**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 операторiв присвоювання, введення, виведення та iнших елементiв мови пропонуються студентом;
4. Семантика усiх елементiв мови пропонується студентом ( у т.ч. операторi циклу та розгалуження).

Результат виконання — специфiкацiя мови програмування.

Специфiкацiя мiстить:

1. Граматику розробленої мови;
2. Опис лексичного складу, синтаксису та семантики усiх елементiв мови
3. Cинтаксичнi дiаграми
4. Приклади програмного коду розробленою мовою.
5. Так званий БАЗОВИЙ ПРИКЛАД програмного коду з використанням УСiХ ЕЛЕМЕНТiВ розробленої мови.

Варiант № 15

Арифметика: цiлi та дiйснi числа, основнi чотири арифметичнi операцiї (додавання, вiднiмання, дiлення та множення), пiднесення до степеня (правоасоцiативна операцiя), дужки

Особливостi: експоненцiйна форма дiйсного числа

iнструкцiя повторення: for (<iд>=<вираз1>; <вiдношення>; <вираз2>)<блок операторiв>

iнструкцiя розгалуження: if <вiдношення> then goto <мiтка>

**Рiшення**

1. Повна граматика розробленої мови:

Program = program ProgName DeclSection DoSection

ProgName = ident

ident = Letter {Letter | Digit }

DeclSection = var DeclarList

DeclarList = Declaration {’,’ Declaration }

Declaration = identtList ’:’ Type

identtList = ident {’,’ ident}

Type = int | real | boolean

DoSection = begin StatementList end

StatementList = Statement {’;’ Statement }

Statement = [Mark ‘:’] Assign | in | Out | ForStatement | Conditional

Assign = ident ’=’ Expression

Expression = ArithmExpression | BoolExpr

BoolExpr = Expression RelOp Expression | true | false

ArithmExpression = [Sign] Term | ArithmExpression ’+’ Term |

ArithmExpression ’-’ Term

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

Factor = ident | Const | ’(’ ArithmExpression ’)’

in = read ’(’ identtList ’)’

Out = write ’(’ identtList ’)’

ForStatement = for indExpr DoBlock

indExpr = ‘(‘ Assign ‘;’ BoolExpr ’;’ Expression ‘)’

DoBlock = Statement ‘;’ | ’begin’ StatementList ’end’

Const = intNumb | RealNumb | BoolConst

Conditional = if BoolExpr then goto Mark

Mark = ident

intNumb = [Sign] Unsignedint

RealNumb = [Sign] UnsignedReal

Sign = ’+’ | ’-’

Unsignedint = Digit {Digit}

UnsignedReal = ’.’ Unsignedint | Unsignedint ’.’ | Unsignedint ’.’

Unsignedint | ExponentForm

ExponentForm = UnsignedReal ‘E’ intNumb

Letter = ’a’ | ’b’ | ’c’ | ’d’ | ’e’ | ’f’ | ’g’ | ’h’ | ’i’ | ’j’ | ’k’ | ’l’ | ’m’ | ’n’ | ’o’ | ’p’ | ’q’ | ’r’ | ’s’ | ’t’ | ’u’ | ’v’ | ’w’ | ’x’ | ’y’ | ’z’ | ’A’ | ’B’ | ’C’ | ’D’ | ’E’ | ’F’ | ’G’ | ’H’ | ’i’ | ’J’ | ’K’ | ’L’ | ’M’ | ’N’ | ’O’ | ’P’ | ’Q’ | ’R’ | ’S’ | ’T’ | ’U’ | ’V’ | ’W’ | ’X’ | ’Y’ | ’Z’

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

BoolConst = true | false

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

2. Опис лексичного складу, синтаксису та семантики усiх елементiв мови

**2.1 Алфавiт**

Програма може мстити текст з використанням таких символiв (character) — лiтер, цифр та спецiальних знакiв.

