Спецификация языка программирования S20

1 Введение

Язык программирования S20 - язык программирования с возможностью использования цикла for и логического оператора if. S20 - императивный язык общего назначения, предусматривает построение монолитных программ.

1.1 Задачи

Разработать язык программирования со следующими возможностями:

арифметика: Основные четыре арифметические операции (сложение, вычитание, деление и умножение), возведение в степень (правоасоциативна операция), скобки.

Инструкция повторения: For <ид> = <выражение> to <выражение> step <выражение> do <список

операторов> next

Инструкция ветвления: If <логическое выражение> then <список операторов> else

<Список операторов> endif

1.2 Обработка

Программа, написанная на языке S20, подается на вход транслятора (компилятора или интерпретатор) для трансформации (перевода, трансляции) до целевой формы. Результат трансляции выполняется в системе времени исполнения (run-time system), для чего принимает входные данные и предоставляет результат выполнения программы.

Трансляция предусматривает фазы лексического, синтаксического и семантического анализа и получения в результате трансляции ПОЛИЗ кода. Далее происходит интерпретация полез. Фазы лексического и синтаксического анализа и интерпретации осуществляются отдельными проходами. Во время синтаксического анализа происходит трансляция кода программы.

1.3 Нотація

метасимвол	значение
=	определяется как
|	альтернатива
[X]	0 или 1 экземпляр x
{X}	0 или более экземпляров x
(X | y)	группировки: любой -Какой с x или y
Zxy	нетерминал
zxy	терминал
'1'	терминал
"1"	терминал

Табл. 1: Принятая нотацыя РБНФ

Для описания языка S20 используется расширенная форма Бэкуса - Наура.

Цепочки, начинающиеся с большой буквы считаются нетерминаламы (нетерминальнимы символами). Терминалы (терминальные символы) - цепочки, начинающиеся с маленькой буквы, или цепочки, которые находятся между одинарными или двойными кавычками. Для графики представления грамматики используются синтаксичні діграмы Вирта.

1.4 Алфавит

Программа может содержать текст с использованием таких символов (character)

- букв, цифр, специальных знаков.

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

Лексический анализ выполняется отдельным проходом, так что не зависит от синтаксического разбора и семантического анализа. Лексический анализатор разбивает текст программы на лексемы. В программе языке S20 могут использоваться лексические элементы, классифицируются как специальные символы, идентификаторы, беззнакововые константы, беззнаковые действительны константы, логические константы и ключевые слова.

2.1 Специальные символы и ключевые 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 '| "Bool"

BoolConst = 'true' | "False"

описание

2. В специальных символов относятся арифметические операторы, операторы отношений, оператор присваивания и знаки пунктуации.

К ключевым словам относятся операторы цикла, операторы условий (разветвления), базовые операторы и типы.

2.1.1 Идентификаторы

описание

Идентификаторы могут быть любой длины. Правописание идентификатора состоит из символа, без учета регистра и любого количества символов или цифр. Ни один идентификатор не должен иметь то же написание, что и любое ключевое слово. Идентификатор обозначает переменную.

Ident = Letter {Letter | Digit}

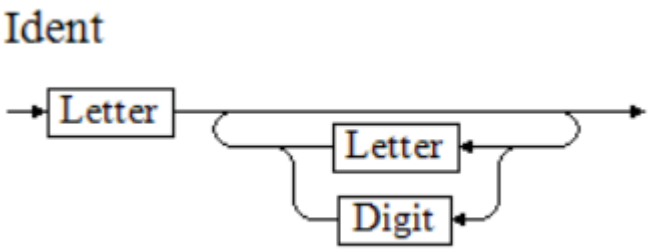
примеры:

X
time
readinteger
WG4
AlterHeatSetting
InquireWorkstationTransformation
InquireWorkstationIdentification

семантика

Элемент, который в фазе лексического анализа может быть определен как идентификатор или как ключевое слово, считается ключевым словом.

Элемент, который в фазе лексического анализа может быть определен как идентификатор или как логическая константа, считается логическим константой.



2.2 Константы

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.

