Programmer10k

Legacy Systems Engineering

They say 10,000 hours to mastery.
I have 10,000 days.

45 · years · professional · software · engineering

THE PROBLEM

Decades-old legacy systems — Fortran, COBOL, Pascal, Ada, and dozens of others — run the world's most critical infrastructure. The engineers who built them have retired. The documentation is incomplete. The business rules are buried in undocumented code.

Automated tools fail because they carry 1980s architectural constraints into modern environments. Traditional rewrite teams fail because they lack the historical context to understand *why* the original code was written the way it was.

THE SOLUTION

45 years of professional software engineering. I find the path through legacy systems — Fortran, COBOL, Pascal, Modula-2, Ada, BASIC, and whatever else your critical infrastructure runs on — and I make the modernisation happen, without losing a single line of business logic.

The AI handles volume and speed. I provide the architectural judgment and validation that no automated tool can replicate — the process thinking and historical context built across five decades of practice.

HOW WE WORK — FIVE PHASES, ZERO SURPRISES

01 PAID DISCOVERY	We never quote blind. Fixed-fee Discovery maps dependencies, assesses complexity, and produces a Feasibility Report with a firm fixed-price quote. Fee credited to full project.
02 AI-ASSISTED ANALYSIS	Specialized tooling reads raw syntax, summarizes program logic, and traces execution flow across hundreds of thousands of lines — in hours, not weeks.
03 HUMAN VALIDATION	45 years of pattern recognition applied to the AI's output. Implicit business rules, edge cases, and architectural quirks that automated tools miss are identified and preserved.
04 ARCHITECTURAL REWRITE	Not machine-translated — hand-crafted. AI drafts are refactored into clean, idiomatic, production-ready code following modern design patterns and your organisation's standards.
05 RIGOROUS TESTING	Characterization Tests built from observed legacy behaviour ensure identical outputs for identical inputs. Full unit test suite targeting 80%+ coverage.

WHY PROGRAMMER10K

PROCESS DEPTH
I find the path through problems that look intractable — the same structured methodology, every engagement, zero surprises.

LANGUAGE BREADTH
Fortran, COBOL, Pascal, Modula-2, Ada, BASIC, RPG, PL/I, and more. Whatever your system is written in, I've worked in it.

ACCURACY
Business logic preserved flawlessly. Characterization tests prove it before a single line goes to production.

TRANSPARENCY
Multi-factor pricing means you only pay for the actual complexity of your specific system — confirmed during Discovery, not guessed upfront.

Ready to modernise without the risk?

Start with a Discovery Phase — flat fee, credited to the full project.

prog10k-5f6pcez9.manus.space

Discovery: $1,500–$5,000 flat · Response: 24 business hours · Min. engagement: $5,000

Fortran · COBOL · Pascal · Modula-2 · Ada · BASIC · RPG · PL/I