<Лука Катерина Анатоліївна> <ТВ-з81>

**Специфiкацiя мови програмування FuriousPeach**

**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. операторів циклу та розгалуження).

Результат виконання — специфікація мови програмування.

**Специфікація містить:**

1. граматику розробленої мови;

2. синтаксичні діаграми;

3. опис лексичного складу, синтаксису та семантики усіх елементів мови; 4. приклади програмного коду розробленою мовою.

5. так званий БАЗОВИЙ ПРИКЛАД програмного коду з використанням УСІХ ЕЛЕМЕНТІВ розробленої мови.

Варіант № 7

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>=<вираз>; <відношення>; <ід>=<вираз>){<cписок операторів>}

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

1. **Вступ**

Мова програмування FuriousPeach призначена слугувати навчальним прикладом для курсу, що має на метi вивчення основ трансляцiї. Назва ”FuriousPeach” в перекладі з англійської мови означає ”Несамовитий персик”. Furious Peach - iмперативна мова загального призначення.

**Завдання**

Розробити Pascal-подiбну мову програмування з оператором (iнструкцiєю) циклу:

for ( <Iдентифiкатор> = <Вираз> ; <Відношення> ; <Iдентифiкатор> = <Вираз>){ <Список операторів> }

**Обробка**

Програма, написана мовою FuriousPeach, подається на вхiд транслятора (компiлятора або iнтерпретатора) для трансформацiї (перекладу, трансляцiї) до цiльової форми (мови). Результат трансляцiї виконується у системi часу виконання (run-time system), для чого приймає вхiднi данi та надає результат виконання програми.

Трансляцiя передбачає фази лексичного, синтаксичного та семантичного аналiзу, а також фазу генерацiї коду. Фази лексичного та синтаксичного аналiзу здiйснюються окремими проходами.

**Нотацiя**

|  |  |
| --- | --- |
| Метасимвол | Значення |
| = | визначається як |
| | | альтернатива |
| [ x ] | 0 або 1 екземпляр x |
| { x } | 0 або бiльше екземплярiв x |
| ( x | y ) | групування: будь - який з x або y |
| Zxy | нетермiнал |
| zxy | термiнал |
| ’1’ | термiнал |
| ”1” | термiнал |

Табл. 1: Прийнята нотацiя РБНФ

Для опису мови FuriousPeach використовується розширена форма Бекуса – Наура. Ланцюжки, що починаються з великої лiтери вважаються нетермiналами (нетермiнальними символами). Термiнали (термiнальнi символи) - ланцюжки, що починаються з маленької лiтери, або ланцюжки,

що знаходяться мiж одинарними або подвiйними лапками. Для графiчного представлення граматики використовуються синтаксичнi дiграми Вiрта.

**Алфавiт**

Програма може мiстити текст з використанням таких символiв (character) - лiтер, цифр, спецiальних знакiв та ознаки кiнця файлу.

Синтаксис

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’

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

SpecSsign = ’.’ | ’,’ | ’:’ | ’;’ | ’(’ | ’)’ | ’=’ | ’+’|’-’ | ’\*’ | ’/’ | ’<’ | ’^’ | ’>’ | ’==’ | ’^’ | ’{’ | ’}’ | WhiteSpace | EndOfLine

WhiteSpace = ’ ’ | ’\t’

EndOfLine = ’\n’ | ’\r’ | ’\r\n’ | ’\n\r’

EndOfFile = ’\0’

**2. Лексика**

Лексичний аналiз виконується окремим проходом, отже не залежить вiд син таксичного розбору та семантичного аналiзу. Лексичний аналiзатор розбиває текст програми на лексеми. У програмi мовою *FuriousPeach* можуть використовуватись лексичнi елементи, що класифiкуються як спецiальнi символи, iдентифiкатори, беззнаковi цiлi константи, беззнаковi дiйснi константи, логiчнi константи та ключовi слова.

**Спецiальнi символи**

Синтаксис

1. SpecSymbols = Arithm | Add | Rel | Bracket | Assign | Punct Arithm = Add | Mult

Add = ’+’ | ’-’

Mult = ’\*’ | ’/’ | div

Pow = ’^’

Rel = ’=’ | ’<= ’ | ’== ’ | ’<’ | ’>’ | ’>=’ | ’<>’ Bracket = ’(’ | ’)’

Assign = ’=’

Punct=’.’|’,’|’:’|’;’|’{’|’}’

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

Обмеження

3. Набiр токенiв, див. табл. 2, мiстить токени add\_op та mult\_op, але не мiстить arith\_op.