ограничения

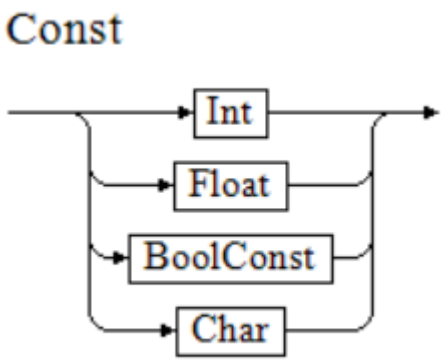
Каждая константа должна иметь тип, а величина константы должна находиться в диапазоне репрезентативных значений для ее типа.

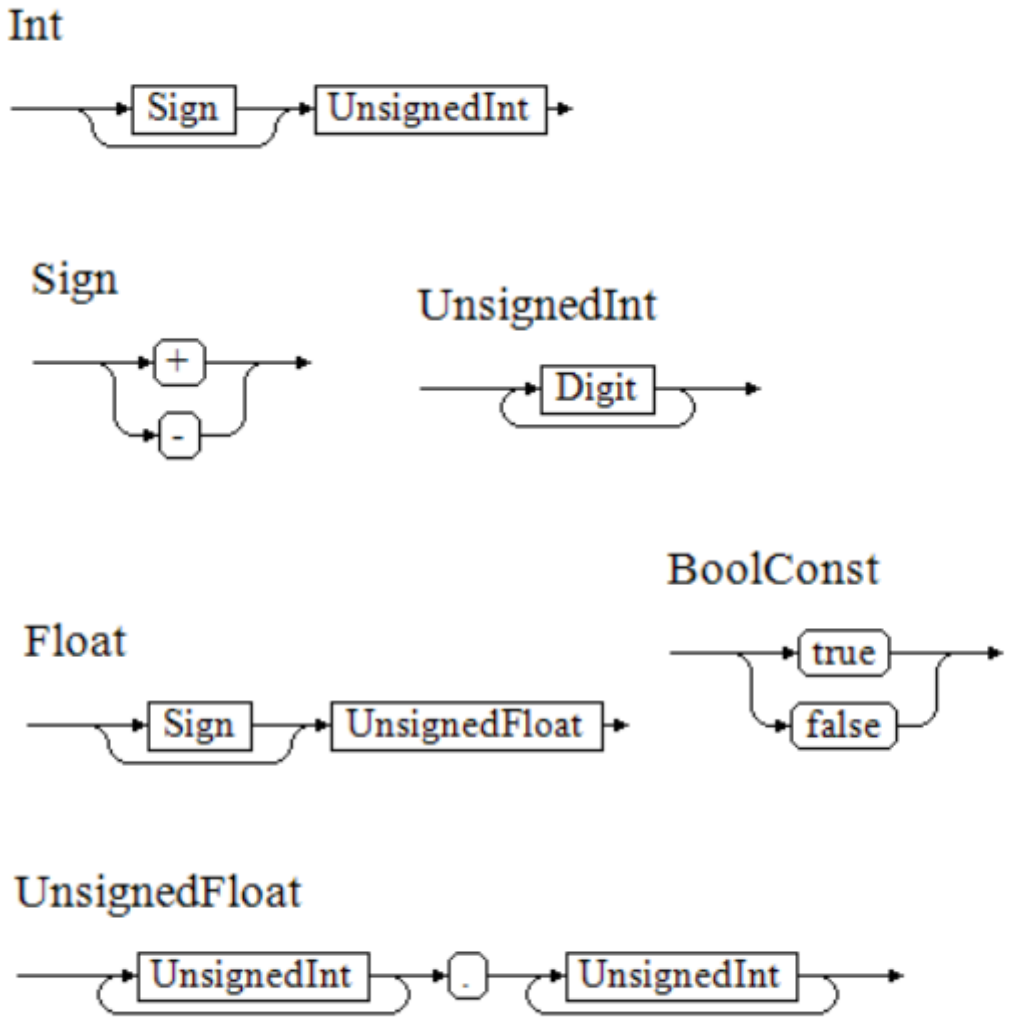
На этапе лексического анализа оказываются только беззнакові цели константы UnsignedInt, беззнакові действительны константы UnsignedFloat и логические константы BoolConst.

семантика

Каждая константа имеет тип, определенный ее формой.

визуальное представление





Синтаксичні диаграммы см. на рис. 1.

