НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Лабораторна робота № 1

ПРОЄКТУВАННЯ ТА СПЕЦИФІКАЦІЯ
ПРОСТОЇ ІМПЕРАТИВНОЇ МОВИ ПРОГРАМУВАННЯ
ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ
(з дисципліни «Побудова компіляторів»)

Звіт студента І курсу, групи ІП-51мн
спеціальності F2 Інженерія програмного забезпечення
Панченко С. В. _____
(Прізвище, ім’я, по батькові) (Підпис)

Перевірив: доцент, к.т.н. Стативка Ю.І. _____
(Посада, науковий ступінь, прізвище та ініціали) (Підпис)

Київ – 2025

Зміст

1 ВСТУП

Мова програмування LAB — імперативна мова загального призначення з LISP-подібним синтаксисом. Назва промовляється як "лаб".

1.1 Обробка

Програма, написана мовою LAB, подається на вхід транслятора для перетворення до цільової форми. Результат трансляції виконується у системі часу виконання, для чого приймає вхідні дані та надає результат виконання програми.

Трансляція передбачає фази лексичного, синтаксичного та семантичного аналізу, а також фазу генерації коду.

1.2 Нотація

Для опису мови LAB використовується граматика ANTLR4. Граматика описує синтаксичні правила мови у форматі, що підтримується генератором парсерів ANTLR4.

2 ЛЕКСИЧНА СТРУКТУРА

Лексичний аналіз виконується окремим проходом, отже не залежить від синтаксичних та семантичних аспектів. Лексичний аналізатор розбиває текст програми на лексеми.

2.1 Пробільні символи

Текст програми мовою LAB складається з лексем, що розпізнаються лексичним аналізатором.

```
WS: [ \t\r\n]+ -> skip;
```

Рисунок 2.1 — Пробільні символи

2.2 Дужки

```
LPAREN: '(';
RPAREN: ')';
```

Рисунок 2.2 — Ліва дужка

Рисунок 2.3 — Права дужка

2.3 Типи даних

Мова LAB підтримує чотири базові типи даних.

```
TYPE: TYPE_INTEGER | TYPE_REAL | TYPE_STRING | TYPE_BOOL;
TYPE_INTEGER: 'int';
TYPE_REAL: 'real';
TYPE_STRING: 'string';
TYPE_BOOL: 'bool';
```