Letter = ’a’ | ’b’ | ’c’ | ’d’ | ’e’ | ’f’ | ’g’ | ’h’ | ’i’ | ’j’ | ’k’ | ’l’ | ’m’ | ’n’ | ’o’ | ’p’ | ’q’ | ’r’ | ’s’ | ’t’ | ’u’ | ’v’ | ’w’ | ’x’ | ’y’ | ’z’ | ’A’ | ’B’ | ’C’ | ’D’ | ’E’ | ’F’ | ’G’ | ’H’ | ’i’ | ’J’ | ’K’ | ’L’ | ’M’ | ’N’ | ’O’ | ’P’ | ’Q’ | ’R’ | ’S’ | ’T’ | ’U’ | ’V’ | ’W’ | ’X’ | ’Y’ | ’Z’

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

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

EndOfLine = ’\n’

**2.2 Спеціальні символи**

SpecSymbols = RelOp | BracketsOp | AssignOp | Punct | AddOp | MultOp | PowOp

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

BracketsOp = ’(’ | ’)’

AssignOp = ’=’

Punct = ’.’ | ’,’ | ’:’ | ’;’

AddOp = ’+’ | ’-’

MultOp = ’\*’ | ’/’

PowOp = ‘^’

До спецiальних символiв належать арифметичнi оператори, оператори вiдношень, оператор присвоювання та знаки пунктуацiї.

**2.3 Iндентифiкатори**

ident = Letter {Letter | Digit }

Першим символом iдентифiкатора може бути тiльки лiтера, наступнi символи, якщо вони є, можуть бути цифрами або лiтерами. Довжина iдентифiкатора не обмежена.

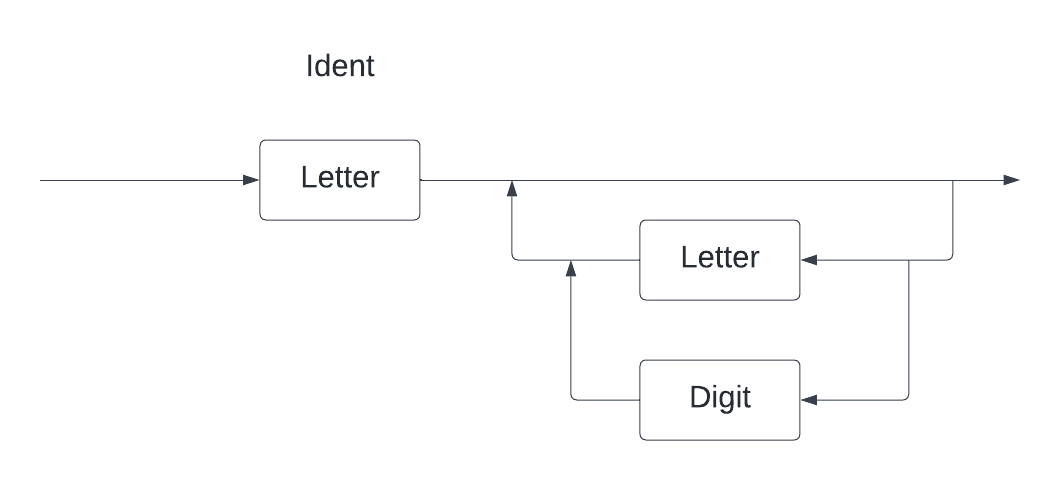
Обмеження:

1. Жоден iдентифiкатор не може збiгатись iз ключовим (вбудованим, зарезервованим) словом або словом true, або false.
2. Елемент, який у фазi лексичного аналiзу може бути визначений як iдентифiкатор або як ключове слово, вважається ключовим словом.
3. Елемент, який у фазi лексичного аналiзу може бути визначений як iдентифiкатор або як логiчна константа, вважається логiчною константою.

Семантика :

iдентифiкатор може позначати змінну, мітку та/або програму.

Синтаксична дiаграма:



Приклади:

a, x1, nagibator2008

**2.4 Константи**

Const = intNumb | RealNumb | BoolConst

intNumb = [Sign] Unsignedint

RealNumb = [Sign] UnsignedReal

Sign = ’+’ | ’-’

Unsignedint = Digit {Digit}

UnsignedReal = ’.’ Unsignedint | Unsignedint ’.’ | Unsignedint ’.’ Unsignedint | ExponentForm

