GW-Basic 64

PRESENTED BY

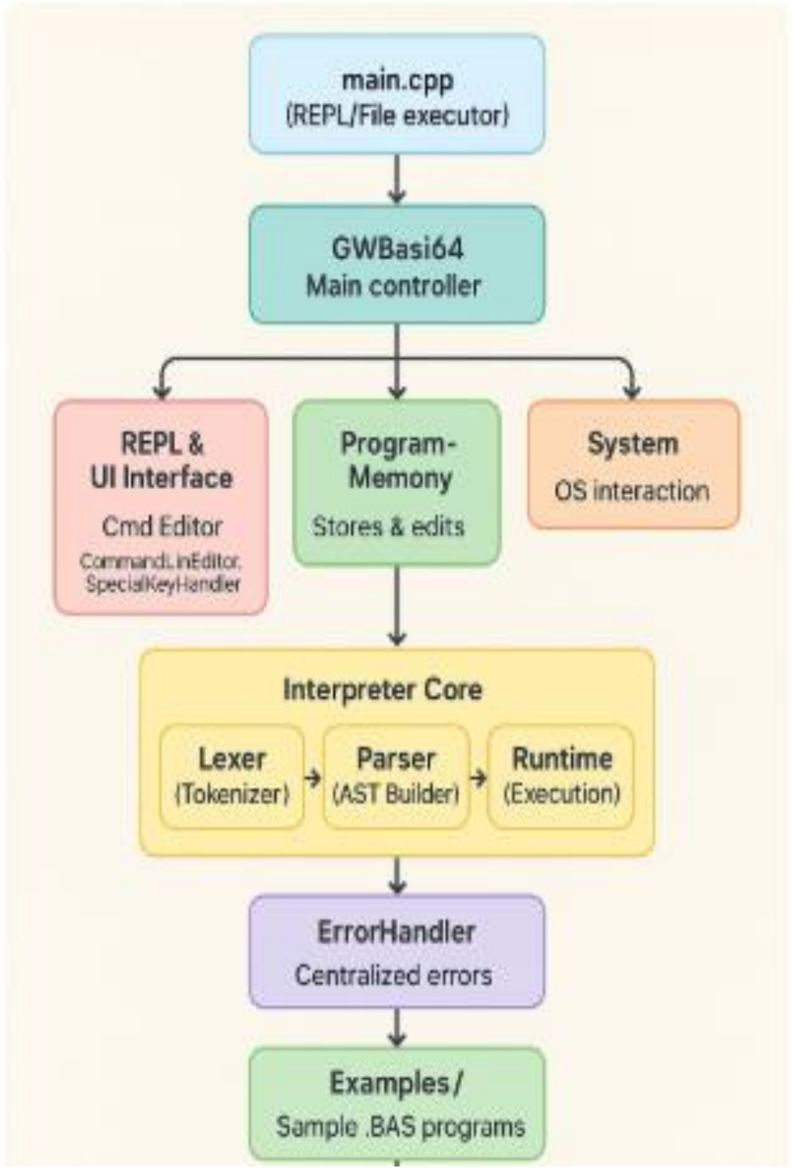
Remin Varghese

EID: 293944

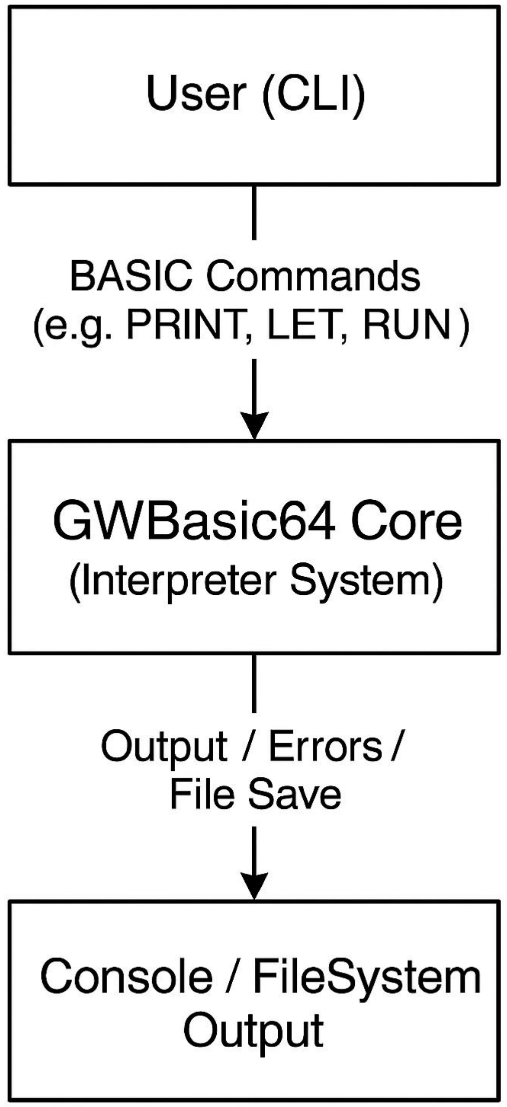
AGENDA

Contributions to – GWBasic64 Project

- Main,
- GWBasic64,
- SystemInterface
- & ErrorHandling Modules



Architectural diagram


User (CLI)
BASIC Commands (e.g. PRINT, LET, RUN)
GWBasic64 Core (Interpreter System)
Output / Errors / File Save
Console / FileSystem Output

Project Overview

Project Ov

What is *gw_basic_64*

A 64-bit variant of the GW-Basic interpreter

Primary goal: Port and extend classic GW-BASIC interpreter with modern architecture

Scope of Integration Task

Connecting modules: *main.cpp*, *gwbasic64.cpp/h*, *systemInterface.cpp/h*, *errorHandler.cpp/h*, plus CMake build files

