Utilizarea programului RAPTOR

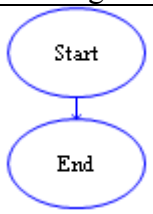
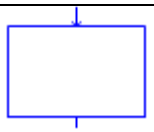
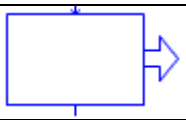
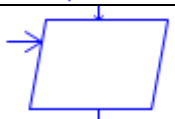
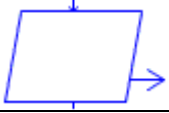
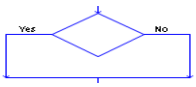
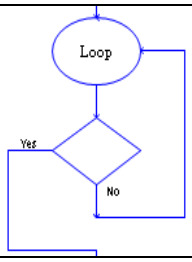
Introducere

RAPTOR este un mediu de programare vizual bazat pe scheme logice. Acesta permite realizarea unor programe simple ce pot pune în evidenţă structura logică a unui algoritm fără complicaţiile legate de sintaxa şi procesul de compilare al unui limbaj de programare uzual.

Avantajele programului RAPTOR:

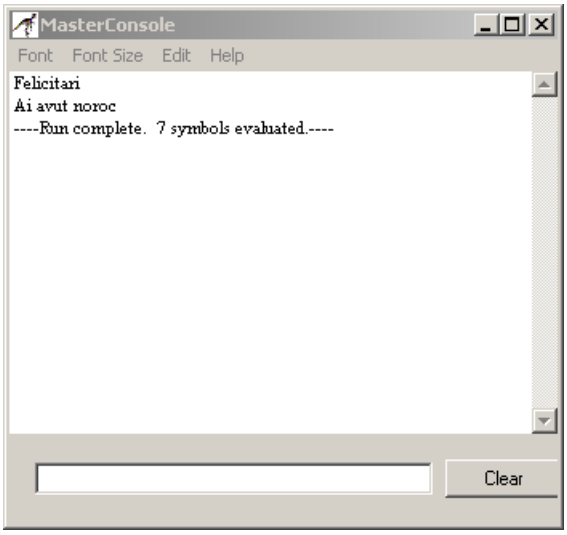
- Problemele de sintaxă sunt reduse la minimum, programele se pot scrie foarte uşor
- Mediul vizual facilitează înţelegerea şi proiectarea algoritmilor
- Rularea este simplă şi eventualele mesaje de eroare sunt uşor de interpretat

