МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

Отчет к практической работе № 8

Тема: **«Конструктор класса. Перегрузка методов класса»**

Дисциплина: **«Алгоритмы и структуры данных»**

|  |  |  |
| --- | --- | --- |
|  |  | **Выполнил:** |
|  |  | Студент группы ПИЖ-б-о-23-1, направление подготовки: 09.03.04 «Программная инженерия»  Панчешный Александр Алексеевич  **Проверил:**  Доцент ДЦРСИЭ  Николаев Евгений Иванович |

Ставрополь 2025

**Лабораторная работа 8.** **Конструктор класса. Перегрузка методов класса**

**Цель работы:** понять принципы работы конструктора

**Ход работы**

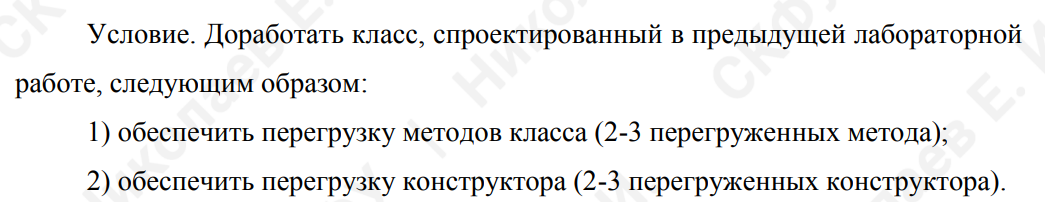


Рисунок 1. Учебная задача

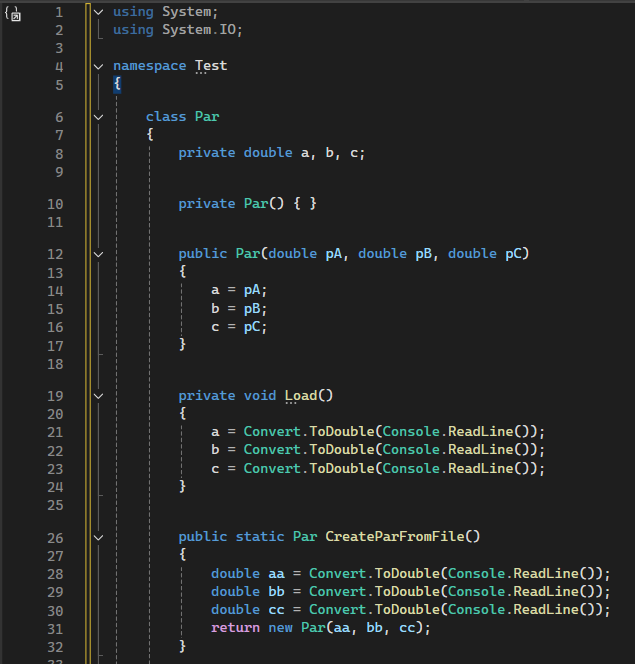




Рисунок 2. Доработка класса Par

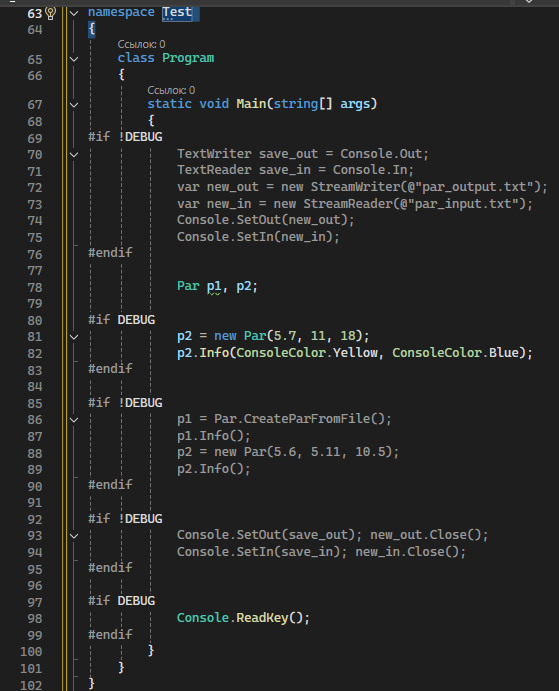
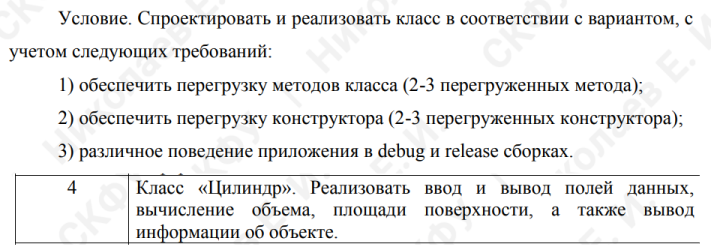


