сотрудников
        List<Employee> employees = new List<Employee>
        {
            new Employee(1, "Иванов Иван Иванович", new DateTime(1990, 5, 15), "м", new DateTime(2015, 6, 1), "Менеджер", 50000),
            new Employee(2, "Петрова Мария Александровна", new DateTime(1985, 3, 10), "ж", new DateTime(2017, 8, 15), "Инженер", 45000),
            new Employee(3, "Сидоров Алексей Васильевич", new DateTime(1992, 7, 20), "м", new DateTime(2020, 1, 10), "Аналитик", 60000),
            new Employee(4, "Кузнецова Ольга Петровна", new DateTime(1980, 12, 5), "ж", new DateTime(2010, 3, 25), "HR", 55000),
            new Employee(5, "Дмитриев Андрей Викторович", new DateTime(1995, 11, 30), "м", new DateTime(2022, 9, 1), "Технический директор",
80000)
        };

        // Вывод информации о всех сотрудниках
        Console.WriteLine("Информация о сотрудниках:");
        PrintEmployeeList(employees);

        // Сортировка по табельному номеру
        Console.WriteLine("\nСортировка по табельному номеру:");
        var sortedByEmployeeID = employees.OrderBy(e => e.EmployeeID).ToList();
        PrintEmployeeList(sortedByEmployeeID);

        // Сортировка по ФИО сотрудника
```

```
        Console.WriteLine("\nСортировка по ФИО сотрудника:");
        var sortedByFullName = employees.OrderBy(e => e.FullName).ToList();
        PrintEmployeeList(sortedByFullName);

        // Сортировка по окладу
        Console.WriteLine("\nСортировка по окладу:");
        var sortedBySalary = employees.OrderBy(e => e.Salary).ToList();
        PrintEmployeeList(sortedBySalary);
    }

    // Метод для вывода списка сотрудников
    static void PrintEmployeeList(List<Employee> employees)
    {
        Console.WriteLine("Табельный номер | ФИО сотрудника              | Дата рождения | Пол | Дата поступления | Должность          | Оклад");
        Console.WriteLine("-----------------------------------------------------------------------------------------------------------");

        foreach (var employee in employees)
        {
            employee.DisplayInfo();
        }

        Console.WriteLine("\n");
    }
}
```

Практическое задание 32

Описать класс, соответствующий заданию. Создать коллекцию из объектов данного класса. Вывести информацию обо всех объектах. Отсортировать список объектов по возрастанию по различным полям. Вывести отсортированные списки в табличном виде.

Поля класса	Задача
Расписание экзаменационной сессии: – дата и время начала; – предмет; – ФИО преподавателя; – группа; – № аудитории.	Отсортировать список объектов по полям: • ФИО преподавателя; • группа; • № аудитории.

```
using System;
using System.Collections.Generic;
using System.Linq;

class ExamSchedule
{
    public DateTime ExamDate { get; set; }       // Дата и время начала экзамена
    public string Subject { get; set; }          // Предмет
    public string TeacherFullName { get; set; }  // ФИО преподавателя
    public string Group { get; set; }            // Группа
    public int RoomNumber { get; set; }          // № аудитории

    // Конструктор для создания расписания экзамена
    public ExamSchedule(DateTime examDate, string subject, string teacherFullName, string group, int roomNumber)
    {
        ExamDate = examDate;
        Subject = subject;
        TeacherFullName = teacherFullName;
        Group = group;
        RoomNumber = roomNumber;
    }

    // Метод для вывода информации о экзамене
    public void DisplayInfo()
    {
        Console.WriteLine($"{ExamDate:dd.MM.yyyy HH:mm,-20} {Subject,-20} {TeacherFullName,-30} {Group,-10} {RoomNumber,-10}");
    }
}

class Program
{
    static void Main()
    {
        // Создание списка экзаменов
        List<ExamSchedule> examSchedules = new List<ExamSchedule>
        {
```