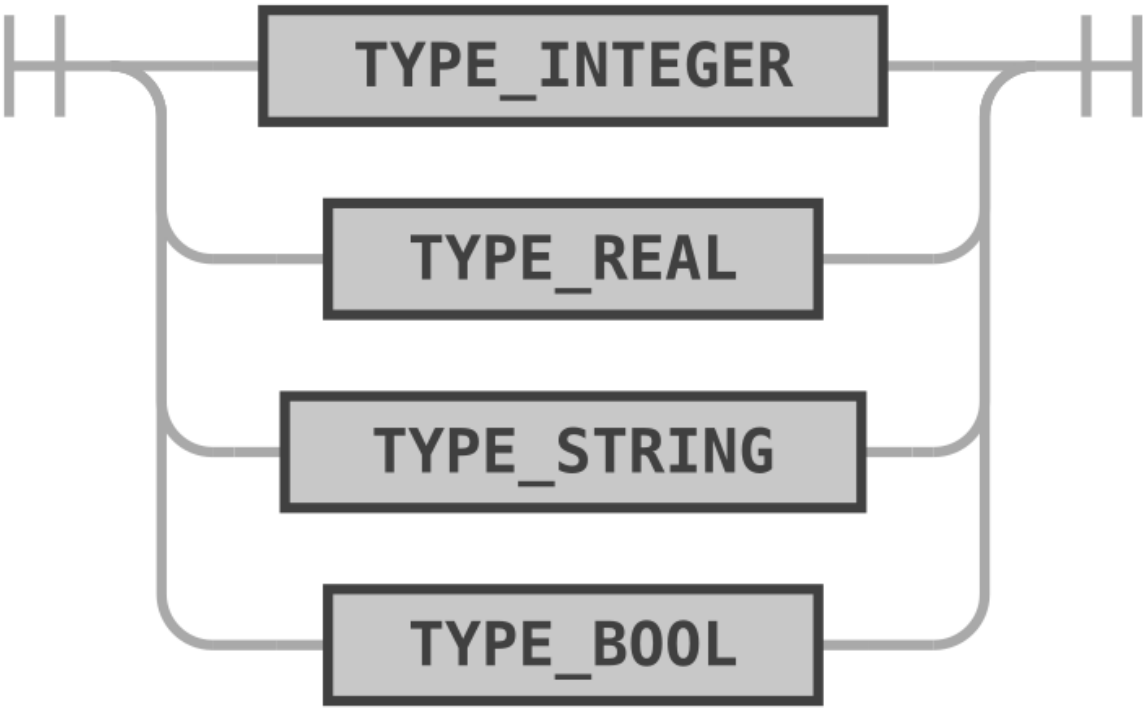

Рисунок 2.4 — Типи даних

2.4 Ключові слова

```
KEYWORD_FN: 'fn';
KEYWORD_LET: 'let';
KEYWORD_IF: 'if';
KEYWORD_WHILE: 'while';
```

Рисунок 2.5 — Ключове слово “fn”

Рисунок 2.6 — Ключове слово “if”

Рисунок 2.7 — Ключове слово “let”

Рисунок 2.8 — Ключове слово “while”

2.5 Літерали

Літерали використовуються для представлення постійних значень.

```
LITERAL: LITERAL_BOOL | LITERAL_REAL | LITERAL_INTEGER |
LITERAL_STRING;
LITERAL_BOOL: 'true' | 'false';
LITERAL_REAL: ('+'|'-')?DIGIT+ '.' DIGIT+;
LITERAL_INTEGER: ('+'|'-')?DIGIT+;
LITERAL_STRING: '"' ( ~["\\\r\n] | '\\' . )* '"';
fragment DIGIT : [0-9];
```

