Специфiкацiя мови програмування Crundras

# Вступ

Представлена тут мова програмування Crundras призначена слугувати піддослідним кроликом для курсу, що має на меті вивчення основ трансляції. Crundras — езотерична мова загального призначення.

## Завдання

Розробити Сі-подiбну мову програмування з оператором (інструкцією) циклу:

for <ід>=<вираз> to <вираз> by <вираз> while (<відношення>)  
<список операторів> rof;

та умовною інструкцією:

if (<відношення>) {<сп. операторів>}

## Обробка

Програма, написана мовою Crundras, подається на вхід транслятора (компілятора або інтерпретатора) для трансформації до цільової форми. Результат трансляції виконується у система 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.

## Нотацiя

Для опису мови Crundras використовується розширена форма Бекуса – Наура. Ланцюжки, що починаються з великої літери вважаються нетермiналами (нетермiнальними 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 бiльше екземплярiв x |
| ( x | y ) | групування: будь -який з x або y |
| Zxy | нетермiнал |
| zxy | термiнал |
| ’1’ | термiнал |
| ”1” | термiнал |

Табл. 1: Прийнята нотація РБНФ

графічного представлення граматики використовуються синтаксичні діаграми Вiрта.

## 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'.

# 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 Crundras можуть використовуватись лексичні елементи, що класифікуються як спеціальні символи, ідентифікатори, беззнакові цілі константи, беззнакові дійсні константи, логічні константи та ключові слова.

## Спецiальнi символи

Синтаксис

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

ArithOp = AddOp | MultOp

AddOp = ’+’ | ’-’

MultOp = ’\*’ | ’/’

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

BracketsOp = ’(’ | ’)’

AssignOp = ’=’

Punct = ’.’ | ’,’ | ’:’ | ’;’

Опис

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

Обмеження

1. Набір токенiв див. табл. ??.

### Ідентифікатори

Синтаксис

1. Ident = Letter {Letter | Digit}.

Опис

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

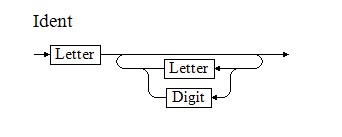
Обмеження

1. Жоден ідентифікатор не може збігатись із ключовим (вбудованим, зарезервованим) словом.

Семантика

1. Елемент, який у фазі лексичного аналізу може бути визначений як ідентифікатор, або як ключове слово, вважається ключовим словом.
2. Елемент, який у фазі лексичного аналізу може бути визначений як ідентифікатор, або як логічна константа, вважається логічною константою.

Візуальне представлення



1. Синтаксична діаграма

Приклади

1. a, x1, time24, as4me17

## Літерали

Синтаксис

Literal = IntegerLiteral | FloatingLiteral | BooleanLiteral.

IntegerLiteral = Digit {Digit}.

FloatingLiteral = IntegerLiteral '.' [IntegerLiteral] | '.'[IntegerLiteral].

BooleanLiteral = true | false.

Обмеження

1. Кожен літерал повинен мати тип, а величина літералу повинна знаходитись у діапазоні репрезентативних значень для її типу.

Семантика

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

Вiзуальне представлення

TODO

1. Синтаксична дiаграма Константи.

Приклади

1. 12, 234, 1.54, 34.567, 23. , .42, true, false

## Ключовi слова

Синтаксис

1. KeyWords = int | float| bool | read |print | for | to | do

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

Табл. 2: Таблиця лексем мови MP2

## Токени

З потоку символiв вхiдної програми на етапi лексичного аналiзу виокремлюються послiдовностi символiв з певним сукупним значенням, — *токени*. Список токенiв мови MP2 див. у табл. ??.

# Типи

Мова Crundras обробляє значення трьох типів: int, float та bool.

1. Цілий тип int може бути представлений оголошеною змінною типу int, або константою IntegerLiteral. Діапазон значень залежить від реалізації.
2. Дійсний тип float може бути представлений оголошеною змінною типу float, або константою FloatingLiteral.
3. Логічний тип bool може бути представлений оголошеною змінною типу bool, або константою BooleanLiteral (true або false). Прийнято false = 0, true != false

# Синтаксис

## Declaration

Оголошення (декларацiї) специфiкує iнтерпретацiю та атрибути набору iдентифiкаторiв.

Синтаксис

1. DeclarationStatement = TypeSpecifier DeclarationList ';'.

DeclarationList = Identifier {',' Identifier}.

