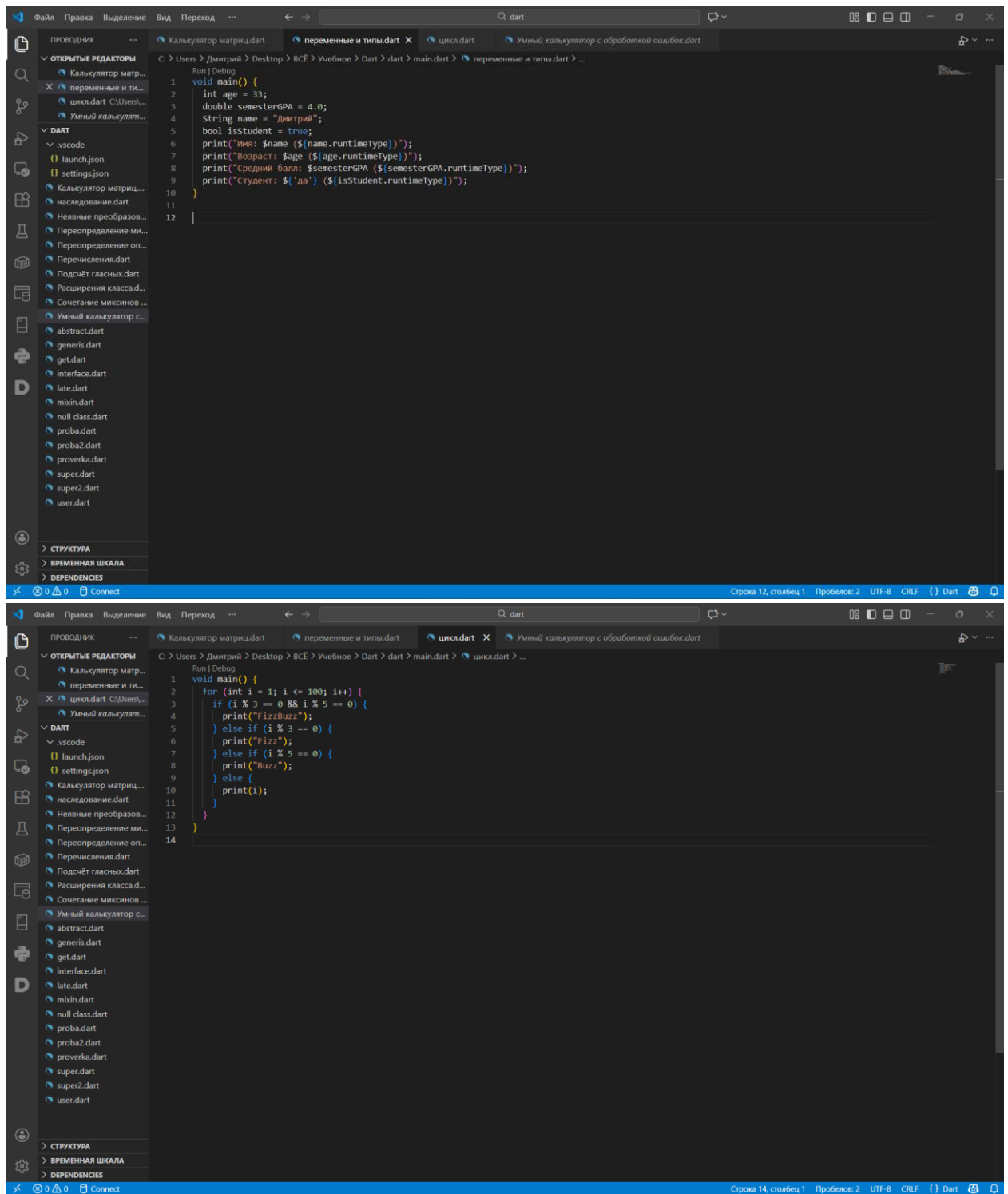
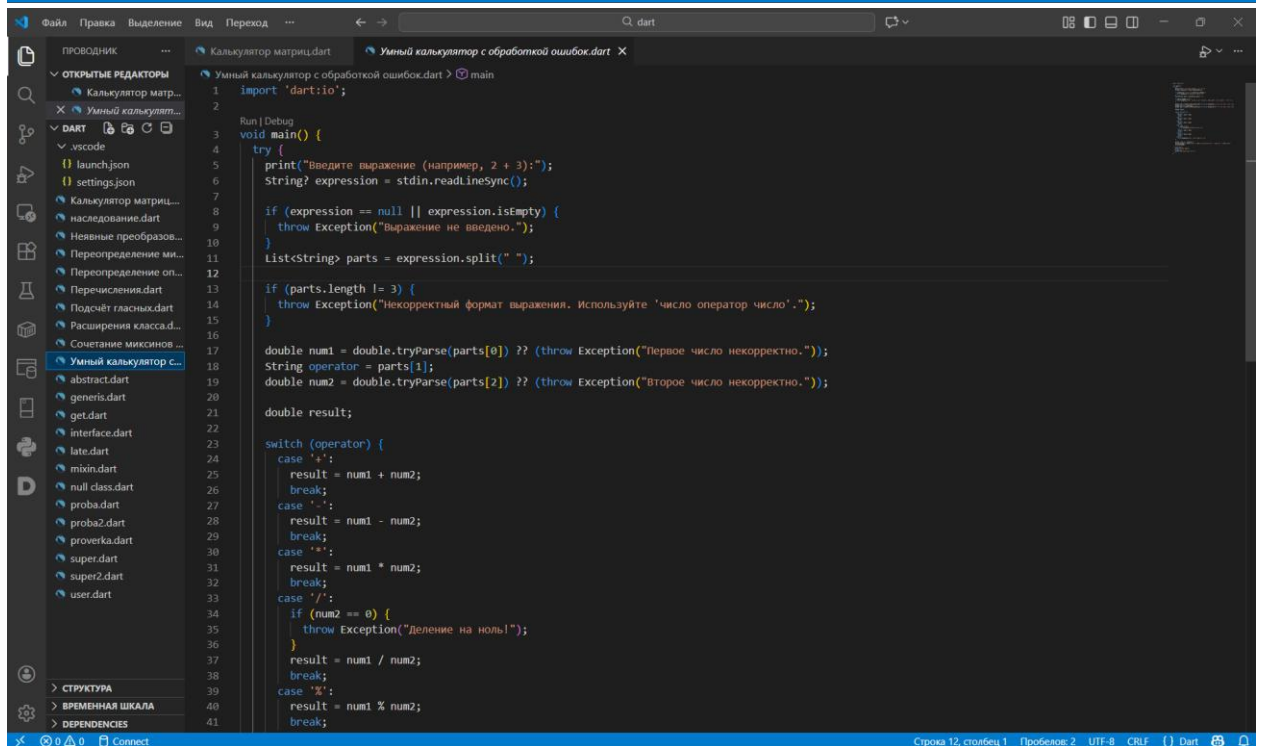
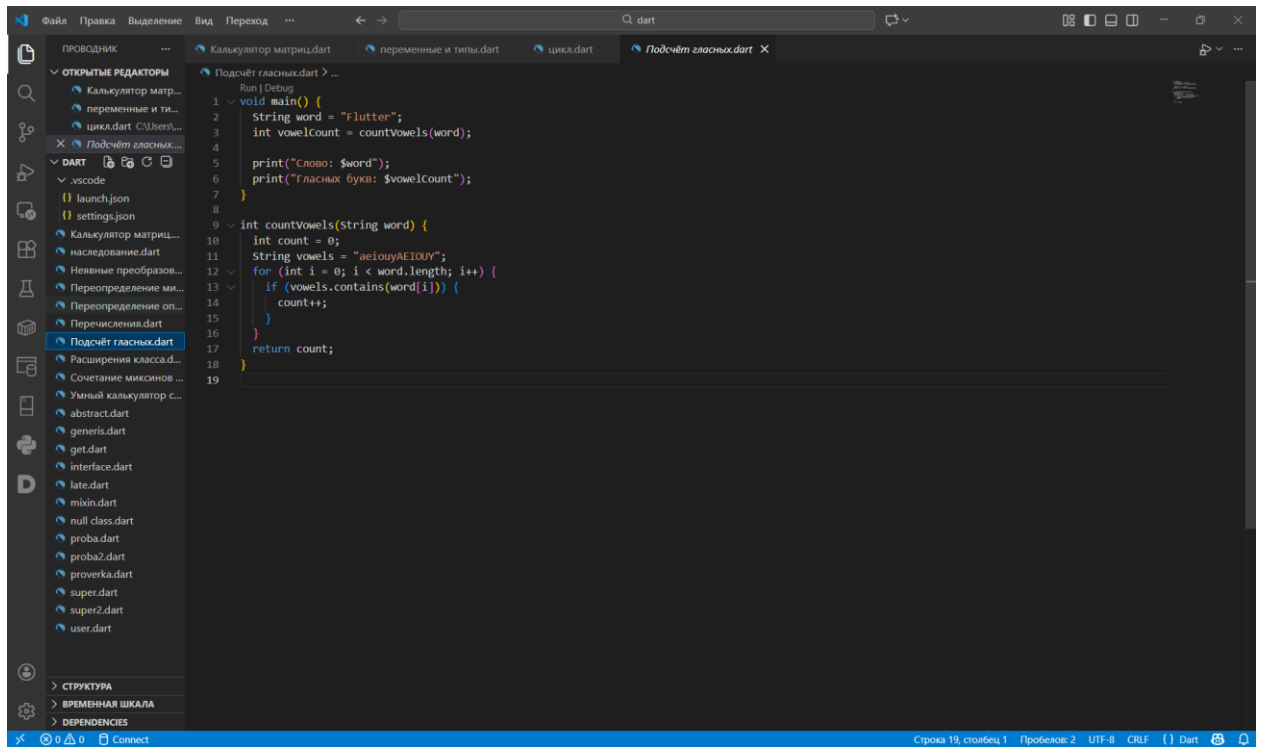


## Практическая работа №1 Знакомство с Dart и Flutter.

Красноперов Дмитрий еозИСИП-207(307)





The screenshot shows the VS Code editor with two files open: `Калькулятор матриц.dart` and `Умный калькулятор с обработкой ошибок.dart`. The active file