Специфікація мови програмування S20

1 Вступ

Мова програмування S20 – мова програмування з можливістю ітерування та використання логічних операторів. S20 – імперативна мова загального призначення, передбачає побудову монолітних програм.

1.1 Завдання

Розробити мову програмування з наступними можливостями:

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

Особливості: експоненційна форма дійсного числа.

Інструкція повторення: for <ід>=<вираз> to <вираз> step <вираз> do <список операторів> next

Інструкція розгалуження: if <логічний вираз>then <список операторів> else <список операторів> endif

1.2 Обробка

Програма, написана мовою S20, подається на вхід транслятора (компілятора або інтерпретатора) для трансформації (перекладу, трансляції) до цільової форми (мови). Результат трансляції виконується у системі часу виконання (run-time system), для чого приймає вхідні дані та надає результат виконання програми.

Трансляція передбачає фази лексичного, синтаксичного та семантичного аналізу, а також фазу генерації коду. Фази лексичного та синтаксичного аналізу здійснюються окремими проходами.

1.3 Нотація

Метасимвол	Значення
=	визначається як
|	альтернатива
[x]	0 або 1 екземпляр x
{x}	0 або більше екземплярів x
(x | y)	групування: будь -який з x або y
Zxy	нетермінал
zxy	термінал
‘1’	термінал
“1”	термінал

Табл. 1: Прийнята нотація РБНФ

Для опису мови S20 використовується розширена форма Бекуса – Наура.

Ланцюжки, що починаються з великої літери вважаються нетерміналами (нетермінальними символами). Термінали (термінальні символи) — ланцюжки, що починаються з маленької літери, або ланцюжки, що знаходяться між одинарними або подвійними лапками. Для графічного представлення граматики використовуються синтаксичні діграми Вірта.

1.4 Алфавіт

Програма може містити текст з використанням таких символів (character)

— літер, цифр, спеціальних знаків та ознаки кінця файлу.

Синтаксис

```
ULetter = "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".
LLetter = "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".
Letter = ULetter | LLetter.
Digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9".
```

2 Лексика

Лексичний аналіз виконується окремим проходом, отже не залежить від синтаксичного розбору та семантичного аналізу. Лексичний аналізатор розбиває текст програми на лексеми. У програмі мовою S20 можуть використовуватись лексичні елементи, що класифікуються як спеціальні символи, ідентифікатори, беззнакові цілі константи, беззнакові дійсні константи, логічні константи та ключові слова.

2.1 Спеціальні символи та ключові слова

Синтаксис

1. SpecSymbols = ArithOp | RelOp | BracketsOp | AssignOp | Punct.
ArithOp = AddOp | MultOp | DegOp.
AddOp = "+" | "-".
MultOp = "*" | "/".
DegrOp = "^".
RelOp = "==" | "!=" | "<=" | ">=" | "<" | ">".
BracketsOp = "(" | ")".
AssignOp = "=".
Punct = "{" | "}" | "." | "," | ":" | ";".

KeyWords= LoopW | CondW | BaseW | BoolW

LoopW = ‘for’ | ‘to’ | ‘step’ | ‘do’ | ‘next’

CondW = ‘if’ | ‘then’ | ‘else’ | ‘endif’

BaseW = ‘main’| ‘int’ | ‘float’

BoolConst = ‘true’ | ‘false’

Опис

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

До ключових слів належать оператори циклу, оператори умов(розгалуження), базові оператори та типи.

Обмеження

3. Набір токенів, див. табл. 2, містить токени add_op та mult_op, але не містить arith_op.

2.1.1 Ідентифікатори

Опис

Ідентифікатори можуть бути будь-якої довжини. Правопис ідентифікатора складається з символу, без урахування регістру та будь-якої кількості символів чи цифр. Жоден ідентифікатор не повинен мати те саме написання, що і будь-яке ключове слово. Ідентифікатор позначає змінну.