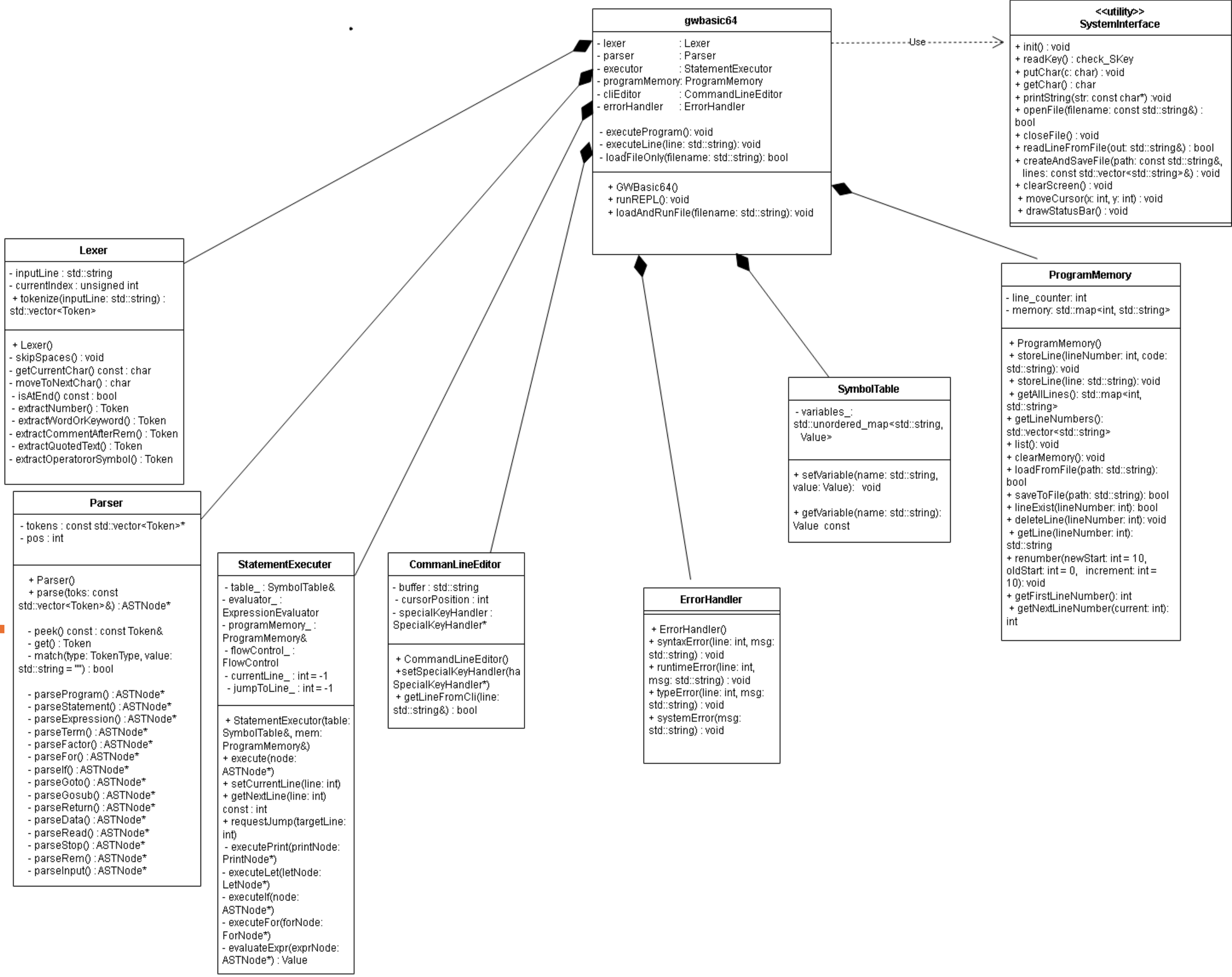
Overview of Modules

- GWBasic64 – a modern interpreter for GW-BASIC-like language.
- Runs in Direct Mode (REPL) and File Mode (.bas).

