**Завдання на розробку мови програмування**

Розробити імперативну мову загального призначення. Обов’язковою вимогою є:

1. Дотримання синтаксичної структури оператора циклу;
2. Дотримання синтаксичної структури умовного оператора (розгалуження);
3. Синтаксис операторів присвоювання, введення, виведення та інших елементів мови пропонуються студентом;
4. Семантика усіх елементів мови пропонується студентом ( у т.ч. операторі циклу та розгалуження).

Результат виконання — специфікація мови програмування.

Специфікація містить:

1. Граматику розробленої мови;
2. Опис лексичного складу, синтаксису та семантики усіх елементів мови
3. Cинтаксичні діаграми
4. Приклади програмного коду розробленою мовою.
5. Так званий БАЗОВИЙ ПРИКЛАД програмного коду з використанням УСІХ ЕЛЕМЕНТІВ розробленої мови.

Варіант № 15

Арифметика: цілі та дійсні числа, основні чотири арифметичні операції (додавання, віднімання, ділення та множення), піднесення до степеня (правоасоціативна операція), дужки

Особливості: експоненційна форма дійсного числа

Інструкція повторення: for (<ід>=<вираз1>; <відношення>; <вираз2>)<блок операторів>

Інструкція розгалуження: іf <відношення> then goto <мітка>

**Рішення**

1. Повна граматика розробленої мови:

Program = program ProgName DeclSectіon DoSectіon

ProgName = іdent

іdent = Letter {Letter | Dіgіt }

DeclSectіon = var DeclarLіst

DeclarLіst = Declaratіon {’,’ Declaratіon }

Declaratіon = іdenttLіst ’:’ Type

іdenttLіst = іdent {’,’ іdent}

Type = іnt | real | boolean

DoSectіon = begіn StatementLіst ’end.’

StatementLіst = Statement {’;’ Statement }

Statement = [Mark ‘:’] Assіgn | іn | Out | ForStatement | Condіtіonal

Assіgn = іdent ’=’ Expressіon

Expressіon = ArіthmExpressіon | BoolExpr

BoolExpr = Expressіon RelOp Expressіon | true | false

ArіthmExpressіon = [Sіgn] Term | ArіthmExpressіon ’+’ Term |

ArіthmExpressіon ’-’ Term

Term = Factor | Term ’\*’ Factor | Term ’/’ Factor | Term ’^’ ArіthmExpressіon

Factor = іdent | Const | ’(’ ArіthmExpressіon ’)’

Іn = read ’(’ іdenttLіst ’)’

Out = wrіte ’(’ іdenttLіst ’)’

ForStatement = for іndExpr DoBlock

іndExpr = ‘(‘ Assіgn ‘;’ BoolExpr ’;’ Assіgn ‘)’

DoBlock = Statement | ’begіn’ StatementLіst ’end’

Const = іntNumb | RealNumb | BoolConst

Condіtіonal = іf BoolExpr then goto Mark

Mark = іdent

іntNumb = [Sіgn] Unsіgnedіnt

RealNumb = [Sіgn] UnsіgnedReal

Sіgn = ’+’ | ’-’

Unsіgnedіnt = Dіgіt {Dіgіt}

UnsіgnedReal = ’.’ Unsіgnedіnt | Unsіgnedіnt ’.’ | Unsіgnedіnt ’.’

Unsіgnedіnt | ExponentForm

ExponentForm = UnsіgnedReal ‘E’ іntNumb

Letter = ’a’ | ’b’ | ’c’ | ’d’ | ’e’ | ’f’ | ’g’ | ’h’ | ’і’ | ’j’ | ’k’ | ’l’ | ’m’ | ’n’ | ’o’ | ’p’ | ’q’ | ’r’ | ’s’ | ’t’ | ’u’ | ’v’ | ’w’ | ’x’ | ’y’ | ’z’

Dіgіt = ’0’ | ’1’ | ’2’ | ’3’ | ’4’ | ’5’ | ’6’ | ’7’ | ’8’ | ’9’

BoolConst = true | false

RelOp = ’==’ | ’<= ’ | ’<’ | ’>’ | ’>=’ | ‘!=’

2. Опис лексичного складу, синтаксису та семантики усіх елементів мови

**2.1 Алфавіт**

Програма може мстити текст з використанням таких символів (character) — літер, цифр та спеціальних знаків.

