Bogdan PRICOP
Gheorghe BĂDĂRĂU

Cap 5. Atomi lexicali. Expresii

- ***Setul de caractere Fortran***
- ***Nume***
- ***Constante***
- ***Constante întregi, reale, complexe, logice şi caracter***
- ***Constante numite. Instrucţiunea PARAMETER***
- ***Variabile. Declararea tipului***
- ***Expresii. Expresii aritmetice***
- ***Expresii caracter***
- ***Expresii de relaţie***
- ***Expresii logice***

Setul de caractere Fortran

Limbajul Fortran este un limbaj scris. Acest limbaj posedă un *alfabet* format din *caractere alfanumerice* şi *caractere speciale*.

Definiţii:

- *caracter este* { *caracter_alfanumeric*| *caracter_special}*
- *caracter_alfanumeric este* { *literă*| _ | cifră}
- *literă este* {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|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}
- cifră *este* {0|1|2|3|4|5|6|7|8|9}
- *caracter special este* {=|+|-|*|/|(|)|,|.|'|"|:|!|%|&|;|<|>|?|$}

Astfel, setul de caractere Fortran este format din cele 26 litere mari şi mici ale alfabetului englezesc, cele 10 cifre arabe 0, ... ,9, semnul subliniere şi un număr de caractere speciale. În orice context compilatorul Fortran interpretează literele mici ca fiind mari.

Atomi lexicali. Separatori

Atomii lexicali sunt cuvinte formate din caracterele alfabetului care, în timpul compilării, pot primi o anumită semnificaţie în funcţie de contextul în care apar. Astfel, atomii lexicali, sunt cele mai mici unităţi ale unei instrucţiuni Fortran care au o anumită semnificaţie. Atomii lexicali pot fi separatori, delimitatori, cuvinte cheie, nume, etichete, constante şi operatori.

Separatorii sunt:

/ ( ) (/ /) = => , : :: ; %

Separatori pot fi şi *blancurile* (spaţiile libere). O constantă sau o etichetă poate fi despărţită de un cuvânt cheie adiacent prin unul sau mai multe blancuri. Un şir de blancuri este echivalent cu un singur blanc. Aceasta ne permite să utilizăm blancurile cum dorim pentru a îmbunătăţi aspectul textului Fortran.

Nume

Un exemplu de atom lexical este ***numele***, sau, cum se mai numeşte uneori, ***nume simbolic***.

Definiţie

nume este *lit* [*char_alfanum*]...

- unde *lit* este literă iar *char_alfanum* este caracter alfanumeric. Numărul maxim de caractere este 31.
- *Exemplul 1*

ION, SUMA, X3X4, ARIA, W65, An_universitar

- *Exemplul* 2 Nume incorecte sunt:

a mic - conţine un blanc

2bcd - primul caracter nu este o literă

+beta - conţine + care nu este caracter alfanumeric

Constante

O constantă este o dată ce îşi păstrează aceeaşi valoare în decursul execuției unui program. Există ***constante literale*** şi ***constante numite***.

Constantele literale se marchează cu o notație ce indică valoarea constantei. În funcție de tipurile intrinseci constantele se împart în constante întregi, reale, complexe, logice şi caracter.

Constantele numite se notează cu un nume și se introduc cu instrucțiunea PARAMETER sau ca variabile cu atributul PARAMETER.

Constante întregi

Definiții

- *const_lit_int_s* *este* [*s*] *șir_cif* [*_kind*]
- *const_lit_int* *este* *șir_cif* [*_kind*]
- șir_cif *este* *cif* [*cif*]...
- *s* *este* {+|-}
- *kind* *este* {*șir_cif* | *nume_const_scal_înt*}

Aici *const-lit-int-s* înseamnă *constantă literală întreagă cu semn*, şir_cif înseamnă şir de cifre, *s* înseamnă semn, *cif* înseamnă cifră, iar *kind* este parametrul kind. Semnul este obligatoriu dacă este negativ și opțional dacă este pozitiv. Dacă parametrul kind lipsește (kind default), o constantă literală întreagă se va scrie ca un număr întreg cu sau fără semn.

