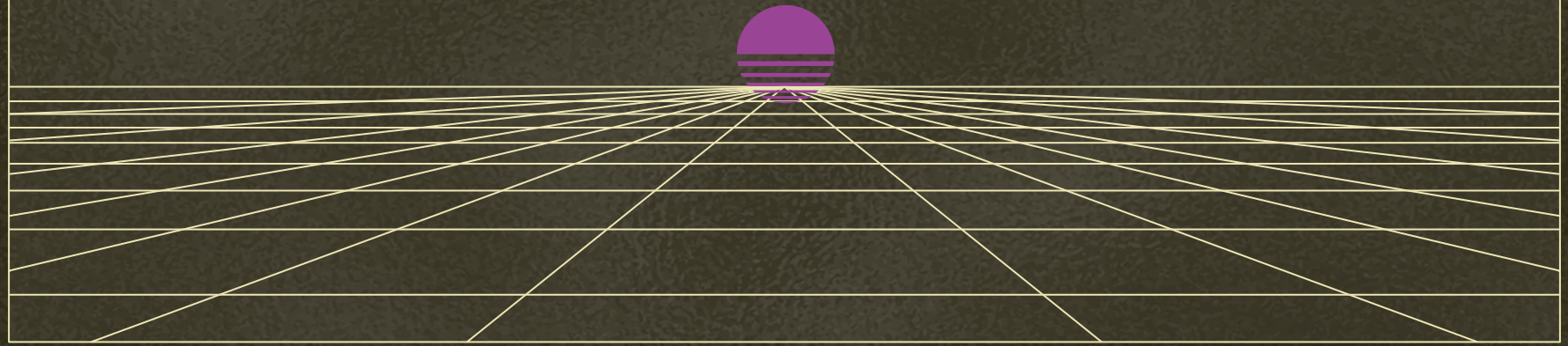
COMPILERS CONSTRUCTION

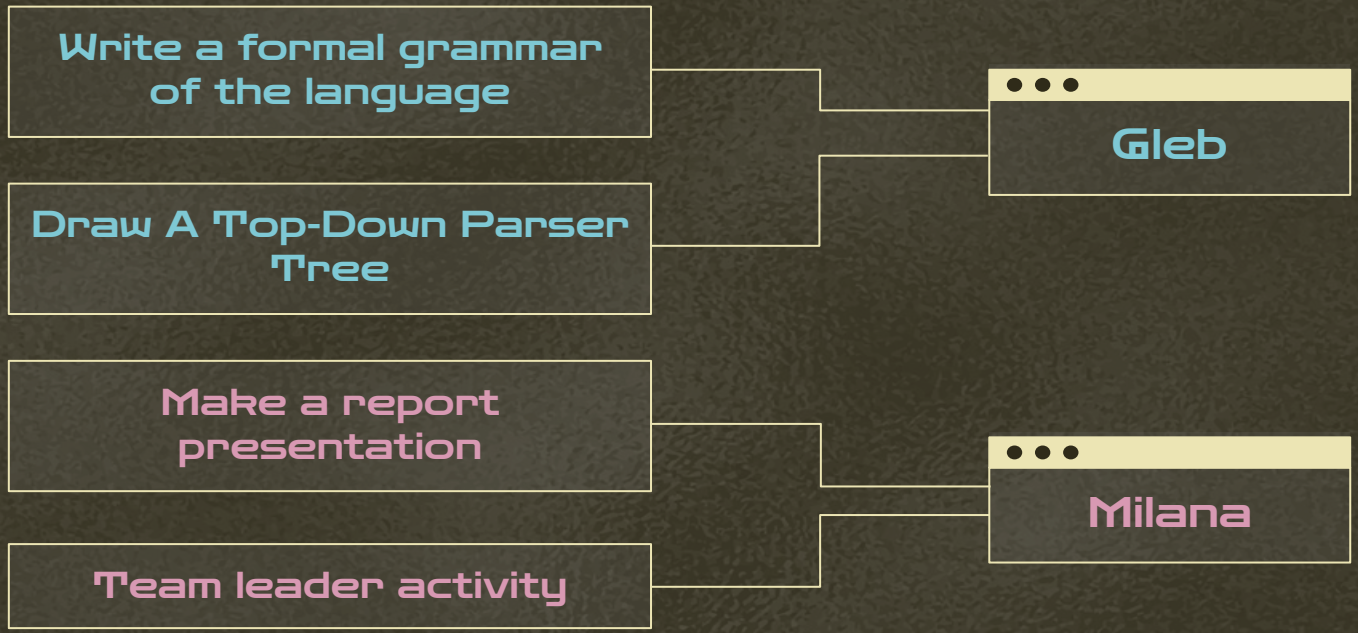
Report 2 - Syntax Analyzer



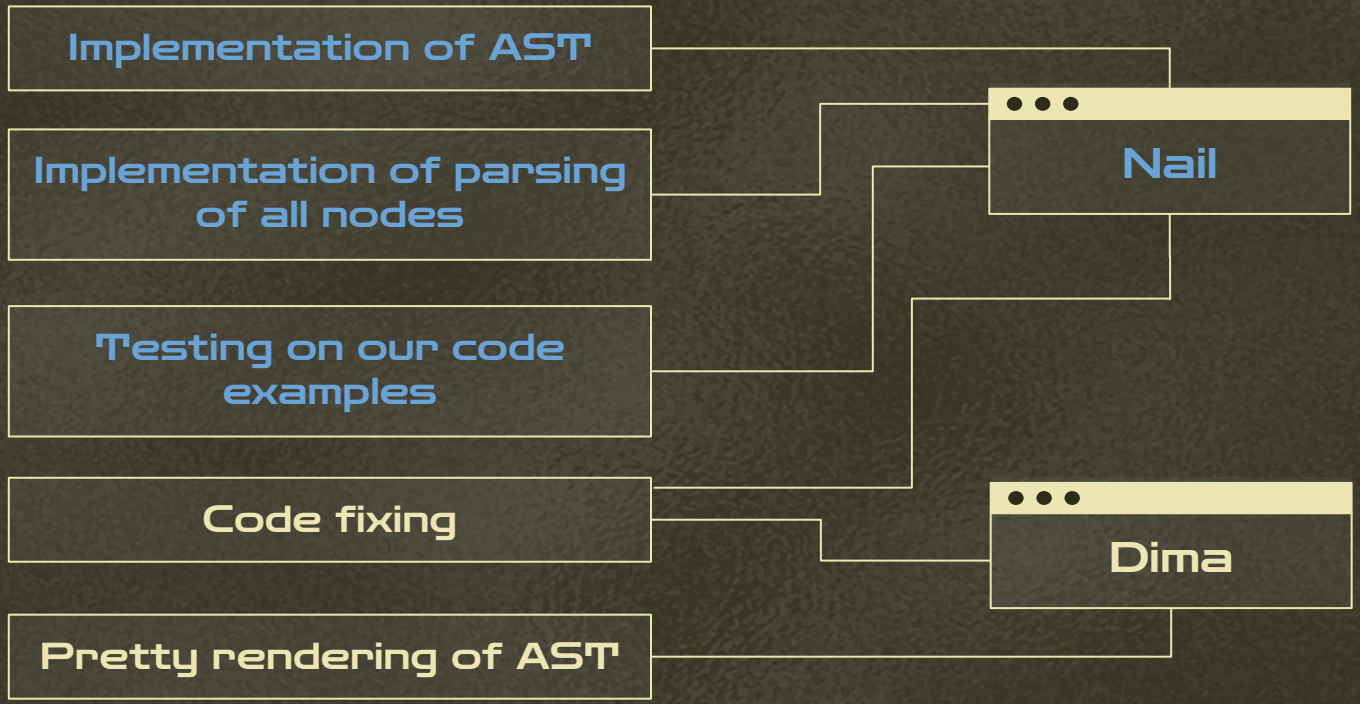
COMPILUL'KI


Gleb
Bugaev
Nail
Minnemulin
Dmitriy
Ekoneshnikov
Milana
Sirozhova

Completed sub-tasks



Completed sub-tasks



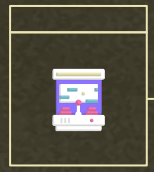
Technology stack

Project: Object-Oriented

Target platform: LLVM

Implementation lang/tool: C++, handwritten parser, CMake to build the project

Target language: LLVM bit-code



Details of Parser Implementations

Tokens representation:
In our compiler, AST nodes are represented as classes derived from a base class Entity. Moreover each node implements the accept method to allow visitor-based traversal.

```cpp
class ProgramDeclaration final : public Entity
{
public:
    std::unique_ptr<ClassName> className;
    std::unique_ptr<ProgramArguments> arguments;

    void accept(Visitor& visitor) const override
    {
        visitor.visit( node: *this);
    }
};
```

```cpp
class Entity
{
public:
    virtual ~Entity() = default;
    virtual void accept(Visitor& visitor) const = 0;

    // Helper function to print indentation
    static void printIndent(const int indent)
    {
        for (int i = 0; i < indent; ++i) std::cout << "  ";
    }
};
```