Ident = Letter {Letter | Digit }

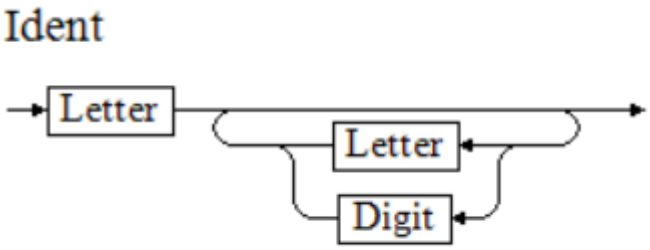
Приклади:

X
time
readinteger
WG4
AlterHeatSetting
InquireWorkstationTransformation
InquireWorkstationIdentification

Семантика

Елемент, який у фазі лексичного аналізу може бути визначений як ідентифікатор або як ключове слово, вважається ключовим словом.

Елемент, який у фазі лексичного аналізу може бути визначений як ідентифікатор або як логічна константа, вважається логічною константою.



2.2 Константи

Синтаксис

Const = ArithmConst | BoolConst.
ArithmConst = Int | Float.
Int = [Sign] UnsignedInt.
Float = [Sign] UnsignedFloat.
Sign = "+" | "-".

UnsignedInt = Digit {Digit}.

UnsignedFloat = UnsignedInt {UnsignedInt}"."UnsignedInt{UnsignedInt}.

BoolConst = true | false.

Обмеження

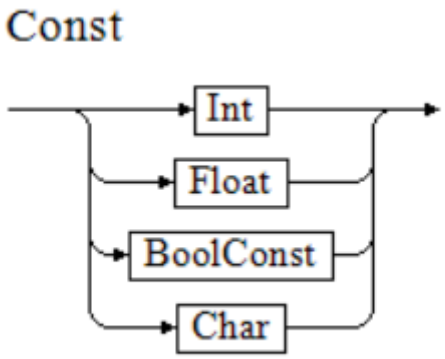
Кожна константа повинна мати тип, а величина константи повинна знаходитись у діапазоні репрезентативних значень для її типу.

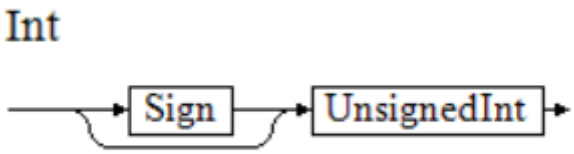
На етапі лексичного аналізу виявляються тільки беззнакові цілі константи UnsignedInt, беззнакові дійсні константи UnsignedFloat та логічні константи BoolConst.

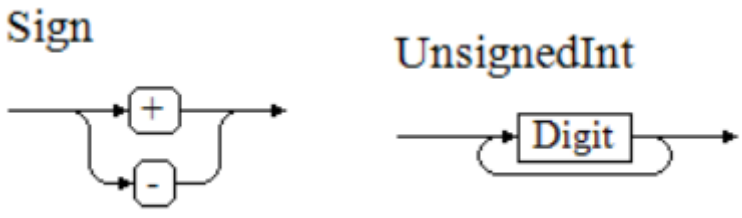
Семантика

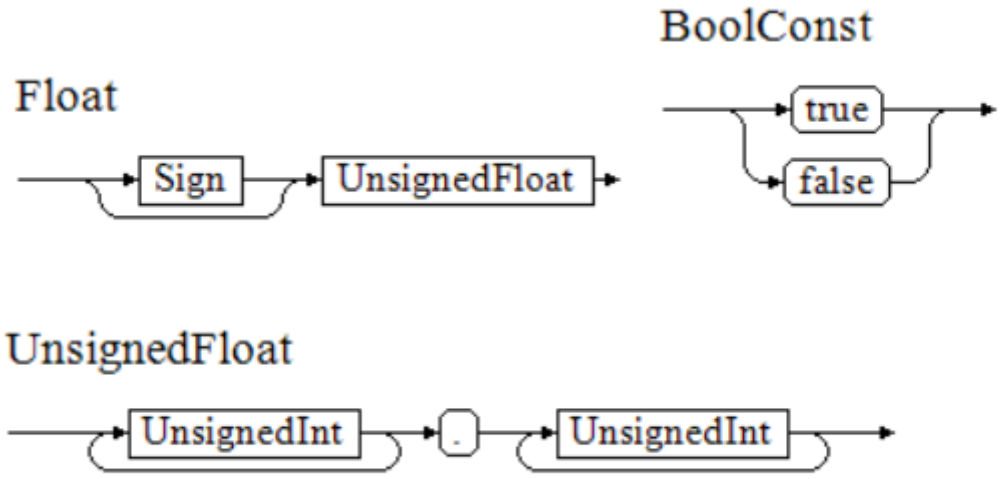
Кожна константа має тип, визначений її формою.