Constante întregi

Constantele sunt interpretate ca fiind în baza 10. O constantă fără semn se presupune că este pozitivă.

- *Exemplu*

Constante întregi

0 1 5320 +25 -3000

Limbajul Fortran permite scrierea unor constante întregi pozitive şi în alte baze diferite de 10. Astfel, avem *constante binare* (baza 2), *constante octale* (baza 8) şi *constante hexazecimale* (baza 16).

Constante reale

Constantele reale în Fortran reprezintă date reale.

Definiții

- *const_lit_reală_s* *este* [*s*] *const-lit-reală*
- *s* *este* {+|-}
- *const_lit_reală* *este* {semnif [E expo] [_kind] | *șir_cif* E ■ ■ expo [_*kind*]}
- semnif *este* {șir_cif. [șir_cif] | .șir_cif}
- *expo* *este* *șir_cif_cu_semn*
- *kind* *este* {şir_cif|*nume_constantă_scalară_întreagă*}

Aici *const_lit_reală_s* înseamnă *constantă literală cu semn, semnif* înseamnă partea semnificativă, E este litera exponent, *expo* înseamnă exponent, şir_cif înseamnă şir de cifre, *kind* este parametrul kind. Semnul - trebuie să apară înaintea unei constante negative; semnul + este opțional în fața unei constante pozitive.

Constante reale

Definiția conține două forme de scriere a constantelor literale reale. Prima formă se numește *forma pozițională*, numită şi *forma în virgulă fixă*, pentru că valoarea fiecărei cifre este determinată de poziția relativă fată de punctul zecimal. În forma poziționalăfără parametru kind o constantă literală reală este:

const_reală_posit *este* {[*s*] *int*. [*frac*] | [*s*] .*frac*}

unde *int* este un șir de cifre ce reprezintă partea întreagă, iar *frac* este un șir de cifre ce reprezintă partea fracționară. Constanta poate să nu aibă cifre în stânga sau în dreapta punctului zecimal, însă partea fracționară și partea întreagă nu pot lipsi simultan.

- *Exemplu*

12. +12. +12.0 0.534 .53 + 0.534 + .534 -3.0 30.02

Constante reale

A doua formă pentru constantele literale reale este *forma exponențială* sau *forma cu virgulă mobilă*. În forma exponențială fără parametru kind o constantă literală reală are forma:

cons_real_exp este *mantisa E expo*

unde *mantisa este* { [s] şir_cifre. [şir_cifre] | [s] . șir_cifres | [s] şir_cifre}

ceea ce înseamnă că mantisa este partea semnificativă. Litera exponent E înseamnă “înmulțit cu 10 la puterea". Exponentul *expo* reprezintă o putere a lui 10 cu care trebuie înmulțită constanta precedentă, întreagă sau reală.

Termenul "virgulă mobilă" provine de la posibilitatea de a muta virgula la mantisă printr-o ajustare corespunzătoare a exponentului.

- *Exemplu:* Numărul 15400 se poate scrie astfel

$$1.54 \times 10^4 = 15.4 \times 10^3 = 0.154 \times 10^5 = 0.0154 \times 10^6 = 154 \times 10^2$$

Constante reale

în formă exponenţiala acest număr îl putem scrie ca o constantă literală reală astfel:

1.54 E04 15.4 E3 0.154 E5 .0154 E6 154. E2 154 E2

Constante complexe

O *constantă literală complexă* reprezintă un număr complex, adică o pereche ordonată de date reale.

Definiţie:

const_lit_complx.	*este*	*(real, imag)*
real	*este*	*{const_lit_int_s|const_lit_real_s}*
imag	*este*	*{const_lit_int_s|const_lit_real_s}*

unde *const_lit_complx* înseamnă *constantă literală complexă*, *const_lit_int_s* înseamnă constantă literală întreagă cu semn, *const_lit_real_s* înseamnă constantă literală reală cu semn, real reprezintă *partea reală a constantei complexe* iar *imag* reprezintă *partea imaginară a constantei complexe*.

