SMART CONTRACT & QA DOCUMENTATION

Contract: Calculator.sol , Structure.sol
Auditor / QA: QA Engineer (Blockchain / EVM)
Audit Type: Manual Review + Functional QA Analysis
Solidity Version: 0.8.30
License: LGPL-3.0-only
Status: QA Documentation – Basic Level

Date: 31/01/2026

---

1. General Overview

This document consolidates the **technical and QA documentation** of two smart contracts developed in Solidity, with the objective of **presenting structure, syntax, data types, events, and basic** functional behaviors on the EVM.

The work approach is oriented toward progressive understanding, where we will eventually use advanced resources that are not yet available, this being the first project in which we work on:

1. Contract structure
2. Execution flow
3. Use of modifiers
4. State management
5. Observability through events
6. QA test case design

1.1 Validation Objective

● Correct Solidity contract structure
● Proper use of data types (uint, int, bool, string, arrays, mappings)
● Expected functional behavior
● Correct event emission
● State persistence and atomicity
● Understanding of visibility (public, internal)
● Clear and traceable QA documentation

1.2 Executive Conclusion

The contracts meet the defined objective, present a clear structure, and allow effective documentation of test cases, execution flows, and Solidity concepts.

The project is suitable as a first example within a progressive series of more complex contracts.

---

2. Scope

2.1 Included

- SPDX License
- Fixed Solidity pragma
- State variables
- Basic and advanced data types
- Modifiers
- Events
- Public and internal functions
- State persistence
- Transaction atomicity
- Manual QA test cases

2.2 Excluded

- Advanced gas optimization
- Access control (roles)
- Pausability
- Upgradeability
- Advanced attacks (reentrancy, MEV)
- Automated testing

All excluded items are out of scope by design, not by omission.

---

3. Methodology

- Line-by-line manual review
- Execution flow analysis
- Validation of pre and post states
- Scenario simulation:

    1. Valid scenarios
    2. Invalid scenarios
    3. Edge cases

- Validation against Solidity ≥ 0.8.x best practices
- Functional QA-focused approach

Contract 1: Calculator.sol

4. Architectural Analysis

4.1 License

- Defined and valid SPDX
- Complies with standards
- Status: Compliant
- Risk: Low

4.2 Solidity Version

- Fixed version: pragma solidity 0.8.30
- Native overflow / underflow protections
- Status: Compliant
- Risk: Low

---

5. State Variables Analysis

Result

- Type: uint256
- Persistent in storage
- Explicit initialization (10)
- Automatic public getter

Use: Demonstrate state persistence and mutability.

---

6. Modifiers

checkNumber(uint256)

Function

- Restricts execution to an exact value (10)
- Demonstrates use of revert
- Applies pre-execution control

Use: Demonstrate modifier control flow.

---

7. Events

Status: Compliant

---

8. Function Analysis

addition (uint256, uint256)

- Deterministic calculation
- Does not modify state
- Emits event

subtraction (uint256, uint256)

- Use of internal function
- Underflow protected by Solidity 0.8.x

subtraction2 (uint256, uint256)

- Demonstrates use of signed integers

multiply (uint256)

- Modifies persistent state
- No range validation

multiply2 (uint256)

● Combined use of modifier + state

---

9. Atomicity

All functions:

● Complete execution or revert
● No intermediate states
● Comply with EVM principles
● Correct post-transaction persistence

Contract 2: Structure.sol

Structure.sol aims to demonstrate the complete structure of a Solidity contract, including:

● Primitive data types
● Arrays
● Mappings
● Structs
● Modifiers
● Events
● Functions with return values

---

10. Components

Numeric Variables:

● uint256, uint16, uint8, uint32
● Use of maximum limits
● Operations between types

Strings:

● Text storage in storage

Booleans:

● Simple logical states

Arrays:

● Static array with initial values

Mapping:

● Key-value association (address → uint)

Struct:

● Definition of composite data structure

---

11. Modifier

Modifier1:

● Pre-execution control
● Logical validation example

---

12. Event

Multiplier:

● Records the result of an operation

---

13. Function Multiplier (uint256, uint256)

● Use of modifier
● Explicit return
● Event emission
● Clear and sequential flow

---

14. Project Structure

/smart-contract-structure/
│
├── contracts/
│ ├── Calculator.sol
│ └── Structure.sol
│
├── docs/

└── smart_contract_&*qa_documentation_es.pdf*
└── *smart_contract*&_qa_documentation_en.pdf
├── qa_test_cases.pdf
└── report_es.pdf
└── report_en.pdf

├── README.md
└── LICENSE

---

15. QA Checklist

- SPDX
- Fixed pragma
- Explicit visibility
- Use of pure / internal
- Events present
- Atomicity
- Correct compilation

---

16. Final Conclusion

Status: OK – Base Level

This repository:

- Is clear
- Is progressive
- Is understandable for QA and developers
- Is documented