Специфікація мови програмування MP2

1 Вступ

Представлена тут мова програмування MP2 призначена слугувати навчальним прикладом для курсу, що має на меті вивчення основ трансляції. MP2 — імперативна мова загального призначення.

1.1 Завдання

Розробити Pascal —подібну мову програмування з оператором (інструкцією) циклу:

```
for <Ідентифікатор> :=  <Вираз> to <Вираз> do <Блок інструкцій>
```

1.2 Обробка

Програма, написана мовою MP2, подається на вхід транслятора (компілятора або інтерпретатора) для трансформації до цільової форми. 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

Для опису мови MP2 використовується розширена форма Бекуса – Наура. Ланцюжки, що починаються з великої літери вважаються нетерміналами (нетермінальними символами). Термінали — ланцюжки, що починаються з маленької літери, або знаходяться між одинарними, або подвійними лапками. Для

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

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:

```
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 = \u0000
```

2 Лексика

Лексичний аналіз виконується окремим проходом, отже не залежить від синтаксичного розбору та семантичного аналізу. Лексичний аналізатор розбиває вихідний текст на лексеми. У програмі мовою MP2 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

Синтаксис

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

   ```
   ArithOp =  AddOp | MultOp
   AddOp = '+' | '-'
   MultOp =  '*' | '/'
   RelOp = '=' | '<= ' | '<' | '>' | '>=' | '<>'
   BracketsOp = '(' | ')'
   AssignOp = ':='
   Punct =  '.' | ',' | ':' | ';'
   ```

Опис

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

Обмеження

3. Набір токенів див. табл. **??**.

2.1.1 Ідентифікатори

Синтаксис

1. `Ident = Letter {Letter | Digit }`

Опис

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

Обмеження

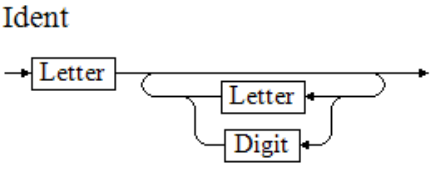
3. Жоден ідентифікатор не може збігатись із ключовим (вбудованим, зарезервованим) словом, або словом `true` або `false`.

Семантика

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

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

Візуальне представлення



6. Синтаксична діаграма

Приклади

7. `a, x1, time24`

2.2 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

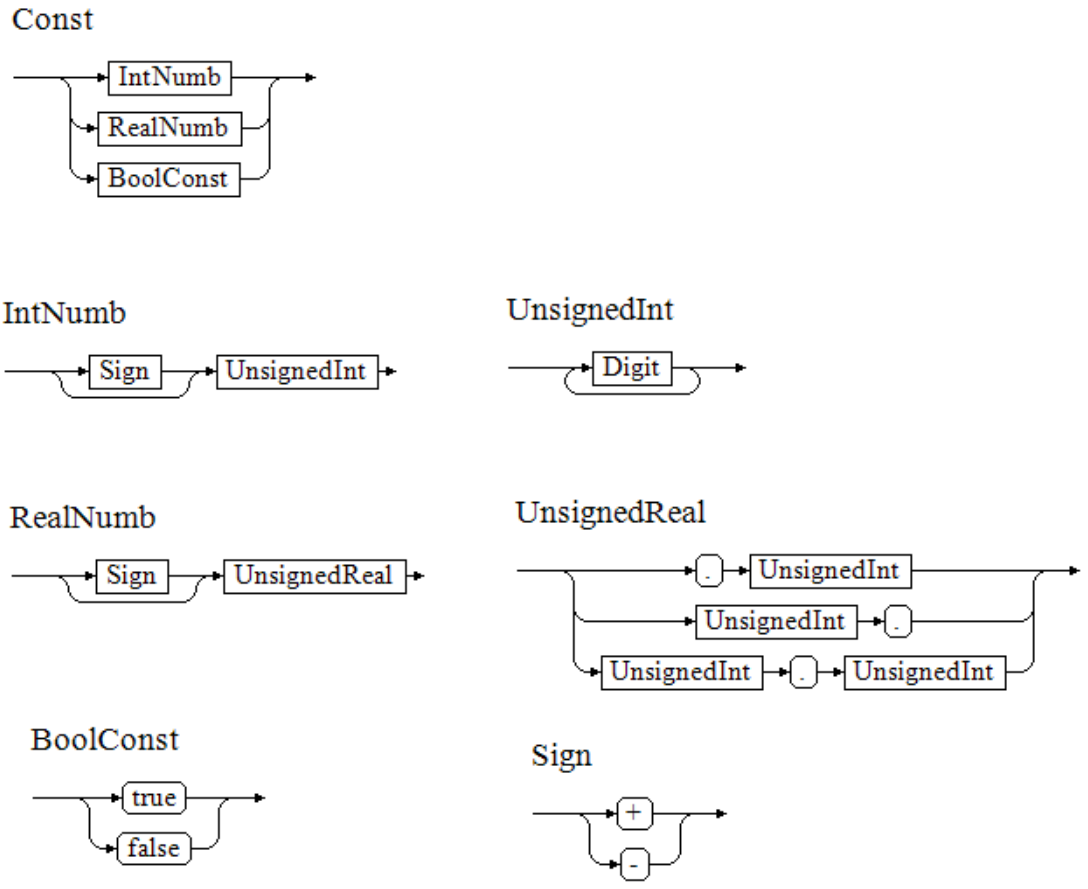
2. Кожна константа повинна мати тип, а величина константи повинна знаходитись у діапазоні репрезентативних значень для її типу.