- *Exemplu:* Numărul complex 2+3i se poate scrie ca o constantă literal complexă astfel:

(2, 3) (2.,3.) (2._1, 3._1) (2._2, 3.) (2._2, 3._2)

Constante logice

O *constantă logică* specifică una din valorile logice, adevărat sau fals.

Definiţie:

const_lit_logic este { .TRUE. [_*kind*] | .FALSE. [_*kind*] }

unde *const_lit_logic* înseamnă constantă literală logică, .TRUE. înseamnă valoarea "adevărat", iar .FALSE. înseamnă valoarea "fals". Parametrul kind este opțional.

- *Exemplu*

.TRUE. .TRUE._3 .TRUE._2 .TRUE._1

Constante caracter

O *constantă caracter* reprezintă un șir de caractere tipăribile ale setului de caractere ce aparține procesorului.

Definiție:

const_lit_caracter este {[knd_] ‘char [char]...’ | [knd_] “char [char]...”}

unde *const_lit_caracter* este *constantă literală caracter*, *char* este unul din caracterele tipăribile, *knd* este parametrul kind. Apostrofurile și ghilimelele din stânga și din dreapta constantei sunt *delimitatori* și nu sunt incluse în valoarea constantei.

Exemplul 1

Constantele 'casa' și "casa"

au valoarea casa, adică șirul de caractere cuprins între delimitatori.

Constante caracter

Exemplul 2

Constantele 'un om' şi 'unom' sunt diferite.

În constantele caracter există diferență între literele mari și mici.

Exemplul 3

Constanta "Ac" este diferită de constanta "ac".

Delimitatorii de un fel pot fi incluși într-un șir mărginit de delimitatori de celălalt fel.

Exemplul 4

Delimitatori incluși în șirul de caractere: "a spus 'salut'" 'a spus "salut" '

Constante numite. Instrucțiunea PARAMETER

În Fortran putem avea şi *constante numite,* constante pe care le notăm cu un nume. Pentru a specifica că este vorba de o constantă numită folosim instrucţiunea PARAMETER.

Definiţie:

instr_parameter este PARAMETER (*name* = *expr* [, *name* = *expr*]...)

unde *name* este un nume de constantă numită, iar *expr* este o expresie ce poate conține doar nume dintr-o instrucţiune PARAMETER anterioară.

Exemplul 1

Instrucțiunea PARAMETER (pi = 3.14159)

definește constanta numită pi cu valoarea 3.14159.

Constante numite. Instrucțiunea PARAMETER

Exemplul 2

Instrucțiunile PARAMETER (i = 10, j = 20)

PARAMETER (ip2 = i + 2)

definesc constantele numite i, j, ip2 ce au valorile 10, 20, 12.

Constantele numite pot fi definite şi cu instrucţiuni de declarare a tipului ce folosesc atributul PARAMETER.

Exemplul 3

Instrucțiunea din exemplul 1 este echivalentă cu instrucțiunea

REAL PARAMETER pi = 3.14159

Variabile. Declararea tipului

O variabilă scalară este un obiect scalar notat cu un ***nume***. Pentru procesor variabila scalară este o celulă de memorie a cărei adresă simbolică este numele variabilei, celulă în care se stochează valoarea variabilei. În decursul execuției programului variabila își poate schimba valoarea.

Majoritatea variabilelor nu au nici o valoare atunci când începe execuția programului. Se spune că aceste variabile sunt *nedefinite*. Excepție o fac variabilele care sunt inițializate cu instrucțiunea DATA sau cu instrucțiuni de declarare a tipului; se zice că aceste variabile sunt *definite*. O variabilă poate căpăta o valoare sau își poate schimba valoarea la execuţia unei instrucțiuni de atribuire sau a unei instrucţiuni de citire. Astfel, o variabilă poate căpăta diferite valori la momente diferite de timp şi în anumite circumstanţe poate deveni nedefinită.

Variabile. Declararea tipului

Pentru a specifica tipul şi unele atribute ce descriu modul de utilizare în program a variabilelor se folosesc instrucţiuni de declarare a tipului.

Şabloane *pentru declararea* ***tipului întreg****:*

- INTEGER var [,var]...
- INTEGER var = expr [, var = expr]...
- INTEGER, PARAMETER var = expr [, var = expr]...

