

DSL to C++ Compiler

This project implements a **custom Domain-Specific Language (DSL) compiler** designed to parse and process arithmetic expressions written in a simplified contract-like syntax. The compiler translates the input DSL into equivalent, executable C++ code. It leverages modern compiler construction tools such as **Flex** (for lexical analysis) and **Bison** (for syntax parsing), along with custom components for AST generation, code evaluation, and code generation.

System Requirements

To build and run the DSL-to-C++ compiler, the following tools and libraries are required:

Development Tools

- **CMake** ≥ 3.10

Used for project configuration and cross-platform build automation.

- **Flex**

Lexical analyzer for tokenizing the DSL input code.

- **Bison** ≥ 2.3

Parser generator used to define the grammar and build the Abstract Syntax Tree (AST).

- **C++ Compiler** (with C++11 support)

Required to compile the generated C++ code and internal modules of the compiler.

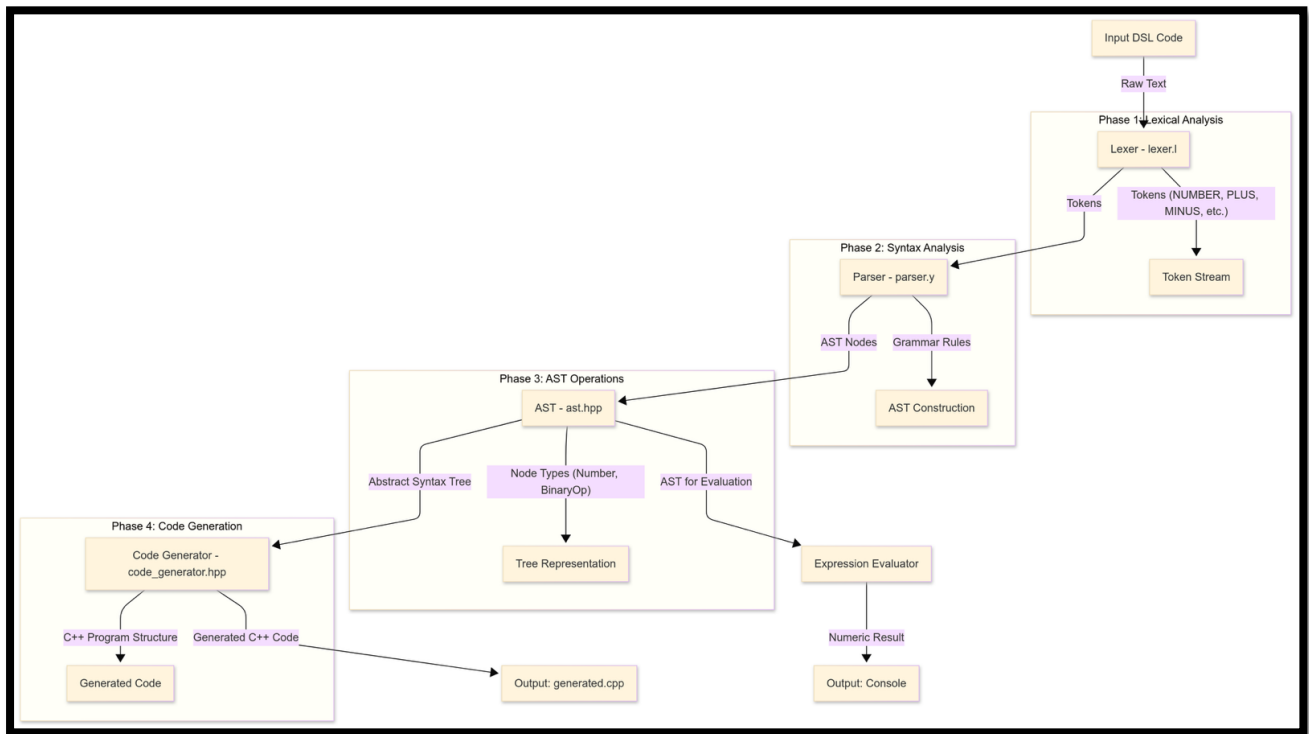
- **Google Test Framework**

Used for unit testing and validation of core components like the lexer, parser, and AST.

PROJECT ARCHITECTURE

```
.
├── CMakeLists.txt           # Main CMake configuration file
├── dsl/                    # Core DSL implementation
│   ├── ast.hpp             # Abstract Syntax Tree definitions
│   └── codegen/            # Code generation components
│       └── code_generator.hpp # C++ code generator
├── src/                   # Source files
│   ├── main.cpp           # Main program entry point
│   ├── lexer/             # Lexical analyzer
│   │   └── lexer.l        # Flex lexer definition
│   ├── parser/            # Parser implementation
│   │   ├── parser.y       # Bison parser definition
│   │   ├── ast.cpp        # AST implementation
│   │   └── globals.cpp    # Global variables/functions
├── test/                  # Test files
│   ├── samples/           # Test case samples
│   │   └── basic_test.cpp # Basic test implementation
│   └── src/               # Test source files
│       └── basic.dsl      # Sample DSL input file
└── build/                 # Build directory (generated)
```

WORKFLOW OF THE PROJECT



Phase	Tool/Component	Role
1	Flex (lexer.l)	Tokenize input DSL
2	Bison (parser.y)	Parse tokens into AST
3	AST (ast.hpp)	Build and evaluate AST
4	Code Generator	Emit C++ code
—	Evaluator	Compute expression results

PREREQUIREMENTS

- 1.To install cmake,bison,flex and google test command
→brew install cmake flex bison googletest
- 2.If google test command is not installed follow the steps
→brew install googletest

BUILDING THE PROJECT

1. Create a build directory:
→mkdir build
→cd build

2. Generate build files:

→ `cmake ..`

3. Build the project:

→ `make`

Running the Compiler

1. Create an input file with expressions:

→ `10*(5-2);`

2. Run the compiler:

→ `./dsl < ../test/src/basic.dsl`