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

Семантика

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

Візуальне представлення



5. Синтаксична діаграма Константи.

Приклади

6. `12, 234, 1.54, 34.567, 23. , true, false`

2.3 Ключові слова

Синтаксис

1. `KeyWords = program | var | begin | end | end. | integer | real | boolean | read | write | for | to | do`

Код	Лексема	Токен	Приклад	Неформальний опис
1		id	a, x1, z12f	ідентифікатор
2		intnum	123, 0, 521	ціле без знаку
3		realnum	.012, 34.76, 876.	дійсне без знаку
4		boolval	true, false	логічне значення
5	program	keyword	program	термінал program
6	var	keyword	var	термінал var
7	begin	keyword	begin	термінал begin
8	end	keyword	end	термінал end
9	end.	keyword	end.	термінал end.
10	integer	keyword	integer	термінал integer
11	real	keyword	real	термінал real
12	boolean	keyword	boolean	термінал boolean
13	read	keyword	read	термінал read
14	write	keyword	write	термінал write
15	for	keyword	for	термінал for
16	to	keyword	to	термінал to
17	do	keyword	do	термінал do
18	:=	assign_op	:=	термінал :=
19	+	add_op	+	термінал +
20	-	add_op	-	термінал -
21	*	mult_op	*	термінал *
22	/	mult_op	/	термінал /
23	div	mult_op	div	термінал div
24	<	rel_op	<	термінал <
25	<=	rel_op	<=	термінал <=
26	=	rel_op	=	термінал =
27	>=	rel_op	>=	термінал >=
28	<>	rel_op	<>	термінал <>
29	(	brackets_op	(	термінал <>
30	)	brackets_op	)	термінал <>
31	.	punct	.	термінал .
32	,	punct	,	термінал ,
33	:	punct	:	термінал :
34	;	punct	;	термінал ;
35		white	␣,\t	
36		eol	\n, \r\n	
37		eof	\u0000	

Табл. 2: Таблиця лексем мови MP2

2.4 Токени

З потоку символів вхідної програми на етапі лексичного аналізу виокремлюються послідовності символів з певним сукупним значенням, — *токени*. Список токенів мови MP2 див. у табл. **??**.

3 Типи

Мова MP2 обробляє значення трьох типів: `integer`, `real` та `boolean`.

1. Цілий тип `integer` може бути представлений оголошеною змінною типу `integer`, або константою `IntNumb`. Діапазон значень залежить від реалізації.

2. Дійсний тип `real` може бути представлений оголошеною змінною типу `real`, або константою `RealNumb`.

3. Логічний тип `boolean` може бути представлений оголошеною змінною типу `boolean`, або константою `BoolConst` (`true` або `false`). Прийнято `false < true`

4 Синтаксис

4.1 Вирази

Синтаксис

1. ```
   Expression =   ArithmExpression | BoolExpr
   BoolExpr = ArithmExpression RelOp ArithmExpression
   ArithmExpression = [ Sign] Term
                      | ArithmExpression '+' Term
                      | ArithmExpression '-' Term
   Term = Factor | Term '*' Factor | Term '/' Factor
   Factor = Ident | Const | '(' ArithmExpression ')'
   ```

Опис

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

3. Розрізняються арифметичні та логічні вирази.

4. Значення, обчислене за арифметичним виразом, має тип `real` або `integer`.

5. Значення, обчислене за логічним виразом, має тип `boolean`.

6. Всі бінарні оператори у виразах є лівоасоціативними.

7. Найвищий пріоритет в унарного мінуса та унарного плюса, далі, у порядку зменшення пріоритету слідують `MultOp`, `AddOp` та `RelOp`.
   ```

8. Послідовність двох або більше операторів з однаковим пріоритетом асоціативна.