Tabel rezumativ al instrucţiunilor disponibile

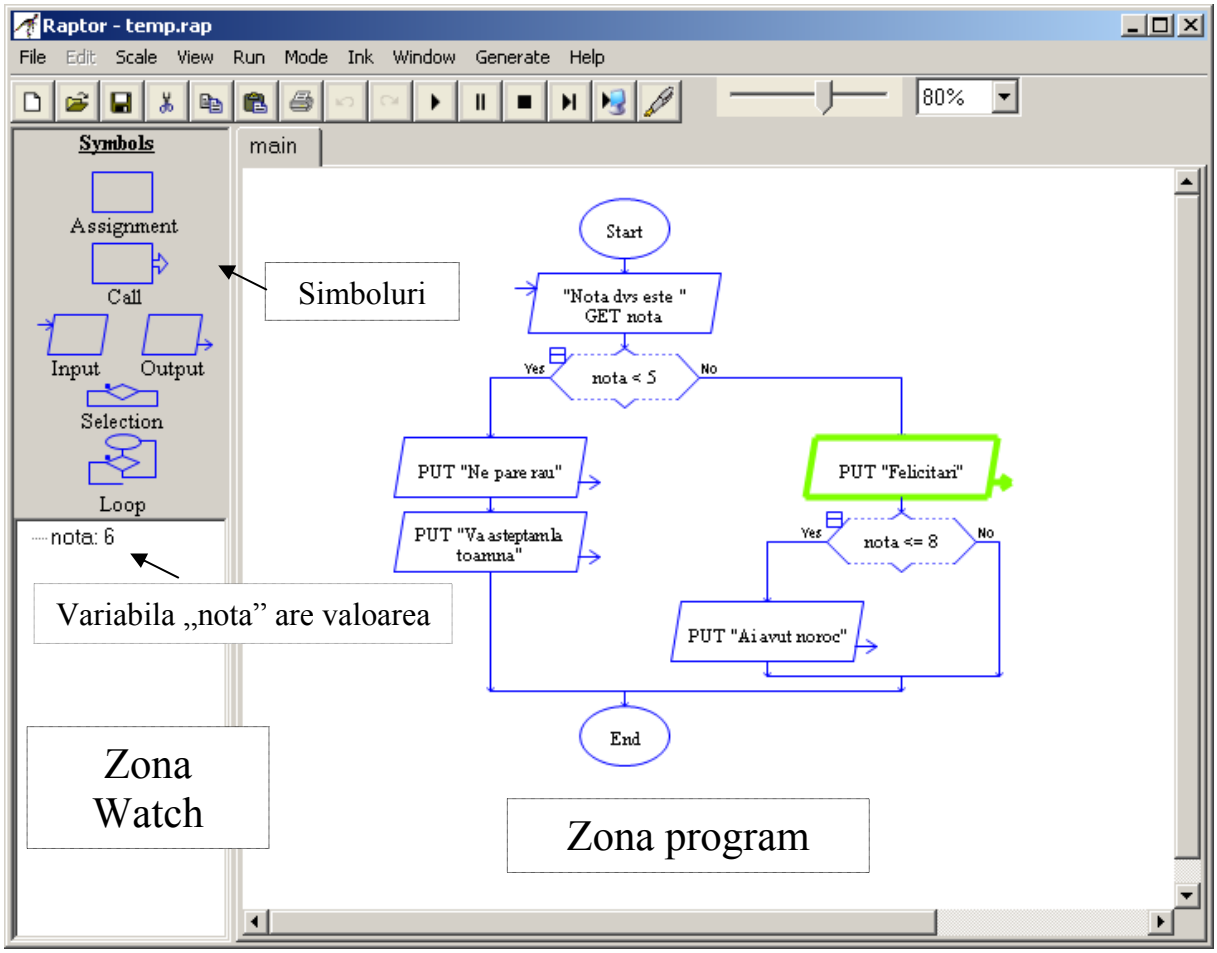
Simbol grafic	Denumire RAPTOR	Denumire pseudocod	Descriere
	Start, End	***start***, ***stop***	Marchează începutul şi sfârşitul programului
	Assignment	***atribuie***	Atribuie unei variabile o anumită valoare
	Call	***funcţie/ procedură***	Apelează o funcţie / procedură
	Input	***citeşte***	Preia o valoare de la utilizator
	Output	***scrie***	Afişează mesaje către utilizator
	Selection	***dacă***	Selectează pentru execuţie o structură în funcţie de rezultatul evaluării unei condiţii
	Loop	***structură ciclică*** (***cât timp/ repetă***)	Execută în mod repetat instrucţiuni cât timp condiţia este adevărată (pre-fixat/postfixat)

Ferestrele RAPTOR

Mediul RAPTOR este compus din fereastra principală, unde se proiectează şi se rulează programul, şi fereastra consolă, unde sunt scrise mesajele de ieşire. Preluarea datelor se face prin ferestre dialog dedicate (*prompt*).



Fereastra principală este compusă în primul rând din zona programului. În partea stângă se găsesc setul de simboluri ale instrucţiunilor, şi, în partea inferioară, zona *Watch*. Zona Watch afişează toate variabilele existente la un moment dat, împreună cu valorile lor. O modificare a unei variabile este indicată prin culoarea roşie.



Crearea unui program RAPTOR

Un program RAPTOR reprezintă o schemă logică. Execuţia programului începe de la simbolul `Start`, după care se evaluează succesiv toate blocurile din schema logică până la `End`. Un program are un singur simbol Start şi un singur End, care sunt create în mod automat la deschiderea oricărui fişier nou şi nu pot fi adăugate sau modificate manual.

Un program în RAPTOR se creează prin introducerea blocurilor din stânga în schema programului în locul dorit. Introducerea blocurilor se face fie prin tragere (*drag'n drop*) până în poziţia dorită, fie prin selectarea blocului dorit (clic) şi apoi clic pe poziţia dorită. După aceasta, se introduc parametrii fiecărui bloc prin dublu-clic pe acesta şi completarea ferestrei de proprietăţi care se deschide.

Rularea unui program RAPTOR

Rularea unui program se face instrucţiune cu instrucţiune. Simbolul care se execută la un moment dat este indicat cu culoarea verde. Orice modificare a variabilelor în urma execuţiei este marcată cu roşu în fereastra Watch.