ExponentForm = UnsignedReal ‘E’ [Sign] intNumb

BoolConst = true | false

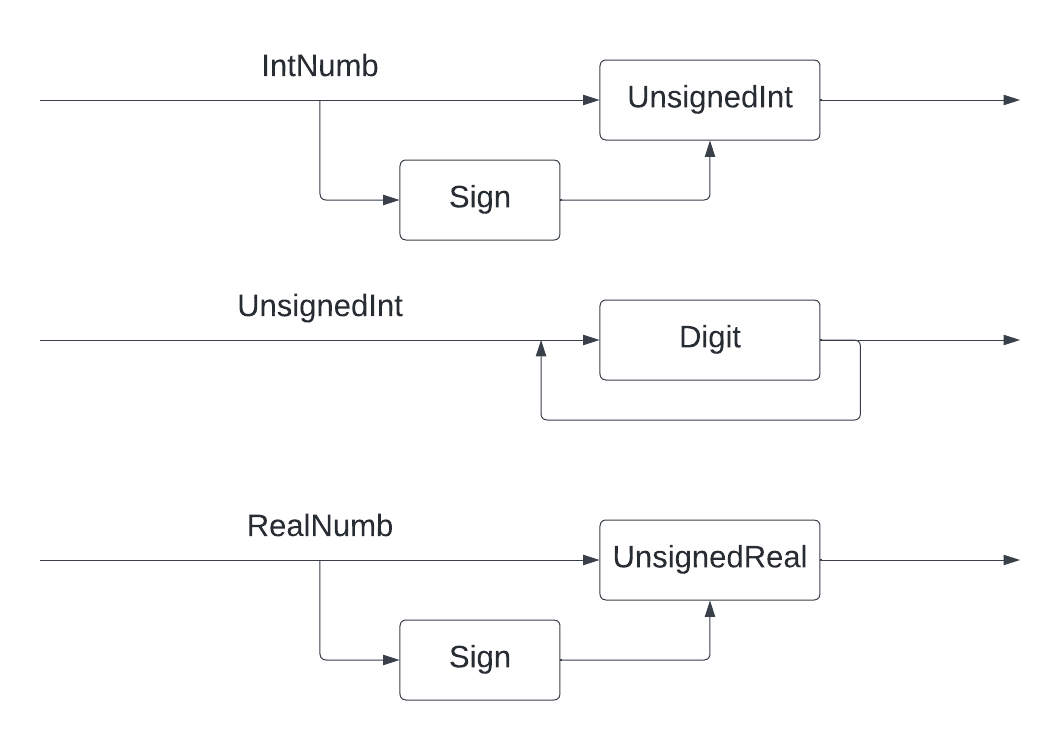
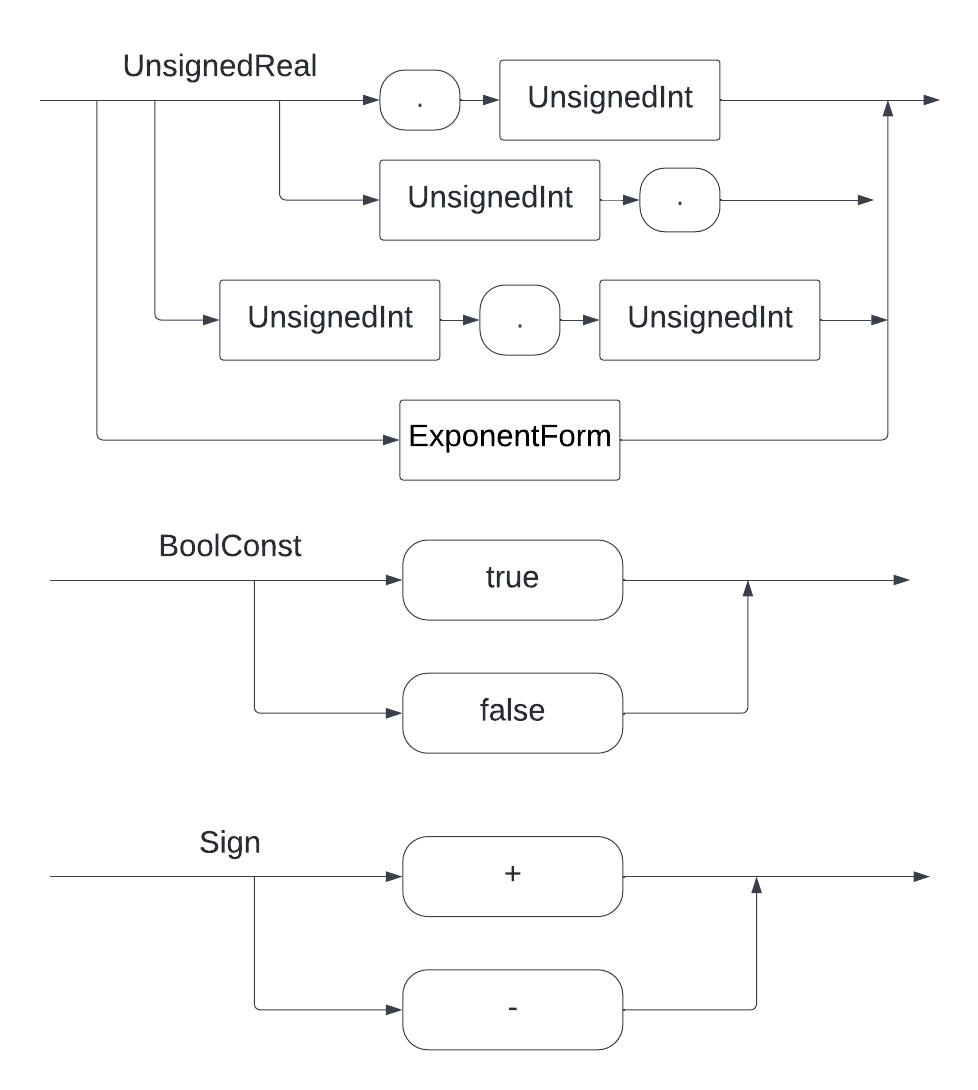
Обмеження:

1. Кожна константа повинна мати тип, а величина константи повинна знаходитись у дiапазонi репрезентативних значень для її типу.
2. На етапi лексичного аналiзу виявляються тiльки беззнаковi цiлi константи Unsignedint, беззнаковi дiйснi константи UnsignedReal та логiчнi константи BoolConst.

Приклади

12, -234, 1.54, +34.567, 23., 2E4, -2.6E2, true, false, 1.2 + 5.E-4

Синтаксичні діаграми:



**2.5 Ключові слова**

KeyWords = program | var | begin | end | int

| real | boolean | read | write | for | if | then | goto

**2.6 Вирази**

Expression = ArithmExpression | BoolExpr

BoolExpr = Expression RelOp Expression | true | false

ArithmExpression = [Sign] Term | ArithmExpression ’+’ Term |

ArithmExpression ’-’ Term

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

Factor = ident | Const | ’(’ ArithmExpression ’)’

Вираз - це послiдовнiсть операторiв i операндiв, що визначає порядок обчислення значення. Розрiзняються арифметичнi та логiчнi вирази:

Значення, обчислене за арифметичним виразом, має тип real або integer. Значення, обчислене за логiчним виразом, має тип boolean.

Всi бiнарнi оператори у виразах цiєї мови лiвоасоцiативнi, окрiм пiднесення до степення, який є правосоцiативним. Найвищий прiоритет в унарного мiнуса та унарного плюса, далi, у порядку зменшення прiоритету слiдують PowOp , MultOp, AddOp та RelOp.

Обмеження :

1. Тип кожної змiнної має бути визначений у роздiлi оголошень.
2. Повторне оголошеної змiнної викликає помилку на етапi (у фазi) трансляцiї.
3. Використання неоголошеної змiнної, викликає помилку на етапi трансляцiї.
4. Використання змiнної, що не набула значення, викликає помилку

Семантика:

1. Кожна константа має тип, визначений її формою та значенням.
2. Змiнна набуває значення в iнструкцiї присвоювання Assign або в iнструкцiї введення in.

Приклади:

Factor:

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

Term:

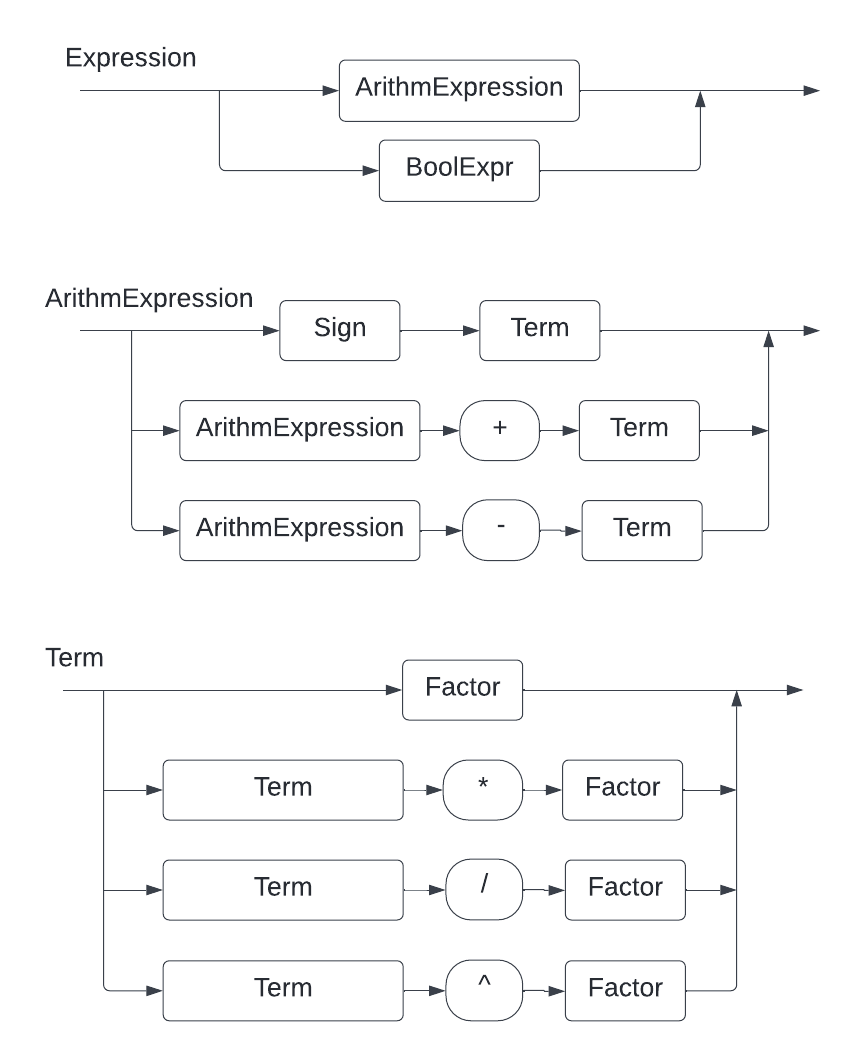
m\*z, 32/(b + 786), k^v

ArithmExpr:

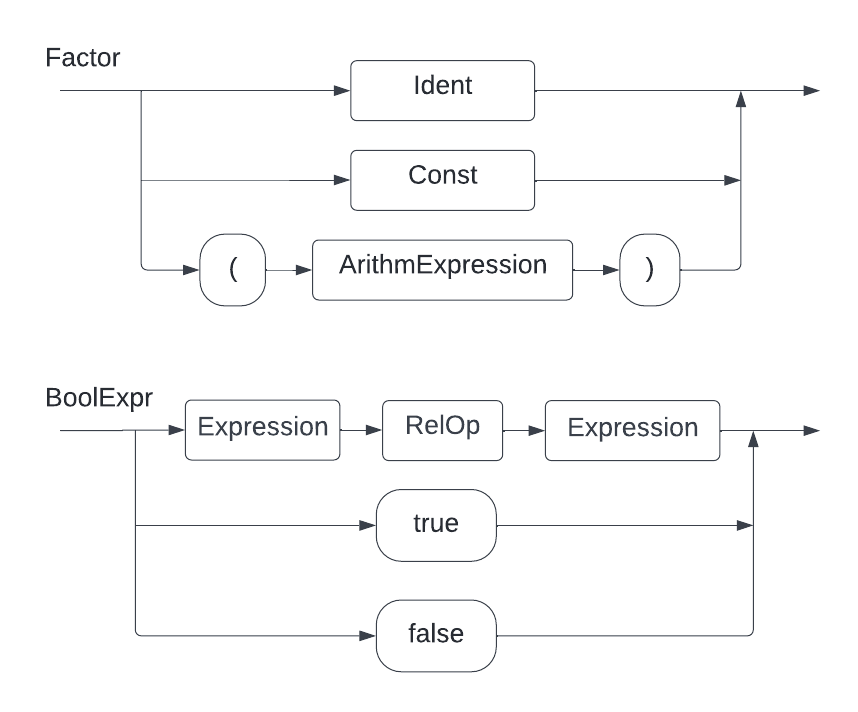
-b, f1 + g, c - 24

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

Синтаксичнi дiаграми:



**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. Розділ оголошень знаходиться перед розділом інструкцій.
2. Кожен ідентифiкатор має бути оголошений i тільки один раз.

Семантика:

1. Оголошення змінної означає виділення пам’яті для зберігання значення декларованого типу.
2. Значення оголошеної змінної залишається невизначеним аж до присвоєння їй значення у інструкції присвоєння або введення.
3. Область видимості змінної (scope) — вся програма.