is `Умный калькулятор с обработкой ошибок.dart`, which contains the following Dart code:

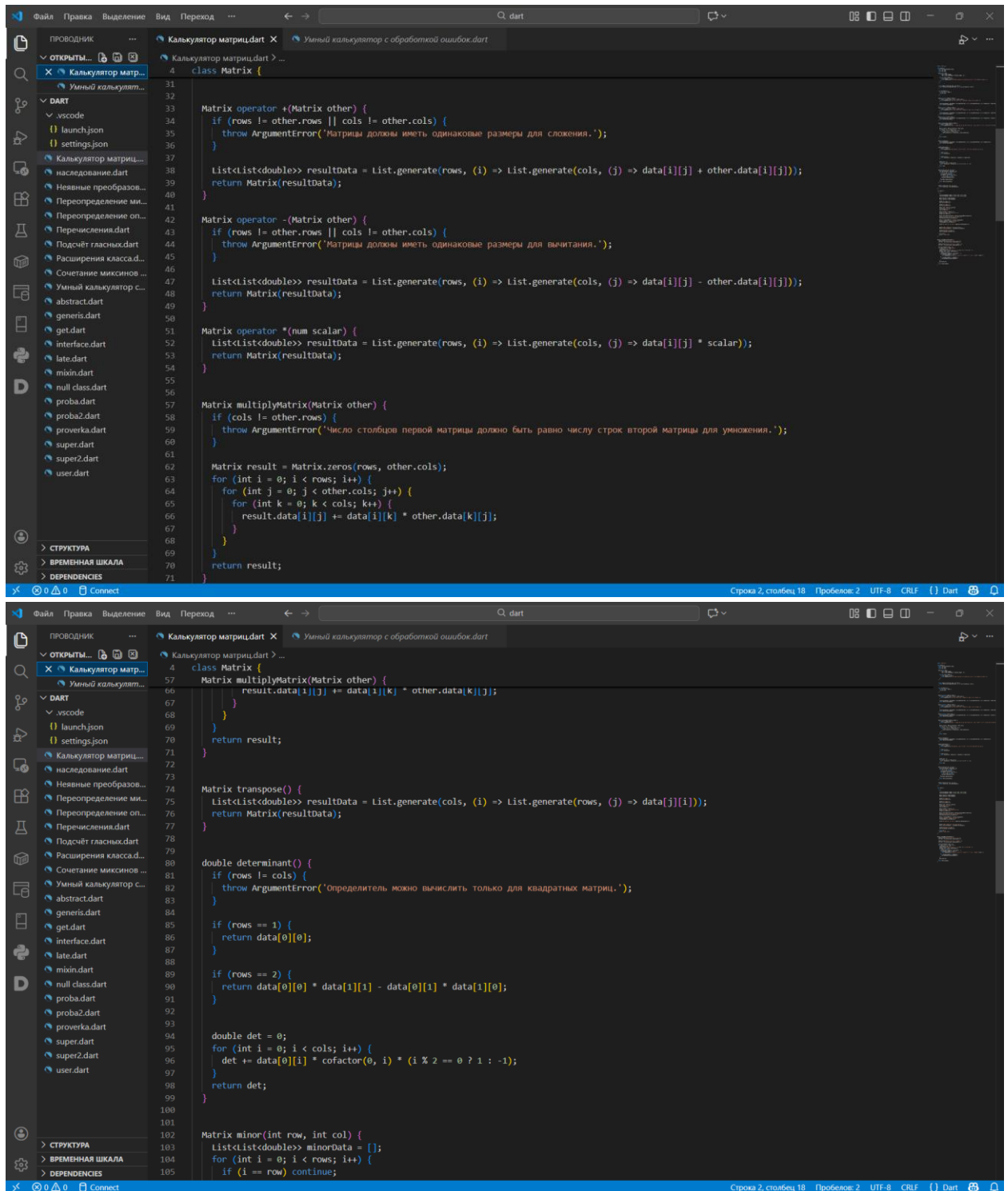
```
void main() {  
    double result;  
  
    switch (operator) {  
        case '+':  
            result = num1 + num2;  
            break;  
        case '-':  
            result = num1 - num2;  
            break;  
        case '*':  
            result = num1 * num2;  
            break;  
        case '/':  
            if (num2 == 0) {  
                throw Exception("Деление на ноль!");  
            }  
            result = num1 / num2;  
            break;  
        case '%':  
            result = num1 % num2;  
            break;  
        default:  
            throw Exception("Некорректный оператор!");  
    }  
  
    print("Результат: $result");  
    String message = result >= 0 ? "Результат положительный" : "Результат отрицательный";  
    print(message);  
}  
  
} catch (e) {  
    print("Ошибка: ${e}");  
} finally {  
    print("Выполнение завершено.");  
}
```

The status bar at the bottom indicates "Строка 12, столбец 1 Пробелов: 2 UTF-8 CRLF () Dart".