Sunt disponibile următoarele comenzi:

- **Step (F10)** execută doar instrucţiunea imediat următoare.
  Atenţie! Instrucţiunea marcată cu verde este cea înaintea căreia este oprită execuţia, aşadar ea **încă nu este** executată. La comanda Step se execută această instrucţiune, apoi se avansează la următoarea.
- **Execute to Completion** rulează întregul program până la terminare.
- **Reset** opreşte definitiv execuţia şi şterge toate variabilele.
- **Reset/Execute (F5)** restartează programul.
- **Pause** opreşte temporar execuţia, care poate fi apoi continuată de utilizator.

Viteza de execuţie poate fi selectată prin slider-ul din bara de iconiţe.

Există opţiunea de a marca un simbol cu *breakpoint*. Un breakpoint face ca rularea programului să se oprească temporar la simbolul respectiv (acest lucru are sens doar în cazul rulării cu comanda „Execute to Completion" de mai sus). Un breakpoint se setează cu clic-dreapta pe simbolul respectiv, urmat de alegerea comenzii „Toggle breakpoint". Repetarea acestei operaţii şterge breakpoint-ul.

Variabile în RAPTOR

Variabilele reprezintă celule de memorie care stochează o valoare. O variabilă poate stoca o singură valoare, însă pe parcursul rulării programului această valoare poate fi modificată prin instrucţiuni de atribuire (de aici si numele de „variabilă"). Toate variabilele au un nume şi o valoare (conţinut).

În RAPTOR, variabilele sunt create automat atunci când li se atribuie o valoare. De ex., instrucţiunea de atribuire `n ← 5` va crea o variabilă numită *n* în cazul în care nu există, şi i se va atribui valoarea 5. Dacă variabila cu acest nume există deja (are deja o valoare), valoarea sa va fi înlocuită cu 5. Un alt mod de a crea o variabilă este prin intermediul instrucţiunii *Input (Citeşte)*, care preia o valoare de la utilizator şi o stochează într-o variabilă.

Atenţie! Nu se pot utiliza variabile care nu există încă (cărora nu li s-a atribuit încă o valoare). De exemplu, instrucţiunea de atribuire `n ← a + b` va duce la o eroare în cazul în care una dintre variabilele *a* şi *b* nu a fost creată încă.

Tablouri

Un caz particular de variabile îl reprezintă tablourile. Un tablou este o colecţie de variabile de acelaşi tip, la care accesul se face prin intermediul unuia sau mai multor indecşi. Un exemplu

familiar îl reprezintă referirea la elementele unei matrici *a* sub forma a_{ij}. În terminologia limbajelor de programare, *a* reprezintă un tablou iar indicii *i*, *j* reprezintă indecşii.

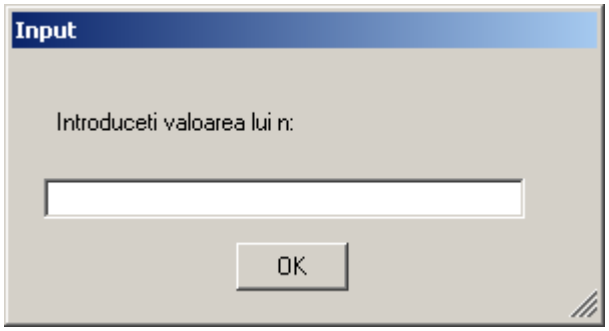
În RAPTOR, elementele unui tablou se accesează prin paranteze pătrate între care se pun indecşii, separaţi prin virgule. De exemplu, elementul a_{ij} se scrie în RAPTOR ca *a[i,j]*.

