INTERPRETOR PENTRU MAŞINA DACICC-1 [1)]

DE

T. RUS
Cluj

În lucrare se descrie interpretorul realizat în limbajul maşinii DACICC — 1, pentru operatorii aritmetici, pornindu-se de la limbajul arborilor. Se prezintă de asemenea schema logică a algoritmului acestui interpretor.

§ 1. PROBLEMA AUTOMATIZĂRII PROGRAMĂRII

Problema programării pentru maşinile cifrice de calcul (m.c.c.) în forma ei simplă poate fi formulată ca şi problema descrierii unui proces dat în termenii operaţiilor elementare ale maşinii, sau, cu alte cuvinte, în limba maşinii. Urmează deci că procesul de rezolvare a unei probleme cu ajutorul unei m.c.c. este în prealabil descris în alţi termeni.

În cazul în care această descriere iniţială a procesului de calcul nu este o descriere într-o limbă formală a algoritmului de rezolvare a problemei respective, prin programare apare în primul rînd problema descrierii într-o limbă formală a acestui proces. În cazul în care procesul de calcul este descris într-o limbă formală, problema programării devine problema traducerii scrierii algoritmului din limba formală în limba maşinii.

În etapa actuală a cercetărilor în domeniul programării pentru maşinile cifrice de calcul, un rol deosebit revine problemei traducerii automate (cu ajutorul maşinii) a algoritmului de rezolvare a unei probleme, scris într-o limbă formală, în limba maşinii, sau în program. Cu alte cuvinte, este din ce în ce mai actuală rezolvarea problemei automatizării programării.

După cum se arată în lucrarea [1], problema automatizării programării se poate formula sub forma următoarelor două probleme :

A) Problema construirii unei limbi formale comode pentru descrierea algoritmilor.

[1)] Lucrarea prezentată la „Colocviul de teoria funcţiilor convexe cu aplicaţii la calculul numeric", ţinut la Cluj în 1—5 iulie 1965.

ST. CERC. MAT., TOM. 18, NR. 6, P. 799—815 BUCUREŞTI, 1966

B) Problema construirii unui algoritm universal, care realizează traducerea oricărui algoritm scris în limba formală construită prin rezolvarea problemei A), în limba unei maşini cifrice de calcul.

Oricare ar fi rezolvarea pe care am adopta-o pentru problema A), adică pentru orice limbă formală aleasă pentru descrierea algoritmilor, se pot indica două moduri de a soluţiona problema B). Unul dintre acestea constă din aşa-numita metodă a interpretării, iar celălalt din aşa-numita metodă a compilării.

În metoda interpretării, algoritmul descris în limba formală introdus în maşină serveşte ca informaţie pentru îndeplinirea acestuia. În acest scop lucrează un anumit program special, numit interpretor, care analizează fiecare unitate elementară de informaţie a algoritmului, asigurînd îndeplinirea operaţiei descrisă de aceasta, precum şi trecerea la o nouă unitate elementară de informaţie.

În metoda compilării, algoritmul descris în limba formală, introdus în maşină, serveşte ca informaţie pentru stabilirea programului algoritmului respectiv în limba maşinii. Acest program executat la maşină asigură îndeplinirea algoritmului respectiv. În scopul realizării traducerii lucrează un anumit program, numit translator, care realizează traducerea algoritmului din limba formală iniţială în limba maşinii.

Pentru eliminarea unor insuficienţe ale ambelor metode, insuficienţe bine precizate în [1], se preconizează utilizarea unei metode combinate, care constă în traducerea descrierii iniţiale a algoritmului din limba formală într-o oarecare limbă intermediară, adaptată specificului limbii unei maşini date. Traducerea obţinută se consideră drept informaţie pentru un interpretor.

În cele ce urmează vom avea în vedere descrierea algoritmului unui interpretor în ipoteza că algoritmul de rezolvare a problemei este deja descris în limbajul arborilor, după cum vom vedea mai jos.

§ 2. LIMBAJUL ARBORILOR

Să considerăm o mulţime M de obiecte matematice, determinată prin apartenenţa elementelor ei la diferite tipuri de elemente matematice cum sînt constantele, variabilele, relaţiile, operaţiile etc.