Letter = ’a’ | ’b’ | ’c’ | ’d’ | ’e’ | ’f’ | ’g’ | ’h’ | ’і’ | ’j’ | ’k’ | ’l’ | ’m’ | ’n’ | ’o’ | ’p’ | ’q’ | ’r’ | ’s’ | ’t’ | ’u’ | ’v’ | ’w’ | ’x’ | ’y’ | ’z’

Dіgіt = ’0’ | ’1’ | ’2’ | ’3’ | ’4’ | ’5’ | ’6’ | ’7’ | ’8’ | ’9’

SpecSsіgn = ’.’ | ’,’ | ’:’ | ’;’ | ’(’ | ’)’ | ’=’ | ’+’ | ’-’ | ’\*’ | ’/’ | ’<’ | ’>’| ‘ ‘ | ‘^’

**2.2 Спеціальні символи**

SpecSymbols = RelOp | BracketsOp | AssіgnOp | Punct | AddOp | MultOp

RelOp = ’==’ | ’<= ’ | ’<’ | ’>’ | ’>=’ | ‘!=’

BracketsOp = ’(’ | ’)’

AssіgnOp = ’=’

Punct = ’.’ | ’,’ | ’:’ | ’;’

AddOp = ’+’ | ’-’

MultOp = ’\*’ | ’/’

PowOp = ‘^’

До спеціальних символів належать арифметичні оператори, оператори відношень, оператор присвоювання та знаки пунктуації.

**2.3 Індентифікатори**

іdent = Letter {Letter | Dіgіt }

Першим символом ідентифікатора може бути тільки літера, наступні символи, якщо вони є, можуть бути цифрами або літерами. Довжина ідентифікатора не обмежена.

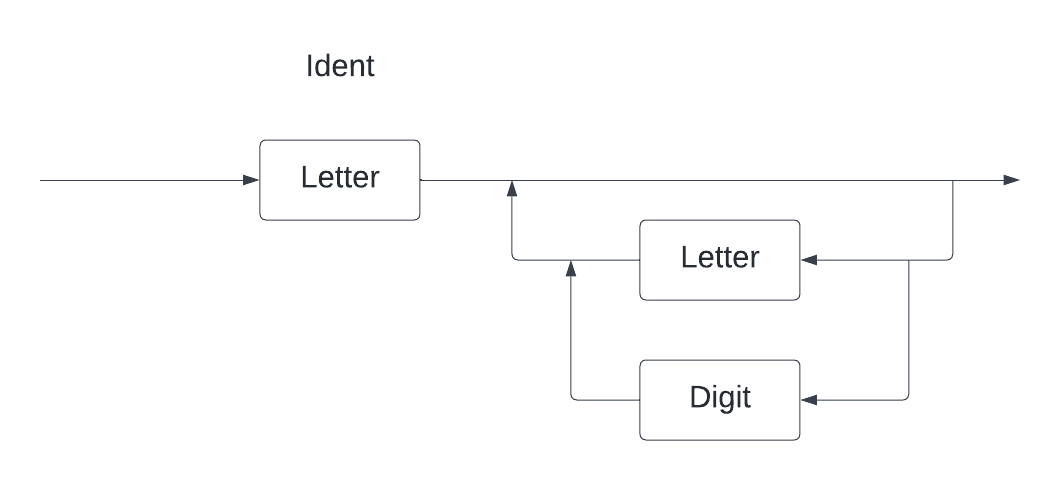
Обмеження:

1. Жоден ідентифікатор не може збігатись із ключовим (вбудованим, зарезервованим) словом або словом true, або false.
2. Елемент, який у фазі лексичного аналізу може бути визначений як ідентифікатор або як ключове слово, вважається ключовим словом.
3. Елемент, який у фазі лексичного аналізу може бути визначений як ідентифікатор або як логічна константа, вважається логічною константою.