примеры

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

2.4 Токены

Лексический анализатор принимает на вход код программы, написанной входной языке. Из потока набора символов выделяются последовательности символов с определенным значением - токены. Каждая лексема имеет свой токен, который обозначает ее назначения. Список токенов языка см. в табл. 2.

код	пример лексем	Токен	Неформальное описание
1	X, xx, x, temp	ident	идентификатор
2	5, 14, 21, 37	int	Целое число (константа) без знака

3	5.5, 14.7, 55.7 14e4, 5e-6	float	Действительное число (константа) без знака
4	true false	bool	Символы true, false для обозначения логического типа
5	main	keyword	символ main
6	bool	keyword	Символ bool, инициализация переменной логического типа
7	int	keyword	Символ int для обозначения целого типа
8	float	keyword	Символ float для обозначения действительного типа
9	if	keyword	Символ if, оператора условия
10	then	keyword	Символ then, оператора условия
11	else	keyword	Символ else, оператора условия
12	endif	keyword	Символ endif, оператора условия
13	for	keyword	Символ for для оператора цикла
14	to	keyword	Символ to, оператора цикла
15	step	keyword	Символ step, оператора цикла
16	do	keyword	Символ do, оператора цикла
18	next	keyword	Символ next,

			оператора цикла
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 ==, сравнение
26	! =	rel_op	Символ! =, Сравнение
27	> =	rel_op	Символ> =, сравнение
28	<=	rel_op	Символ <=, сравнение
29	>	rel_op	Символ>, сравнение
30	<	rel_op	Символ <сравнения
31	(	brackets_op	символ (
32	)	brackets_op	символ)
34	;	end_row	символ;
35	\ n	nl	конец строки
36	\ 32 (""), \ t	ws	пробельные символы
37	{	brackets_op	символ {
38	}	brackets_op	символ}
39	,	coma	Симлов,

Табл. 2: Таблица лексем языка

3 Типы

Язык S20 поддерживает значение трех типов: int, float и bool.

1. Целый тип int может быть представлен объявленной переменной типа int. Диапазон значений - [-2147483648; 2147483647].

2. Действительный тип Float может быть представлен объявленной переменной типа

float. Диапазон значений - [$3.4e^{-38}$; $3.4e^{38}$]

3. Логический тип bool может быть представлен объявленной переменной типа bool (true или false).