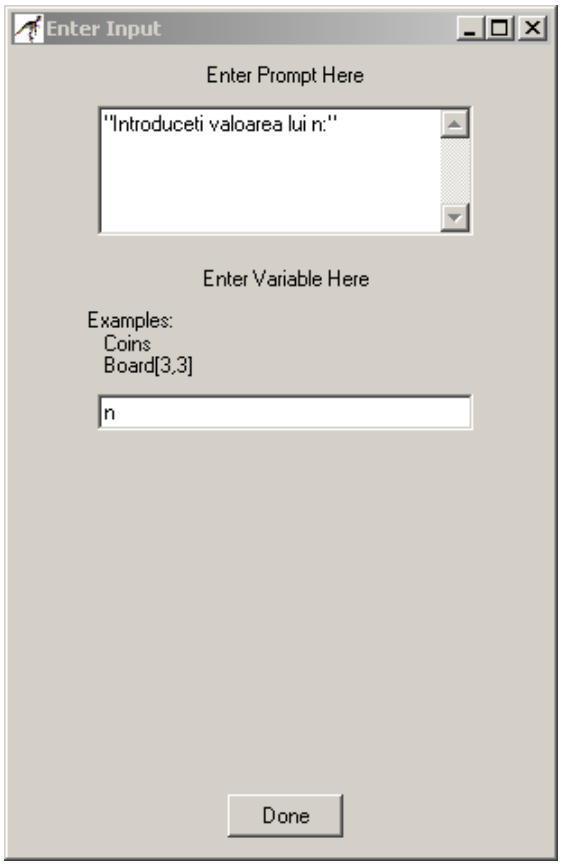
Un tablou se creează în RAPTOR la fel ca o variabilă, adică direct prin atribuire. Numerotarea elementelor începe de la numărul 1. Astfel, *a[1,1]* reprezintă elementul de pe prima linie şi prima coloană .

Instrucţiuni RAPTOR

Input (Citeşte) - Preia date de la utilizator
Instrucţiunea Input (Citeşte) se foloseşte pentru a prelua date de la utilizator. Aceasta determină apariţia unei ferestre în care utilizatorului i se cere să introducă datele necesare.

Prin dublu-clic pe blocul Input (Citeşte) se introduce mesajul afişat utilizatorului şi numele variabile în care va fi stocată valoarea introdusă de acesta. Mesajul destinat utilizatorului trebuie introdus între ghilimele.

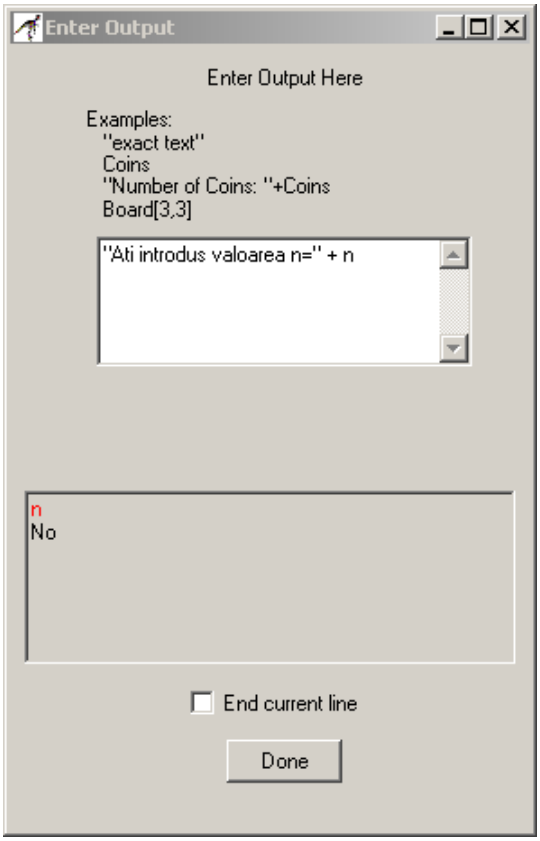


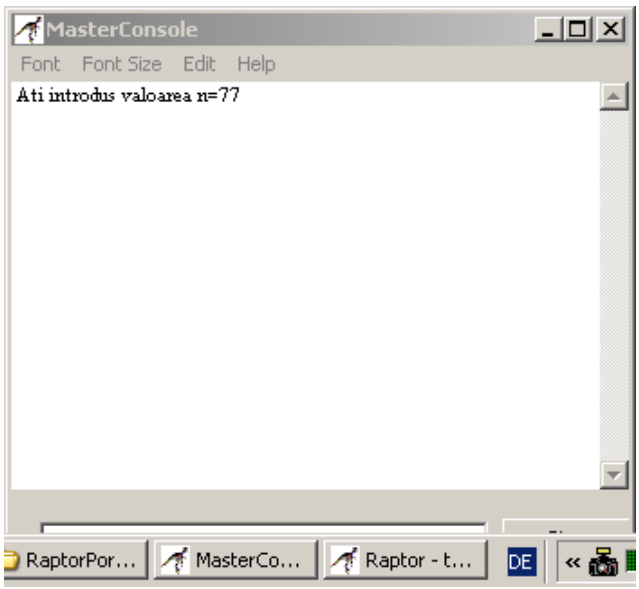


Output (Scrie) – Afişează mesaje şi date

Instrucţiunea Output (Scrie) se foloseşte pentru a afişa mesaje către utilizator (de ex. rezultatele unui calcul). Mesajul este afişat în fereastra consolă (Master Console).