Modules:

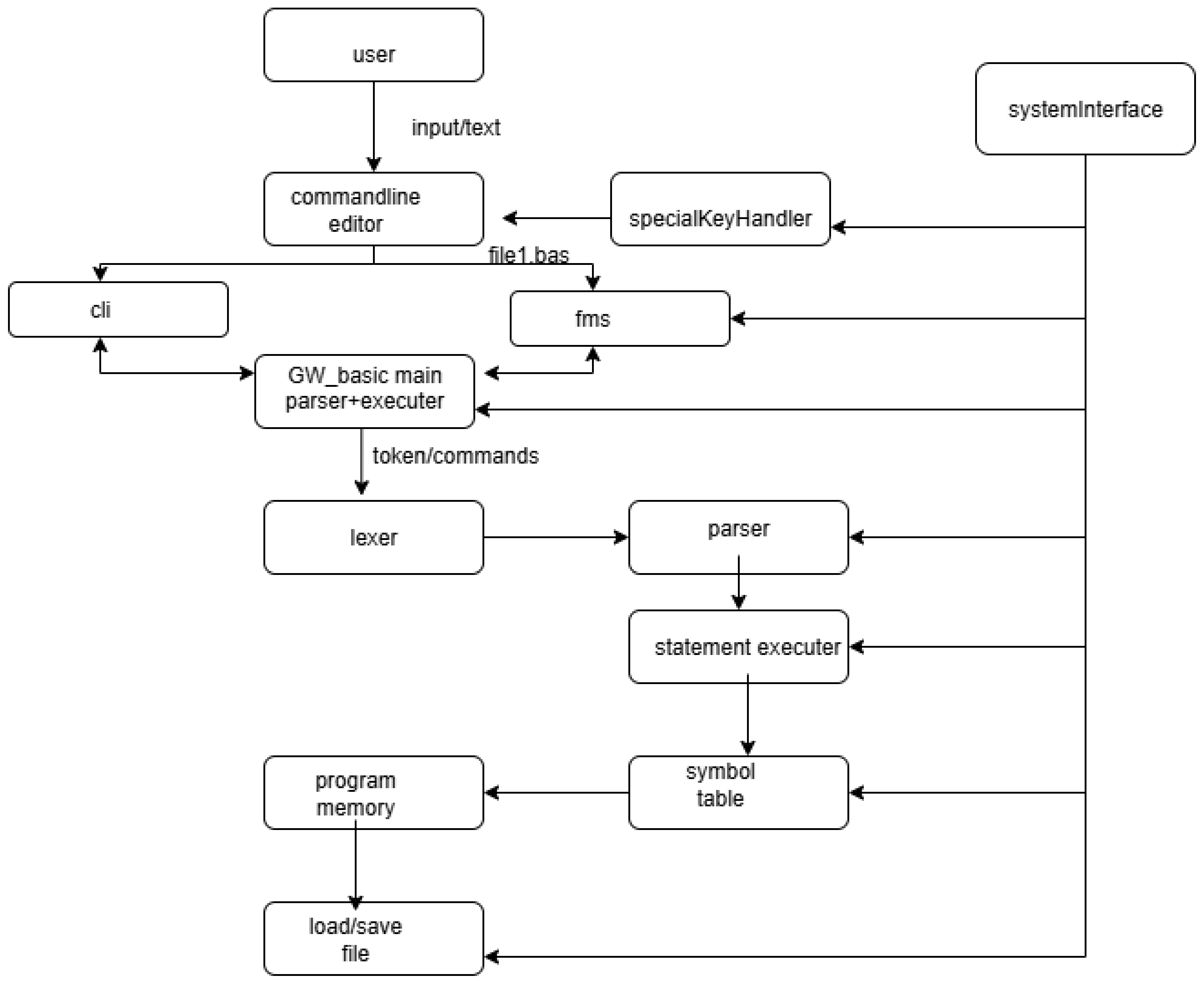
- main.cpp
- GWBasic64 class (core orchestrator)
- SystemInterface (I/O Abstraction)
- ErrorHandler (Error Reporting)