4 Синтаксис

4.1 Выражения

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 ")".
```

описание

Выражение - это последовательность операторов і операндов, определяющий порядок вычисления значения.

Различаются арифметические и логические выражения.

Значение, вычисленное по арифметическим выражением, имеет тип float или int.

Значение, вычисленное по логическим выражением, имеет тип bool.

Самый высокий приоритет DegrOp, дальше, в порядке

уменьшение приоритета следуют MultOp, AddOp и RelOp.

Последовательность двух или более операторов с одинаковым приоритетом ассоциативной.

ограничения

Повторное объявленной переменной вызывает ошибку на этапе (в фазе) трансляции.

Использование необъявленной переменной, вызывает ошибку на этапе трансляции.

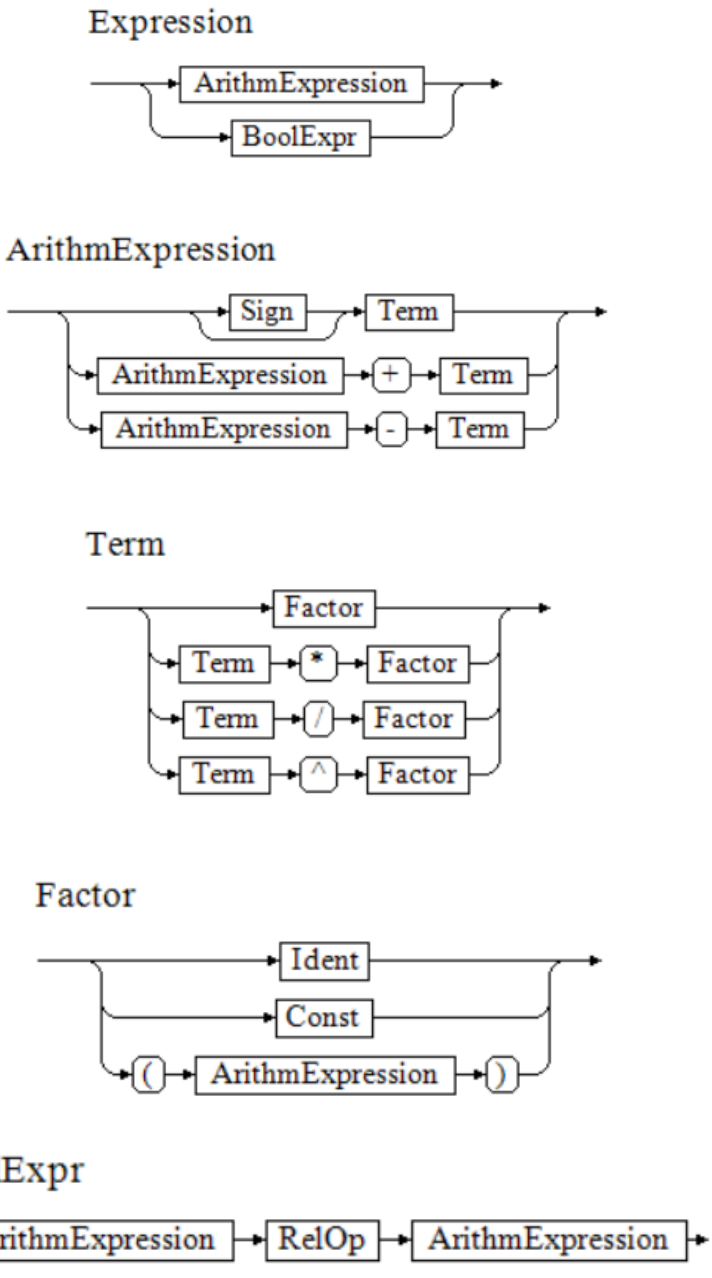
Использование переменной, не приобрела значение, вызывает ошибку.

семантика

Каждая константа имеет тип, определенный ее формой и значением.

Переменная принимает значение в инструкции присваивания Assign или в инструкции введение Inp.

Визуальное представление:



4.2 Операторы

4.2.1 Арифметические операторы

синтаксис

1. AddOp = '+' | '-'

MultOp = '*' | '/'

DegrOp = "^" | "^^".

оператор	операция	типы операндов	Тип результата
+	Добавление	int, float	float, если хоть один float, иначе int
-	вычитание	int, float	
*	умножения	int, float	
/	Диленные	int, float	float
^	Степень	int, float	float, если хоть один float, иначе int

Табл. 3: бинарные арифметические операторы

оператор	операция	типы операнда	Тип результата
+	идентичность	int float	int float
-	изменения Знака	int float	int float

Табл. 4: унарные арифметические операторы

оператор	операция	типы операнда	Тип результата
^^	экспоненциальная форма действительного числа	float	float

семантика

Тип результата при применении бинарных операторов см. табл. 3.

Тип результата при применении унарных операторов см. табл. 4.

Деления на ноль вызывает ошибку.

пример

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

4.2.2 Операторы отношение

синтаксис

RelOp = '==' | "! = '| '<=' | '<' | '>' | '> ='

ограничения

Значение обоих операндов должны быть либо числовыми (типа int или float).

Результат всегда имеет тип bool.

семантика

Если один из операндов имеет тип int, а другой - float, то значение типа int приводится к типу float.

пример

5. x1 + 3 <1, 0.5 * 2.34> = 52, 424! = A14

5 Объявления

синтаксис

