Bogdan PRICOP
Gheorghe BĂDĂRĂU

Cap 4. Tipuri de date intrinseci

- ***Date scalare***
- ***Date întregi***
- ***Date logice***
- ***Date reale***
- ***Necesități de stocare***

Date scalare

În limbajul Fortran pentru a descrie datele ce urmează a fi prelucrate se folosește terminologia ***obiect - date*** (data object). Datele cele mai simple sunt ***datele scalare***, care sunt date fără structură.

O dată scalară prezintă o ***notație*** şi o ***valoare***. În timpul execuției programului fiecare dată scalară este stocată într-un anumit mod într-o celulă de memorie. În limbajele de programare există şi ***date compuse*** ce prezintă o altă structură. De exemplu, în Fortran, datele compuse sunt tablourile.

Fiecărei date i se asociază un ***tip***; tipul datei este determinat de natura ei, de notaţia folosită pentru indentificare, de modul de stocare în memorie şi de modul de efectuare a operaţiilor.

Date scalare

Majoritatea limbajelor de programare posedă anumite ***tipuri de date intrinseci***, adică anumite tipuri de date predefinite: date întregi, date reale, date logice şi date caracter. Limbajul Fortran acceptă 5 tipuri de date intrinseci: întregi, reale, complexe, logice şi caracter.

Într-un program cu datele de un anumit tip putem să efectuăm diferite sarcini:

- pentru a calcula, folosim date reale sau complexe
- pentru a număra, folosim date întregi
- pentru a lua decizii, folosim date logice
- pentru a explica, folosim date caracter.

Date întregi

Pentru un computer dat ce lucrează cu un anumit compilator ***datele de tip întreg***, ***datele întregi***, sau prescurtat, ***întregii***, sunt elemente ale sistemului numerelor întregi. ***Sistemul numerelor întregi*** este o submulţime de *n* numere cuprinse într-un anumit interval al mulțimii numerelor întregi binecunoscute din matematică:

$$min \leq n \leq max$$

Pentru a descrie reprezentarea și comportarea datelor de un anumit tip se folosește un ***model de reprezentare***. În Fortran se folosește următorul ***model de reprezentare a datelor întregi***:

$$s \times \sum_{k=0}^{q-1} x_k \cdot B^k$$

unde *B* este *radix* (baza), un număr mai mare decât 1, *q* este un întreg pozitiv numit *digits* care reprezintă *numărul de cifre semnificative ale bazei*, *s* reprezintă semnul (+ sau -).

Date logice

Datele de *tip logic,* sau de *tip Boolean* definesc registrul mărimilor logice ce nu pot avea decât doua valori **TRUE** (adevărat) şi **FALSE** (fals). Numele "Boolean" provine de la numele lui George Boole (1815-1864) care a pus bazele algebrei logice. Cu datele logice se pot efectua ***operaţiile logice*** :

∨ disjuncţie, OR, sau	∧ conjuncţie, AND, şi	¬ negaţie, NOT, nu

Valorile expresiilor $p \vee q$, $p \wedge q$ şi $\neg p$ sunt definite în continuare unde p şi q sunt operanzi logici.

p	q	$p \vee q$	$p \wedge q$	$\neg p$
False	False	False	False	True
True	False	True	False	False
False	True	True	False	True
True	True	True	True	False

Date logice

Din definițiile operațiilor logice prezentate anterior, putem demonstra următoarele legi:

1. Legile de comutativitate

$$p \vee q = q \vee p$$
$$p \wedge q = q \wedge p$$

2. Legile de asociativitate

$$(p \vee q) \vee r = p \vee (q \vee r)$$
$$(p \wedge q) \wedge r = p \wedge (q \wedge r)$$

3. *Legile de distributivitate*

$$(p \wedge q) \vee r = (p \vee r) \wedge (q \vee r)$$
$$(p \vee q) \wedge r = (p \wedge r) \vee (q \wedge r)$$

4. *Legile lui De Morgan*

$$\neg (p \vee q) = \neg p \wedge \neg q$$
$$\neg (p \wedge q) = \neg p \vee \neg q$$

Date reale

La baza analizei matematice se află multimea ***R*** a numerelor reale. Computerele nu pot lucra cu numere reale, ele pot lucra numai cu aproximații finite ale acestora. Există mai multe metode de aproximație a numerelor reale cu reprezentări finite. Metoda cea mai folosită este aproximația numerelor reale prin *sistemul numerelor reale* numit şi ***sistemul numerelor cu virgulă flotantă*** (floating-point system). În programare *datele reale* sunt elemente ale sistemului $\hat{R}$ al numerelor cu virgulă flotantă. În sistemul numerelor cu virgulă flotantă un număr real x se aproximează cu un număr $\hat{x}$ ce se poate scrie cu ajutorul a două numere întregi M şi E , fiecare din ele conținând un număr finit de cifre, astfel:

$$\hat{x} = M \times B^E$$

B se numeşte ***baza*** reprezentării sistemului numerelor cu virgulă flotantă, M ***mantisă*** iar E se numeşte ***exponent***.

Date reale

Denumirea de "virgulă flotantă" provine de la faptul că același număr se poate scrie mutând poziția virgulei şi ajustând corespunzător exponentul. De exemplu, dacă $B = 10$, avem $10{,}56 \times 10^3 = 1{,}056 \times 10^4 = 0{,}1056 \times 10^5 = 1056 \times 10^1$.

Mulțimea $\hat{R}$ nu este o mulțime infinită. Numere din această mulțime nu sunt dispuse în mod uniform pe axa numerelor reale. Cu datele reale se pot efectua operațiile aritmetice, însă aritmetica numerelor cu virgulă flotantă, diferă de aritmetica numerelor reale. Nu este posibil să descriem în mod absolut exact operațiile aritmetice efectuate cu date reale.

Rezultatele calculelor cu date reale, depind de tipul problemei şi de algoritmul ales. În cel mai bun caz rezultatele obținute vor fi aproximații cu erori obligatorii. Evaluarea acestor erori, ce sunt consecința înlocuirii conținutului real cu o mulţime finită de reprezentanți, este o problemă dificilă şi este obiect al *analizei numerice*

Necesități de stocare

Compilatorul FTN 90 (Nag, 1997) acceptă următoarele tipuri intrinseci de date scalare:

i) întreg: INTEGER, INTEGER ([K1ND =] 1), INTEGER([KIND =] 2) şi INTEGER ([KIND =] 3)

ii) real: REAL, REAL ([KIND =] 1) şi REAL ([KIND =] 2)

iii) complex: COMPLEX, COMPLEX ([KIND =] 1), COMPLEX ([KIND =] 2)

iv) logic: LOGICAL, LOGICAL ([KIND =] 1), LOGICAL ([KIND =] 2), LOGICAL ([KIND =] 3)

v) caracter (CHARACTER *n)

Compilatorul FTN 90 impune următoarele cerințele de stocare:

Necesități de stocare

Tip	Bytes
INTEGER	4
INTEGER (KIND = 1)	1
INTEGER (KIND = 2)	2
INTEGER (KIND = 3)	4
REAL	4
REAL (KIND = 1)	4
REAL (KIND = 2)	8
LOGICAL	4
LOGICAL (KIND = 1)	1
LOGICAL (KIND = 2)	2
LOGICAL (KIND = 3)	4
COMPLEX (KIND = 1)	8
COMPLEX (KIND = 2)	16
CHARACTER*n	n

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