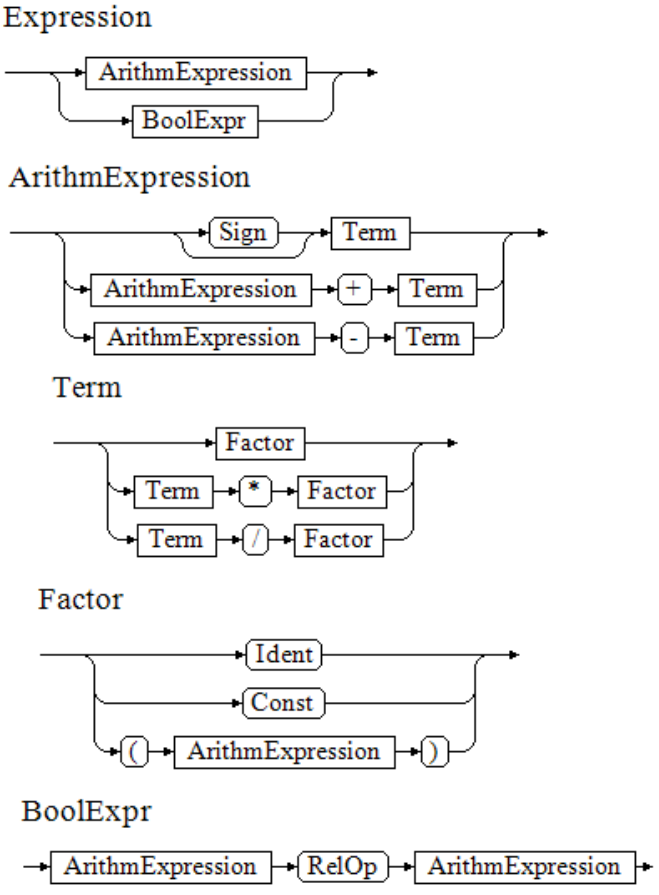
Обмеження

9. Використання змінної, з не визначеним на момент обчислення виразу значенням, викликає помилку.

Семантика

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

Візуальне представлення



11. Синтаксична діаграма.

Приклади

12. Factor:

```
x, 12, (a + 234)
```

13. Term:

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

14. ArithmExpr:

```
-b, f1 + g, c - 24
```

15. BoolExpr:

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

4.2 Оператори

Опис залежності типу результату від типу операндів

4.2.1 Арифметичні оператори

Бінарні
Унарні

4.2.2 Оператори відношення

5 Програма

Кожна програма починається з термінала `program` та ідентифікатора програми. Ідентифікатор ніяк не використовується у програмі. Далі, після термінала `var` розміщується список декларацій, де вказуються ідентифікатори змінних та призначені їм типирис. ??. Програма не може містити неоголошену змінну.

```
Program = program ProgName
          var DeclarList
          begin  StatementList  'end.'
ProgName = Ident
```

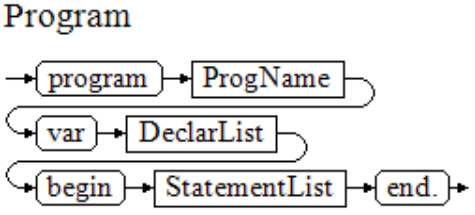


Рис. 1: Програма

Список декларацій `DeclarList` — це одна, або більше декларацій. Розділь-ником декларацій є крапка з комою (`;`); оголошення (декларація) — це список змінних, з роздільником комою (`,`), наступним символом двокрапка `:` та типом `Type`, див. рис. ??.