Приклад:

var

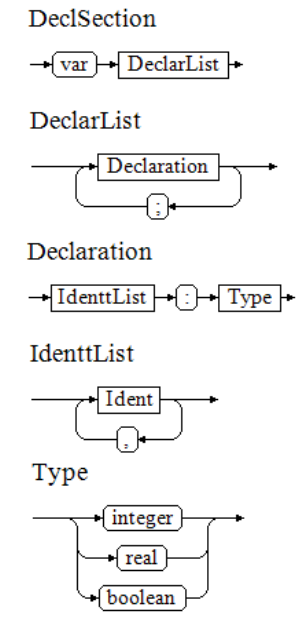
x1, z, a : real;

c7, b : boolean;

k, m, r, w : int;

k: exponential

Синтаксичні дiаграми:



**2.8 iнструкцiї**

Синтаксис:

1. DoSection = begin StatementList ’end.’

StatementList = Statement {’;’ Statement }

Statement = [Mark ‘:’] Assign | in | Out | ForStatement | Conditional

Опис

iнструкцiї (statements) визначають алгоритмiчнi дiї, якi мають бути виконанi у програмi.

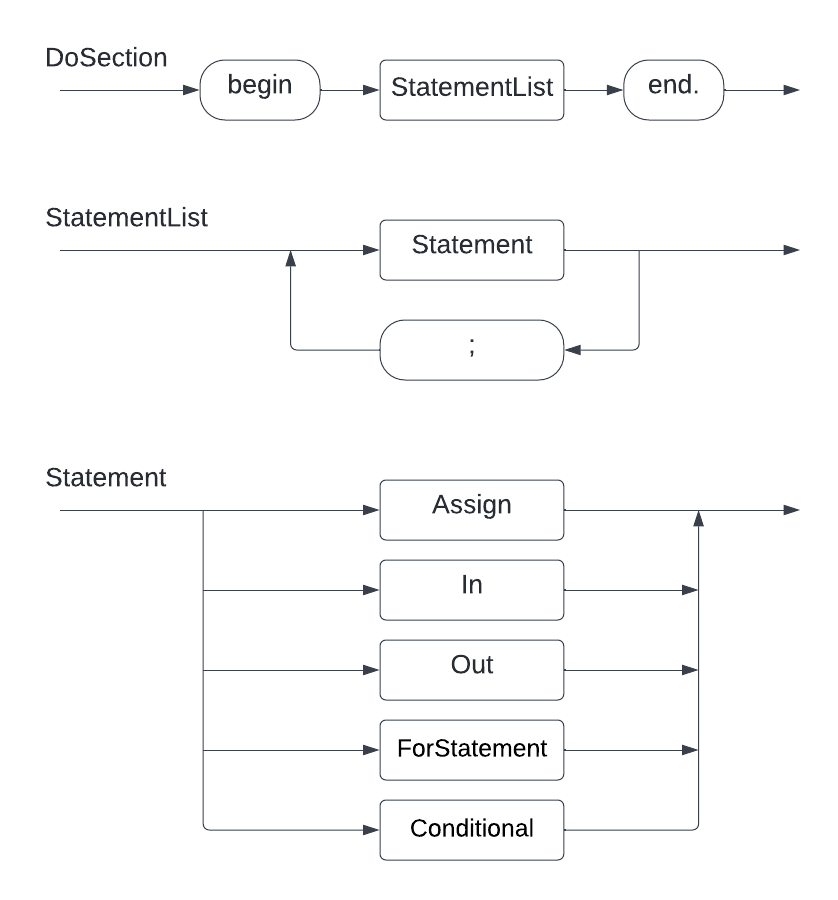
Обмеження

У роздiлi iнструкцiй має бути не менше однiєї iнструкцiї.

Приклад

begin read(z,a,b); x = (z + a)/( z+b ) end.

Синтаксичнi дiаграми:



Синтаксис

Assign = ident ’=’ Expression

Опис

Значення, якi можуть використовуватись у лiвiй та правiй частинах iнструкцiї присвоювання називають l-значенням та r-значенням (або lvalue та rvalue, або left-value та right-value).

Обмеження

Тип змiнної з iдентифiкатором ident повинен збiгатись з типом значенням виразу Expression (типом r-значення).

Семантика

l-значення має тип вказiвника на мiсце зберiгання значеннязмiнної з iдентифiкатором ident. r-значення має тип значення, обчисленого за виразом Expression.

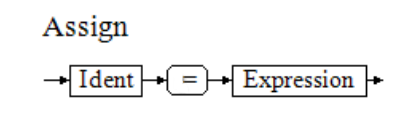
Приклад

f5 = 3/4+1.23,

b7 = 2+3 < 7 / 5,

a = b7

Синтаксична дiаграма:



**2.9 iнструкцiя введення**

Синтаксис

in = read ’(’ identtList ’)’.