Приклад

int a, b; bool d;

## Вирази

Синтаксис

Опис

1. Вираз - це послiдовнiсть операторiв i операндiв, що визначає порядок обчислення значення.
2. Розрiзняються арифметичнi та логiчнi вирази.
3. Значення, обчислене за арифметичним виразом, має тип real або integer.
4. Значення, обчислене за логiчним виразом, має тип boolean.
5. Всi бiнарнi оператори у виразах є лiвоасоцiативними.
6. Найвищий прiоритет в унарного мiнуса та унарного плюса, далi, у порядкузменшення прiоритету слiдують MultOp, AddOp та RelOp.
7. Послiдовнiсть двох або бiльше операторiв з однаковим прiоритетом асоцiативна.

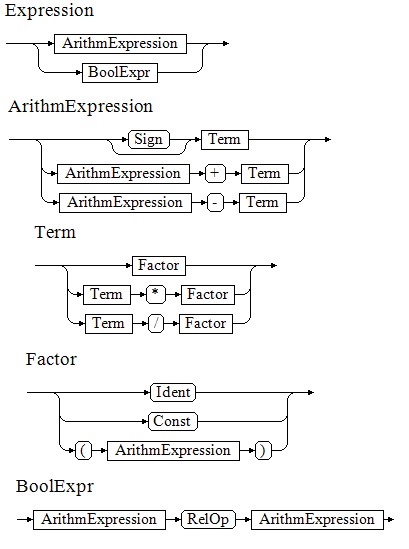
Обмеження

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

Семантика

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

Вiзуальне представлення



1. Синтаксична дiаграма.

Приклади

1. Factor:

x, 12, (a + 234)

1. Term:

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

1. ArithmExpr:

-b, f1 + g, c - 24

1. BoolExpr:

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

## Оператори

Опис залежностi типу результату вiд типу операндiв

### Арифметичнi оператори

Бiнарнi

Унарнi

4.2.2 Оператори вiдношення

# Програма

Кожна програма починається з термiнала program та iдентифiкатора програми. Iдентифiкатор нiяк не використовується у програмi. Далi, пiсля термiнала var розмiщується список декларацiй, де вказуються iдентифiкатори змiнних та призначенi їм типирис. ??. Програма не може мiстити неоголошену змiнну.

Program = program ProgName

var DeclarList begin StatementList ’end.’

ProgName = Ident

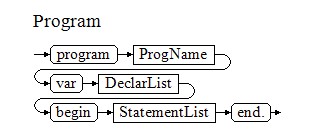


Рис. 1: Програма

Список декларацiй DeclarList — це одна, або бiльше декларацiй. Роздiльником декларацiй є крапка з комою (;); оголошення (декларацiя) — це список змiнних, з роздiльником комою (,), наступним символом двокрапка : та типом Type, див. рис. ??.

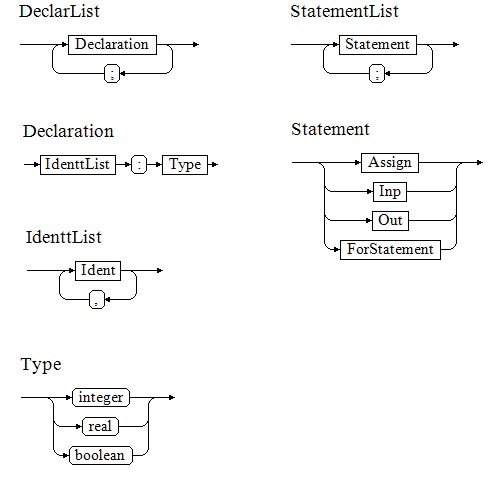


Рис. 2: Списки оголошень та iнструкцiй

Решта програми - це список iнструкцiй StatementList мiж термiналами begin та end. (з крапкою). Список iнструкцiй StatementList — одна чи бiльше iнструкцiй з роздiльником ; (крапка з комою). введення, виведення, або повторення, рис. ??.

# Iнструкцiї

Iнструкцiї (Statements) визначають алгоритмiчнi дiї, якi мають бути виконанi. За винятком зазначених далi випадкiв, iнструкцiї виконуються послiдовно.

## Оператор (iнструкцiя) присвоювання

Синтаксис

1. Assign = Ident ’:=’ Expression

Обмеження

1. Тип змiнної з iдентифiкатором Ident має вiдповiдати типу виразу праворуч оператора :=.
2. ...

Семантика

1. ...
2. ...

Приклад

1. ...

## Iнструкцiя введення

Синтаксис

1. ...

Обмеження

1. ...
2. ...

Семантика

1. ...
2. ...

Приклад

1. ...

## Iнструкцiя виведення

Синтаксис

1. ...

Обмеження

1. ...
2. ...Семантика
3. ...
4. ...

Приклад

1. ...

## Оператор циклу

Синтаксис

1. ...

Обмеження

1. ...
2. ...Семантика
3. ...
4. ...

Приклад

1. ...

# Повна граматика мови Crundras

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