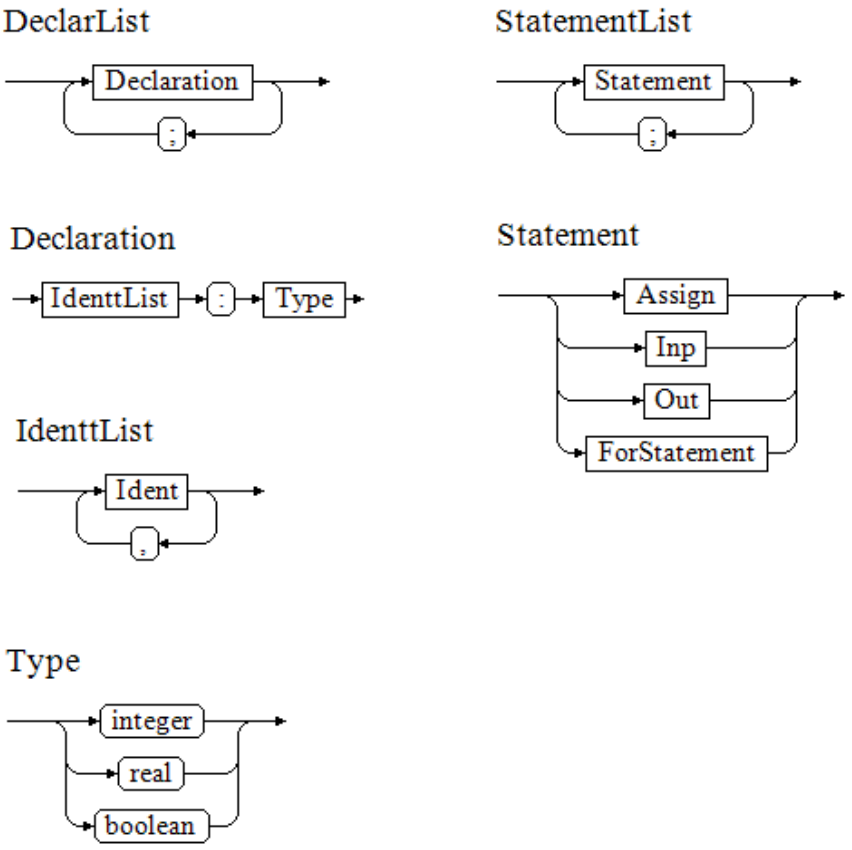


Рис. 2: Списки оголошень та інструкцій

Решта програми - це список інструкцій `StatementList` між терміналами `begin` та `end.` (з крапкою). Список інструкцій `StatementList` — одна чи більше інструкцій з роздільником `;` (крапка з комою). введення, виведення, або повторення, рис. **??**.

6 Оголощення

Оголошення (декларації) специфікує інтерпретацію та атрибути набору ідентифікаторів.
Синтаксис

1. ...

Обмеження

2. ...

3. ...

Семантика

4. ...

5. ...

Приклад

6. ...

7 Інструкції

Інструкції (Statements) визначають алгоритмічні дії, які мають бути виконані. За винятком зазначених далі випадків, інструкції виконуються послідовно.

7.1 Оператор (інструкція) присвоювання

Синтаксис

1. `Assign = Ident ':=' Expression`

Обмеження

2. Тип змінної з ідентифікатором `Ident` має відповідати типу виразу праворуч оператора `:=`.

3. ...

Семантика

4. ...

5. ...

Приклад

6. ...

7.2 Інструкція введення

Синтаксис

1. ...

Обмеження

2. ...

3. ...

Семантика

4. ...

5. ...

Приклад

6. ...

7.3 Інструкція виведення

Синтаксис

1. ...

Обмеження

2. ...

3. ...

Семантика

4. ...

5. ...

Приклад

6. ...

7.4 Оператор циклу

Синтаксис

1. ...

Обмеження

2. ...

3. ...

Семантика

4. ...

5. ...

Приклад

6. ...

8 Повна граматика мови MP2

```
Program = program ProgName
          var DeclarList
          begin  StatementList  'end.'
ProgName = Ident
Ident = Letter {Letter | Digit }
DeclarList = Declaration {';' Declaration }
Declaration = IdenttList ':' Type
IdenttList = Ident {',' Ident}
Type = integer | real | boolean
StatementList = Statement {';' Statement }
Statement = Assign | Inp | Out | ForStatement
Assign = Ident ':=' Expression
Expression =   ArithmExpression | BoolExpr
BoolExpr = ArithmExpression RelOp ArithmExpression
ArithmExpression = Term
                  | ArithmExpression '+' Term
                  | ArithmExpression '-' Term
Term = Factor | Term '*' Factor | Term '/' Factor
Factor = Ident | Const | '(' ArithmExpression ')'
Inp = read '(' IdenttList ')'
Out = write '(' IdenttList ')'
ForStatement = for IndExpr do DoBlock
IndExpr = Ident ':='  ArithmExpression to ArithmExpression
DoBlock = Statement | 'begin' StatementList 'end'
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
RelOp = '=' | '<= ' | '<' | '>' | '>=' | '<>'
```