Mulţimea simbolurilor care notează elementele lui M o notăm cu A. În continuare prin N notăm mulţimea numerelor întregi şi pozitive.

Considerăm funcţia δ definită pe A şi cu valori în N, $\delta: A \to N$, pentru care presupunem că mulţimea valorilor ei este finită, şi anume o considerăm ca fiind numerele $0, 1, \ldots, p$.

Spunem că elementele n_1 şi n_2, $n_1, n_2 \in A$ sînt echivalente dacă $\delta(n_1) = \delta(n_2)$.

În felul acesta mulţimea A este împărţită de către funcţia δ în clasele de echivalenţă $A_1, A_2, \ldots, A_p$. În clasa de echivalenţă A_i intră elementele $n \in A$ pentru care $\delta(n) = i$. Elementele lui A_i se vor numi în continuare operaţii i-nare ale mulţimii M. Mulţimea A_i se consideră

descompusă în trei submulţimi disjuncte A_i^1, A_i^2, A_i^3, care corespund tipurilor de operaţii: legături, functori şi relaţii.

Pentru cazul $i = 0$, elementele lui A_0 se numesc operaţii cu 0-argumente şi sînt de fapt constantele, variabilele, şi alte obiecte matematice ce diferă de tipul operaţie şi care intră în mulţimea M.

Elementele de bază ale limbajului arborilor sînt aşa-numiţii arbori marcaţi. Aceştia se definesc în felul următor:

DEFINIŢIA 1. Se numeşte arbore marcat un graf conex şi orientat care nu conţine cicluri şi care satisface condiţiile:

1. Există un singur nod în care nu intră nici o săgeată. Acesta se va numi rădăcina arborelui.

2. Oricare nod al arborelui este marcat de un element al mulţimii A după următoarea regulă: dacă din nodul respectiv pleacă i săgeţi, atunci el este marcat de un element al clasei A_i, $i = 0, 1, \ldots, p$. Nodurile marcate de elementele clasei A_0 se vor numi vîrfuri.

3. În fiecare nod al arborelui intră cel mult o săgeată.

Vom nota în continuare arborii cu litera a (cu indici dacă este cazul), iar nodurile cu (n), unde n este simbolul care marchează nodul respectiv.

În mulţimea arborilor astfel definiţi vom introduce acum o operaţie pe care o numim operaţie de compunere. Ea se defineşte în felul următor: fie arborii $a_1, a_2, \ldots, a_p$ daţi şi fie elementul $n \in A_p$. Prin compunerea arborilor $a_1, a_2, \ldots, a_p$ cu ajutorul elementului $n \in A_p$ înţelegem arborele a, a cărei rădăcină este marcată de elementul n şi din care pleacă p săgeţi, care intră respectiv în rădăcinile arborilor $a_1, a_2, \ldots, a_p$. Notăm acest arbore prin $a = ((n), a_1, a_2, \ldots, a_p)$.

DEFINIŢIA 2. Spunem că două noduri oarecare (n_p) şi (n_q) ale unui arbore a sînt legate printr-un drum dacă există în arborele a o mulţime de noduri $\{(n_0), (n_1), \ldots, (n_l)\}$ care satisface condiţiile:

1. $(n_p) = (n_0)$, $(n_q) = (n_l)$.

2. $(n_i) \to (n_{i+1})$ sau $(n_{i+1}) \to (n_i)$, $i = 0, 1, \ldots, l-1$.

Numărul l va fi numit lungimea drumului dintre nodurile (n_p) şi (n_q) şi va fi notat cu $l = l[(n_p), (n_q)]$.

DEFINIŢIA 3. Se numeşte subarbore al unui arbore a o mulţime de noduri $\{(n_1), (n_2), \ldots, (n_p)\}$ ale arborelui a în care există un singur nod (n') cu proprietatea că este legat de toate celelalte noduri ale mulţimii prin drumuri de lungime $l \geqslant 1$. Nodul (n') se va numi rădăcina subarborelui respectiv.

Subarborii unui arbore a de rădăcină (n) pentru care avem $l[(n), (n')] = 1$ se vor numi subarbori principali ai lui a.