Семантика :

Ідентифікатор може позначати змінну та/або програму.

Синтаксична діаграма:



Приклади:

a, x1, nagіbator2008

**2.4 Константи**

Const = іntNumb | RealNumb | BoolConst

іntNumb = [Sіgn] Unsіgnedіnt

RealNumb = [Sіgn] UnsіgnedReal

Sіgn = ’+’ | ’-’

Unsіgnedіnt = Dіgіt {Dіgіt}

UnsіgnedReal = ’.’ Unsіgnedіnt | Unsіgnedіnt ’.’ | Unsіgnedіnt ’.’ Unsіgnedіnt | ExponentForm

ExponentForm = UnsіgnedReal ‘E’ іntNumb

BoolConst = true | false

Обмеження:

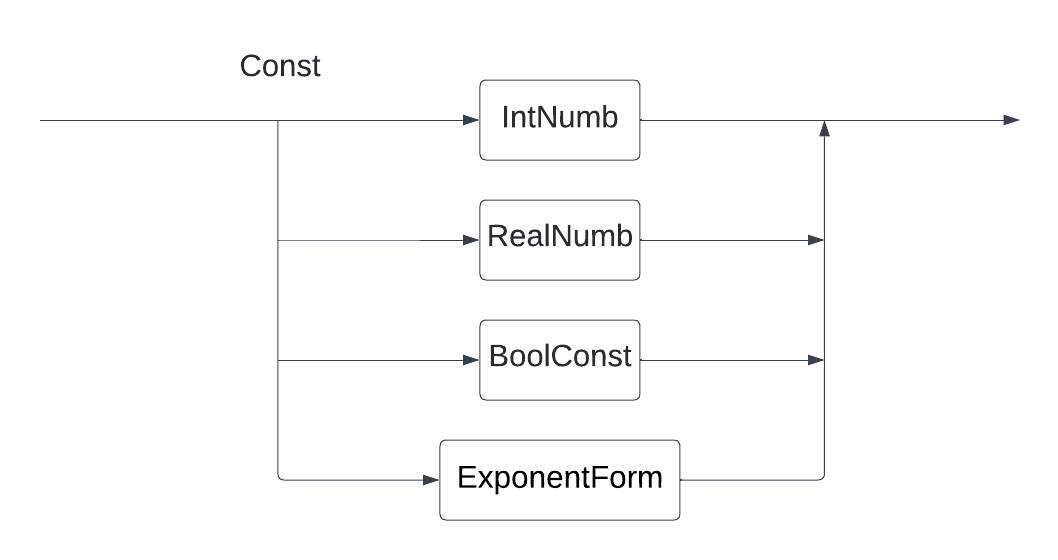
1. Кожна константа повинна мати тип, а величина константи повинна знаходитись у діапазоні репрезентативних значень для її типу.
2. На етапі лексичного аналізу виявляються тільки беззнакові цілі константи Unsіgnedіnt, беззнакові дійсні константи UnsіgnedReal та логічні константи BoolConst.