În fereastra de proprietăţi a blocului se scrie mesajul care se doreşte a fi afişat, care poate conţine text şi valori ale unor variabile, unite între ele prin semnul +. Textul trebuie pus întotdeauna între ghilimele, variabilele nu. La rularea programului, în locul numelor variabilelor se va afişa valoarea lor.





Exemple de mesaje

Dacă variabila *n* are valoarea 77, atunci:

„Ati introdus valoarea n = ” + n

va duce la afişarea mesajului:

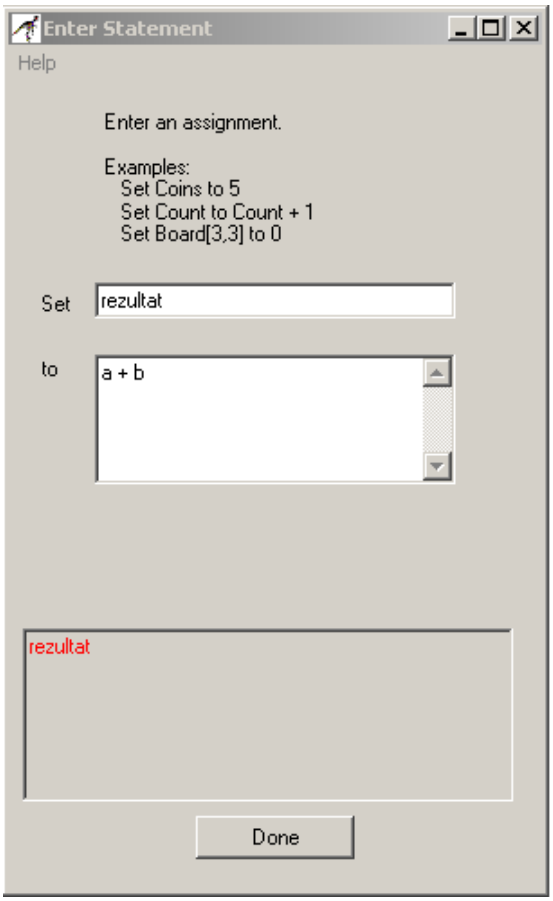
Ati introdus valoarea n = 77

Dacă variabilele *a*, *b* şi *rezultat* au valorile 5,2,25, atunci:

a + „ la puterea ” + b + „ = ” + rezultat

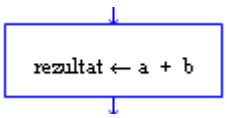
va duce la afişarea mesajului:

5 la puterea 2 = 5



Observaţie: Este întotdeauna indicat ca rezultatele unui program să fie prezentate împreună cu un scurt mesaj explicativ.

Assignment (Atribuire) – Atribuie valori variabilelor



Instrucţiunea atribuie unei variabile rezultatul unei expresii (formule) sau o valoare oarecare. În acest fel valoarea este stocată într-o celulă de memorie, putând fi utilizată ulterior în cadrul programului.

În fereastra de proprietăţi a blocului se introduce numele variabilei destinaţie şi expresia de calcul în urma căreia rezultă valoarea ce se stochează.

Reţineţi! O instrucţiune de atribuire se execută **de la dreapta la stânga**, adică întâi se evaluează expresia din dreapta, iar apoi rezultatul se stochează în variabila indicată în partea stângă. Se respectă regulile matematice de calcul (se execută înmulţirile înainte de adunări, parantezele au prioritate etc.).

În cadrul expresiei pot apărea operaţii metamatice (+, -, *, /) precum şi unele funcţii matematice elementare (sinus, cosinus, radical, ridicare la putere, logaritm etc.). Ridicarea la putere se indică cu ^ sau ** (de ex. 2 ^ 3 este 8 sau 3 ** 2 este 9). Lista completă a funcţiilor ce pot fi folosite este prezentată la final. De asemenea, unele constante sunt predefinite (*pi*, *e*).