**Iдентифiкатори**

Синтаксис

1. Ident = Letter {Letter | Digit }

Опис

2. Першим символом iдентифiкатора може бути тiльки лiтера, наступнi сим воли, якщо вони є, можуть бути цифрами або лiтерами. Довжина iденти фiкатора не обмежена.

3. Iдентифiкатор може позначати змiнну та програму.

Обмеження

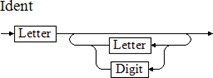
4. Жоден iдентифiкатор не може збiгатись iз ключовим (вбудованим, заре зервованим) словом або словом true, або false.

Семантика

5. Елемент, який у фазi лексичного аналiзу може бути визначений як iден тифiкатор або як ключове слово, вважається ключовим словом.

6. Елемент, який у фазi лексичного аналiзу може бути визначений як iден тифiкатор або як логiчна константа, вважається логiчною константою.

Вiзуальне представлення



6. Синтаксична дiаграма

Приклади

7. a, x1, time24

**Константи**

Синтаксис

1. Const = IntNumb | RealNumb | BoolConst

IntNumb = [Sign] UnsignedInt

RealNumb = [Sign] UnsignedReal

Sign = ’+’ | ’-’

UnsignedInt = Digit {Digit}

UnsignedReal = ’.’

UnsignedInt | UnsignedInt

’.’ | UnsignedInt ’.’

UnsignedInt

BoolConst = true | false

Обмеження

2. Кожна константа повинна мати тип, а величина константи повинна зна ходитись у дiапазонi репрезентативних значень для її типу.

3. На етапi лексичного аналiзу виявляються тiльки беззнаковi цiлi констан ти UnsignedInt, беззнаковi дiйснi константи UnsignedReal та логiчнi константи BoolConst.

Семантика

4. Кожна константа має тип, визначений її формою.

Вiзуальне представлення

5. Синтаксичнi дiаграми див. на рис. 1.

Приклади

6. 11, 134, 1.84, 84.567, 23. , true, false

**Ключовi слова**

Синтаксис

1. KeyWords = program | int | real | bool | in | out | for | if | else | then | goto |

**Токени**

З потоку символiв вхiдної програми на етапi лексичного аналiзу виокремлю ються послiдовностi символiв з певним сукупним значенням, - токени. Список

токенiв мови FuriousPeach див. у табл. 2. Єдиний приклад засвiдчує унiкальнiсть лексеми. Текст "символ . . . ", наприклад "символ program", означає, що вказаний символ (symbol) є термiналом.

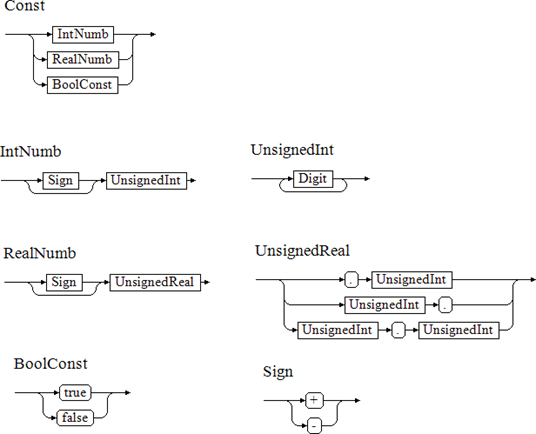


Рис. 1: Константи

|  |  |  |  |
| --- | --- | --- | --- |
| Код | Приклади лексем | Токен | Неформальний опис |
| 1 | a, x1, z12f | id | iдентифiкатор |
| 2 | 123, 0, 521 | intnum | цiле без знаку |
| 3 | .012, 34.76, 876. | realnum | дiйсне без знаку |
| 4 | true, false | boolval | логiчне значення |
| 5 | program | keyword | символ program |
| 6 | endl | keyword | символ endl |
| 7 | int | keyword | символ int |
| 8 | real | keyword | символ real |
| 9 | bool | keyword | символ bool |
| 10 | input | keyword | символ input |
| 11 | print | keyword | символ print |
| 12 | for | keyword | символ for |
| 13 | if | keyword | символ if |
| 14 | goto | keyword | символ goto |
| 15 | then | keyword | символ then |
| 16 | else | keyword | символ else |
| 17 | = | assign\_op | символ = |
| 18 | + | add\_op | символ + |
| 19 | - | add\_op | символ - |
| 20 | \* | mult\_op | символ \* |
| 21 | / | mult\_op | символ / |