The screenshot shows the VS Code editor with two files open: `Калькулятор матриц.dart` and `Умный калькулятор с обработкой ошибок.dart`. The active file is `Калькулятор матриц.dart`, which contains the following Dart code:

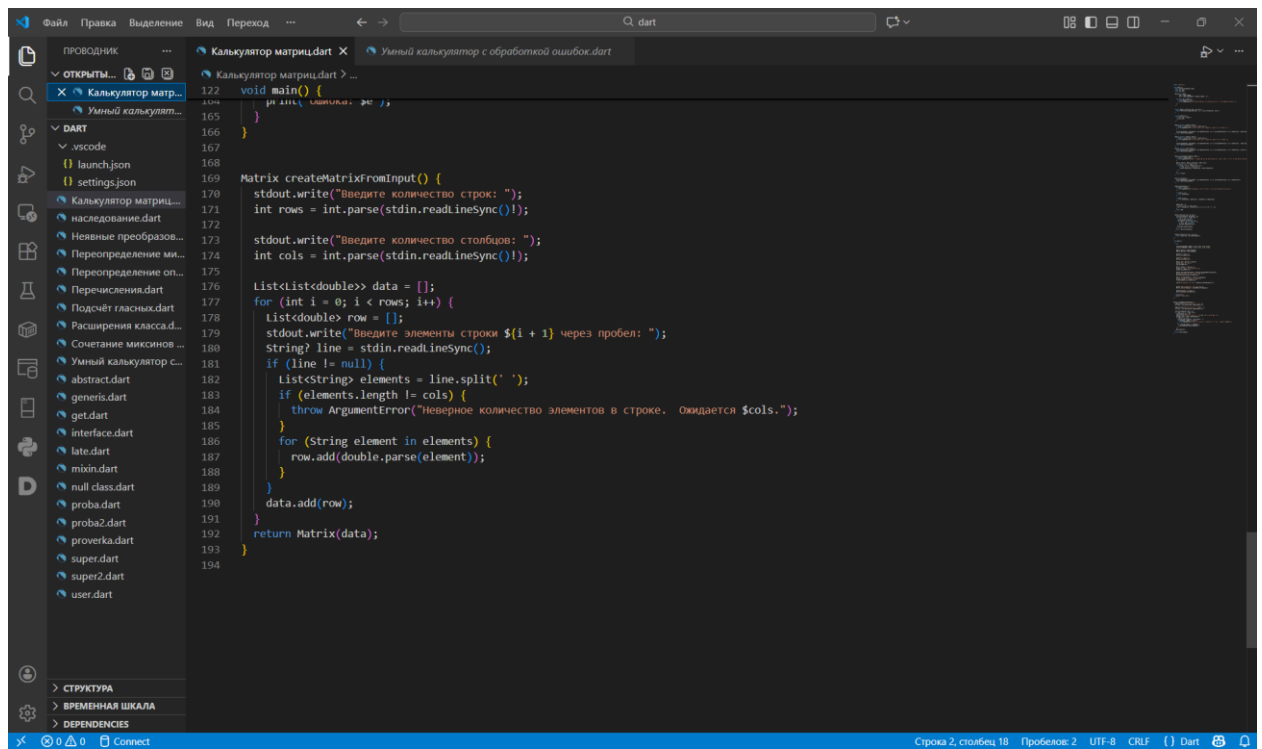
```
import 'dart:io';  
  
class Matrix {  
    final List<List<double>> data;  
    final int rows;  
    final int cols;  
  
    Matrix(this.data)  
        : rows = data.length,  
          cols = data.isNotEmpty ? data[0].length : 0 {  
        for (var row in data) {  
            if (row.length != cols) {  
                throw ArgumentError("Матрица должна быть прямоугольной (все строки одинаковой длины).");  
            }  
        }  
    }  
  
    factory Matrix.zeros(int rows, int cols) {  
        return Matrix(List.generate(rows, (_) => List.filled(cols, 0.0)));  
    }  
  
    void printMatrix() {  
        for (var row in data) {  
            print(row);  
        }  
    }  
  
    Matrix operator +(Matrix other) {  
        if (rows != other.rows || cols != other.cols) {  
            throw ArgumentError("Матрицы должны иметь одинаковые размеры для сложения.");  
        }  
  
        List<List<double>> resultData = List.generate(rows, (i) => List.generate(cols, (j) => data[i][j] + other.data[i][j]));  
        return Matrix(resultData);  
    }  
  
    Matrix operator -(Matrix other) {  
        if (rows != other.rows || cols != other.cols) {  
            throw ArgumentError("Матрицы должны иметь одинаковые размеры для сложения.");  
        }  
  
        List<List<double>> resultData = List.generate(rows, (i) => List.generate(cols, (j) => data[i][j] - other.data[i][j]));  
        return Matrix(resultData);  
    }  
}
```

The status bar at the bottom indicates "Строка 2, столбец 18 Пробелов: 2 UTF-8 CRLF () Dart".