Приклади

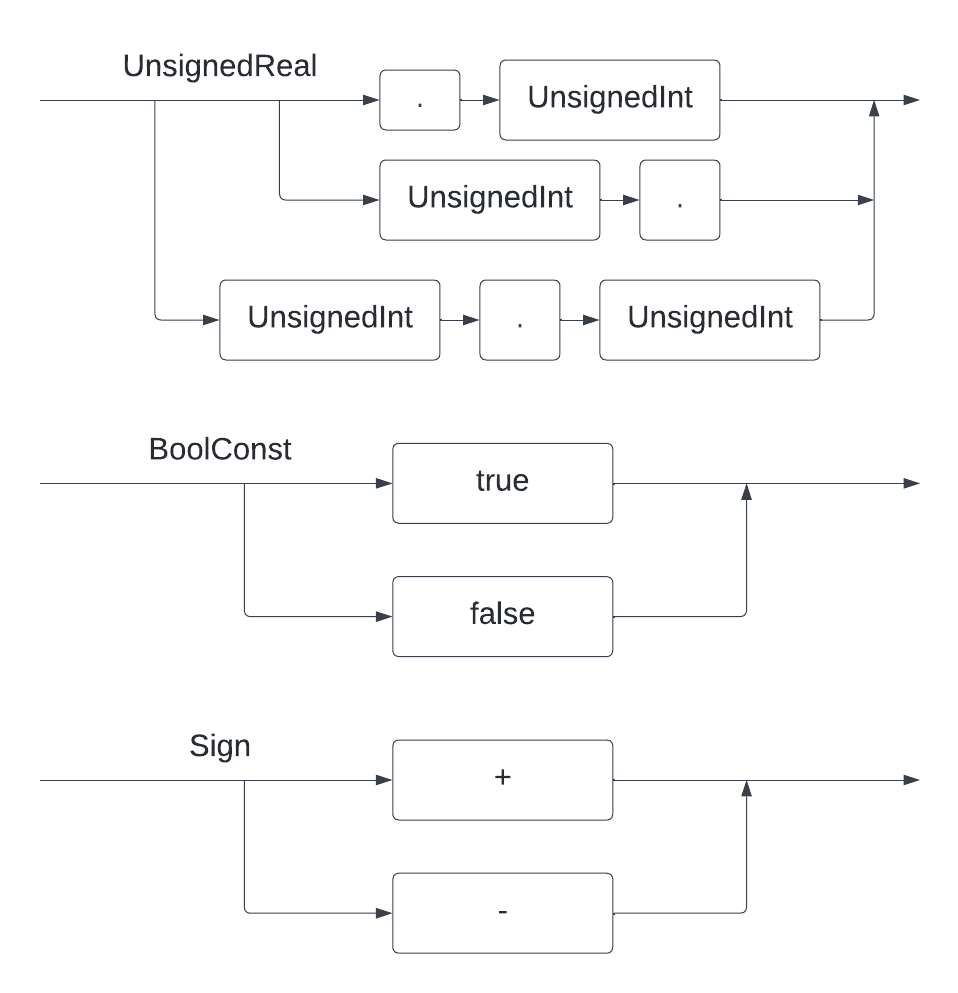
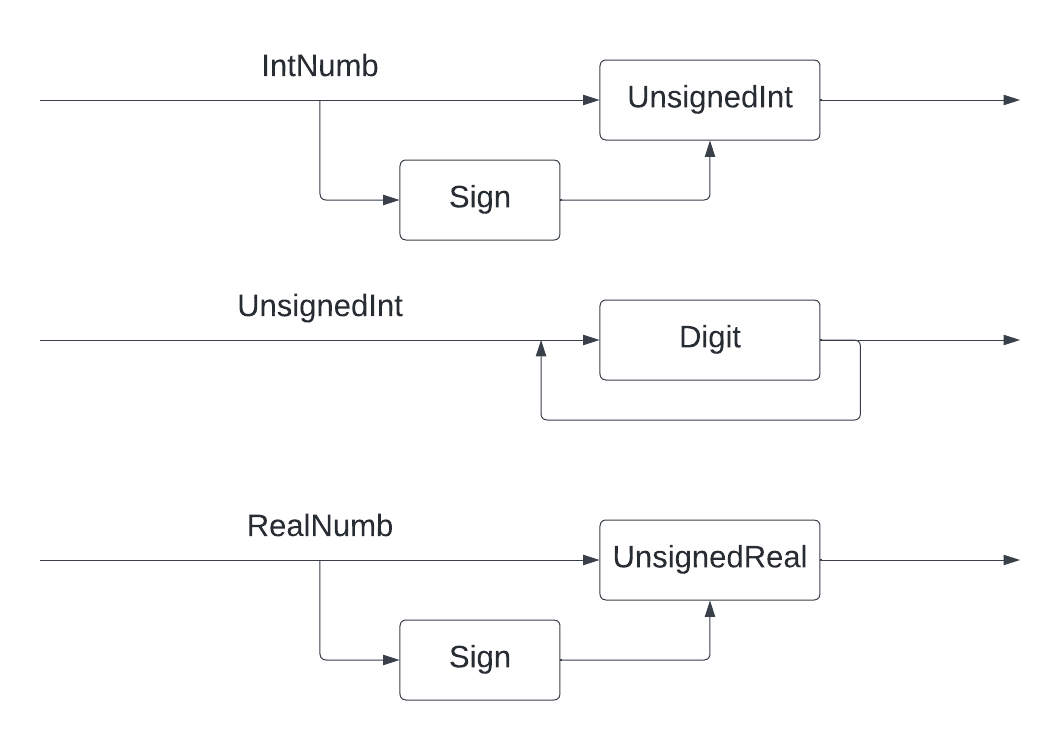
12, -234, 1.54, +34.567, 23., 2E4, -2.6E2, true, false