DEFINIŢIA 4. Spunem că nodurile $(n_1), (n_2), \ldots, (n_p)$ ale unui arbore a formează nivelul k al acestuia, dacă, (n) fiind rădăcina arborelui a, avem $l[(n), (n_i)] = k$, $i = 1, 2, \ldots, p$, şi pentru orice nod (n') al arborelui a care nu aparţine mulţimii $\{(n_1), (n_2), \ldots, (n_p)\}$ avem $l[(n), (n')] \neq k$.

În afara noţiunilor deja definite, se mai pot defini şi alte noţiuni legate de arbori şi care fac din aceştia elemente cu care se poate opera uşor. Dintre acestea mai amintim doar noţiunea de egalitate a doi arbori.

DEFINIȚIA 5. Spunem că doi arbori a_1 și a_2 sînt egali dacă sînt îndeplinite următoarele condiții :

1. Numărul nivelelor arborelui a_1 este egal cu numărul nivelelor arborelui a_2.

2. Numărul nodurilor de pe nivelul k al arborelui a_1 este egal cu numărul nodurilor de pe nivelul k al arborelui a_2, oricare ar fi $0 \leqslant k \leqslant K$, K fiind numărul nivelelor celor doi arbori.

3. Dacă $n_1, n_2, \ldots, n_p$ sînt simbolurile care marchează nodurile $(\ )_1, (\ )_2, \ldots, (\ )_p$, de pe nivelul k al arborelui a_1 și dacă $n'_1, n'_2, \ldots, n'_p$ sînt simbolurile care marchează nodurile $(\ )'_1, (\ )'_2, \ldots, (\ )'_p$ de pe nivelul k al arborelui a_2, ordinea în care sînt luate nodurile pe nivelul k în cei doi arbori fiind aceeași, atunci $n_i = n'_i$, $i = 1, 2, \ldots, p$, egalitatea fiind aici înțeleasă în sens grafic [2].

Dacă condiția 3 de mai sus se înlocuiește cu condiția ca n_i și n'_i, $i = 1, 2, \ldots, p$, să aparțină aceleiași clase de echivalență a lui A, atunci obținem o altă noțiune de egalitate pe care o numim egalitate slabă.

DEFINIȚIA 6. Spunem că arborii a_1 și a_2 sînt egali (slab egali) pînă la nivelul k dacă subarborii lor, care au și ca rădăcini, rădăcinile arborilor a_1 și a_2 respective, și posedă un număr de k nivele, sînt egali (slab egali).

Pentru ca arborele să devină un obiect cu care să se poată opera ușor cu mașina, se reprezintă cu ajutorul unei matrici, care nu este altceva decît un masiv bidimensional de informație.

Fie pentru aceasta arborele a dat. Reprezentăm acest arbore cu ajutorul unei matrici $H = (\alpha_{ij})$, astfel ca fiecărui nod al arborelui a să-i corespundă o linie a matricei H, și anume în felul următor : dacă $(n) \in a$, atunci acestuia îi corespunde linia $\alpha_{i1}\ \alpha_{i2} \ldots \alpha_{ik+2}$ a matricei H, în care $\alpha_{i1} = \delta(n)$, $\alpha_{i2} = n$, $\alpha_{i3}, \ldots, \alpha_{ik+2}$ fiind un șir de indici care iau valorile $0, 1, \ldots, p$ și care determină poziția nodului (n) în arborele a. Pentru aceasta $\alpha_{i3}, \alpha_{i4}, \ldots, \alpha_{ik+2}$ se citește în felul următor :

Dacă $\alpha_{i3} = \alpha_{i4} = \ldots = \alpha_{ip} = 0$, atunci nodul respectiv se găsește pe nivelul $p + 1$ al arborelui a, iar dacă (n_0) este rădăcina arborelui a, atunci $\alpha_{i3}, \alpha_{i4}, \ldots, \alpha_{ik+2}$ indică drumul care leagă pe (n_0) de (n), drum pe care îl putem reprezenta sub forma $(n_0) \to (n_1) \to \ldots \to (n)$, unde (n_i) este nodul de pe nivelul i al arborelui a și anume al $\alpha_{i\,p+i+1}$-lea nod de pe nivelul 1 al subarborelui arborelui a care are ca rădăcină nodul (n_{i-1}) al arborelui a, fiind situat pe nivelul $i - 1$ al acestuia.