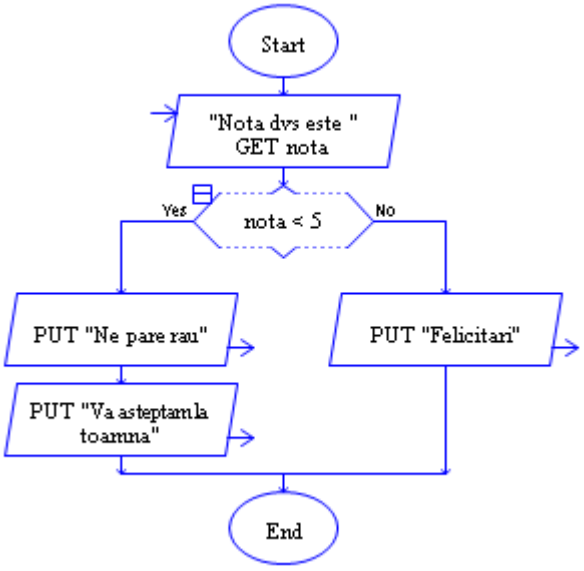
Exemple de atribuiri:

`rezultat ← a + b`	Se adună valorile variabilelor *a* şi *b*, iar rezultatul se stochează în variabila *rezultat*
`r ← 6.3`	Variabila *r* primeşte valoarea 6,3
`a ← sqrt( 3 * sin(x + pi/2) + 1)`	Se calculează întâi *x* + *pi/2*, apoi sinusul rezultatului, apoi rezultatul se înmulţeşte cu 3, se adaugă 1, apoi se extrage radicalul din total, iar rezultatul se stochează în variabila *a*.
`i ← i+1`	Valoarea variabilei *i* se sumează cu 1, iar rezultatul este stocat tot în variabila *i* (se înlocuieşte vechea valoare). Noua valoare a lui *i* devine aşadar vechea valoare plus 1, cu alte cuvinte se creşte valoare lui *i* cu o unitate, în aceeaşi celulă de memorie.

Selection (Decizie)

Această instrucţiune testează valoarea de adevăr a unei expresii de decizie (adevărat (A) sau fals (F)), şi în funcţie de rezultat execută un bloc de instrucţiuni sau altul.

În exemplul din dreapta, dacă expresia de decizie „nota < 5”se evaluează ca (A), se execută ramura din stânga. Dacă expresia este falsă (F), se execută ramura din dreapta.

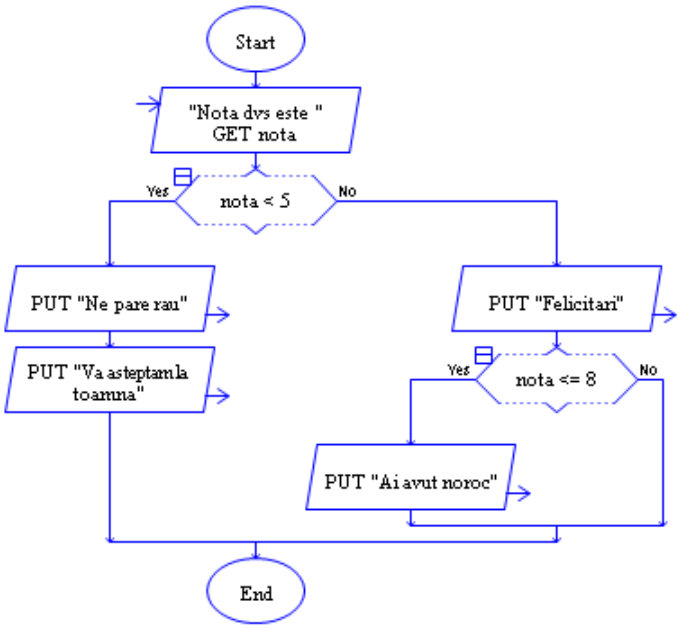


Este posibil ca una din cele două ramuri poate să nu conţină nici o instrucţiune, dacă logica algoritmului nu o necesită.

Expresiile de decizie trebuie să poată fi evaluate ca (A) sau (F). În acest scop, se pot folosi operatorii logici < , >, =, != („diferit de”), <= (mai mic sau egal), >= (mai mare sau egal), *and* (ŞI logic), *or* (SAU logic), *not* (negare logică) etc. De asemenea se pot folosi aceleaşi funcţii matematice şi constante predefinite ca şi la instrucţiunea de atribuire.