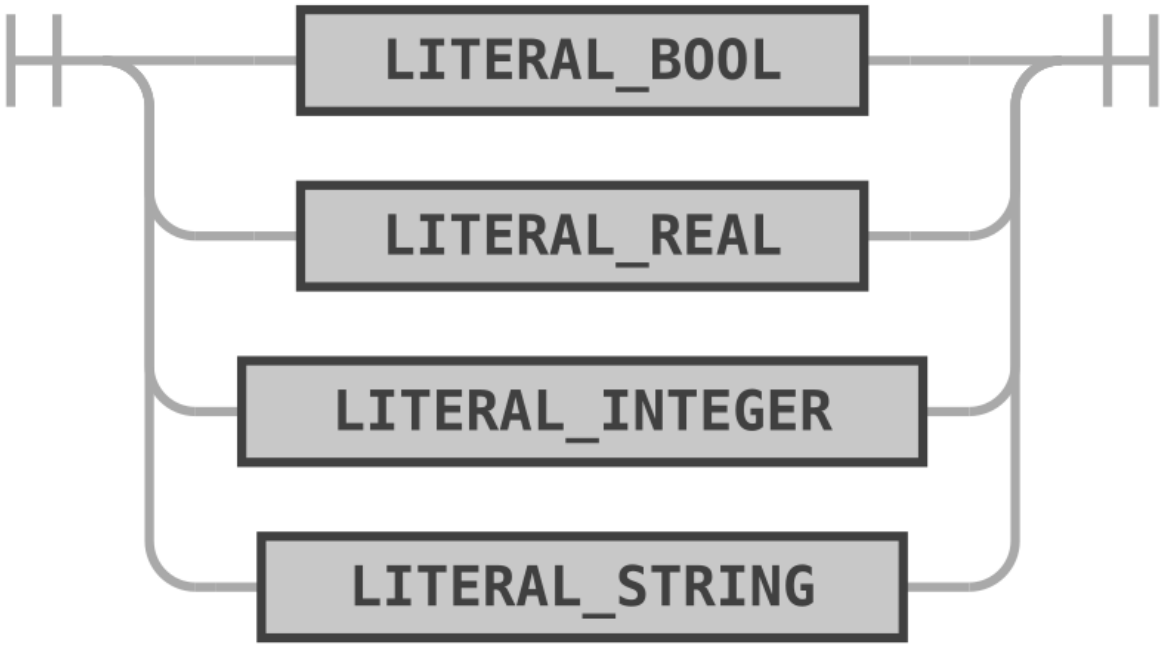

Рисунок 2.9 — Літерали

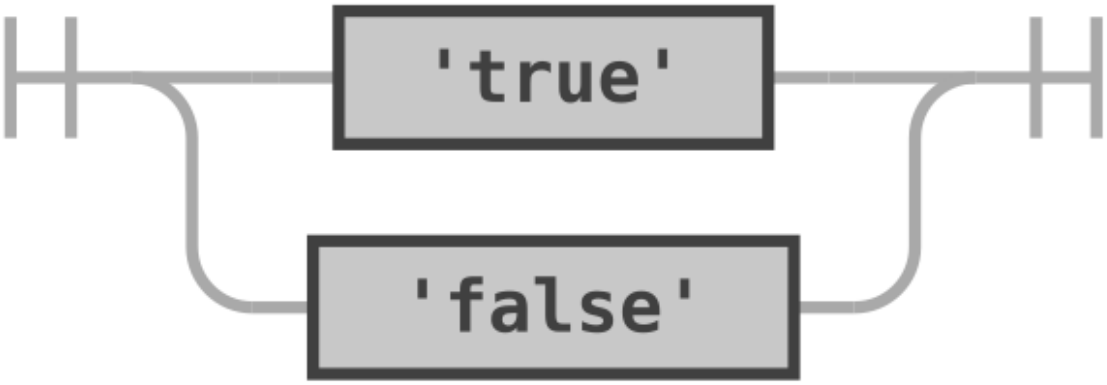

Рисунок 2.10 — Булевий літерал

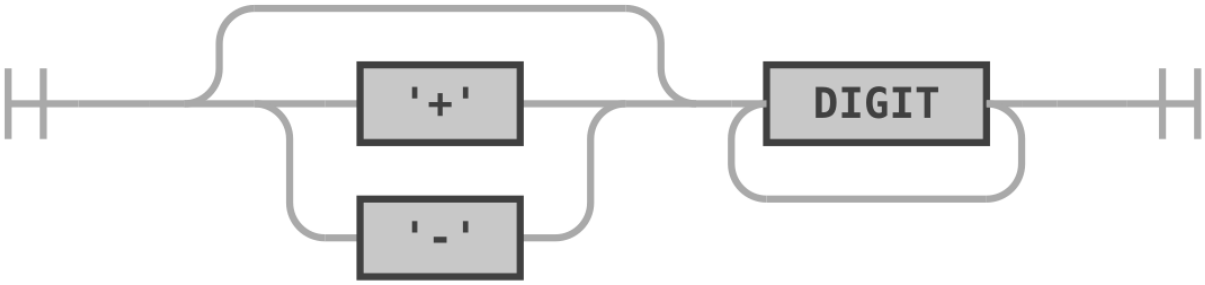

Рисунок 2.11 — Ціле число

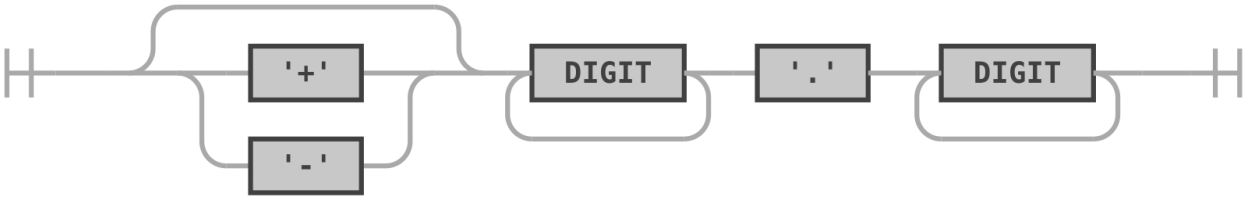

Рисунок 2.12 — Дійсне число

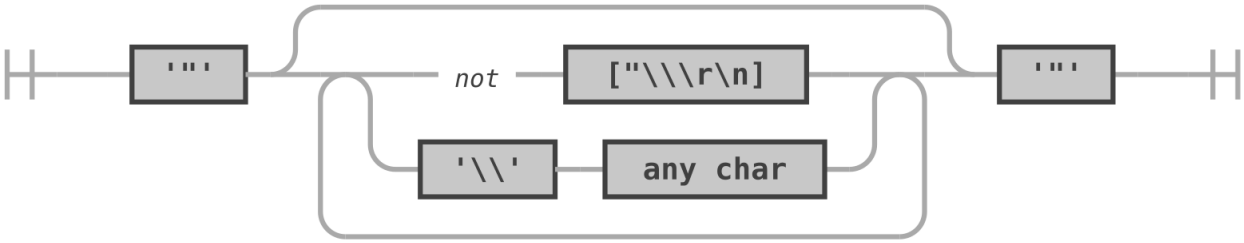

Рисунок 2.13 — Рядок

Рисунок 2.14 — Цифра

2.6 Ідентифікатори

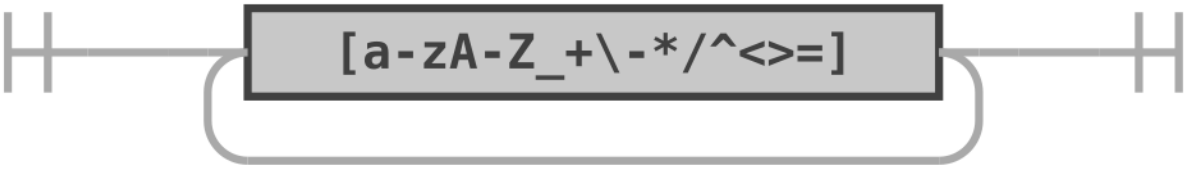

Рисунок 2.15 — Ідентифікатор

3 СИНТАКСИЧНА СТРУКТУРА

3.1 Програма

Програма складається з одного або більше визначень функцій.

```
program: funcDef+;
```

Рисунок 3.1 — Програма

3.2 Визначення функції

Кожна функція має ім'я, список параметрів (може бути порожнім) та тіло функції. Змінні, оголошені всередині функції, мають локальну область видимості. Параметри функції також є локальними змінними. Програма повинна містити функцію з іменем `main`, яка служить точкою входу для виконання програми.

```
funcDef: LPAREN KEYWORD_FN ID LPAREN funcParamList? RPAREN
statementsList RPAREN;
funcParamList: funcParam+;
funcParam: LPAREN ID TYPE RPAREN;
```