Візуальне представлення









Синтаксичні діаграми див. на рис. 1.

Приклади

6.12, 234, 1.54, 34.567, 23, true, false

2.4 Токени

З потоку символів вхідної програми на етапі лексичного аналізу виокремлюються послідовності символів з певним сукупним значенням, — токени. Список токенів мови див. у табл. 2. Єдиний приклад засвідчує унікальність лексеми.

Код	Приклад лексем	Токен	Неформальный опис
1	X, xx, x, temp	ident	Ідентифікатор
2	5, 14, 21, 37	int	Ціле число без знаку
3	5.5, 14.7, 55.7	float	Дійсне без знаку
4	true, false	bool	Логічне без знаку
5	main	keyword	Символ main
6	bool	keyword	Символ bool, ініціалізація змінної логічного типу
7	int	keyword	Символ int, ціле число
8	float	keyword	Символ float, дійсне число
9	if	keyword	Символ if, для оператора умови

10	then	keyword	Символ then, для оператора умови
11	else	keyword	Символ else, для оператора умови
12	endif	keyword	Символ endif, для оператора умови
13	for	keyword	Символ for, для оператора циклу
14	to	keyword	Символ to, для оператора циклу
15	step	keyword	Символ step, для оператора циклу
16	do	keyword	Символ do, для оператора циклу
18	next	keyword	Символ next, для оператора циклу
19	+	add_op	Символ +
20	-	add_op	Символ -
21	*	mult_op	Символ *
22	/	mult_op	Символ /
23	^	degr_op	Символ ^
24	=	assign_op	Символ =
25	==	rel_op	Символ == , порівняння
26	!=	rel_op	Символ != , порівняння
27	>=	rel_op	Символ >= , порівняння
28	<=	rel_op	Символ <= , порівняння
29	>	rel_op	Символ > , порівняння
30	<	rel_op	Символ < , порівняння
31	(	brackets_op	Символ (
32	)	brackets_op	Символ)
33	:	punct	Символ :
34	;	punct	Символ ;
35	\n, ;\n	eol	Кінець рядка
36	\32, \t	ws	Пробільні символи
37	{	punct	Символ {

38	}	punct	Символ }
39	^^	degr_op	Символ e

Табл. 2: Таблиця лексем мови

3 Типи

Мова S20 підтримує значення трьох типів: int, float та bool.

1. Цілий тип int може бути представлений оголошеною змінною типу int. Діапазон значень — [-2147483648; 2147483647].

2. Дійсний тип Float може бути представлений оголошеною змінною типу float. Діапазон значень — [$3.4e^{-38}$; $3.4e^{38}$]

3. Логічний тип bool може бути представлений оголошеною змінною типу bool (true або false). Прийнято false.

4 Синтаксис

4.1 Вирази

Синтаксис

```
Expression = ArithmExpression | BoolExpr.
BoolExpr = ArithmExpression RelOp ArithmExpression.
ArithmExpression = [ Sign] Term | ArithmExpression "+" Term |
ArithmExpression "-" Term.
Term = Factor | Term "*" Factor | Term "/" Factor | Term "^" Factor.
Factor = Ident | Const | "(" ArithmExpression ")".
```

Опис

Вираз - це послідовність операторів і операндів, що визначає порядок обчислення значення.

Розрізняються арифметичні та логічні вирази.

Значення, обчислене за арифметичним виразом, має тип float або int.

Значення, обчислене за логічним виразом, має тип bool.

Найвищий пріоритет DegrOp, далі, у порядку зменшення пріоритету слідують MultOp, AddOp та RelOp.