Details of Parser Implementations

Lookahead (Peek Token):

The parser constantly checks the next token in the stream to determine what kind of expression to parse next.

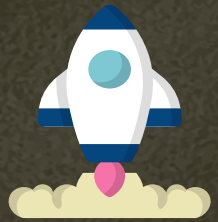
Recursive Parsing:

If the parser detects that it needs to handle a sub-expression, it calls itself recursively to build the correct AST structure for that sub-expression.

Error Handling:

The parser throws exceptions when it encounters unexpected tokens.

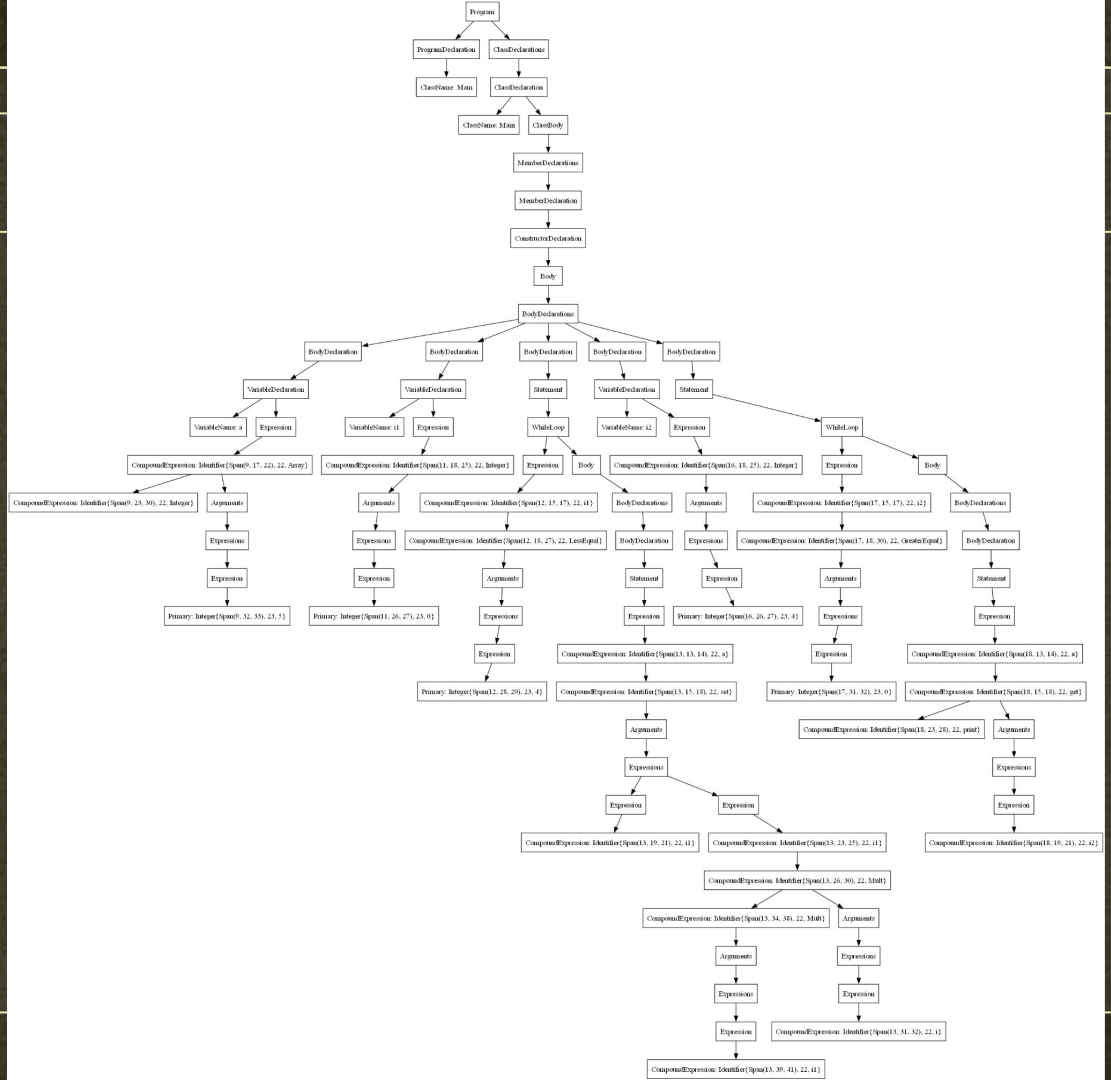
Examples



GitHub

Test

```
// Array test
// Creating and printing the array of cubes of integers from 0 to 4
Program : Main
class Main is
    this ( ) is
        var a : Array [ Integer ] ( 5 )
        var i1 : Integer ( 0 )
        while i1 . LessEqual ( 4 ) loop
            a . set ( i1 , i1 . Mult ( i ) . Mult ( i1 ) )
        end
        var i2 : Integer ( 4 )
        while i2 . GreaterEqual ( 0 ) loop
            a . get ( i2 ) . print ( )
        end
    end
end
```


Program
ProgramDeclaration
ClassDeclarations
ClassName: Main
ClassDeclaration
ClassName: Main
ClassBody
MemberDeclarations
MemberDeclaration
ConstructorDeclaration
Body
BodyDeclarations
BodyDeclaration
BodyDeclaration
BodyDeclaration
BodyDeclaration
BodyDeclaration
VariableDeclaration
VariableDeclaration
Statement
VariableDeclaration
Statement
VariableName: a
Expression
VariableName: i1
Expression
WhileLoop
VariableName: i2
Expression
WhileLoop
CompoundExpression: Identifier{Span(9, 17, 22), 22, Array}
CompoundExpression: Identifier{Span(11, 18, 25), 22, Integer}
Expression
Body
CompoundExpression: Identifier{Span(16, 18, 25), 22, Integer}
Expression
Body