Construirea matricei H se face cu ajutorul unei metode recursive [6] care pornește de la linia atașată rădăcinii arborelui a și care este de forma $\delta(n_0)\ n_0\ 00\ldots01$.

Operația de compunere a arborilor se transpune asupra matricilor în felul următor : fie arborii $a_1, a_2, \ldots, a_p$ și fie $n \in A$. Considerăm arborele $a = ((n), a_1, a_2, \ldots, a_p)$ compus din $a_1, a_2, \ldots, a_p$ cu ajutorul lui n. Dacă $H_1, H_2, \ldots, H_p$, sînt matricile care reprezintă arborii $a_1, a_2, \ldots, a_p$, atunci în lucrarea [6] se construiesc algoritmii C_1 și C_2 cu proprietățile că

$$C_1(n;\ H_1,\ H_2, \ldots, H_p) = H$$

$$C_2(H) = n;\ H_1,\ H_2, \ldots, H_p.$$

Limbajul arborilor va consta deci din mulțimea $\mathfrak{A}$ a arborilor marcați de elementele mulțimii A în care este definită o operație de compunere. Elementele de bază ale acestui limbaj vor fi formate din arborii formați dintr-un singur nod, marcat de elemente ale clasei A_0.

După cum se arată în lucrarea [4], pentru orice cuvînt bine construit în limba formală construită în [3] se poate construi un algoritm care transformă acest cuvînt într-un arbore. În lucrarea [5] se definesc patru operații asupra arborilor și se descriu doi algoritmi în această limbă cu ajutorul operațiilor definite, algoritmi care rezolvă unele probleme nearitmetice care pot fi formulate (formalizate) în limbajul arborilor. Ca și un caz particular pentru acești algoritmi este indicat și algoritmul unui interpretor. Lucrarea de față va cuprinde descrierea amănunțită a acestui interpretor, care a fost realizat (tradus) în limba mașinii DACICC—1.

§ 3. ALGORITMUL UNUI INTERPRETOR AL FORMULELOR ARITMETICE

Să considerăm formula $y = f(x)$ și să presupunem că funcția $f(x)$ poate fi reprezentată cu ajutorul unui arbore. După cum am mai spus, în lucrarea [6] se indică o metodă de codificare a unui arbore cu ajutorul unei matrici. Pentru clarificarea expunerii să considerăm exemplul următor :

$$y = \frac{(x+1)\ (x-1)}{\sqrt{x^2+1}}.$$

Arborele care reprezintă această funcție este reprezentat în figura 1. Matricea care codifică arborele din figura 1 este următoarea :

2	:	0	0	0	0	1
2	×	0	0	0	1	1
1	$\sqrt{\ }$	0	0	0	1	2
2	−	0	0	1	1	1
2	+	0	0	1	1	2
2	+	0	0	1	2	1
0	1	0	1	1	1	1
0	x	0	1	1	1	2
0	1	0	1	1	2	1
0	x	0	1	1	2	2
0	1	0	1	2	1	1
2	π	0	1	2	1	2
0	x	1	2	1	2	1
0	2	1	2	1	2	2

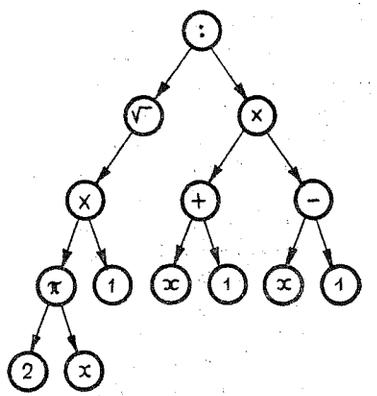


Fig. 1

În lucrarea [6], prima coloană a acestei matrici s-a numit vector grad, a doua coloană s-a numit vector nod, iar celelalte coloane formează așa-numita matrice de indici.