```
Decl = Type Declaration "=" (Float | Int | Ident | BoolConst) ",".
Declaration = Ident.
Type = int | float | bool.
```

описание

Объявления (декларация) - набор идентификаторов, которые могут быть использованы в программе.

Объявления идентификатор означает объявление переменной.

ограничения

Каждый идентификатор должен быть объявлен і только один раз.

семантика

Объявления переменной означает выделение памяти для хранения значения

декларируемого типа.

Значение объявленной переменной должно сразу быть известно.

Область видимости переменной (scope) - вся программа.

визуальное представление

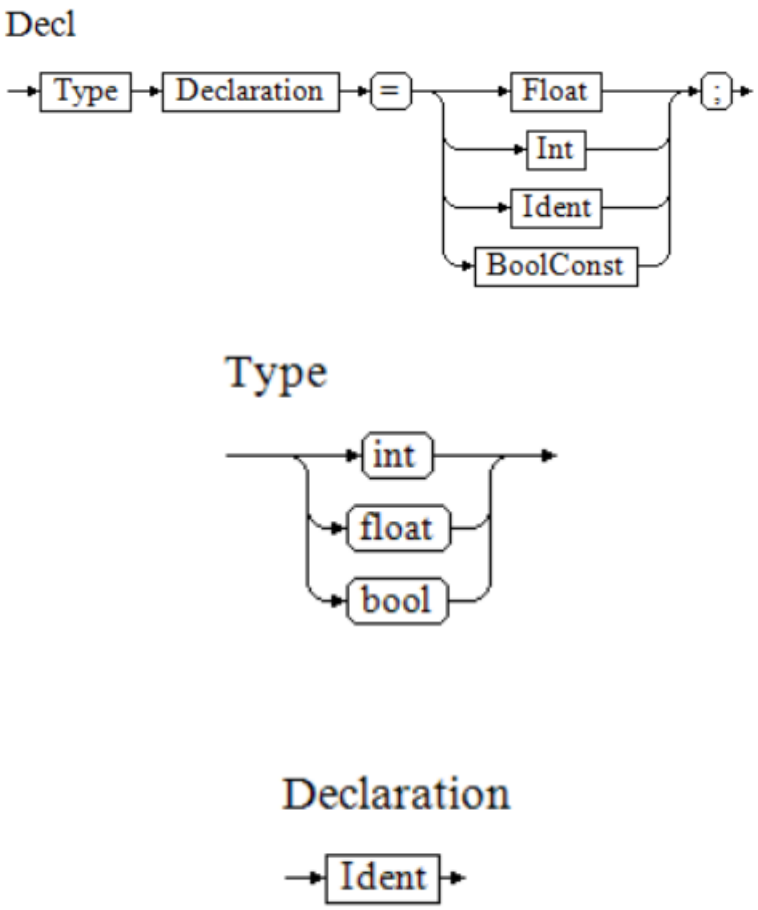


Рис. 2: Раздел объявлений

Синтаксичкские диаграммы см. на рис. 2.

пример

```
int a = 5;
float Temp1 = 1.4e4;
bool temp = False;
```

6. Инструкции

синтаксис

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

описание

Инструкции определяют алгоритмические действия, которые должны быть выполнены в программе.

ограничения

В разделе инструкций должно быть не менее одной инструкции.

семантика

визуальное представление

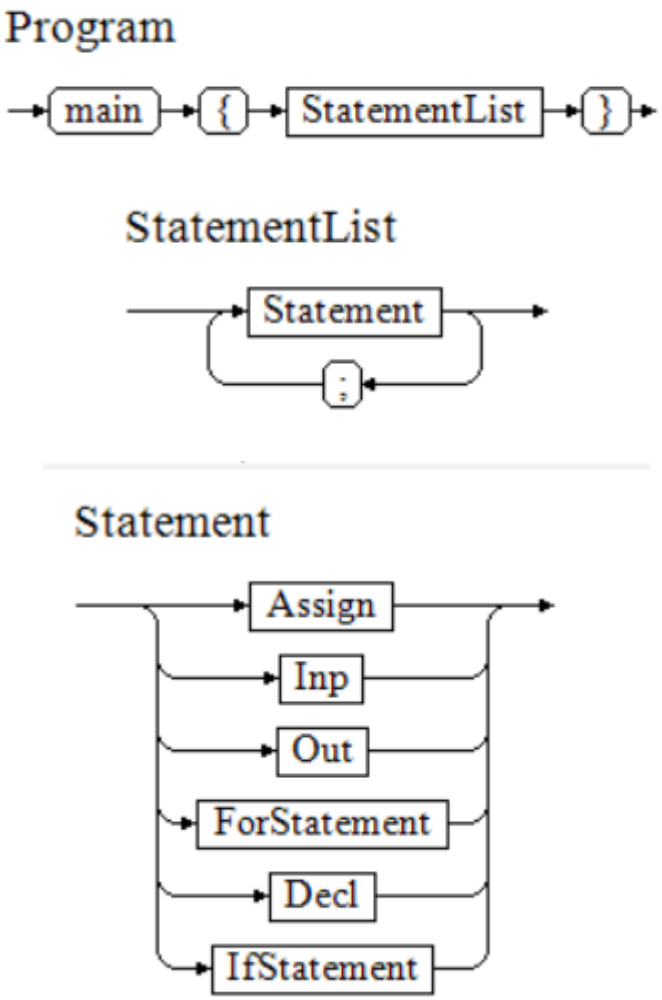


Рис. 3: Раздел инструкций

6. Синтаксические диаграммы см. на рис. 3.

пример