Class Diagram



gwbasic64
- lexer : Lexer
- parser : Parser
- executor : StatementExecutor
- programMemory: ProgramMemory
- cliEditor : CommandLineEditor
- errorHandler : ErrorHandler
- executeProgram(): void
- executeLine(line: std::string): void
- loadFileOnly(filename: std::string): bool
+ GWBasic64()
+ runREPL(): void
+ loadAndRunFile(filename: std::string): void
Use
<<utility>>
SystemInterface
+ init() : void
+ readKey() : check_SKey
+ putChar(c: char) : void
+ getChar() : char
+ printString(str: const char*) :void
+ openFile(filename: const std::string&) : bool
+ closeFile() : void
+ readLineFromFile(out: std::string&) : bool
+ createAndSaveFile(path: const std::string&, lines: const std::vector<std::string>&) : void
+ clearScreen() : void
+ moveCursor(x: int, y: int) : void
+ drawStatusBar() : void
Lexer
- inputLine : std::string
- currentIndex : unsigned int
+ tokenize(inputLine: std::string) : std::vector<Token>
+ Lexer()
- skipSpaces() : void
- getCurrentChar() const : char
- moveToNextChar() : char
- isAtEnd() const : bool
- extractNumber() : Token
- extractWordOrKeyword() : Token
- extractCommentAfterRem() : Token
- extractQuotedText() : Token
- extractOperatororSymbol() : Token
Parser
- tokens : const std::vector<Token>*
- pos : int
+ Parser()
+ parse(toks: const std::vector<Token>&) : ASTNode*
- peek() const : const Token&
- get() : Token
- match(type: TokenType, value: std::string = "") : bool
- parseProgram() : ASTNode*
- parseStatement() : ASTNode*
- parseExpression() : ASTNode*
- parseTerm() : ASTNode*
- parseFactor() : ASTNode*
- parseFor() : ASTNode*
- parseIf() : ASTNode*
- parseGoto() : ASTNode*
- parseGosub() : ASTNode*
- parseReturn() : ASTNode*
- parseData() : ASTNode*
- parseRead() : ASTNode*
- parseStop() : ASTNode*
- parseRem() : ASTNode*
- parseInput() : ASTNode*
StatementExecuter
- table_ : SymbolTable&
- evaluator_ : ExpressionEvaluator
- programMemory_ : ProgramMemory&
- flowControl_ : FlowControl
- currentLine_ : int = -1
- jumpToLine_ : int = -1
+ StatementExecutor(table: SymbolTable&, mem: ProgramMemory&)
+ execute(node: ASTNode*)
+ setCurrentLine(line: int)
+ getNextLine(line: int) const : int
+ requestJump(targetLine: int)
- executePrint(printNode: PrintNode*)
- executeLet(letNode: LetNode*)
- executeIf(node: ASTNode*)
- executeFor(forNode: ForNode*)
- evaluateExpr(exprNode: ASTNode*) : Value
CommanLineEditor
- buffer : std::string
- cursorPosition : int
- specialKeyHandler : SpecialKeyHandler*
+ CommandLineEditor()
+setSpecialKeyHandler(handler: SpecialKeyHandler*)
+ getLineFromCli(line: std::string&) : bool
ErrorHandler
+ ErrorHandler()
+ syntaxError(line: int, msg: std::string) : void
+ runtimeError(line: int, msg: std::string) : void
+ typeError(line: int, msg: std::string) : void
+ systemError(msg: std::string) : void
SymbolTable
- variables_: std::unordered_map<std::string, Value>
+ setVariable(name: std::string, value: Value): void
+ getVariable(name: std::string): Value const
ProgramMemory
- line_counter: int
- memory: std::map<int, std::string>
+ ProgramMemory()
+ storeLine(lineNumber: int, code: std::string): void
+ storeLine(line: std::string): void
+ getAllLines(): std::map<int, std::string>
+ getLineNumbers(): std::vector<std::string>
+ list(): void
+ clearMemory(): void
+ loadFromFile(path: std::string): bool
+ saveToFile(path: std::string): bool
+ lineExist(lineNumber: int): bool
+ deleteLine(lineNumber: int): void
+ getLine(lineNumber: int): std::string
+ renumber(newStart: int = 10, oldStart: int = 0, increment: int = 10): void
+ getFirstLineNumber(): int
+ getNextLineNumber(current: int): int