Pentru scopul pe care îl urmărim în continuare, este suficient să considerăm numai vectorul nod şi vectorul grad al matricei care codifică un arbore oarecare a. Să notăm atunci vectorul nod prin $N = (n_1, n_2, \ldots, n_l)$ iar vectorul grad prin $G = (g_1, g_2, \ldots, g_l)$.

Se vede imediat că, dat fiind arborele a, vectorii N şi G se obţin în felul următor: n_1 este simbolul care marchează rădăcina arborelui a, iar $g_1 = \delta(n_1)$; n_2 este simbolul care marchează primul nod de pe nivelul unu al arborelui a, numărat de la dreapta spre stînga, iar $g_2 = \delta(n_2)$; n_3 este următorul nod de pe nivelul unu numărat în acelaşi fel, $g_3 = \delta(n_3)$, ş.a.m.d. pînă se epuizează toate nodurile de pe nivelul unu. Procesul continuă pe urmă prin considerarea următorului nivel al arborelui a, ş.a.m.d. pînă ce se epuizează toate nivelele arborelui a.

Facem acum observaţia că valoarea funcţiei δ este determinată şi de arborele în care apare nodul (n) care intră prin n în domeniul funcţiei δ. De exemplu, pentru simbolul $+$ admitem că avem $2 \leqslant \delta(+) < \infty$.

În general, admitem că avem $r \leqslant \delta(n) < \infty$.

TEOREMĂ. *Oricare ar fi arborele a, el este complet caracterizat de vectorii N şi G.*

Demonstraţia acestei teoreme revine la a arăta că dat fiind arborele a, vectorii N şi G sînt unic determinaţi, şi invers, dat fiind vectorii N şi G, arborele a este unic determinat. Prima parte a acestei afirmaţii urmează imediat din construcţia celor doi vectori. Pentru demonstrarea celei de-a doua părţi vom descrie algoritmul de construire a arborelui a din vectorii N şi G.

Primul pas al acestui algoritm constă în construirea rădăcinii arborelui a, care va fi marcată de simbolul n_1, şi din care vor pleca $\delta(n_1)$ săgeţi (fig. 2).

Al doilea pas al algoritmului constă din construirea rădăcinilor subarborilor principali ai lui a. Să numerotăm aceşti subarbori cu indicii $1, 2, \ldots, \delta(n_1)$ şi să-i notăm cu $a_1, a_2, \ldots, a_{\delta(n_1)}$.

Rădăcina lui a_1 va fi marcată de simbolul n_{1+1} deci de n_2 iar numărul săgeţilor care vor pleca din ea va fi $\delta(n_2)$. În general, rădăcina lui a_i, $1 \leqslant i \leqslant \delta(n_1)$ va fi marcată de simbolul n_{i+1}, iar numărul săgeţilor care pleacă din acest nod va fi $\delta(n_{i+1})$, $i = 1, 2, \ldots, \delta(n_1)$.

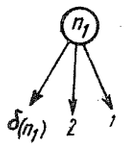


Fig. 2

Cu nodurile deja construite se procedează în mod analog ca şi cu nodul marcat de n_1, considerîndu-le în ordine de la dreapta la stînga. Se vede uşor că după un număr finit de aplicări ale paşilor unu şi doi ai algoritmului descris mai sus, se vor epuiza toate componentele vectorului N şi arborele obţinut coincide cu arborele a iniţial.

Înainte de a trece la descrierea algoritmului interpretorului să construim algoritmul de descompunere al vectorilor nod şi grad ai unui arbore a în rădăcina acestui arbore şi în vectorii nod şi grad ai subarborilor principali ai lui a. Acest algoritm este de altfel un caz particular al algoritmului C_2 despre care vorbeam mai sus.

Fie deci arborele a şi fie $N = (n_1, n_2, \ldots, n_l)$ şi $G = (g_1, g_2, \ldots, g_l)$ vectorii nod şi grad ai acestuia.

Primul pas al algoritmului constă în marcarea elementelor n_1 şi g_1 care sînt respectiv simbolul ce marchează rădăcina arborelui a şi numărul săgeţilor care pleacă din această rădăcină.