Рисунок 3. Использование класса Par с перегруженными методами

**I. Задание и исходный код индивидуального задания:**



using System;

using System.IO;

namespace CylinderApp

{

class Cylinder

{

private double radius;

private double height;

// Конструкторы

public Cylinder()

{

radius = 1;

height = 1;

}

public Cylinder(double radius)

{

this.radius = radius;

this.height = 1;

}

public Cylinder(double radius, double height)

{

this.radius = radius;

this.height = height;

}

// Методы ввода данных

public void Input()

{

#if DEBUG

Console.WriteLine("Введите радиус цилиндра:");

radius = double.Parse(Console.ReadLine());

Console.WriteLine("Введите высоту цилиндра:");

height = double.Parse(Console.ReadLine());

#else

try

{

using (StreamReader reader = new StreamReader("input.txt"))

{

radius = double.Parse(reader.ReadLine());

height = double.Parse(reader.ReadLine());

}

}

catch (Exception ex)

{

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

}

#endif

}

public void Input(double radius, double height)

{

this.radius = radius;

this.height = height;

}

// Методы вывода данных

public void Output()

{

#if DEBUG

Console.WriteLine($"Радиус: {radius}, Высота: {height}");

#else

try

{

using (StreamWriter writer = new StreamWriter("output.txt"))

{

writer.WriteLine($"{radius} {height}");

}

}

catch (Exception ex)

{

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

}

#endif

}

public void Output(string filePath)

{

try

{

using (StreamWriter writer = new StreamWriter(filePath))

{

writer.WriteLine($"Радиус: {radius}, Высота: {height}");

}

}

catch (Exception ex)

{

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

}

}

// Методы вычисления

public double CalculateVolume()

{

return Math.PI \* Math.Pow(radius, 2) \* height;

}

public double CalculateSurfaceArea()

{

return 2 \* Math.PI \* radius \* (radius + height);

}

// Метод вывода информации

public void Info()

{

Console.WriteLine("Информация о цилиндре:");

Console.WriteLine($"Радиус: {radius}");

Console.WriteLine($"Высота: {height}");

Console.WriteLine($"Объем: {CalculateVolume()}");

Console.WriteLine($"Площадь поверхности: {CalculateSurfaceArea()}");

}

}

class Program

{

static void Main(string[] args)

{

Cylinder cylinder1 = new Cylinder();

cylinder1.Input();

cylinder1.Output();

cylinder1.Info();

Cylinder cylinder2 = new Cylinder(5);

cylinder2.Info();

Cylinder cylinder3 = new Cylinder(3, 10);

cylinder3.Output("cylinder\_info.txt");

Console.ReadKey();

}

}

}

**II. Контрольные вопросы**

**1. Что такое статический класс? Что такое статический метод?**

* **Статический класс:** Это класс, который не может быть инстанцирован (создан как объект). Он может содержать только статические члены (поля, методы, свойства). Статические классы используются для группировки связанных методов и полей, которые не требуют создания экземпляра класса.
* **Статический метод:** Это метод, который принадлежит самому классу, а не его экземплярам. Он может быть вызван без создания объекта класса. Статические методы используются для реализации функций, которые не зависят от состояния конкретного объекта.