|  |  |  |  |
| --- | --- | --- | --- |
| 22 | div | mult\_op | символ div |
| 23 | < | rel\_op | символ < |
| 24 | <= | rel\_op | символ <= |
| 25 | == | rel\_op | символ == |
| 26 | >= | rel\_op | символ >= |
| 27 | <> | rel\_op | символ <> |
| 28 | ^ | pow\_op | символ ^ |
| 29 | ( | brackets\_op | символ <> |
| 30 | ) | brackets\_op | символ <> |
| 31 | . | punct | символ . |
| 32 | , | punct | символ , |
| 33 | : | punct | символ : |
| 34 | ; | punct | символ ; |
| 35 | { | punct | символ { |
| 36 | } | punct | символ } |
| 37 | \32,\t | ws | пробiльнi символи |
| 38 | \n, \r\n | eol | кiнець рядка |
| 39 | \0 | eof | кiнець файлу |

Табл. 2: Таблиця лексем мови FuriousPeach

**3. Типи**

Мова FuriousPeachпiдтримує значення трьох типiв: int, real та bool.

1. Цiлий тип integer може бути представлений оголошеною змiнною типу int, або константою IntNumb. Дiапазон значень - [ -32768; 32767 ].

2. Дiйсний тип real може бути представлений оголошеною змiнною типу real, або константою RealNumb. Дiапазон значень - [3.4x10-38 ; 3.4x1038 ].

3. Логiчний тип boolean може бути представлений оголошеною змiнною типу bool, або константою BoolConst (true або false). Прийнято false < true

**4. Синтаксис**

**Вирази**

Синтаксис

1. Expression = ArithmExpression | BoolExpr

BoolExpr = ArithmExpression Rel ArithmExpression

ArithmExpression = [ Sign] Term

| ArithmExpression ’+’ Term

7

| ArithmExpression ’-’ Term

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

| | Term ’^’ Factor

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

Опис

2. Вираз - це послiдовнiсть операторiв i операндiв, що визначає порядок об числення значення.

3. Розрiзняються арифметичнi та логiчнi вирази.

4. Значення,обчислене за арифметичнимвиразом,має тип real або int. 5. Значення, обчислене за логiчним виразом, має тип bool.

6. Всi бiнарнi оператори у виразах цiєї мови лiвоасоцiативнi.

7. Найвищий прiоритет в унарного мiнуса та унарного плюса, далi, у порядку зменшення прiоритету слiдують Mult, Add та Rel.

8. Послiдовнiсть двох або бiльше операторiв з однаковим прiоритетом асоцiативна.

9. Тип кожної змiнної має бути визначений у роздiлi оголошень.

10. Повторне оголошеної змiнної викликає помилку на етапi (у фазi) транс ляцiї.

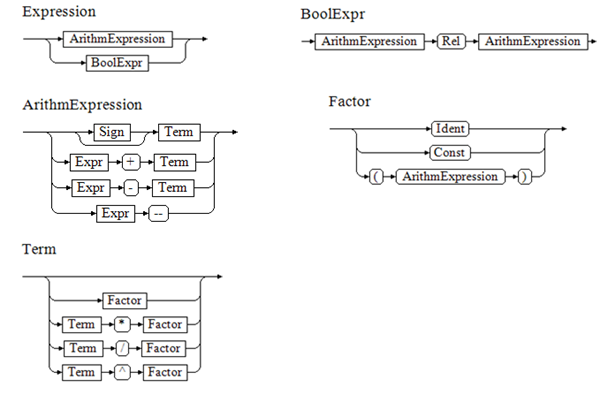
11. Використання неоголошеної змiнної, викликає помилку на етапi трансля цiї.