```
            new ExamSchedule(new DateTime(2024, 6, 12, 9, 0, 0), "Математика", "Иванов Иван Иванович", "Группа 101", 205),
            new ExamSchedule(new DateTime(2024, 6, 13, 10, 0, 0), "Физика", "Петрова Мария Александровна", "Группа 102", 302),
            new ExamSchedule(new DateTime(2024, 6, 14, 11, 0, 0), "Программирование", "Сидоров Алексей Васильевич", "Группа 101", 305),
            new ExamSchedule(new DateTime(2024, 6, 15, 14, 0, 0), "Химия", "Кузнецова Ольга Петровна", "Группа 103", 207),
            new ExamSchedule(new DateTime(2024, 6, 16, 15, 0, 0), "Биология", "Дмитриев Андрей Викторович", "Группа 102", 210)
        };

        // Вывод информации обо всех экзаменах
        Console.WriteLine("Информация о расписании экзаменов:");
        PrintExamScheduleList(examSchedules);

        // Сортировка по ФИО преподавателя
        Console.WriteLine("\nСортировка по ФИО преподавателя:");
        var sortedByTeacher = examSchedules.OrderBy(e => e.TeacherFullName).ToList();
        PrintExamScheduleList(sortedByTeacher);

        // Сортировка по группе
        Console.WriteLine("\nСортировка по группе:");
        var sortedByGroup = examSchedules.OrderBy(e => e.Group).ToList();
        PrintExamScheduleList(sortedByGroup);

        // Сортировка по номеру аудитории
        Console.WriteLine("\nСортировка по номеру аудитории:");
        var sortedByRoom = examSchedules.OrderBy(e => e.RoomNumber).ToList();
        PrintExamScheduleList(sortedByRoom);
    }

    // Метод для вывода списка расписаний экзаменов
    static void PrintExamScheduleList(List<ExamSchedule> examSchedules)
    {
        Console.WriteLine("Дата и время       | Предмет             | Преподаватель                | Группа    | Аудитория");
        Console.WriteLine("-----------------------------------------------------------------------------------------------");

        foreach (var exam in examSchedules)
        {
            exam.DisplayInfo();
        }

        Console.WriteLine("\n");
    }
}
```

Практическое задание 33

Создайте список целых чисел. Заполните его *n* случайными числами из диапазона [*a, b*]. Используя технологию LINQ, выполните следующие действия:

- Создать список из положительных элементов и отсортировать его по возрастанию.
- Найти сумму положительных элементов, значения которых состоят из двух цифр.
- Подсчитать количество тех элементов, значения которых по модулю превосходят 10 и кратны 5.
- Найти максимальный нечетный элемент.
- Определить, есть ли отрицательные элементы, кратные 3.
- Найти первый отрицательный нечетный элемент; в случае его отсутствия вывести соответствующее сообщение.

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
    {
        Random random = new Random();

        // Заданные границы диапазона для случайных чисел
        int a = -50;
        int b = 50;
        int n = 20; // Количество случайных чисел в списке

        // Создание списка случайных чисел из диапазона [a, b]
        List<int> numbers = Enumerable.Range(0, n) // Генерация списка из n элементов
```

```
                    .Select(_ => random.Next(a, b + 1)) // Заполнение случайными числами
                    .ToList();

        // Вывод списка случайных чисел
        Console.WriteLine("Список случайных чисел:");
        numbers.ForEach(num => Console.Write(num + " "));
        Console.WriteLine();

        // 1. Создание списка положительных чисел и сортировка их по возрастанию
        var positiveSorted = numbers.Where(num => num > 0) // Отбираем только положительные числа
                                    .OrderBy(num => num) // Сортируем по возрастанию
                                    .ToList();
        Console.WriteLine("\nПоложительные элементы, отсортированные по возрастанию:");
        positiveSorted.ForEach(num => Console.Write(num + " "));
        Console.WriteLine();

        // 2. Нахождение суммы положительных элементов, которые имеют две цифры
        var sumTwoDigitPositive = numbers.Where(num => num > 0 && num >= 10 && num <= 99) // Фильтруем только положительные двузначные
числа
                                         .Sum(); // Суммируем их
        Console.WriteLine("\nСумма положительных элементов с двумя цифрами: " + sumTwoDigitPositive);

        // 3. Подсчет количества элементов, по модулю превосходящих 10 и кратных 5
        var countModulusOver10AndDivisibleBy5 = numbers.Count(num => Math.Abs(num) > 10 && num % 5 == 0); // Фильтруем элементы, которые по
модулю больше 10 и кратны 5
        Console.WriteLine("\nКоличество элементов, которые по модулю превосходят 10 и кратны 5: " + countModulusOver10AndDivisibleBy5);

        // 4. Нахождение максимального нечетного элемента
        var maxOdd = numbers.Where(num => num % 2 != 0) // Отбираем нечетные числа
                            .Max(); // Находим максимальное значение
        Console.WriteLine("\nМаксимальный нечетный элемент: " + maxOdd);

        // 5. Проверка наличия отрицательных чисел, которые кратны 3
        bool hasNegativeDivisibleBy3 = numbers.Any(num => num < 0 && num % 3 == 0); // Проверяем наличие отрицательных чисел, кратных 3
        Console.WriteLine("\nЕсть ли отрицательные элементы, кратные 3: " + (hasNegativeDivisibleBy3 ? "Да" : "Нет"));

        // 6. Нахождение первого отрицательного нечетного числа, если оно существует
        var firstNegativeOdd = numbers.FirstOrDefault(num => num < 0 && num % 2 != 0); // Находим первое отрицательное нечетное число
        if (firstNegativeOdd != 0) // Если число не равно 0 (это означает, что элемент найден)
        {
            Console.WriteLine("\nПервый отрицательный нечетный элемент: " + firstNegativeOdd);
        }
        else
        {
            Console.WriteLine("\nОтсутствует первый отрицательный нечетный элемент.");
        }

        // 7. Отображение чисел, которые кратны 3
        var multiplesOf3 = numbers.Where(num => num % 3 == 0) // Отбираем все числа, кратные 3
                                  .ToList();
        Console.WriteLine("\nЧисла, которые кратны 3:");
        multiplesOf3.ForEach(num => Console.Write(num + " "));
        Console.WriteLine();
    }
}
```