Рисунок 3.2 — Функція

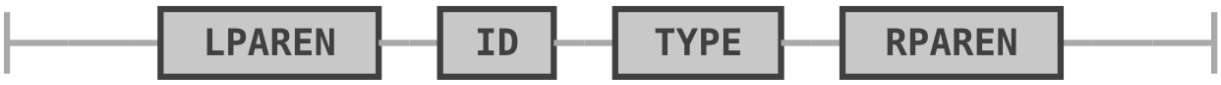

Рисунок 3.3 — Параметр функції

Рисунок 3.4 — Список параметрів функції

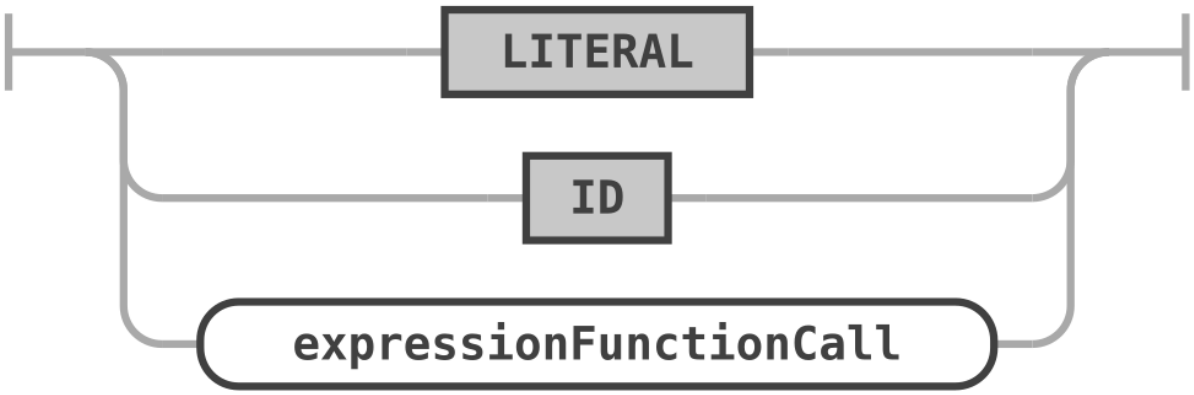

Рисунок 3.5 — Аргумент виклику функції

Приклад:

```
(fn main ((a int) (b real) (c string)) (
    // тіло функції
))
```

3.3 Інструкції

Тіло функції складається з інструкцій.

```
statement
  : statementVariableDeclaration
  | statementIf
  | statementWhile
  | expressionFunctionCall
  ;

statementsList: LPAREN statement* RPAREN;
```

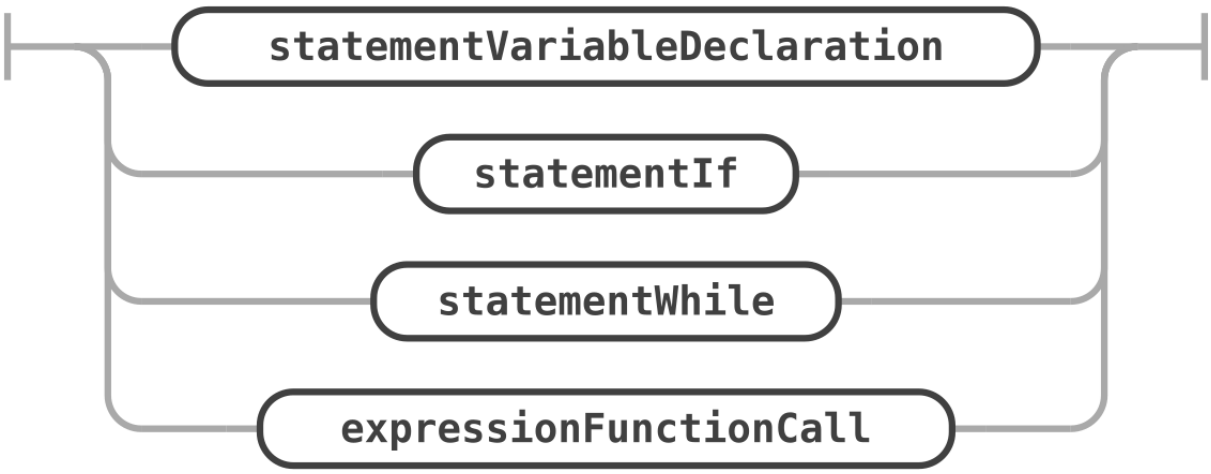

Рисунок 3.6 — Інструкція

3.3.1 Оголошення змінних

```
statementVariableDeclaration: LPAREN KEYWORD_LET ID
functionCallArgument RPAREN;
```