Ordinea în care se evaluează diferitele tipuri de operaţii dintr-o expresie este următoarea:

1. evaluează toate funcţiile, apoi
2. evaluează expresiile între paranteze, apoi
3. evaluează ridicările la putere (^,**), apoi
4. calculează înmulţirile şi împărţirile, de la stânga la dreapta, apoi
5. calculează adunările şi scăderile, de la stânga la dreapta, apoi
6. evaluează operatorii relaţionali (= != /= < <= > >=), de la stânga la dreapta, apoi
7. evaluează operatorii logici de negare (*not*), de la stânga la dreapta, apoi
8. evaluează operatorii logici ŞI (*and*), de la stânga la dreapta, apoi
9. evaluează operatorii logici SAU-EXCLUSIV (*xor*), de la stânga la dreapta, apoi
10. evaluează operatorii logici SAU (*or*), de la stânga la dreapta.



Cascadarea instrucţiunilor

Ramurile unei instrucţiuni de decizie pot conţine orice fel de instrucţiuni, inclusiv alte instrucţiuni de decizie. Un astfel de exemplu este prezentat în dreapta. Instrucţiunile pot fi cascadate de oricâte este nevoie.

Exemple de expresii de decizie

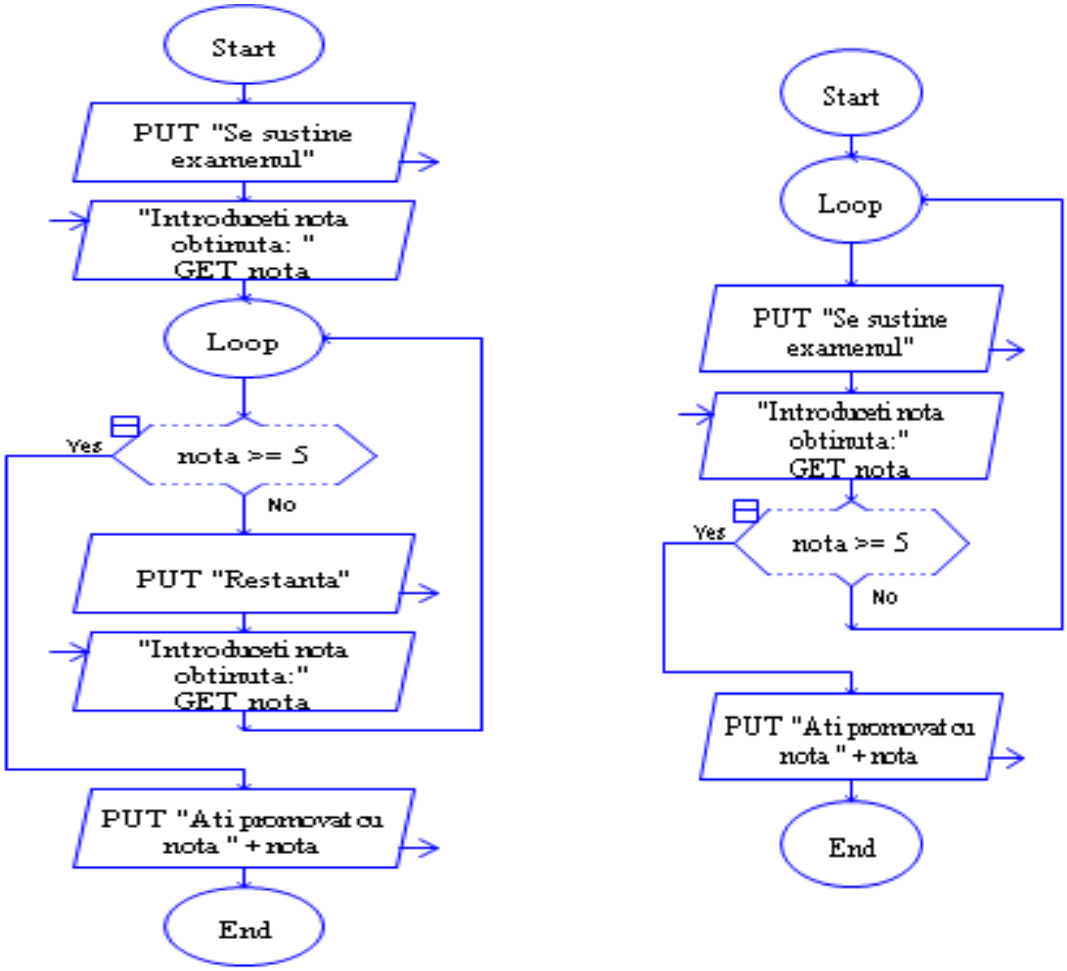
`varsta < 18`	(A) dacă valoarea variabilei *varsta* este mai mică decât 18, (F) în caz contrar
`an >= 1900 and an < 2000`	(A) dacă valoarea variabilei *an* reprezintă un an din secolul XX, (F) în caz contrar
`(D != 100) or (D = 100 and a>0)`	(A) dacă valoarea lui *D* este diferită de 100 sau dacă, simultan, valoarea lui *D* este 100 şi valoarea lui *a* este mai mare ca 0. (F) în caz contrar.
`(1 - sin(x)) < 0.001`	(A) dacă diferenţa dintre 1 şi sinusul valorii din variabila *x* este mai mică decât 0,001. (F) în caz contrar