```
main {
        int a = 0;
        a = 5;
        echo (A)
}
```

6.1 Оператор (инструкция) присваивания

синтаксис

Assign = Ident '=' Expression

описание

Значения, которые могут использоваться в левой и правой частях инструкции присваивания называют l-значением и r-значением (или lvalue и rvalue, или left-value и right-value).

ограничения

Тип переменной с идентификатором Ident должен совпадать с типом r-значение, однако значение типа int из правой части может записаться в переменную с типом float.

семантика

l-значение имеет тип указателя на место хранения значения переменной с

идентификатором Ident.

r-значение имеет тип значения, вычисленного по выражению Expression.

визуальное представление

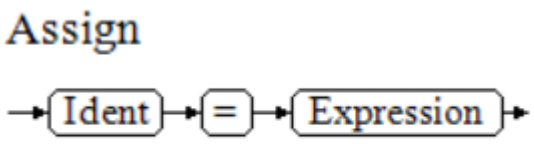


Рис. 4 Оператор присваивания

5. синтаксических диаграмму см. на рис. 4.

пример

6. Xx = 3/4 + 1.23;

   b7 = 2 + 3 <4;

6.2 Инструкции ввода

синтаксис

Inp = read "(" DeclarList ")"

описание

Значения вводятся с клавиатуры.

Введение каждого отдельного значения подтверждается клавишу Enter.

ограничения

Отличие типа введенного значения от типа переменной вызывает ошибку.

визуальное представление

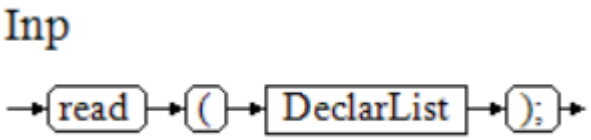


Рис. 5: Инструкция введение

Синтаксическую диаграмму см. на рис. 5.

пример

read (a1, v2, len7)

6.3 Инструкция вывода

синтаксис

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

описание

Все значения списке выводятся в одну строку консоли.

ограничения

Вывод переменной с неопределенным значением выводит слово 'undefined'.

визуальное представление

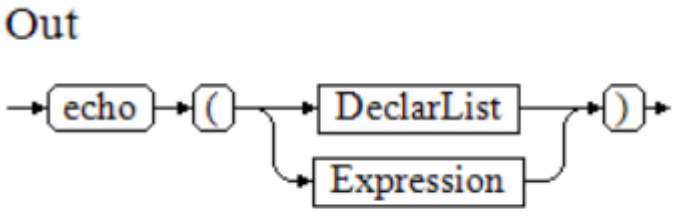


Рис. 6: Инструкция вывода

Синтаксическую диаграмму см. на рис. 6.

пример

echo (c, x1, f5)

6.4 Оператор цикла (инструкция повторения)

синтаксис

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

описание

Тело оператора цикла StatementListвыполняется один или более раз.

ограничения

Параметр цикла Ident - переменная численного типа.

значение ArithmExpression1, ArithmExpression2 и ArithmExpression3 имеют численный тип.

семантика