Послідовність двох або більше операторів з однаковим пріоритетом асоціативна.

Обмеження

Повторне оголошеної змінної викликає помилку на етапі (у фазі) трансляції.

Використання неоголошеної змінної, викликає помилку на етапі трансляції.

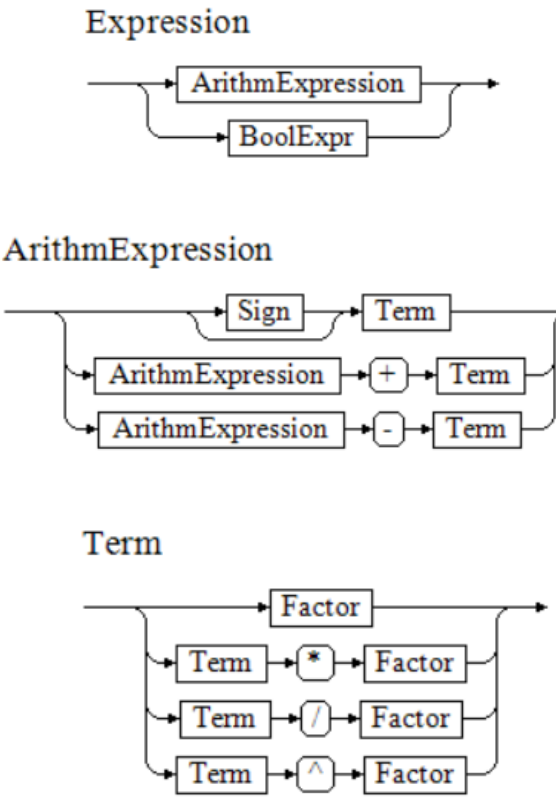
Використання змінної, що не набула значення, викликає помилку.

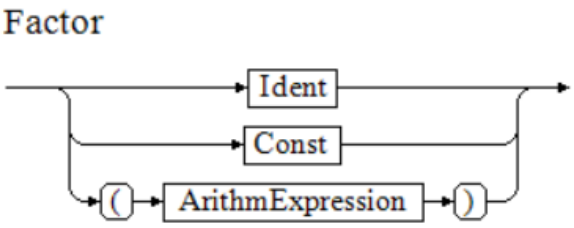
Семантика

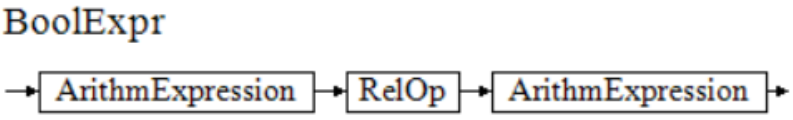
Кожна константа має тип, визначений її формою та значенням.

Змінна набуває значення в інструкції присвоювання Assign або в інструкції введення Inp.

Візуальне представлення:







4.2 Оператори

4.2.1 Арифметичні оператори

Синтаксис

1. AddOp = ’+’ | ’-’

   MultOp = ’*’ | ’/’

   DegrOp = "^" | “^^”.

Оператор	Операція	Типи операндів	Тип результату
+	Додавання	int, float	float, якщо хоч один float, інакше int
-	Віднімання	int, float	
*	Множення	int, float	
/	Ділення	int, float	float
^	Степінь	int, float	float, якщо хоч один float, інакше int

Табл. 3: бінарні арифметичні оператори

Оператор	Операція	Типи операнда	Тип результату
+	Ідентичність	int float	int float
-	Зміни Знаку	int float	int float

Табл. 4: унарні арифметичні оператори

Оператор	Операція	Типи операнда	Тип результату
^^	експоненційна форма дійсного числа	float	float

Семантика

Тип результату при застосуванні бінарних операторів див. табл. 3.

Тип результату при застосуванні унарних операторів див. табл. 4.

Ділення на нуль викликає помилку.

Приклад

6. 1.234 * a13 / 45.67 - 3.4 + 6, 7 / 8,5^9

4.2.2 Оператори відношення

Синтаксис

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

Обмеження

Значення обох операндів мають бути або числовими (типу int або float), або логічними (типу bool).