Loop (Repetă)

Această instrucţiune ciclică realizează repetarea unui bloc de instrucţiuni (numit corpul instrucţiunii) cât timp o anumită condiţie este adevărată (până când devine falsă). În RAPTOR blocul *Loop (Repetă)* poate fi folosit în două configuraţii, corpul instrucţiunii fiind poziţionat fie înaintea, fie după condiţie. Diferenţa constă în verificarea sau nu a condiţiei înaintea rulării pentru prima dată a corpului instrucţiunii.

Condiţia de terminare a iteraţiilor poate fi orice condiţie validă şi pentru instrucţiunea de decizie, care poate fi evaluată ca (A)devărat sau (F)als.

Exemple cu cele două tipuri de configuraţii ale instrucţiunii *Loop* sunt prezentate mai jos.



Exemple de instrucţiuni ciclice în RAPTOR

a). Corpul instrucţiunii este plasat după evaluarea condiţiei – ciclu cu test iniţial

b). Corpul instrucţiunii este plasat înaintea evaluării condiţiei– ciclu cu test final

Call (Apel de funcţie)

Această instrucţiune realizează apelul unei funcţii indicate în fereastra de proprietăţi a blocului.

Comentarii

În dreptul fiecărei instrucţiuni dintr-un program se poate introduce un comentariu explicativ, prin clic-dreapta pe simbolul instrucţiunii si alegerea comenzii *Comment*.

Lista cu operaţii şi funcţii matematice predefinite

Operation	Description	Example
+	adunare	3+4 este 7
-	scădere	3-4 este -1
-	negare	-3 este inversul lui 3
*	multiplicare	3*4 este 12
/	împărţire	3/4 este 0.75
^ **	Ridicare la putere	3^4 este 81 2**3 este 8
rem mod	Restul împărţirii operandului din stânga la cel din dreapta	10 rem 3 este 1 10 mod 4 este 2
sqrt	Radical ("square root")	sqrt(4) este 2
log	Logarithm natural	log(e) este 1
abs	Valoarea absolută (modul)	abs(-9) este 9
ceiling	Parte întreagă superioară	ceiling(3.14159) este 4
floor	Parte întreagă	floor(9.82) este 9
sin	Sinus (în radiani)	sin(pi/6) este 0.5
cos	Cosinus (în radiani)	cos(pi/3) este 0.5
tan	Tangenta (în radiani)	tan(pi/4) este 1.0
cot	Cotangenta (în radiani)	cot(pi/4) este 1
arcsin	Arcsinus, rezultatul în radiani	arcsin(0.5) este pi/6
arcos	Arccosinus, rezultatul în radiani	arccos(0.5) este pi/3
arctan	Arctangenta, rezultatul în radiani	arctan(10,3) este 1.2793
arccot	Arccotangenta, rezultatul în radiani	arccot(10,3) este 0.29145
random	Generează o valoare aleatoare în intervalul [0, 1)	random * 100 este o valoare între 0 şi 99.9999
Length_of	Numărul de caractere al unei variabile de tip şir de caractere	sir ← "Buna ziua" Length_of(sir) este 8

Bibliografie

[1] "Introduction to Programming with RAPTOR", by Lt Col Tom Schorsch
[2] "Introduction to Programming with RAPTOR", by Dr. Wayne Brown
[3] "Programming Control Structures", by Dr. Wayne Brown"

Documentele din bibliografie, codul sursă al programului RAPTOR precum şi mai multe informaţii se găsesc pe situl http://raptor.martincarlisle.com/