Практическое задание 34

Создайте список целых чисел. Заполните его *n* случайными числами из диапазона [*a, b*]. Используя технологию LINQ, выполните следующие действия:

- Создать список из элементов, кратных 3, и отсортировать его по возрастанию.
- Найти сумму элементов, значения которых кратны 4.
- Подсчитать количество тех элементов, значения которых положительны и не превосходят 20.
- Найти минимальный по модулю элемент.
- Определить, есть ли отрицательные элементы, кратные 5.
- Найти первый нечетный элемент; в случае его отсутствия вывести соответствующее сообщение.

```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
```

```
    static void Main()
    {
        Random random = new Random();

        // Заданные границы диапазона для случайных чисел
        int a = -50;
        int b = 50;
        int n = 20; // Количество случайных чисел в списке

        // Создание списка случайных чисел из диапазона [a, b]
        List<int> numbers = Enumerable.Range(0, n) // Генерация списка из n элементов
                                      .Select(_ => random.Next(a, b + 1)) // Заполнение случайными числами
                                      .ToList();

        // Вывод списка случайных чисел
        Console.WriteLine("Список случайных чисел:");
        numbers.ForEach(num => Console.Write(num + " "));
        Console.WriteLine();

        // 1. Создание списка из чисел, кратных 3, и сортировка его по возрастанию
        var multiplesOf3 = numbers.Where(num => num % 3 == 0) // Отбираем только числа, кратные 3
                                  .OrderBy(num => num) // Сортируем по возрастанию
                                  .ToList();
        Console.WriteLine("\nЭлементы, кратные 3, отсортированные по возрастанию:");
        multiplesOf3.ForEach(num => Console.Write(num + " "));
        Console.WriteLine();

        // 2. Нахождение суммы элементов, которые кратны 4
        var sumMultiplesOf4 = numbers.Where(num => num % 4 == 0) // Фильтруем числа, кратные 4
                                     .Sum(); // Суммируем их
        Console.WriteLine("\nСумма элементов, кратных 4: " + sumMultiplesOf4);

        // 3. Подсчет количества элементов, которые положительные и не превосходят 20
        var countPositiveNotGreaterThan20 = numbers.Count(num => num > 0 && num <= 20); // Фильтруем положительные числа, которые не превосходят 20
        Console.WriteLine("\nКоличество положительных элементов, которые не превосходят 20: " + countPositiveNotGreaterThan20);

        // 4. Нахождение минимального по модулю элемента
        var minAbsValue = numbers.Select(num => Math.Abs(num)) // Преобразуем в список их абсолютных значений
                                 .Min(); // Находим минимальное значение
        Console.WriteLine("\nМинимальный по модулю элемент: " + minAbsValue);

        // 5. Проверка наличия отрицательных элементов, кратных 5
        bool hasNegativeDivisibleBy5 = numbers.Any(num => num < 0 && num % 5 == 0); // Проверяем наличие отрицательных элементов, кратных 5
        Console.WriteLine("\nЕсть ли отрицательные элементы, кратные 5: " + (hasNegativeDivisibleBy5 ? "Да" : "Нет"));

        // 6. Нахождение первого нечетного элемента
        var firstOdd = numbers.FirstOrDefault(num => num % 2 != 0); // Находим первый нечетный элемент
        if (firstOdd != 0) // Если элемент не равен 0, значит, он найден
        {
            Console.WriteLine("\nПервый нечетный элемент: " + firstOdd);
        }
        else
        {
            Console.WriteLine("\nОтсутствуют нечетные элементы.");
        }

        // 7. Элементы, кратные 5
        var multiplesOf5 = numbers.Where(num => num % 5 == 0) // Отбираем числа, кратные 5
                                  .OrderBy(num => num) // Сортируем по возрастанию
                                  .ToList();
        Console.WriteLine("\nЭлементы, кратные 5, отсортированные по возрастанию:");
        multiplesOf5.ForEach(num => Console.Write(num + " "));
        Console.WriteLine();
    }
}
```