La pasul al doilea al algoritmului se construieşte vectorul nod şi vectorul grad al subarborelui principal al lui a care are ca rădăcină cel mai de la dreapta nod de pe nivelul 1 al arborelui a. Să notăm cu N_1 şi G_1 aceşti vectori. Pentru construirea lor să mai considerăm numerele variabile h şi h' care pentru început presupunem că sînt $h = 1$ şi $h' = 1$.

Elementele lui N_1 şi G_1 se vor nota în ordine cu n_i^1 şi g_i^1 şi se obţin astfel:

$$n_1^1 = n_{h+1}; \quad g_1^1 = g_{h+1}.$$

Se calculează apoi valorile lui h şi h' după formulele:

$$g = \sum_{i=1}^{h'} g_{h+i},$$

$$h' = g,$$

$$h = 1 + \sum_{i=1}^{h'} g_i.$$

Elementele următoare ale vectorilor N_1 şi G_1 vor fi atunci

$$n_2^1 = n_{h+1}, n_3^1 = n_{h+2}, \ldots, n_s^1 = n_{h+h'}$$

$$g_2^1 = g_{h+1}, g_3^1 = g_{h+2}, \ldots, g_s^1 = g_{h+h'}.$$

Acest proces continuă atîta timp cît numărul $g = \sum_{i=1}^{h'} g_{h+i}$ este diferit de zero.

Dacă $g = 0$, procesul se opreşte, iar componentele $n_1^1, n_2^1, \ldots, n_s^1$ şi $g_1^1, g_2^1, \ldots, g_s^1$ obţinute reprezintă vectorii nod şi grad N_1 şi G_1 căutaţi. Această afirmaţie rezultă din următoarele două observaţii:

Observaţia 1. Deoarece h iniţial este 1 şi din numerotarea nodurilor arborelui a înseamnă că n_{h+1} este simbolul care marchează rădăcina primului subarbore principal al lui a. Analog pentru g_{h+1}.

Observaţia 2. Numărul $h = 1 + \sum_{i=1}^{h'} g_i$ reprezintă numărul nodurilor lăsate la o parte, pe nivelele superioare nivelului pe care se găseşte nodul marcat de n_{h+1}, iar numărul $g = \sum_{i=1}^{h'} g_{h+i}$ reprezintă numărul săgeţilor care pleacă din nodurile pe care deja le-am parcurs în procesul construirii subarborelui respectiv.

Procesul descris mai sus se repetă pentru valorile iniţiale ale lui h şi $h' = 1, 2, \ldots, g_1$, obţinînd respectiv vectorii N_1, G_1; N_2, G_2; ...; N_{g1}, G_{g1}, care reprezintă de fapt subarborii principali $a_1, a_2, \ldots, a_{g1}$ ai arborelui a.

Cu ajutorul algoritmului descris mai sus, vom trece acum la descrierea algoritmului interpretorului. Fie pentru aceasta N_1 și G_1 vectorii nod și grad ai arborelui care reprezintă formula pe care dorim s-o calculăm. Să notăm cu n numărul nivelelor acestui arbore și cu l numărul nodurilor sale.

Algoritmul interpretorului va fi compus din trei pași. La primul pas se descompun vectorii N_1 și G_1 care reprezintă arborele a în vectorii N_1^1, G_1^1; N_1^2, G_1^2; ... ; $N_1^{q_1}$, $G_1^{q_1}$, care reprezintă subarborii principali ai arborelui a, și în elementele n_1 și g_1 care reprezintă simbolul care marchează rădăcina lui a și numărul săgeților care pleacă din această rădăcină. n_1 și g_1 vor servi în continuare ca prime elemente ale unor vectori $'N_1$ și $'G_1$, ce vor reprezenta vectorul nod și vectorul grad al arborelui a' obținut din a în urma unei analize făcută de algoritmul de mai sus asupra vectorilor N_1 și G_1.