Опис

1. Значення вводяться з клавiатури.
2. Введення кожного окремого значення пiдтверджується клавiшею Enter.

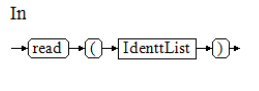
Обмеження

Вiдмiннiсть типу введеного значення вiд типу змiнної викликає помилку.

Приклад

read(a1,v2,len7)

Вiзуальне представлення:



**2.10 iнструкцiя виведення**

Синтаксис

Out = write ’(’ identtList ’)’.

Опис

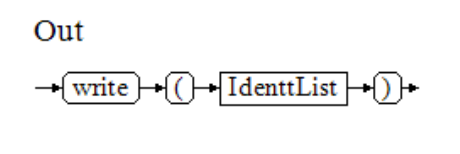
1. Всi значення списку виводяться у один рядок консолi.
2. Кожне значення виводиться у форматi ident=значення.

Обмеження

Виведення змiнної з невизначеним значенням викликає помилку.

Вiзуальне представлення:

Приклад



write(c,x1,f5, ‘\n’ )

**2.11 Оператор циклу**

Синтаксис

ForStatement = for indExpr DoBlock

indExpr = ‘(‘ Assign ‘;’ BoolExpr ’;’ Expression ‘)’

DoBlock = Statement ‘;’ | ’begin’ StatementList ’end’

Assign = ident ’=’ Expression

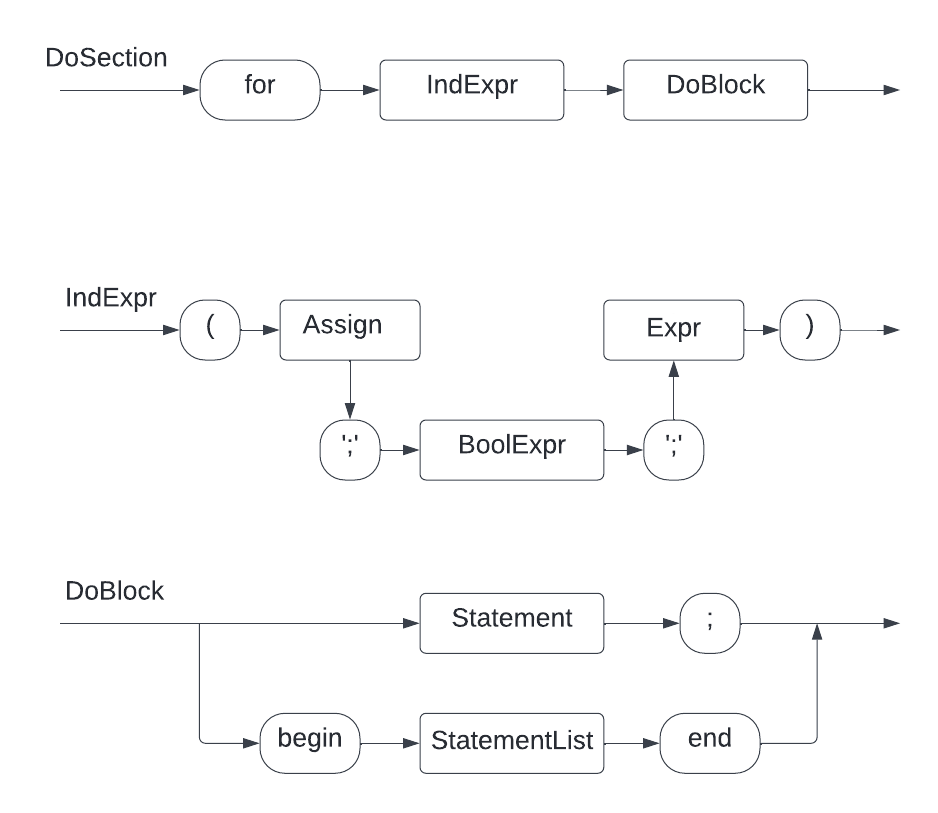
Опис

Тiло оператора циклу DoBlock виконується один або бiльше разiв залежно вiд становища блоку indExpr.