Практическое задание 35

Определите делегат, принимающий параметр целого типа и возвращающий значение логического типа. Создайте массив целых чисел из *n* элементов, заполненный случайными числами из диапазона [-20, 20].
Определите методы Print (выводит элементы массива), Sum (подсчитывает сумму элементов), которые в качестве первого параметра принимают массив, а в качестве второго – соответствующее лямбда-выражение.

Используя лямбда-выражения как аргументы этих методов:

- вывести: 1) все элементы массива; 2) четные элементы массива;
- подсчитать сумму отрицательных нечетных элементов массива.

```
using System;
using System.Linq;

class Program
{
    // Делегат, принимающий целое число и возвращающий логическое значение
    delegate bool MyPredicate(int num);

    // Метод для вывода элементов массива, фильтруемых лямбда-выражением
    static void Print(int[] array, MyPredicate predicate)
    {
        // Проходим по всем элементам массива
        foreach (var num in array)
        {
            // Если элемент соответствует условию, заданному лямбда-выражением (predicate)
            if (predicate(num))
            {
                // Выводим элемент
                Console.Write(num + " ");
            }
        }
        // Переход на новую строку после вывода элементов
        Console.WriteLine();
    }

    // Метод для подсчета суммы элементов массива, удовлетворяющих условию лямбда-выражения
    static int Sum(int[] array, MyPredicate predicate)
    {
        // Используем LINQ для фильтрации массива по условию и затем суммируем элементы
        return array.Where(num => predicate(num)).Sum();
    }

    static void Main()
    {
        // Создаем объект генератора случайных чисел
        Random random = new Random();

        // Размер массива
        int n = 20;

        // Создаем массив случайных чисел в диапазоне от -20 до 20
        int[] numbers = Enumerable.Range(0, n)  // Генерация последовательности от 0 до n-1
                                  .Select(_ => random.Next(-20, 21)) // Для каждого элемента генерируем случайное число
                                  .ToArray(); // Преобразуем результат в массив

        // Выводим все элементы массива
        Console.WriteLine("Все элементы массива:");
        Print(numbers, num => true); // Лямбда-выражение, которое всегда возвращает true (выводим все элементы)

        // Выводим четные элементы массива
        Console.WriteLine("\nЧетные элементы массива:");
        Print(numbers, num => num % 2 == 0); // Лямбда-выражение для проверки, что число четное (num % 2 == 0)

        // Подсчитываем сумму отрицательных нечетных элементов массива
        int sumNegativeOdd = Sum(numbers, num => num < 0 && num % 2 != 0); // Лямбда-выражение для проверки, что число отрицательное и нечетное
        Console.WriteLine("\nСумма отрицательных нечетных элементов массива: " + sumNegativeOdd);
    }
}
```

Практическое задание 36

Создайте текстовый файл целых чисел (значения вводятся с клавиатуры). Проанализировав в программе созданный файл, создайте еще 2 файла:

- содержащий отрицательные числа и их сумму,
- содержащий положительные числа и их сумму.

