

---

# ЗАСТОСУВАННЯ КОНТЕКСТНИХ УМОВ ДЛЯ ПОБУДОВИ ОПЕРАЦІЙНОЇ СЕМАНТИКИ В РЕАЛІЗАЦІЇ МОВИ ПРОГРАМУВАННЯ

*Виконав: студент КН-4*

*Білогрудов Даніїл Вячеславович*

*Науковий керівник: к. фіз.-мат. наук, доцент*

*Бублик Володимир Васильович*



# ПОСТАНОВКА ЗАДАЧІ

1. Дослідити теоретичні основи формалізації мов програмування, а також різні підходи до визначення операційної семантики.
2. Розглянути основні складові компоненти інтерпретатора та їхню роль у трансляції вихідного коду та роль контекстних умов на етапі інтерпретації програми.
3. Розробити і протестувати інтерпретатор згідно з попередньо визначеною граматиною та застосуванням правил структурної операційної семантики.

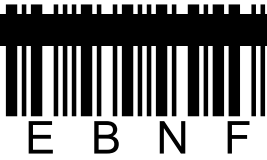
## СТАН ПРОБЛЕМИ

- Контекстно-вільні граматики не здатні покрити усі аспекти специфікації мови програмування.
- Потреба у додатковій перевірці на контекстні умови програми (типізація, області видимості змінних).
- Фокусування не на **формі** програми (синтаксис), а **значенні** (семантика).

*«Безбарвні зелені ідеї сплять несамовито»*

# ЗМІСТ РОБОТИ

- Розгляд нотації та принципів застосування структурної операційної семантики за Гордоном Плоткіном у порівнянні з іншими підходами до формальної семантики мов програмування.
- Опис фаз інтерпретації (лексичний, синтаксичний, семантичний аналізи, виконавець), підходи до реалізації абстрактного синтаксичного дерева (взірець Компонувальник) і семантичної перевірки (взірець Відвідувач + символна таблиця).
- Визначення контекстно-вільної граматики у нотації Бекуса-Науера і правил структурної операційної семантики для розробленої мови програмування.



# ГРАМАТИКА

`<prog> ::= <stmts>`

`<stmts> ::= <stmt> ";" ( <stmt> ";" )*`

`<stmt> ::= "var" <var_ident> ":" <type> [ "==" <expr> ]`  
| `<var_ident> "==" <expr>`  
| `"for" <var_ident> "in" <expr> ".." <expr> "do" <stmts> "end" "for"`  
| `"read" <var_ident> | "print" <expr>`  
| `"if" <expr> "do" <stmts> [ "else" <stmts> ]"end" "if"`

`<expr> ::= <opnd> <op> <opnd> | [ <unary_opnd> ] <opnd>`

`<opnd> ::= <int> | <string> | <var_ident> | "(" <expr> ")"`

`<type> ::= "int" | "string" | "bool"`

`<var_ident> ::= <ident>`

`<reserved_keyword> ::= "var" | "for" | "end" | "in" | "do" | "read" | "print"`  
| `"int" | "string" | "bool" | "assert" | "if" | "else"`

## ПОТРЕБА У КОНТЕКСТІ

```
var n : str;  
for n in 0..2  
do  
    k := k + 1;  
end for;
```

# ПОТРЕБА У КОНТЕКСТІ

```
var n : str;  
for n in 0..2  
do  
    k := k + 1;  
end for;
```

Ідентифікатор, але якого  
типу?

Чи змінна  
оголошена?

# ОПЕРАЦІЙНА СЕМАНТИКА

$$\frac{\Gamma(x) = (\tau, \_) \quad \Gamma \vdash e : \tau}{\Gamma, \sigma \rightarrow \Gamma[x \mapsto (\tau, \text{val}(e, \sigma))], \sigma}$$

Присвоєння значення до змінної тільки, якщо вона оголошена

$$\frac{\Gamma \vdash e : \text{bool}}{\Gamma, \sigma \vdash \text{if } e \text{ do } s_1 \text{ [else } s_2 \text{] end if}}$$

Умовою розгалуження має бути тільки булевий вираз

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \text{ op } e_2 : \tau}$$

Операнди мають бути одного і того самого типу

$$\frac{\Gamma, \sigma \vdash \text{var } x : \tau \quad x \notin \text{dom}(\Gamma)}{\Gamma[x \mapsto (\tau, \text{undefined})], \sigma}$$

Змінна має бути унікальна в межах області видимості



# ВИКОРИСТАНІ ТЕХНОЛОГІЇ

Мова, якою написаний інтерпретатор: C#.

Фреймворк для автоматичного тестування: MSTest.