La al doilea pas al algoritmului se verifică dacă vectorii N_1^i, G_1^i obținuți la primul pas reprezintă un arbore cu două nivele. Dacă, de exemplu, N_1^i, G_1^i reprezintă un astfel de arbore, atunci avem :

$$N_1^i = (n_1^i,\ n_2^i, \ldots, n_{l_i}^i)$$

$$G_1^i = (g_1^i,\ g_2^i, \ldots, g_{l_i}^i)$$

$$g_1^i = l_i - 1.$$

În acest caz operația n_1^i se execută asupra mărimilor $n_2^i, n_3^i, \ldots, n_{l_i}^i$ și se trece rezultatul sub forma vectorului nod și a vectorului grad al unui arbore cu un singur nivel, în dreapta simbolului n_1 și respectiv g_1 pe care le-am păstrat la început.

Dacă nici unul dintre vectorii N_1^i, G_1^i nu îndeplinește condiția $g_1^i = l_i - 1$, se trece la primul pas al algoritmului pentru vectorii N_1^i, G_1^i, $i = 1, 2, \ldots, g_i$, ale căror elemente n_1^i, g_1^i se așează în dreapta șirului construit, ale cărui prime elemente erau n_1 și g_1.

Acest proces continuă de n ori după care se face $n := n - 1$.

Pasul al treilea al algoritmului este acela corespunzător operației de calcul pe care credem că nu este necesar să-l descriem. Subliniem doar că dacă acest pas se înlocuiește cu unul corespunzător algoritmului de construire a comenzilor operațiilor, interpretorul devine translator.

După ce s-a făcut $n := n - 1$, se trece la primul pas al algoritmului pentru vectorii $'N_1$, $'G_1$, cu care se procedează analog ca și cu N_1, G_1.

Procesul se oprește atunci cînd avem $n = 1$. Rezultatul apare sub forma vectorului nod și a vectorului grad al unui arbore format dintr-un singur nod.

Pentru ilustrarea algoritmului descris mai sus să considerăm următorul exemplu : să se calculeze cu ajutorul interpretorului valoarea expresiei

$$E = \frac{(a+b)(a-b)}{2ab} + \frac{b^2 - a^2}{2ab} + \frac{a-b}{ab}.$$

Arborele care reprezintă această expresie este prezentat în figura 3. Vectorii nod și grad corespunzători arborelui din figura 3 sînt

$N_1 = (+, :, :, :, -, \times, -, \times, \times, \times, a, b, a, b, \pi, \pi, a, b,$
$2, -, +, a, b, 2, a, 2, b, 2, a, b, a, b)$

$G_1 = (3, 2, 2, 2, 2, 2, 2, 3, 2, 3, 0, 0, 0, 0, 2, 2, 0, 0, 0,$
$2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).$

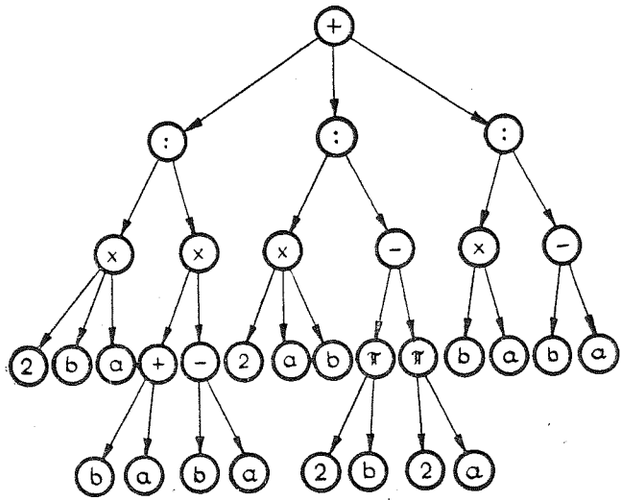


Fig. 3

Am notat cu π operația de ridicare la putere.