CompoundExpression: Identifier{Span(9, 23, 30), 22, Integer}
Arguments
Arguments
CompoundExpression: Identifier{Span(12, 15, 17), 22, i1}
BodyDeclarations
Arguments
CompoundExpression: Identifier{Span(17, 15, 17), 22, i2}
BodyDeclarations
Expressions
Expressions
CompoundExpression: Identifier{Span(12, 18, 27), 22, LessEqual}
BodyDeclaration
Expressions
CompoundExpression: Identifier{Span(17, 18, 30), 22, GreaterEqual}
BodyDeclaration
Expression
Expression
Arguments
Arguments
Statement
Expression
Arguments
Statement
Primary: Integer{Span(9, 32, 33), 23, 5}
Primary: Integer{Span(11, 26, 27), 23, 0}
Expression
Arguments
Expressions
Primary: Integer{Span(16, 26, 27), 23, 4}
Expression
Arguments
Expressions
Expression
Expression
Primary: Integer{Span(12, 28, 29), 23, 4}
CompoundExpression: Identifier{Span(13, 13, 14), 22, a}
Primary: Integer{Span(17, 31, 32), 23, 0}
CompoundExpression: Identifier{Span(18, 13, 14), 22, a}
CompoundExpression: Identifier{Span(13, 15, 18), 22, set}
CompoundExpression: Identifier{Span(18, 15, 18), 22, get}
Arguments
CompoundExpression: Identifier{Span(18, 23, 28), 22, print}
Arguments
Expressions
Expressions
Expression
Expression
Expression
CompoundExpression: Identifier{Span(13, 19, 21), 22, i1}
CompoundExpression: Identifier{Span(13, 23, 25), 22, i1}
CompoundExpression: Identifier{Span(18, 19, 21), 22, i2}
CompoundExpression: Identifier{Span(13, 26, 30), 22, Mult}
CompoundExpression: Identifier{Span(13, 34, 38), 22, Mult}
Arguments
Arguments
Expressions
Expressions
Expression
Expression
Expression
CompoundExpression: Identifier{Span(13, 31, 32), 22, i}
CompoundExpression: Identifier{Span(13, 39, 41), 22, i1}

THANKS!