Перед первой итерации вычисляются значения ArithmExpression1, ArithmExpression2 и ArithmExpression3.

если значение ArithmExpression1≤ ArithmExpression2С шагом ArithmExpression3выполняем.

Перед первой итерации параметр цикла Ident принимает значение

ArithmExpression1.

После первой и последующих итераций параметр цикла Ident принимает значение Ident = Ident +ArithmExpression3.

StatementList выполняется, если значение Ident∈ [ArithmExpression1; ArithmExpression2].

визуальное представление

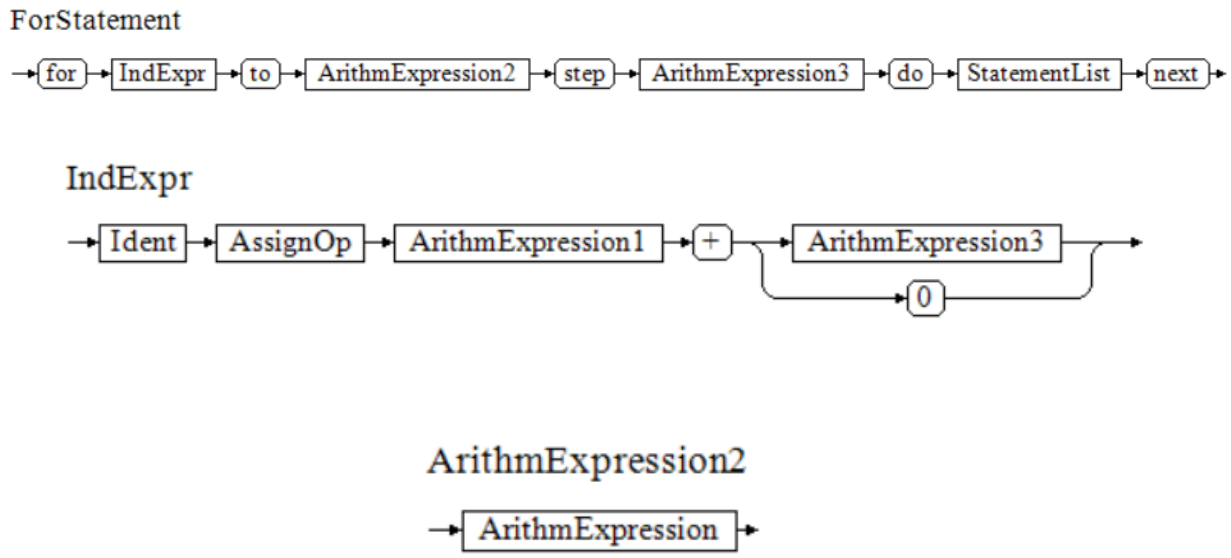


рис. 7

Синтаксические диаграммы см. на рис. 7.

пример

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

6.5 Оператор ветвления

синтаксис

IfStatement = if CheckExpression "then" DoSomething1 "else" DoSomething2 "endif".

CheckExpression = Expression.

DoSomething1 = StatementList.

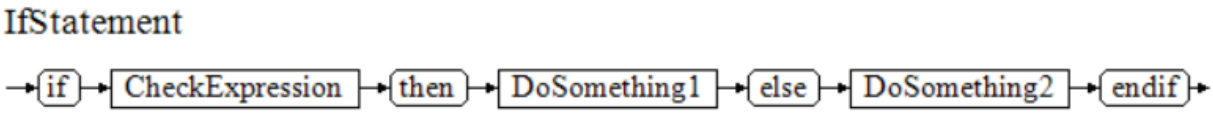
DoSomething2 = StatementList.

описание

Тело оператора ветвления выполняется один раз.

Набор выполняемых действий зависит он значение выражения CheckExpression

В случае получения значения true выполняется блок then DoSomething1,иначе - else DoSomething2



7 Программа

синтаксис

Program = main "{" StatementList "}".

описание

Каждая программа начинается с main и открывающей фигурной скобки.

Каждая программа заканчивается закрывающей скобкой.

В конце каждой инструкции нужно поставить "; \ n" или ","

визуальное представление

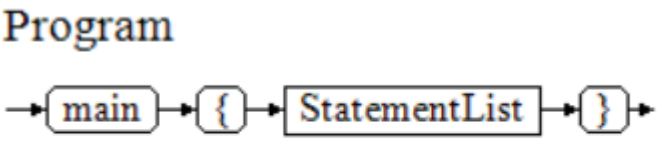


Рис. 8

ограничения

При использовании несуществующей инструкции возникнет ошибка

пример

```
main {
    int a = 5,
    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 Полная грамматика языка S20

Decl = Type Declaration "=" (Float | Int | Ident | BoolConst) ",".

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] UnsignedFloat "e" [Sign] Int.

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 | EndRow | Nl.

ArithOp = AddOp | MultOp | DegOp.

AddOp = "+" | "-".

MultOp = "*" | "/".

DegrOp = "^".

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

AssignOp = "=".

Punct = ".".

EndRow = ",".

Nl = "\ n".

Ws = "\ t" | "".

BracketsOp = "(" | ")" | "{" | "}".