Выведите содержимое каждого файла на экран.

```
using System.IO; // Добавьте это пространство имен для работы с файлами

using System;
using System.Collections.Generic;
using System.IO; // Подключаем пространство имен для работы с файлами

class Program
{
    static void Main()
    {
        // Создаем и заполняем файл с числами, введенными с клавиатуры
        string filePath = "numbers.txt"; // Указываем путь к файлу, куда будут записаны числа
        List<int> numbers = new List<int>(); // Создаем список для хранения чисел, введенных пользователем

        Console.WriteLine("Введите целые числа. Введите 'end' для завершения:");

        // Ввод чисел с клавиатуры, пока не введено "end"
        while (true)
        {
            string input = Console.ReadLine(); // Считываем строку с клавиатуры
            if (input.ToLower() == "end") // Если введено "end", выходим из цикла
                break;

            if (int.TryParse(input, out int number)) // Пробуем преобразовать введенную строку в целое число
            {
                numbers.Add(number); // Если преобразование успешно, добавляем число в список
            }
            else
            {
                Console.WriteLine("Некорректный ввод, попробуйте снова."); // Если введено не число, выводим сообщение об ошибке
            }
        }

        // Сохраняем все введенные числа в файл
        File.WriteAllLines(filePath, numbers.ConvertAll(n => n.ToString())); // Конвертируем числа в строки и записываем их в файл

        Console.WriteLine("\nЧисла успешно записаны в файл " + filePath); // Выводим сообщение о том, что числа успешно записаны в файл

        // Создаем два списка для хранения отрицательных и положительных чисел
        List<int> negativeNumbers = new List<int>();
```

```
    List<int> positiveNumbers = new List<int>();
    int negativeSum = 0; // Переменная для подсчета суммы отрицательных чисел
    int positiveSum = 0; // Переменная для подсчета суммы положительных чисел

    // Чтение чисел из списка и разделение их на положительные и отрицательные
    foreach (int number in numbers) // Перебираем все числа из списка
    {
        if (number < 0) // Если число отрицательное
        {
            negativeNumbers.Add(number); // Добавляем его в список отрицательных чисел
            negativeSum += number; // Добавляем число к общей сумме отрицательных чисел
        }
        else if (number > 0) // Если число положительное
        {
            positiveNumbers.Add(number); // Добавляем его в список положительных чисел
            positiveSum += number; // Добавляем число к общей сумме положительных чисел
        }
    }

    // Записываем отрицательные числа и их сумму в файл
    string negativeFilePath = "negative_numbers.txt"; // Путь к файлу для отрицательных чисел
    File.WriteAllLines(negativeFilePath, negativeNumbers.ConvertAll(n => n.ToString())); // Записываем все отрицательные числа в файл
    File.AppendAllText(negativeFilePath, "\nСумма отрицательных чисел: " + negativeSum); // Добавляем в файл сумму отрицательных чисел

    // Записываем положительные числа и их сумму в файл
    string positiveFilePath = "positive_numbers.txt"; // Путь к файлу для положительных чисел
    File.WriteAllLines(positiveFilePath, positiveNumbers.ConvertAll(n => n.ToString())); // Записываем все положительные числа в файл
    File.AppendAllText(positiveFilePath, "\nСумма положительных чисел: " + positiveSum); // Добавляем в файл сумму положительных чисел

    // Вывод содержимого файла с отрицательными числами и их суммой на экран
    Console.WriteLine("\nСодержимое файла с отрицательными числами и их суммой:");
    Console.WriteLine(File.ReadAllText(negativeFilePath)); // Читаем и выводим содержимое файла с отрицательными числами

    // Вывод содержимого файла с положительными числами и их суммой на экран
    Console.WriteLine("\nСодержимое файла с положительными числами и их суммой:");
    Console.WriteLine(File.ReadAllText(positiveFilePath)); // Читаем и выводим содержимое файла с положительными числами
  }
}
```