Результат завжди має тип bool.

Семантика

Якщо один з операндів має тип int, а інший — float, то значення типу int приводиться до типу float.

Приклад

5. x1+3 < 1, 0.5*2.34 >= 52, 424 != a14

5 Оголошення

Синтаксис

```
Decl = Type DeclarList “;”.
DeclarList = Declaration {"," Declaration}.
Declaration = Ident.
Type = int | float | bool.
```

Опис

Оголошення (декларація) специфікує набір ідентифікаторів, які можуть бути використані у програмі.

Оголошення ідентифікатор означає оголошення змінної.

Обмеження

Кожен ідентифікатор має бути оголошений і тільки один раз.

Семантика

Оголошення змінної означає виділення пам’яті для зберігання значення декларованого типу.

Значення оголошеної змінної залишається невизначеним аж до присвоєння їй значення у інструкції присвоєння або введення.

Область видимості змінної (scope) — вся програма.

Візуальне представлення

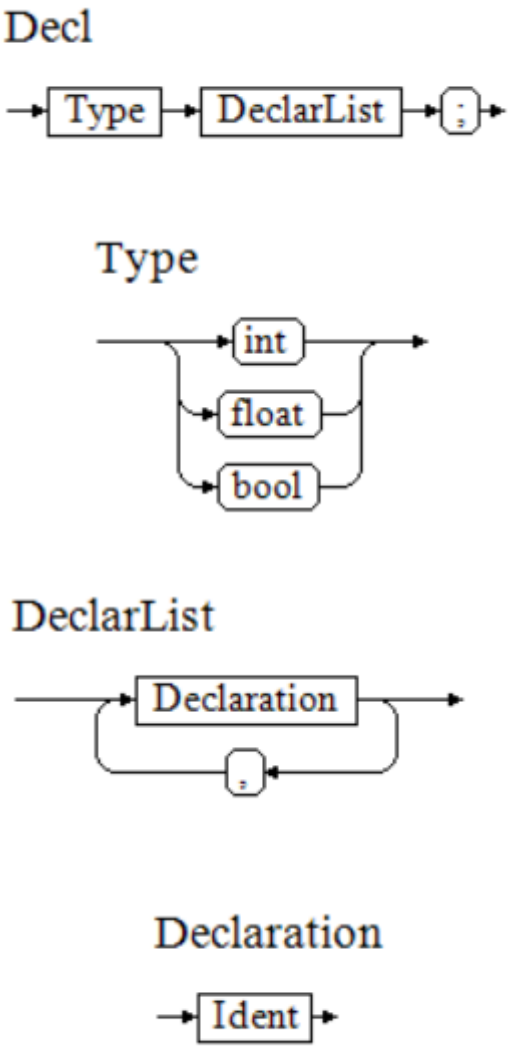


Рис. 2: Розділ оголошень

Синтаксичні діаграми див. на рис. 2.

Приклад

```
int a;
float Temp1, temP2;
bool temp;
```

6. Інструкції

Синтаксис

```
Program = main "{" StatementList "}".
StatementList = Statement {";" Statement}.
Statement = Assign | Inp | Out | ForStatement | Decl | IfStatement.
```

Опис

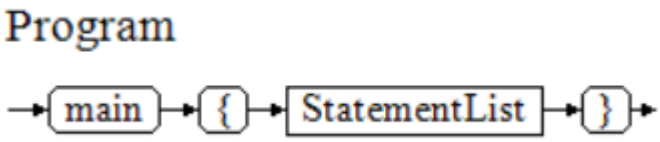
Інструкції визначають алгоритмічні дії, які мають бути виконані у програмі.

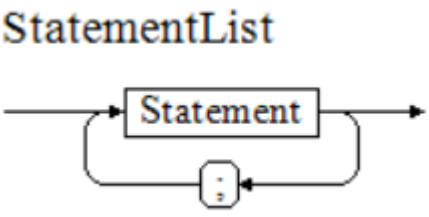
Обмеження

У розділі інструкцій має бути не менше однієї інструкції.