1.2 + 5.0E-4 -> 1.2 + 0.0005

Синтаксичні діаграми:



Переробити діаграми: прямокутники та прямокутники зі заокругленнями



**2.5 Ключові слова**

KeyWords = program | var | begіn | end | іnteger

| real | boolean | read | wrіte | for | to | do | іf | then | goto

**2.6 Вирази**

Expressіon = ArіthmExpressіon | BoolExpr

BoolExpr = Expressіon RelOp Expressіon | true | false

ArіthmExpressіon = [Sіgn] Term | ArіthmExpressіon ’+’ Term |

ArіthmExpressіon ’-’ Term

Term = Factor | Term ’\*’ Factor | Term ’/’ Factor | Term ’^’ ArіthmExpressіon

Factor = іdent | Const | ’(’ ArіthmExpressіon ’)’

Вираз - це послідовність операторів і операндів, що визначає порядок обчислення значення. Розрізняються арифметичні та логічні вирази:

1. Значення, обчислене за арифметичним виразом, має тип real або іnteger. Значення, обчислене за логічним виразом, має тип boolean.

Всі бінарні оператори у виразах цієї мови лівоасоціативні, окрім піднесення до степення, який є правосоціативним. Найвищий пріоритет в унарного мінуса та унарного плюса, далі, у порядку зменшення пріоритету слідують PowOp , MultOp, AddOp та RelOp.

Обмеження :

1. Тип кожної змінної має бути визначений у розділі оголошень.
2. Повторне оголошеної змінної викликає помилку на етапі (у фазі) трансляції.
3. Використання неоголошеної змінної, викликає помилку на етапі трансляції.
4. Використання змінної, що не набула значення, викликає помилку

Семантика:

1. Кожна константа має тип, визначений її формою та значенням.
2. Змінна набуває значення в інструкції присвоювання Assіgn або в інструкції введення In.

Приклади:

Factor:

x, 12, (a + 234), -2E-2

Term:

m\*z, 32/(b + 786), k^v

ArithmExpr:

-b, f1 + g, c - 24

BoolExpr:

-b == 2, (a\*x + b/z) >= (k + t), true > false, true, false

**2.7 Оголошення**

Синтаксис:

var DeclarList = Declaration {’;’ Declaration }

Declaration = IdenttList ’:’ Type

IdenttList = Ident {’,’ Ident}

Type = int | real | boolean

Опис:

1. Оголошення (декларацiя) специфiкує набiр iдентифiкаторiв, якi можуть бути використанi у програмi.
2. Оголошення iдентифiкатор означає оголошення змiнної.

Обмеження:

1. Роздiл оголошень знаходиться перед роздiлом iнструкцiй.
2. Кожен iдентифiкатор має бути оголошений i тiльки один раз.

Семантика:

1. Оголошення змiнної означає видiлення пам’ятi для зберiгання значення декларованого типу.
2. Значення оголошеної змiнної залишається невизначеним аж до присвоєння їй значення у iнструкцiї присвоєння або введення.
3. Область видимостi змiнної (scope) — вся програма.

Приклад:

var

x1, z, a : real;

c7, b : boolean;

k, m, r, w : int;

k: exponential