Exemplu

- INTEGER i1
- INTEGER I, J, K
- INTEGER i1 =10
- INTEGER X1=1, X2=2, X3=3, X4=4
- INTEGER, PARAMETER masa = 2

Variabile. Declararea tipului

Şabloane *pentru declararea* ***tipului real****:*

- REAL var [, var] ...
- REAL var = expr [, var = expr]...
- REAL, PARAMETER var = expr [, var = expr]...

Exemplu

- REAL a, b, c, d
- REAL gama
- REAL beta = 3.0
- REAL, PARAMETER pi = 3.14159

Variabile. Declararea tipului

Șabloane *pentru declararea* ***tipului complex****:*

- COMPLEX var [, var]...
- COMPLEX var = expr [, var = expr]...
- COMPLEX, PARAMETER var = expr [, var = expr]...

Exemplu

- COMPLEX AX, BX
- COMPLEX, PARAMETER s = (1.,0.)

Variabile. Declararea tipului

Șabloane *pentru declararea* ***tipului logic****:*

- LOGICAL var [, var]...
- LOGICAL var = expr [, var = expr]...
- LOGICAL, PARAMETER var = expr [, var = expr]...

Exemplu

- LOGICAL test
- LOGICAL g1, g2, g3
- LOGICAL h1 = .TRUE. , h2 = .FALSE.

Variabile. Declararea tipului

Şabloane *pentru declararea* ***tipului caracter****:*

- CHARACTER [([LEN =] len)] var [,var]...
- CHARACTER * len [,] var [, var]...
- CHARACTER var [, var] ...
- CHARACTER var = expr (, var = expr]...

Exemplu

- CHARACTER (80) LINIE
- CHARACTER (LEN = 80) LINIE
- CHARACTER*80 LINIE
- CHARACTER A, S, C
- CHARACTER*1 K1='o', K2='p', K3='q'
- CHARACTER r1*2 = "xy", r2*2="xy"

Declararea implicită

Într-o unitate de program tipul variabilelor se poate descrie prin instrucțiuni de declarare a tipului. În lipsa instrucțiunilor de declarare a tipului (default), prin convenție, numele ce încep cu una dintre literele I, J, K, L, M, N, litere mici sau mari, reprezintă variabile de tip întreg, iar numele ce încep cu orice altă literă reprezintă variabile de tip real.

Pentru a schimba asociația între tip și primul caracter al numelui se folosește instrucțiunea IMPLICIT.

Exemplu

```
IMPLICIT REAL (D)
IMPLICIT REAL (A - C, S, T, U - Z)
IMPLICIT INTEGER I, J, K, O
IMPLICIT CHARACTER*2 (W)
IMPLICIT LOGICAL (L)
```

Declararea implicită

Instrucțiunile specifică că toate numele ce încep cu litera D sunt numere reale; toate numele ce încep cu una din literele A, B, C, S, T, U, V, W, X, Y, Z reprezintă variabile reale; toate numele ce încep cu una din literele I, J, K, O reprezintă variabile întregi, toate variabilele ce încep cu litera W sunt variabile caracter cu lungimea 2 și numele ce încep cu litera L reprezintă variabile logice.

Astfel, instrucțiunea IMPLICIT atribuie tipul specificat tuturor numelor ce încep cu litera specificată sau cu una din literele din domeniul de litere specificat. Dacă se folosește **IMPLICIT NONE**, atunci toate numele variabilelor trebuie să apară în mod explicit în instrucţiuni de declarare a tipului; omiterea unui nume de variabilă va cauza o eroare de compilare.

Expresii aritmetice

În Fortran o expresie indică a serie de calcule sau de manipulări ale datelor. O expresie este formată din operanzi (datele de prelucrat) şi operatori (operații de prelucrare).

O expresie scalară numerică are ca rezultat o valoare de tip întreg, real sau complex. Expresiile scalare numerice se construiesc cu operanzi numerici de tip real, întreg sau complex, paranteze şi operatori aritmetici.