Data Flow Diagram



Responsibilities

Designed & implemented entry point (main.cpp):

- Handles command-line arguments, initializes system, runs REPL or file execution.

Developed the GWBasic64 class:

- Ties together Lexer, Parser, Executor, ProgramMemory, and CLI.
- Supports commands like RUN, LIST, NEW, SAVE, LOAD, AUTO, DELETE, etc.

Implemented SystemInterface:

- Unified Console I/O, File I/O, and Key Handling for Windows/MiniOS.

Implemented Error Handling Framework:

- Standardized syntax, runtime, type, and system errors.

Design Rationale

Separation of Concerns:

SystemInterface abstracts OS-specific I/O → making it portable between Windows and MiniOS.

GWBasic64 handles modular interpreter logic

Robust Error Handling:

Centralized ErrorHandler prevents crashes, provides user-friendly messages.

Extensibility:

Modular commands (e.g., RUN, LIST) allow new features without major rewrites.

main.cpp Flow

Features:

- Validates input arguments.
- Switches between REPL & File Execution.
- Top-level error catching with try/catch.

Why This Design?

- Keeps the entry point clean & focused.
- Ensures unknown exceptions don’t crash the interpreter.

GWBasic64 Class Responsibilities

Core Functions:

- runREPL(): Interactive shell.
- loadAndRunFile(): Executes .BAS files.
- executeProgram(), executeLine(): Delegates to Lexer, Parser, Executor.

Design Choice:

- Acts as the “controller” in the interpreter’s architecture.

SystemInterface

Provides abstracted platform I/O:

- Keyboard input (detects special keys).
- Console output & status bar.
- File operations: openFile, readLine, saveFile.

Why abstraction?

- Intended to Support MiniOS & Windows without changing interpreter logic.

ErrorHandler Implementation

Centralized error reporting:

- syntaxError(), runtimeError(), typeError(), systemError().

Design Benefit:

- Consistent error messages.
- Clear separation between detection and display.
- Supports Direct Mode (-1) and Program Mode (line numbers).

Key Achievements

- Stability: Robust error catching at all levels.
- User Experience: REPL design with command set & status bar improves usability.
- Extensibility: Modular design allows adding new GW-BASIC commands easily.

Summary & Conclusion

- Objective: seamlessly integrate modules into robust system
- Approach: clear separation, centralized error handling, modular build
- Achievements: functional interpreter, stable integration, clean build

Reference

- GW-BASIC User's Manual.pdfPortable Executable – Wikipedia
- GW-BASIC – Wikipedia
- Microsoft gihub repo archived GW-Basic source code
- Lowlevel.eu
- Wikiosdev.org

THANK YOU