12. Використання змiнної, що не набула значення, викликає помилку. Семантика

13. Кожна константа має тип, визначений її формою та значенням.

14. Змiнна набуває значення в iнструкцiї присвоювання Assign або в iнстру кцiї введення Input.

Вiзуальне представлення

15. Синтаксична дiаграма.

Приклади

16. Factor:

x, 12, (a + 234)

17. Term:

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

18. ArithmExpr:

-b, f1 + g, c - 24

19. BoolExpr:

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

**Оператори**

**Арифметичнi оператори**

Синтаксис

1. Add = ’+’ | ’-’

Mult = ’\*’ | ’/’ | div | ’^’

Обмеження

2. Оператор цiлочислового дiлення div застосовується тiльки до операндiв типу integer.

Семантика

3. Тип результату при застосуваннi бiнарних операторiв див. табл. 3. 4. Тип результату при застосуваннi унарних операторiв див. табл. 4. 5. Дiлення на нуль викликає помилку.

Приклад

6. 1.234\*x1/45.67, -3.4+6, 7/8, 12 div 5, 2^2

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Операцiя | Типи операндiв | Тип результату |
| + | додавання | int або real | real,  якщо хоч один real, iнакше int |
| - | вiднiмання | int або real |
| \* | множення | int або real |
| / | дiлення | int або real | real |
| div | цiлочисельне дiлення | int | int |
| ^ | Піднесення до  степені | int | int |

Табл. 3: Бiнарнi арифметичнi оператори

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор | Операцiя | Типи операнда | Тип результату |
| + | iдентичностi | int  real | int  real |
| - | змiни знаку | int  real | int  real |

Табл. 4: Унарнi арифметичнi оператори

**Оператори вiдношення**

Синтаксис

1. Rel = ’==’ | ’<=’ | ’<’ | ’>’ | ’>=’ | ’<>’

Обмеження

2. Значення обох операндiв мають бути або числовими (типу int або real), або логiчними (типу bool).

3. Результат завжди має тип bool.

Семантика

4. Якщо один з операндiв має тип int, а iнший - real, то значення типу int приводиться до типу real.

Приклад

5. x1+3 < 1, 0.5\*2.34 <= 15 div 7, (z-3>0) <> (2>=m), true > false

**5. Оголошення**

Синтаксис

1. DeclarList = Declaration { Declaration }

Declaration = Type ’ ’ IdenttList

IdenttList = Ident {’,’ Ident}

Type = int | real | bool

Опис

2. Оголошення (декларацiя) специфiкує набiр iдентифiкаторiв, якi можуть бути використанi у програмi.

3. Оголошення iдентифiкатор означає оголошення змiнної.

Обмеження

4. Роздiл оголошень знаходиться перед роздiлом iнструкцiй.

5. Кожен iдентифiкатор має бути оголошений i тiльки один раз. Семантика

6. Оголошення змiнної означає видiлення пам’ятi для зберiгання значення декларованого типу.

7. Значення оголошеноїзмiнноїзалишаєтьсяневизначенимаждоприсвоєння їй значення у iнструкцiї присвоєння або введення.

8. Область видимостi змiнної (scope) - вся програма.

Вiзуальне представлення

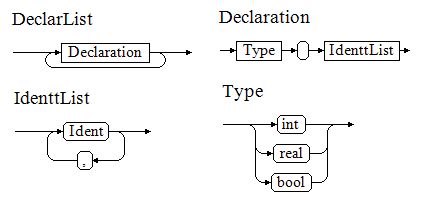


Рис. 2: Роздiл оголошень

9. Синтаксичнi дiаграми див. на рис. 2.

Приклад

real x1, z, a

bool c7, b

int k, m, r, w

**6. Роздiл iнструкцiй**

Синтаксис

1. DoSection = StatementList

StatementList = Statement { Statement }

Statement = Assign | Input | Print | ForStatement

Опис

2. Iнструкцiї (statements) визначають алгоритмiчнi дiї, якi мають бути вико нанi у програмi.

Обмеження

3. У роздiлi iнструкцiй має бути не менше однiєї iнструкцiї.

Семантика

5. Iнструкцiї виконуюються послiдовно у порядку запису (оскiльки мова FuriousPeachне мiстить нi умовних операторiв, нi операторiв переходу).