```
class Matrix {  
    Matrix minor(int row, int col) {  
        List<List<double>> minorData = [];  
        for (int i = 0; i < rows; i++) {  
            if (i == row) continue;  
            List<double> minorRow = [];  
            for (int j = 0; j < cols; j++) {  
                if (j == col) continue;  
                minorRow.add(data[i][j]);  
            }  
            minorData.add(minorRow);  
        }  
        return Matrix(minorData);  
    }  
  
    double cofactor(int row, int col) {  
        return minor(row, col).determinant();  
    }  
}  
  
void main() {  
    try {  
        List<List<double>> data1 = [[1.0, 2.0], [3.0, 4.0]];  
        List<List<double>> data2 = [[5.0, 6.0], [7.0, 8.0]];  
  
        Matrix matrix1 = Matrix(data1);  
        Matrix matrix2 = Matrix(data2);  
  
        print("Матрица 1:");  
        matrix1.printMatrix();  
  
        print("Матрица 2:");  
        matrix2.printMatrix();  
  
        Matrix sum = matrix1 + matrix2;  
        print("Сумма матриц:");  
        sum.printMatrix();  
    } catch (e) {  
        print("Ошибка: $e");  
    }  
}
```

```
matrix1.printMatrix();  
  
print("Матрица 2:");  
matrix2.printMatrix();  
  
Matrix sum = matrix1 + matrix2;  
print("Сумма матриц:");  
sum.printMatrix();  
  
Matrix product = matrix1 * 2;  
print("Матрица 1, умноженная на 2:");  
product.printMatrix();  
  
Matrix matrixMultiplication = matrix1.multiplyMatrix(matrix2);  
print("Умножение матрицы 1 на матрицу 2:");  
matrixMultiplication.printMatrix();  
  
Matrix transposedMatrix = matrix1.transpose();  
print("Транспонированная матрица 1:");  
transposedMatrix.printMatrix();  
  
print("Определитель матрицы 1: ${matrix1.determinant()}");  
  
print("\nВвод данных для новой матрицы:");  
Matrix matrixFromInput = createMatrixFromInput();  
  
print("Введенная матрица:");  
matrixFromInput.printMatrix();  
  
} catch (e) {  
    print("Ошибка: $e");  
}  
  
Matrix createMatrixFromInput() {  
    stdout.write("Введите количество строк: ");  
    int rows = int.parse(stdin.readLineSync());  
    stdout.write("Введите количество столбцов: ");  
    int cols = int.parse(stdin.readLineSync());  
    List<List<double>> data = [];  
    for (int i = 0; i < rows; i++) {  
        List<double> row = [];  
        for (int j = 0; j < cols; j++) {  
            row.add(double.parse(stdin.readLineSync()));  
        }  
        data.add(row);  
    }  
    return Matrix(data);  
}
```



<https://github.com/Dmitry-Bit8/Mobile-development>