Семантика

Візуальне представлення





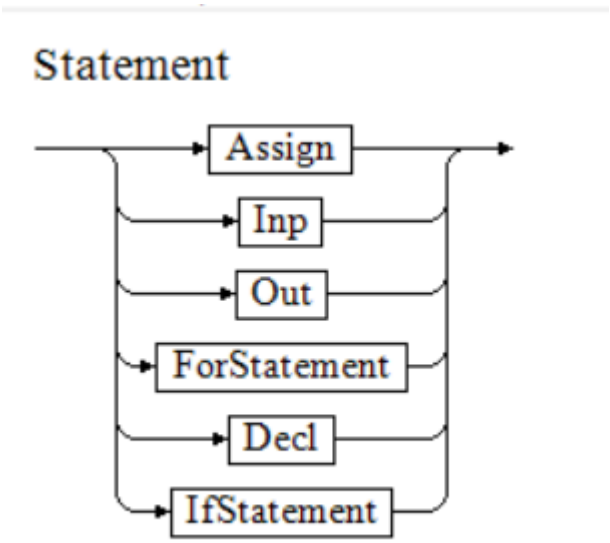


Рис. 3: Розділ інструкцій

6. Синтаксичні діаграми див. на рис. 3.

Приклад

```
main{
        int a;
        a = 5;
        echo (a);
}
```

6.1 Оператор (інструкція) присвоювання

Синтаксис

Assign = Ident ’=’ Expression

Опис

Значення, які можуть використовуватись у лівій та правій частинах інструкції присвоювання називають l-значенням та r-значенням (або lvalue та rvalue, або left-value та right-value).

Обмеження

Тип змінної з ідентифікатором Ident повинен збігатись з типом r-значення.

Семантика

l-значення має тип вказівника на місце зберігання значення змінної з ідентифікатором Ident.

r-значення має тип значення, обчисленого за виразом Expression.

Візуальне представлення

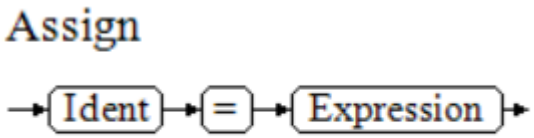


Рис. 4: Оператор присвоювання

5. Синтаксичну діаграму див. на рис. 4.

Приклад

6. Xx = 3/4+1.23;

   b7 = 2+3 < 4;

6.2 Інструкція введення

Синтаксис

Inp = read “(“DeclarList “)”

Опис

Значення вводяться з клавіатури.

Введення кожного окремого значення підтверджується клавішею Enter.

Обмеження

Відмінність типу введеного значення від типу змінної викликає помилку.

Візуальне представлення

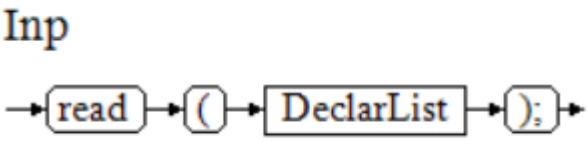


Рис. 5: Інструкція введення

Синтаксичну діаграму див. на рис. 5.

Приклад

```
read(a1,v2,len7) ;
```

6.3 Інструкція виведення

Синтаксис

Out = echo "(" (DeclarList | Expression) ")".

Опис

Всі значення списку виводяться у один рядок консолі.

Обмеження

Виведення змінної з невизначеним значенням виводить слово ‘undefined’.

Візуальне представлення

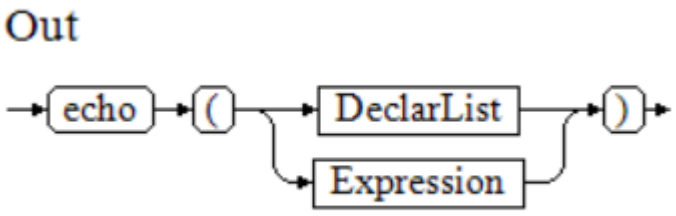


Рис. 6: Інструкція виведення

Синтаксичну діаграму див. на рис. 6.