Вiзуальне представлення

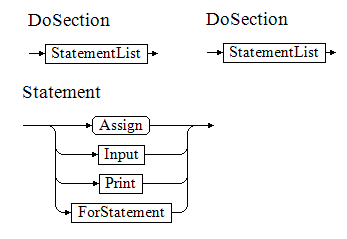


Рис. 3: Роздiл iнструкцiй

6. Синтаксичнi дiаграми див. на рис. 3.

Приклад

7. print(z,a,b)

x = (z + a)/( z+b )

**Оператор (iнструкцiя) присвоювання**

Синтаксис

1. Assign = Ident ’=’ Expression

Опис

2. 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

3. Тип змiнної з iдентифiкатором Ident повинен збiгатись з типом r-значення.

Семантика

4. l-значення має тип вказiвника на мiсце зберiгання значення змiнної з iдентифiкатором Ident.

5. r-значення має тип значення, обчисленого за виразом Expression. Вiзуальне представлення



Рис. 4: Оператор присвоювання

5. Синтаксичну дiаграму див. на рис. 4.

Приклад

6. f = 3/4+1.23, b7 = 2+3 < 7 div 5

**Iнструкцiя введення**

Синтаксис

1. Input = input ’(’ IdenttList ’)’.

Опис

2. Значення вводяться з клавiатури.

3. Введення кожного окремого значення пiдтверджується клавiшею Enter. Обмеження

4. Вiдмiннiсть типу введеного значення вiд типу змiнної викликає помилку. Вiзуальне представлення

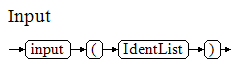


Рис. 5: Iнструкцiя введення

5. Синтаксичну дiаграму див. на рис. 5.

Приклад

6. input(a1,v2,len7)

**Iнструкцiя виведення**

Синтаксис

1. Print = print ’(’ IdenttList ’)’.

Опис

2. Всi значення списку виводяться у один рядок консолi.

3. Кожне значення виводиться у форматi \tIdent=значення. Обмеження

4. Виведення змiнної з невизначеним значенням викликає помилку. Вiзуальне представлення

5. Синтаксичну дiаграму див. на рис. 6.

Приклад

6. print(c,x1,f5)

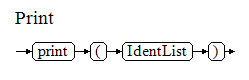


Рис. 6: Iнструкцiя виведення

**Оператор циклу (iнструкцiя повторення)**

Синтаксис

1. ForStatement = for ’(’ IndExpr ’)’ ’{’ DoBlock ’}’ IndExpr = Ident ’=’ ArithmExpression1 ’;’ BoolExpr ’;’ ArithmExpression2

ArithmExpression1 = ArithmExpression

ArithmExpression2 = Ident Add Step

DoBlock = Statement | StatementList

Опис

2. Тiло оператора циклу DoBlock виконується один або бiльше разiв. Обмеження

3. Параметр циклу Ident - змiнна типу int.

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

Семантика

5. Перед першою iтерацiєю обчислюється значення ArithmExpression1 та BoolExpr.

6. Якщо значення BoolExpr == true, то обчислюється значення

ArithmExpression2. Якщо значення BoolExpr == false, то цикл закінчує роботу.

7. Перед першою iтерацiєю параметр циклу Ident приймає значення ArithmExpression1.

8. Пiсля першої та наступних iтерацiй параметр циклу Ident приймає зна чення Ident = Ident Add Step.

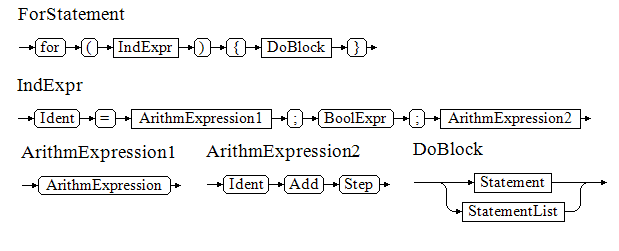
9. DoBlock виконується , якщо значення

Ident ∈ [ArithmExpression1; IntNumb] .

**Вiзуальне представлення**

10. Синтаксичнi дiаграми див. на рис.

