**Federal State Autonomous Educational**

**Institution National Research University**

**"Higher School of Economics"**

Moscow Institute of Electronics and Mathematics named after A.N. Tikhonov (MIEM HSE)

Department of Computer Engineering

Course: Algorithms and Programming

**TECHNICAL DOCUMENTATION**

Developer: Poptsova Dasha BIV243

Documentation Language: English

Moscow 2025

**Оглавление**

[**1. General Information** 3](#_Toc200786243)

[**1.1 Purpose of the Product** 3](#_Toc200786244)

[**1.2 Scope of Application** 3](#_Toc200786245)

[**1.3 Development Justification** 3](#_Toc200786246)

[**2. Developer Information** 3](#_Toc200786247)

[**3. Technical Description** 3](#_Toc200786248)

[**3.1 System Architecture** 3](#_Toc200786249)

[**3.2 Implementation Details** 4](#_Toc200786250)

[**3.3 Technologies Used** 4](#_Toc200786251)

[**4. Algorithm Logic** 5](#_Toc200786252)

[**4.1 Commission Calculation Algorithm** 5](#_Toc200786253)

[**4.2 End‑to‑End Example Calculation** 5](#_Toc200786254)

[**5. Safety Considerations** 5](#_Toc200786255)

[**6. Standards Compliance** 5](#_Toc200786256)

[**7. Appendices** 6](#_Toc200786257)

[**8. Development Recommendations** 6](#_Toc200786258)

**1. General Information**

**1.1 Purpose of the Product**

The Bank Transfer Optimizer CLI is a C++‑based console application that computes the minimal commission fee for transferring funds between correspondent banks. It supports:

* Variable commission schemes combining fixed and percentage‑based fees.
* Optional splitting of the transfer amount into up to five equal parts to bypass steep fee thresholds.
* Routing through at most one intermediary bank, constrained by actual correspondent links.

**1.2 Scope of Application**

The tool is intended for:

* Business analysts evaluating intra‑country liquidity routes.
* Back‑office operators who need fee forecasts before executing transfers.
* Academic use in graph‑theory or financial‑optimization courses.

**1.3 Development Justification**

Manual evaluation of commissions across many banks is error‑prone and time‑consuming. Automating the calculation guarantees repeatability, reveals non‑obvious optimal paths, and simplifies what‑if analysis when commission tables change.

**2. Developer Information**

Organization: Investing Company IT Invest

**3. Technical Description**

**3.1 System Architecture**

+---------------------------+

| CSV Data Sources |

| • Banks1.csv |

| • Commissions.csv |

+---------------------------+

↓

+---------------------------+

| Parser Layer |

| • splitCsvLine() |

| • loadBanks() |

| • CommissionTable::load()|

+---------------------------+

↓

+---------------------------+

| Graph Builder |

| • buildGraph() |

| • Edge weight function |

+---------------------------+

↓

+---------------------------+

| Solver |

| • dijkstraLimited() |

| • findBestSplitRoute() |

+---------------------------+

↓

+---------------------------+

| CLI Interface |

| • interactive main() |

+---------------------------+

Each layer is isolated by clear interfaces, facilitating unit testing and future refactoring.

**3.2 Implementation Details**

* **CSV Parsing:** Custom state‑machine parser that honours RFC 4180 quoting rules and escapes "".
* **Data Structures:**
  + std::unordered\_map<std::string, Rules> – commission lookup tables.
  + std::vector<std::vector<Edge>> – adjacency list storing dynamic edge weights.
* **Graph Generation:**
  + A directed edge A → B exists iff the correspondent sets of A and B intersect.
  + Edge weight w(x) = outFee\_A(x) + inFee\_B(x), where x is the (possibly split) part amount.
* **Shortest Path:** Modified Dijkstra that tracks (node, hops) pairs and rejects paths with hops > 2 (≤ 1 intermediary).
* **Splitting Logic:** For each k ∈ [1,5] the solver re‑computes edge weights for x = amount / k, runs Dijkstra once, then multiplies the path cost by k. The global minimum over k is selected.
* **Error Handling:**
  + Unknown bank names → graceful message and non‑zero exit code.
  + Amount ≤ 0 → validation error.
* **Build Flags:** -std=c++17 -O2 -Wall -Wextra -pedantic.

**3.3 Technologies Used**

- C++ Standard Library (vector, map, string, fstream, sstream)

- Linux/Windows terminal

- g++ or clang++ compiler

**4. Algorithm Logic**

# **4.1 Commission Calculation Algorithm**

1. Determine Rule: For a given amount *x*, select the first rule whose threshold ≥ x; if none, take the last rule.
2. Compute Fee: fee = fixedFee + percent × x.
3. Edge Weight: Sum the output fee of the sender bank and the input fee of the receiver bank.
4. Path Cost: Sum of edge weights along the path.
5. Split Evaluation: Repeat steps 1–4 for every k = 1…5, choosing the minimum k × pathCost\_k.

### **4.2 End‑to‑End Example Calculation**

Transfer 1 500 ₽ from Bank B to Bank E

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | Part (₽) | Path | Cost per Part (₽) | Total (₽) |
| 1 | 1 500 | B → H → E | 34 + ∞ ← fails (no split) | 315 |
| 2 | 750 | B → H → E | 34 | 68 |
| 3 | 500 | B → C → E | 32 | 96 |

**5. Safety Considerations**

* **Input Sanitisation:** All user inputs are trimmed, length‑checked, and validated against loaded bank lists.
* **Memory Safety:** Smart pointers avoided; RAII and containers guarantee automatic cleanup.
* **Overflow Prevention:** Amounts stored in double, but validated to 0 < amount ≤ 1e13 to avoid double precision loss.
* **Denial‑of‑Service:** Dijkstra’s priority‑queue operations are bounded by O(E log V); with ≤ 100 banks runtime stays < 1 ms.

**6. Standards Compliance**

The project follows documentation and software standards as follows:

- ESKD – documentation structure and detailed description

- ESPD – algorithmic and logic structure

- ESTD / ESTPP – technical architecture and scalability framework

**7. Appendices**

* **A. CSV Schema Definitions**
  + Banks1.csv: "BankName","Country","Correspondents"
  + Commissions.csv: "BankName","Direction","Threshold","FixedFee","Percent"
* **B. Command‑Line Options**
  + Currently interactive; roadmap includes flags --from, --to, --amount, --max‑split.
* **C. Unit‑Test Coverage Report** (generated by ctest --output‑on‑failure).

**8. Development Recommendations**

1. **Plugin Architecture:** Extract fee‑calculation policy into a strategy interface to support tiered or currency‑dependent fees.
2. **Currency Expansion:** Add Currency column to both CSV files; parameterise commission tables by currency key.
3. **REST API Wrapper:** Provide FastAPI micro‑service around the core solver for integration with web dashboards.
4. **Advanced Splitting:** Implement dynamic‑programming search for arbitrary split vectors, controlled by a time budget.
5. **Cross‑Border Extension:** Re‑introduce Borders.csv with FX spread and border fees; model as multi‑layer graph.