

ЛАБОРАТОРНАЯ РАБОТА №2

Разработка классов для работы с табулированными функциями.

**по курсу
Объектно-ориентированное программирование**

Группа 6204-010302D

Студент: Е.Д.Васильев.

**Преподаватель: Борисов Дмитрий
Сергеевич.**

Задание 1: Создание пакета functions

Ход выполнения: Создан пакет functions для организации структуры проекта и логической группировки связанных классов.

Результат: Пакет functions создан

Задание 2: Создание класса FunctionPoint

Ход выполнения: Создан класс FunctionPoint, объект которого описывает одну точку табулированной функции с координатами (x, y). Объявлены поля private x и private y. Было реализовано 3 конструктора. Реализованы геттеры и сеттеры, необходимые для реализации инкапсуляции.

Конструкторы:

- FunctionPoint(double x, double y) - точка с заданными координатами
- FunctionPoint(FunctionPoint point) - конструктор копирования
- FunctionPoint() - конструктор по умолчанию

Результат: Класс успешно создан. Соблюден принцип инкапсуляции.

Задание 3: Создание класса TabulatedFunction

Ход выполнения: Создан класс для работы с табулированными функциями. Точки хранятся в массиве, упорядоченном по координате x. Реализованы 2 конструктора.

Конструкторы:

- TabulatedFunction(double leftX, double rightX, int pointsCount) - значения функции равны 0.
- TabulatedFunction(double leftX, double rightX, double[] values) - заданные значения функции

Результат: Конструкторы корректно создают объекты табулированных функций.

Задание 4: Методы работы с функцией

Ход выполнения: Реализованы методы для работы с областью определения и вычисления значений функции.

Методы:

- `getLeftDomainBorder()` - левая граница области определения
- `getRightDomainBorder()` - правая граница области определения
- `getFunctionValue(double x)` - значение функции в точке с помощью линейной интерполяции

Результат: Реализованные методы корректно работают, включая обработку точек вне области определения.

Задание 5: Методы для работы с точками

Ход выполнения: Реализованы методы для доступа и модификации точек функции.

Методы:

- `getPointsCount()` - количество точек
- `getPoint(int index)` - копия точки (инкапсуляция)
- `setPoint(int index, FunctionPoint point)` - замена точки
- Геттеры и сеттеры для координат x и y

Результат: Обеспечены: модификация точек с сохранением порядка и безопасный доступ к данным.

Задание 6: Изменение количества точек

Ход выполнения: Реализованы методы для изменения структуры табулированной функции.

Методы:

- `deletePoint(int index)` - удаление точки
- `addPoint(FunctionPoint point)` - добавление точки

Результат: Методы эффективно работают с оптимизацией использования памяти. С помощью них можно добавлять и удалять точки.

Задание 7: Тестирование

Ход выполнения: Создан класс Main для тестирования функциональности.

Что было сделано в main:

1. Создание функции $y = x^2 + 2$
2. Проверка значений в различных точках
3. Тестирование интерполяции
4. Проверка операций с точками
- 5.

Вывод значений в консоль:

(0,0; 2,0)

(2,5; 8,3)

(5,0; 27,0)

(7,5; 58,3)

(10,0; 102,0)

Проверка значений функции в различных точках

$f(-5,0) = \text{NaN}$

$f(0,0) = 2,0$

$f(2,5) = 8,3$

$f(5,0) = 27,0$

$f(7,5) = 58,3$

$f(10,0) = 102,0$

$f(12,0) = \text{NaN}$

Замена значения функции во второй точке на 100

(0,0; 2,0)

(2,5; 100,0)

(5,0; 27,0)

(7,5; 58,3)

(10,0; 102,0)

Добавляем новую точку (5.5; 30.25)

(0,0; 2,0)

(2,5; 100,0)

(5,0; 27,0)

(5,5; 30,3)

(7,5; 58,3)

(10,0; 102,0)

Удаляем точку с индексом 2

(0,0; 2,0)

(2,5; 100,0)

(5,0; 27,0)

(5,5; 30,3)

(10,0; 102,0)