**2. Что такое конструктор класса?**

**Ответ:**

Конструктор класса - это специальный метод, который вызывается при создании нового объекта класса. Он используется для инициализации полей объекта и выполнения других необходимых действий при создании объекта. Конструктор имеет имя, совпадающее с именем класса, и не имеет возвращаемого значения.

**3. Что такое перегрузка методов?**

Перегрузка методов - это возможность создания нескольких методов с одним и тем же именем в одном классе, но с разными параметрами. Компилятор выбирает, какой метод вызвать, на основе типов и количества аргументов, переданных при вызове метода.

**4. Может ли один конструктор класса вызывать другой конструктор? Прежде чем отвечать попробуйте реализовать такой вызов в своем разработанном классе.**

Да, один конструктор класса может вызывать другой конструктор с помощью ключевого слова this. Это позволяет избежать дублирования кода при инициализации объекта.

**Пример:**

public class MyClass

{

private int x;

private int y;

public MyClass() : this(0, 0) // Вызов конструктора с двумя параметрами

{

}

public MyClass(int x, int y)

{

this.x = x;

this.y = y;

}

}

**5. Сколько перегрузок может иметь метод класса?**

Метод класса может иметь сколько угодно перегрузок, пока каждая перегрузка имеет уникальную сигнатуру (разное количество или типы параметров).

**6. Для каких целей следует перегружать конструктор класса?**

Перегрузка конструкторов класса используется для:

* Предоставления различных способов инициализации объекта с разными наборами параметров.
* Установки значений по умолчанию для полей объекта.
* Вызова других конструкторов для избежания дублирования кода.

**7. Что такое сигнатура метода?**

Сигнатура метода - это комбинация имени метода и типов его параметров. Сигнатура метода используется компилятором для различения перегруженных методов.

**8. Какие ошибки присутствуют в объявлении класса?**

public class Vector

{

float x;

float y;

float z;

Vector(float pX, float pY, float pZ)

{

xpx; ypY; zpZ;

}

public Vector() : this(0, 0, 0)

{

}

}

**Ответ:**

* **Опечатки в конструкторе Vector(float pX, float pY, float pZ):** xpx, ypY, zpZ должны быть x = pX, y = pY, z = pZ.
* **Отсутствует модификатор доступа для конструктора Vector(float pX, float pY, float pZ):** Конструкторы должны иметь модификатор доступа, обычно public.
* **Поля x, y, z объявлены без модификатора доступа:** По умолчанию поля класса имеют модификатор доступа private, поэтому они доступны только внутри класса Vector.

**9. Дано определение класса:**

public class Человек

{

int Полныхлет;

string ФИО;

public Человек(int pAge, string pFIO)

{

Полныхлет = pAge;

ФИО = pFIO;

}

}

**и фрагмент использования класса в функции Main():**

static void Main(string[] args)

{

Человек чел = new Человек(3, "");

чел.Полныхлет = 5;

чел.ФИО = "Николаев Е.И.";

}

**Вопрос:** Какие ошибки компиляции существуют в коде?

**Ответ:**

* **Отсутствует модификатор доступа для поля Полныхлет:** По умолчанию поля класса имеют модификатор доступа private, поэтому поле Полныхлет доступно только внутри класса Человек. В методе Main происходит попытка обратиться к этому полю напрямую, что вызовет ошибку компиляции.
* **Несоответствие типов в конструкторе:** В конструкторе Человек параметр pAge имеет тип int, а поле Полныхлет также имеет тип int. Однако, в строке Полныхлет = pAge; происходит попытка присвоить значение pAge полю Полныхлет без ошибки, так как типы совпадают.

**10. Укажите ошибки (если есть) в объявлении класса:**

class Person {

private String name;

private int age;

public Person(String pName, int pAge)

{

name = pName;

age = pAge;

}

public static String GetName()

{

return name;

}

}

**Ответ:**

* **Ошибка в методе GetName():** Метод GetName() объявлен как static, но пытается вернуть нестатическое поле name. Статические методы могут обращаться только к статическим членам класса.
* **Использование String вместо string:** В C# рекомендуется использовать string (псевдоним для System.String) вместо String. Хотя оба варианта работают, string считается более предпочтительным.