Семантика

1. Перед першою iтерацiєю параметр циклу Assign1 приймає значення та обчислюються значення BoolExpr.
2. Якщо значення true, то виконується один раз тіло циклу DoBlock.
3. Якщо значення false, то цикл завершується.
4. Після першої та наступних ітерацій параметр циклу Ident приймає значення Expression

Вiзуальне представлення



Приклад

for (x = 3 ; x < 10 ; x = x + 1)

begin

x = x ^ 2 ;

write(x);

end;

**2.12 Оператор розгалуження**

Синтаксис

Conditional = if BoolExpr then goto Mark

Mark = ident

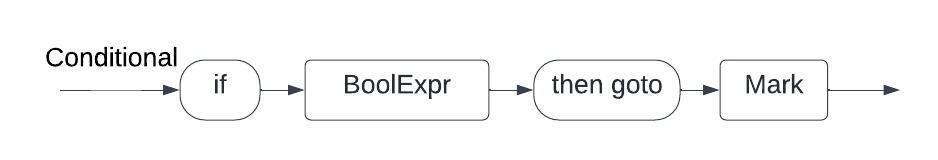
Опис

Перехiд до вiдмiтки Mark виконується один раз залежно вiд становища блоку BoolExpr.

Семантика

1. Спершу обчислюються значення BoolExpr.
2. Якщо значення true, то виконується перехід до відмітки Mark. Повинна існувати відмітка на яку переходить оператор розгалуження в блоці DoBlock.
3. Якщо значення false, то оператор розгалуження заверується

Дiаграма



Приклади

if 1 > c then goto 15

if true then goto c

**3. Програма**

Синтаксис

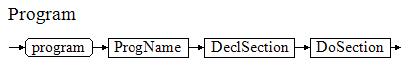
Program = program ProgName DeclSection DoSection

ProgName = ident

Опис

Кожна програма починається з термiнала program та iдентифiкатора програми, за iдентифiкатором програми — (єдиний) роздiл оголошень, далi — (єдиний) роздiл iнструкцiй. iдентифiкатор програми нiяк не використовується у програмi.

Вiзуальне представлення



**4. Токени**

|  |  |  |  |
| --- | --- | --- | --- |
| Код | Приклади лексем | Токен | Неформальний опис |
| 1 | Ab , g , Gg3, h1hj4l | id | ідентифікатор |
| 2 | 123, 0, 4632 | int | ціле без знака |
| 3 | 1.2, .44, 222. | real | дійсне без знака |
| 4 | true, false | bool | логічне значення |
| 5 | program | keyword | символ program |
| 6 | var | keyword | символ var |
| 7 | begin | keyword | символ begin |
| 8 | end | keyword | символ end |
| 9 | end. | keyword | символ end. |
| 10 | int | keyword | символ int |
| 11 | boolean | keyword | символ boolean |
| 12 | read | keyword | символ read |
| 13 | write | keyword | символ write |
| 14 | for | keyword | символ for |
| 15 | if | keyword | символ if |
| 16 | then | keyword | символ then |
| 17 | goto | keyword | символ goto |
| 18 | = | assign\_op | символ = |
| 19 | + | add\_op | символ + |
| 20 | - | add\_op | символ - |
| 21 | \* | mult\_op | символ \* |
| 22 | / | mult\_op | символ / |
| 23 | ^ | pow\_op | символ ^ |
| 24 | < | rel\_op | символ < |
| 25 | <= | rel\_op | символ <= |
| 26 | == | rel\_op | символ == |
| 27 | > | rel\_op | символ > |
| 28 | >= | rel\_op | символ >= |
| 29 | != | rel\_op | символ != |
| 30 | ( | brackets\_op | символ ( |
| 31 | ) | brackets\_op | символ ) |
| 32 | . | punct | символ . |
| 33 | , | punct | символ , |
| 34 | : | punct | символ : |
| 35 | ; | punct | символ ; |
| 36 | \n | eol | кінець рядка |

**4. Базовий приклад програми**

program modifiedFactorial

var

n , x , result :int;

modifiedResult :real;

repeat :boolean

begin

toRead: read(n);

if n <= 0 then goto toRead;

result = 1;

for (x = 1; x < n ; x + 1)

begin

result = result \* (x+1);

end;

write(result, ‘\n’);

modifiedResult = ( result / 10^(n+1) ) - 1.E-4;

write(modifiedResult, ‘\n’);

If 100 >= modifiedResult then goto toRead;

read(repeat);

if repeat == true then goto toRead;

endlessLoop :repeat = false;

If repeat != false then goto endlessLoop;

repeat = modifiedResult >= 50;

write(repeat);

end