Приклад

```
echo(c,x1,f5);
```

6.4 Оператор циклу (інструкція повторення)

Синтаксис

ForStatement = for IndExpr to ArithmExpression2 step ArithmExpression3 do StatementList next.
IndExpr = Ident AssignOp ArithmExpression1 "+" (ArithmExpression3 | "0").
ArithmExpression1 = ArithmExpression.
ArithmExpression2 = ArithmExpression.

ArithmExpression3 = ArithmExpression.Опис

Опис

Тіло оператора циклу StatementList виконується один або більше разів.

Обмеження

Параметр циклу Ident — змінна типу int.

Значення ArithmExpression1, ArithmExpression2 та ArithmExpression3 мають тип int або приводяться до типу int.

Семантика

Перед першою ітерацією обчислюються значення ArithmExpression1, ArithmExpression2 та ArithmExpression3.

Якщо значення ArithmExpression1 ≤ ArithmExpression2, з кроком ArithmExpression3 виконуємо.

Перед першою ітерацією параметр циклу Ident приймає значення ArithmExpression1.

Після першої та наступних ітерацій параметр циклу Ident приймає значення Ident = Ident + ArithmExpression3.

StatementList виконується, якщо значення Ident ∈ [ArithmExpression1; ArithmExpression2].

Візуальне представлення

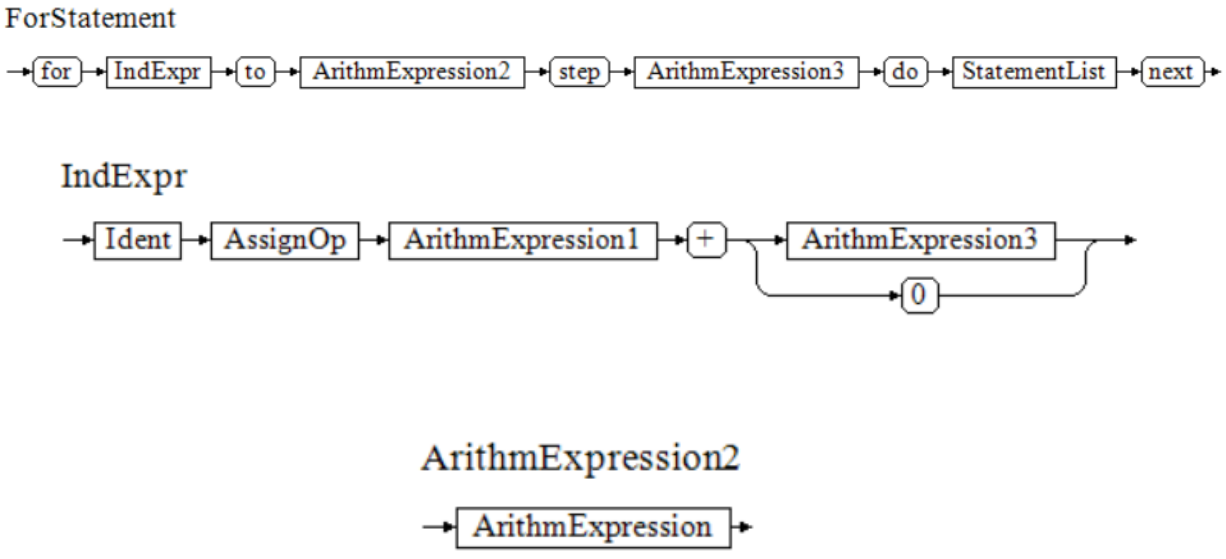


рис. 7

Синтаксичні діаграми див. на рис. 7.

Приклад

```
for i = 3  to 5*64 step 4/2 do echo(i) next
```

6.5 Оператор розгалуження

Синтаксис

IfStatement = if CheckExpression (then DoSomething1 | else DoSomething2) endif.

CheckExpression = Expression.

DoSomething1 = StatementList.

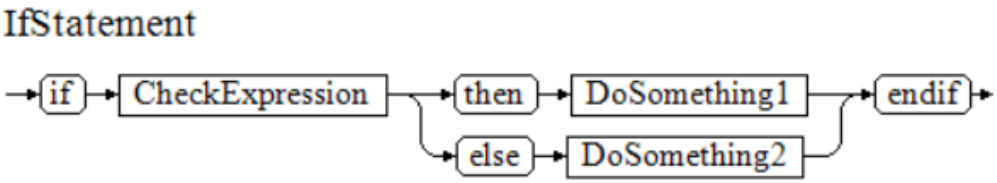
DoSomething2 = StatementList.

