Arabic Compiler

1. Abstraction

An Arabic-language compiler designed to parse and execute programs written in Arabic syntax. This project aims to create a compiler that can understand Arabic keywords, variables, and control structures like loops and conditionals, tailored for Arabic-speaking developers.

2. Features

- Arabic Syntax Parsing: Supports Arabic keywords for variable declarations, loops, conditionals, and function calls.

- Flexible Parser: The parser is designed with extendable rules, enabling easy addition of new syntax features.

- Error Handling: Detailed error messages for syntax errors during parsing.

- Unit Testing: Uses xUnit for easy unit testing and modular addition of new functionality.

3. Tokens

Tokens	Regex
BOOLEAN	^(صح|خطأ)$
LOOP	^(طالما|من)$
RANGE	^(الي)$
PRINT	^(اطبع)$
DATATYPE	^(صحيح|عائم|مزدوج|كلمة|متغير)$
else_stmt	^(اخر)$
if_stmt	^(اذا)$
IN	^(في)$
ID	(^[\u0600-\u06FF_][\w]*$)\|(^[A-Za-z_][\w]*$)
NUM	^(-|\+)?(\d+)(\.(\d+))?([eE][-\+]?\d+)?$
SEMICOLON	;
(	\(
)	\)
{	{
}	}
[	\[
]	\]
BITSOP	(\\||&)
COMPARISONOP	^(<|>|<=|>=|==|\!=)$
ASSIGNOP	^(=)$
MATHOP	(\+|/|-|*|\^|%)
COMMA	(,)

4. Rules

Program	➡	Declaration | Loop | ConditionStmt | Func_Return | Assign | Print
Declaration	➡	"DATATYPE" "ID" (FunDeclaration | VarDeclaration ";")
FunDeclaration	➡	"(" Params ")" "{" Program "}"
Params	➡	("DATATYPE" "ID" ("DATATYPE" "ID" | "," Params))?
VarDeclaration	➡	"ASSIGNOP" Exp
Exp	➡	("ID" | "NUM") (("MATHOP" | "BITSOP") Exp)?
Loop	➡	("LOOP" "(" Condition ")" "{" Program "}") | "LOOP" "(" "DATATYPE" "ID" "IN" "(" Range ")" ")" "{" Program "}"
Condition	➡	Exp "COMPARISONOP" Exp
Range	➡	Exp "Range" Exp
ConditionStmt	➡	"if_stmt" "(" Condition ")" "{" Program "}" ("else_stmt" "{" Program "}")?
Range	➡	"return" Exp ";"
Assign	➡	"ID" "ASSIGNOP" Exp ";"
Print	➡	"PRINT" "(" ("ID" | "NUM") ")" ";"