7. Приклад

Рис. 7: Оператор циклу

11. for ( i = 0 ; i < 10; i=i+1)

{print(i)}

12. for ( i = 11; i>0; i=i-1){

z=11-i

print(z)

}

**Умовний оператор**

Синтаксис

11. IfStatement = if BoolExpr ’then ’ ’goto’ Ident1 [ ’else’ ’goto’ Ident2]

Ident1 = Letter {Letter | Digit}

Ident2 = Letter {Letter | Digit}

Опис

12. Тiло оператора DoBlock1 та DoBlock2 виконується один раз.

Обмеження

13. Параметр BoolExpr - змiнна типуbool або приводиться до типу bool.

Семантика

14. Передпершоюiтерацiєюобчислюються значенняBoolExpr.

15. Якщо значення BoolExpr == true, то виконується інструкція then ≤

goto. Якщо значення BoolExpr == false, то виконується інструкція else .

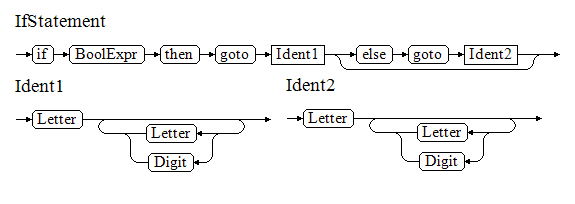
16. DoBlock1 виконується , якщо значення BoolExpr∈ true.

17. DoBlock2 виконується , якщо значення BoolExpr∈ false.

Вiзуальне представлення

18. Синтаксичнi дiаграми див. на рис. 7.

Приклад

Рис. 7: Умовний оператор

int i = 0

input(i)

if i > 1 then goto

print(i+1)

else goto

print(i)

**7 Програма**

Синтаксис

1. Program = program ProgName

DeclSection

DoSection

ProgName = Ident

Опис

2. Кожна програма починається з термiнала program та iдентифiкатора про грами; за iдентифiкатором програми - (єдиний) роздiл оголошень; далi - (єдиний) роздiл iнструкцiй.

3. Iдентифiкатор програми нiяк не використовується у програмi. Вiзуальне представлення

3. Синтаксичну дiаграму див. на рис. 8.

Приклад



Рис. 8: Програма

6. program counter

int a

real b

input(a)

for (int i = 1; i<a; i=i+1):

input(b)

if b > 0 then goto

print(i,b)

**8. Повна граматика мови FuriousPeach**

Program = program ProgName DeclSection DoSection ProgName = Ident

Ident = Letter {Letter | Digit }

DeclSection = DeclarList

DeclarList = Declaration { Declaration } Declaration = Type ’ ’ IdenttList

IdenttList = Ident {’,’ Ident}

Type = int | real | bool

DoSection = StatementList

StatementList = Statement {’;’ Statement } Statement = Assign | Input | Print | ForStatement Assign = Ident ’=’ (Expression | BoolExpr) Expression = ArithmExpression | BoolExpr BoolExpr = ArithmExpression Rel ArithmExpression ArithmExpression = [Sign] Term

| ArithmExpression ’+’ Term

| ArithmExpression ’-’ Term

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

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

Input = input ’(’ IdenttList ’)’

Print = print ’(’ IdenttList ’)’

ForStatement = for ’(’ IndExpr ’)’ ’{’

DoBlock ’}’

IndExpr = Ident ’=’ ArithmExpression1 ’;’

BoolExpr ’;’ ArithmExpression2

ArithmExpression1 = ArithmExpression

ArithmExpression2 = Ident Add Step

DoBlock = Statement | StatementList

IfStatement = if BoolExpr ’then ’ ’goto’ Ident1 | ’else’ ’goto’ Ident2

Ident1 = Letter {Letter | Digit }

Ident2 = Letter {Letter | Digit }

Const = IntNumb | RealNumb | BoolConst

IntNumb = [Sign] UnsignedInt

RealNumb = [Sign] UnsignedReal

Sign = ’+’ | ’-’

UnsignedInt = Digit {Digit}

UnsignedReal = ’.’ UnsignedInt

| UnsignedInt ’.’

| UnsignedInt ’.’ UnsignedInt

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’

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

Rel = ’==’ | ’<= ’ | ’<’ | ’>’ | ’>=’ | ’<>’