Опис

Тіло оператора розгалуження виконується один раз.

Набір виконуваних дій залежить він значення виразу CheckExpression

У разі отримання значення true виконується блок then DoSomething1, інакше - else DoSomething2



7 Програма

Синтаксис

Program = main "{" StatementList "}".

Опис

Кожна програма починається з main та відкриваючої фігурної дужки.

Кожна програма закінчується закриваючою дужкою.

В кінці кожної інструкції потрібно поставити “;\n” або “;”

Візуальне представлення

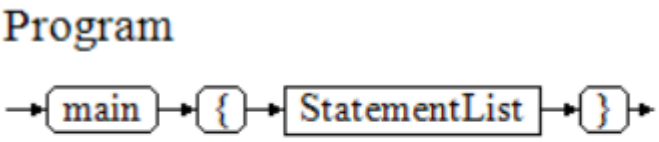


Рис. 8

Обмеження

В разі використання неіснуючої інструкції виникне помилка

Приклад

```
main{
    int a, b;
    float a1;

    a = -56*6;
    b = 86^^6;
    a1 = b*0.67;
    for id=a to 50+224 step 1^4 do
        if(i > 0)
        then
            echo(i)
        else
            echo(a1*i)
        endif
    next
}
```

8 Повна граматика мови S20

Decl = Type DeclarList ";".

DeclarList = Declaration {"," Declaration}.

Declaration = Ident.

Ident = Letter {Letter | Digit }.

Program = main "{" StatementList "}".

StatementList = Statement {";" Statement}.

Statement = Assign | Inp | Out | ForStatement | Decl | IfStatement.

Assign = Ident AssignOp Expression.

Inp = read "("DeclarList ")".

Out = echo "(" (DeclarList | Expression) ")".

ForStatement = for IndExpr to ArithmExpression2 step ArithmExpression3 do StatementList next.

IndExpr = Ident AssignOp ArithmExpression1 "+" (ArithmExpression3 | "0").

ArithmExpression1 = ArithmExpression.

ArithmExpression2 = ArithmExpression.

ArithmExpression3 = ArithmExpression.

Expression = ArithmExpression | BoolExpr.

BoolExpr = ArithmExpression RelOp ArithmExpression.

ArithmExpression = [Sign] Term | ArithmExpression "+" Term | ArithmExpression "-" Term.

Term = Factor | Term "*" Factor | Term "/" Factor | Term "^" Factor.

Factor = Ident | Const | "(" ArithmExpression ")".

IfStatement = if CheckExpression (then DoSomething1 | else DoSomething2) endif.

CheckExpression = BoolExpr.

DoSomething1 = StatementList.

DoSomething2 = StatementList.

Const = ArithmConst | BoolConst.

ArithmConst = Int | Float.

Int = [Sign] UnsignedInt.

Float = [Sign] UnsignedFloat.

Sign = "+" | "-".

UnsignedInt = Digit {Digit}.

UnsignedFloat = UnsignedInt {UnsignedInt}"."UnsignedInt{UnsignedInt}.

KeyWords = LoopW | CondW | BaseW | BoolW.

LoopW = for | to | step | do | next.

CondW = if | then | else | endif.

BaseW = main | Type.

Type = int | float | bool.

ULetter = "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".

LLetter = "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".

Letter = ULetter | LLetter.

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

BoolConst = true | false.

SpecSymbols = ArithOp | RelOp | BracketsOp | AssignOp | Punct.

ArithOp = AddOp | MultOp | DegOp.

AddOp = "+" | "-".

MultOp = "*" | "/".

DegrOp = "^".

RelOp = "==" | "!=" | "<=" | ">=" | "<" | ">".

BracketsOp = "(" | ")".

AssignOp = "=".

Punct = "{" | "}" | "." | "," | ":" | ";".