Operatorii aritmetici sau *operatorii numerici* sunt:

- ** - ridicare la putere
- * - înmulțire
- / - împărțire
- - - scădere
- + - adunare

Expresii aritmetice

Exemplu

Operatori binari: a+b; a-2; c*d; *x/y*

Exemplul

În paranteze + şi - sunt operatori unari:

a + (-2.); c *(+D); y * (-3)

Precedența

Operator	Precedența	În context de precedență egală
**	Cea mai mare	dreapta la stânga
* sau /	-	stânga la dreapta
+ sau – unar	-	stânga la dreapta
+ sau – binar	-	stânga la dreapta

Expresii caracter

Expresiile scalare caracter conțin operanzi ce pot fi: constante caracter, variabile caracter. Există un singur operator caracter, *operatorul de concatenare //* ce are ca efect combinarea a doi operanzi caracter într-un singur rezultat caracter.

Operanzii trebuie să aibă același parametru kind. Parametrul lungime al concatenării reprezintă suma lungimilor operanzilor.

Exemplu
'AB' // 'CDE'
produce constanta caracter 'ABCDE'. Operanzii 'AB', 'CDE' au lungimile 2, respectiv 3 iar rezultatul va avea lungimea 5.

Expresii caracter

Parantezele nu afectează evaluarea unei expresii caracter.

Exemplu

Următoarele expresii caracter sunt echivalente:

("ABC" // "DE") // "F"

"ABC" // ("DE") // "F"

"ABC" // "DE" // "F'

Dacă un operand caracter dintr-o expresie caracter conține blancuri, blancurile sunt incluse în valoarea expresiei caracter.

Exemplu

"ABC*b*" //„DE*b*" //"F"

are valoarea:

"ABC*b*DE*b*F"

Aici cu *b* am notat caracterul blanc.

Expresii de relație

Expresiile de relație, numite și *expresii relaționale*, compară valorile a două expresii numerice sau caracter. *Operatorii de* relaţie, sau cum se mai numesc, *operatorii relaţionali* sunt:

Operator	Operația
.LT. sau <	Mai mic decât
.LE. sau <=	Mai mic decât sau egal
.EQ. sau ==	Egal
.NE. sau /=	Neegal
.GT. sau >	Mai mare decât
.GE. sau >=	Mai mare decât sau egal

Expresii logice

Operatorii logici sunt:

Operator	Operația
.NOT.	Negație
.AND.	Conjuncție
.OR.	Disjuncție
.EQV.	Echivalență
.NEQV.	Neechivalență

Rezultatul evaluării unei expresii logice este de tip logic, constanta logică .TRUE. sau constanta logică .FALSE. Operatorii .AND., .OR., .EQV., .NEQV. sunt operatori binari; ei se scriu între operatori de tip logic.

Expresii logice

Operatorul .NOT. este operator unar și precedă operandul. Operatorii .NOT., .AND. şi .OR au semnificația din logica matematică. Semnificația operațiilor .EQV. şi .NEQV. este arătată în continuare:

a	b	a.EQV.b	a.NEQV.b
TRUE	TRUE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE

Expresii logice

Precedența operatorilor logici este prezentată în continuare:

Operator	Precedența	În context de precedență egală
.NOT.	Cea mai mare	
.AND.	-	stânga la dreapta
.OR.	-	stânga la dreapta
.EQV. sau .NEQV.	-	stânga la dreapta

Când două operații consecutive sunt de precedență egală, operația din stânga se efectuează prima. Doi operatori .NOT. nu pot fi adiacenți. Operatorul .NOT. poate să apară lângă un alt operator logic.

Exemple de expresii logice

.NOT. a; a .OR. b .AND. .NOT. c;

((.NOT. d) .AND. . TRUE.) .NEQV. (a .OR. c)

Bibliografie

- *Octavian PETRUȘ, Fortran 90/95, Limbaj și Tehnici de programare, Editura Universității Tehnice "Gheorghe Asachi" din Iași, 2001*
- Romeo CHELARIU, Sisteme de operare şi limbaje de programare (Îndrumar de laborator), http://www.sim.tuiasi.ro/wp-content/uploads/Chelariu-indrumar-solp.pdf, 2004
- https://ro.wikipedia.org