Рисунок 3.7 — Оголошення змінної

Приклад:

```
(let d (concat c "Hello KPI"))
(let i 0)
```

3.3.2 Умовна інструкція

```
statementIf: LPAREN KEYWORD_IF expressionFunctionCall statementsList
statementsList RPAREN;
```

Рисунок 3.8 — Умовна інструкція

Приклад:

```
(if (== 55 104) (
    (display "TRUE")
) (
    (display "FALSE")
))
```

3.3.3 Інструкція циклу

```
statementWhile: LPAREN KEYWORD_WHILE expressionFunctionCall
statementsList RPAREN;
```

Рисунок 3.9 — Інструкція циклу

Приклад:

```
(while (le i 5) (
    (display (* 2 i))
    (set i (+ i 1))
))
```

3.3.4 Виклики функцій

Виклики функцій мають префіксну нотацію. Аргументами можуть бути літерали, ідентифікатори або інші виклики функцій.

```
expressionFunctionCall: LPAREN ID functionCallArgument* RPAREN;
functionCallArgument: LITERAL | ID | expressionFunctionCall;
```

Рисунок 3.10 — Виклик фнукції

Приклад:

```
(display "Hello")
(+ 2 3)
(* 2 (- a (/ i b)))
(concat c "Hello KPI")
```

```
grammar lab;

WS: [ \t\r\n]+ -> skip;

TYPE: TYPE_INTEGER | TYPE_REAL | TYPE_STRING | TYPE_BOOL;
TYPE_INTEGER: 'int';
TYPE_REAL: 'real';
TYPE_STRING: 'string';
TYPE_BOOL: 'bool';

KEYWORD_FN: 'fn';
KEYWORD_LET: 'let';
KEYWORD_IF: 'if';
KEYWORD_WHILE: 'while';

LPAREN: '(';
RPAREN: ')';

statement
 : statementVariableDeclaration
 | statementIf
 | statementWhile
 | expressionFunctionCall
 ;

functionCallArgument: LITERAL | ID | expressionFunctionCall;
expressionFunctionCall: LPAREN ID functionCallArgument* RPAREN;
statementVariableDeclaration: LPAREN KEYWORD_LET ID
functionCallArgument RPAREN;
statementsList: LPAREN statement* RPAREN;
statementIf: LPAREN KEYWORD_IF expressionFunctionCall statementsList
statementsList RPAREN;
statementWhile: LPAREN KEYWORD_WHILE expressionFunctionCall
statementsList RPAREN;

funcParam: LPAREN ID TYPE RPAREN;
funcParamList: funcParam+;
funcDef: LPAREN KEYWORD_FN ID LPAREN funcParamList? RPAREN
statementsList RPAREN;

program: funcDef+;

LITERAL: LITERAL_BOOL | LITERAL_REAL | LITERAL_INTEGER |
LITERAL_STRING;
LITERAL_BOOL: 'true' | 'false';
LITERAL_REAL: DIGIT+ '.' DIGIT* | '.' DIGIT+;
```

```
LITERAL_INTEGER: DIGIT+;
LITERAL_STRING: '"' ( ~["\\\r\n] | '\\' . )* '"';

ID : [a-zA-Z_+\-*/^<>=]+;
fragment DIGIT : [0-9];
```

5 ПРИКЛАД ПРОГРАМИ

```
(fn otherfunc () (
    (display (** 2 6))
))

(fn main ((a int) (b real) (c string)) (
    (let d (concat c "Hello KPI"))
    (let dd true)
    (if (== 55 104) (
        (display (concat d "I AM TRUE"))
    ) (
        (display (concat d "I AM FALSE"))
    ))
    (let i 0)
    (while (le i 5) (
        (display (* 2 (- a (/ i b))))
        (set i (+ i 1))
    ))
))
```