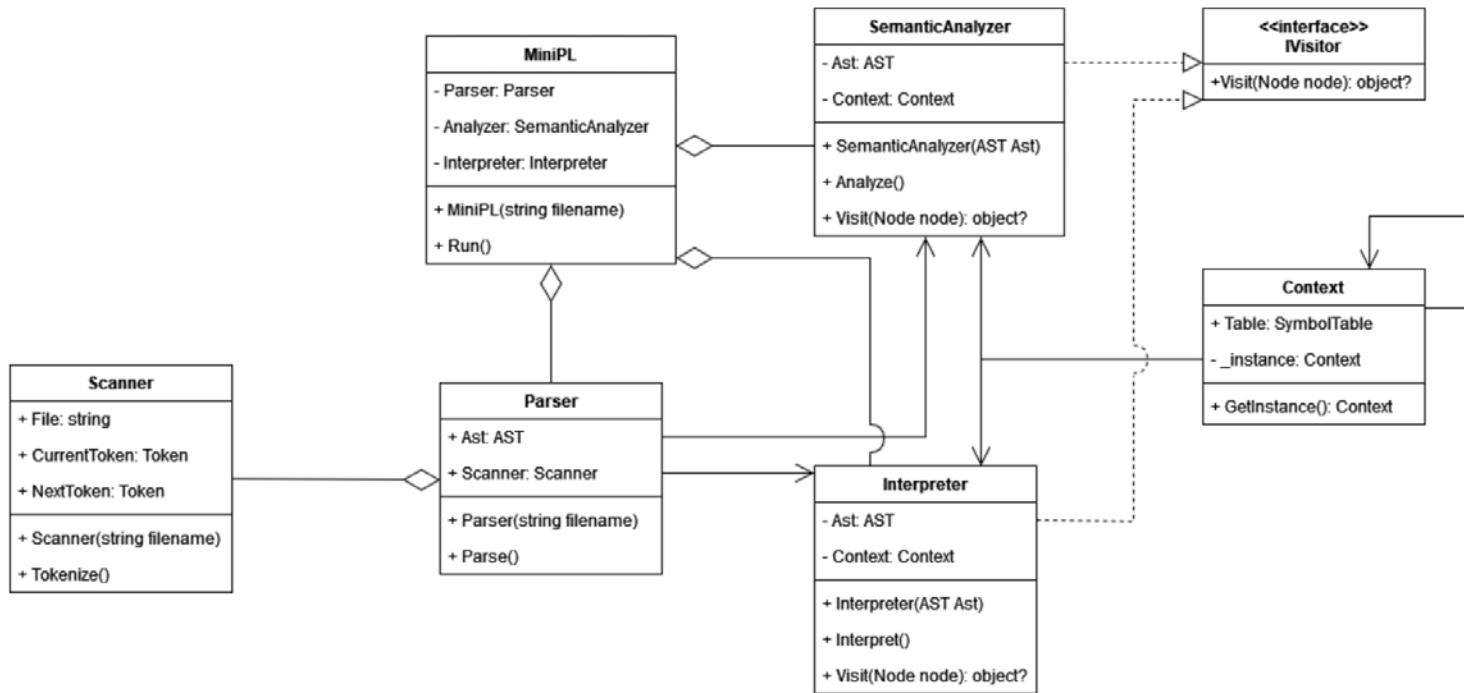
```
namespace MiniPL
{
    /* Semantic Analyzer is the part of the MiniPL interpreter which uses the Visitor
    pattern
    * to check the semantic correctness of the program, in particular variables
    declaration
    * and usage, and type matching.
    */
    public class SemanticAnalyzer : IVisitor
    {
        private readonly AST Ast;
        private readonly Context Context;

        private readonly List<string> boolOperators = new() {
            TFS(TokenType.EQ), TFS(TokenType.LT), TFS(TokenType.GT)
        };

        // list of exceptions for the statement mode recovery
        private readonly List<MiniPLException> exceptions = new();

        public SemanticAnalyzer(AST ast)
        {
            Ast = ast;
            Context = Context.GetInstance();
        }
    }
}
```

# РЕЗУЛЬТАТИ



UML-діаграма інтерпретатора мови програмування MPL

## РЕЗУЛЬТАТИ

```
print "Fibonacci series\n";
print "Enter num: ";
var n : int;
read n;
var i : int;
var prev : int := 0;
var curr : int := 1;
print prev;
print "\n";
print curr;
print "\n";
for i in 2..n do
    var next : int := prev + curr;
    print next;
    print "\n";
    prev := curr;
    curr := next;
end for;
```

Приклад програми мовою програмування MPL

# РЕЗУЛЬТАТИ

```
VAR          var          Ln: 1  Cl: 1
IDENTIFIER  X            Ln: 1  Cl: 5
COLON       :            Ln: 1  Cl: 7
INT         int          Ln: 1  Cl: 9
ASSIGN      :=          Ln: 1  Cl: 13
INT_LITERAL 4            Ln: 1  Cl: 16
PLUS       +            Ln: 1  Cl: 18
LPAREN     (            Ln: 1  Cl: 20
INT_LITERAL 6            Ln: 1  Cl: 21
MUL        *            Ln: 1  Cl: 23
INT_LITERAL 2            Ln: 1  Cl: 25
RPAREN     )            Ln: 1  Cl: 26
SEMICOLON  ;            Ln: 1  Cl: 27
PRINT      print        Ln: 2  Cl: 1
IDENTIFIER  X            Ln: 2  Cl: 7
SEMICOLON  ;            Ln: 2  Cl: 8
EOF        ;            Ln: 2  Cl: 9
ProgNode
  StmtNode
    DeclNode
      IdentNode [X]
      TypeNode [int]
      LRExpNode
        OpndNode
          IntNode [4]
          OpNode [+]
          OpndNode
            LRExpNode
              OpndNode
                IntNode [6]
                OpNode [*]
                OpndNode
                  IntNode [2]
          PrintNode
            LExprNode
              OpndNode
                IdentNode [X]
```

16

Демонстрація роботи інтерпретатора з додатковою інформацією про результати лексичного і синтаксичного аналізів

# РЕЗУЛЬТАТИ

```
SemanticError: Variable is not declared on line 1 column 5  
for str in str..str do end for;  
      ^
```

```
SemanticError: Variable is not declared on line 1 column 12  
for str in str..str do end for;  
      ^
```

```
SemanticError: Variable is not declared on line 1 column 17  
for str in str..str do end for;  
      ^
```

```
SemanticError: Variable is not declared on line 2 column 5  
for i in (2 = 2)..10 do end for;  
      ^
```

```
SemanticError: Variable type mismatch (expected int, got bool) on line 2 column 16  
for i in (2 = 2)..10 do end for;  
      ^
```

```
SemanticError: Variable type mismatch (expected int, got string) on line 4 column 6  
x := "2";  
   ^
```

Коректна обробка аварійних ситуацій  
на етапі семантичного аналізу

# РЕЗУЛЬТАТИ

Test run finished: 56 Tests (56 Passed, 0 Failed, 0 Skipped) run in 125 ms

Test	Duration	Traits	Error Message
✓ MPLTests (56)	29 ms		
✓ MPLTests (56)	29 ms		
✓ InterpreterTest (13)	25 ms		
✓ Interpret_ArithmeticOperations_ProduceCorrectResult (ariphmetics.mpl,-4,15,6,14,9)	22 ms		
✓ Interpret_IllegalVariableUsage_ThrowRuntimeError (2)	< 1 ms		
✓ Interpret_InputNonIntValue_ThrowRuntimeError (3)	< 1 ms		
✓ Interpret_ValidPrograms_ExecuteProgram (5)	2 ms		
✓ Interpret_VariableDeclarationAndAssignment_CorrectResult (2)	1 ms		
✓ ParserTest (17)	2 ms		
✓ Parse_IllegalToken_ThrowSyntaxError (4)	1 ms		
✓ Parse_ValidPrograms_BuildAST (5)	< 1 ms		
✓ Parse_ValidStatements_BuildAST (6)	< 1 ms		
✓ Parse_NestedOperations_ValidExpr (nested_operations.mpl)	1 ms		
✓ Parse_TestProgram_ValidAST (valid_ast.mpl)	< 1 ms		
✓ ScannerTest (17)	2 ms		
✓ Tokenize_IncorrectProgram_ThrowLexicalError (8)	< 1 ms		
✓ Tokenize_ValidPrograms_GenerateTokens (5)	< 1 ms		
✓ _ReadFile_NonExistingFile_ThrowFileNotFoundException (non_existing_file.mpl) ..	< 1 ms		
✓ Tokenize_BlankSpacesAndComments_Ignore (spaces_and_comments.mpl)	2 ms		
✓ Tokenize_FileWithAllTokens_AllTokensRecognized (all_valid_tokens.mpl)	< 1 ms		
✓ Tokenize_StringWithEscapeChars_ProcessCorrectly (escape_char_string.mpl)	< 1 ms		
✓ SemanticTest (9)	< 1 ms		
✓ Analyze_VariableDeclaration_VariablesInTable (var_declaration.mpl,a,b,str,i)	< 1 ms		
✓ Analyze_SemanticErrorousProgram_ThrowSemanticError (3)	< 1 ms		
✓ Analyze_ValidPrograms_AnalysisPerformedCorrectly (5)	< 1 ms		

**Group Summary**

MPLTests

Tests in group: 56

Total Duration: 29 ms

Outcomes

✓ 56 Passed

Результати автоматичного тестування

## ВИСНОВОК

- *Розглянуто* підходи до формальної семантики мов програмування, головним чином **структурної операційної семантики**.
- *Визначено* **контекстні умови** згідно зі специфікацією мови програмування MPL і запропоновано підходи до їх кодової обробки.
- *Розроблено і протестовано* **інтерпретатор** мови програмування MPL.

**ДЯКУЮ ЗА  
УВАГУ!**