La primul pas, algoritmul interpretorului aplicat lui N_1 și G_1 va construi următorii vectori :

$'N_1 = (+,$

$'G_1 = (3,$

și

$N_1^1 = (:, -, \times, a, b, a, b,)$

$G_1^1 = (2, 2, 2, 0, 0, 0, 0)$

$N_1^2 = (:, -, \times, \pi, \pi, a, b, 2, a, 2, b, 2)$

$G_1^2 = (2, 2, 3, 2, 2, 0, 0, 0, 0, 0, 0, 0)$

$N_1^2 = (:, \times, \times, -, +, a, b, 2, a, b, a, b)$

$G_1^2 = (2, 2, 3, 2, 2, 0, 0, 0, 0, 0, 0, 0).$

Deoarece pentru nici unul dintre aceşti vectori condiţia $g_1^i = l_i - 1$ nu este îndeplinită, cu vectorii N_1^i, G_1^i, pentru $i = 1, 2, 3$, se procedează ca şi cu vectorii N_1 şi G_1. În felul acesta avem :

$$'N_1 = (+, :, :, :,$$
$$'G_1 = (3, 2, 2, 2,$$
$$N_1^{11} = (-, a, b)$$
$$G_1^{11} = (2, 0, 0,)$$
$$N_1^{12} = (+, a, b,)$$
$$G_1^{12} = (2, 0, 0)$$
$$N_1^{21} = (-, \pi, \pi, a, 2, b, 2)$$
$$G_1^{21} = (2, 2, 2, 0, 0, 0, 0)$$
$$N_1^{22} = (\times, a, b, 2,)$$
$$G_1^{22} = (3, 0, 0, 0)$$
$$N_1^{31} = (\times, -, +, a, b, a, b)$$
$$G_1^{31} = (2, 2, 2, 0, 0, 0, 0)$$
$$N_1^{32} = (\times, a, b, 2)$$
$$G_1^{32} = (3, 0, 0, 0).$$

La următorul pas al algoritmului se constată că există vectori pentru care $g_1^i = l_i - 1$. Trecînd la pasul al treilea al algoritmului pentru aceşti vectori, adică la pasul de calcul, avem

$$'N_1 = (+, :, :, :, r_1, r_2, -, r_3, x, r_4,$$
$$'G_1 = (3, 2, 2, 2, 0, 0, 2, 0, 2, 0,$$

unde $r_1 = a - b$, $r_2 = a + b$, $r_3 = 2ab$, $r_4 = 2\,ab$. Vectorii N_1^{21}, G_1^{21}, N_1^{31}, G_1^{31} se descompun apoi cu ajutorul primului pas al algoritmului în :

$$N_1^{211} = (\pi, a, 2)$$
$$G_1^{211} = (2, 0, 0)$$
$$N_1^{212} = (\pi, b, 2)$$
$$G_1^{212} = (2, 0, 0)$$
$$N_1^{311} = (-, a, b),$$
$$G_1^{311} = (2, 0, 0)$$
$$N_1^{312} = (+, a, b)$$
$$G_1^{312} = (2, 0, 0).$$

La următorul pas al aplicării algoritmului, deoarece toţi vectorii obişnuiţi la pasul precedent satisfac condiţia $g_1^i = l_i - 1$, se vor executa numai operaţii de calcul. În urma acestora vom avea

$$'N_1 = (+, :, :, :, r_1, r_2, -, r_3, \times, r_4, r_5, r_6, r_7, r_8)$$

$$'G_1 = (3, 2, 2, 2, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0).$$

Arborele reprezentat de vectorii $'N_1$ şi $'G_1$ este prezentat în figura 4.

Procedînd cu $'N_1$ şi $'G_1$ în mod absolut analog ca şi cu N_1 şi G_1, după aplicarea algoritmului de mai sus de încă trei ori se obţine soluţia problemei.

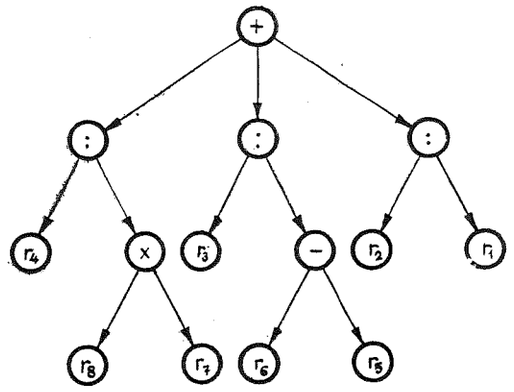


Fig. 4

Pentru descrierea schemei logice a programului algoritmului descris mai sus, ne vom folosi de noţiunea de algoritm adresă [7] precum şi de următoarea distribuţie a memoriei :

Notăm cu $\alpha_1, \alpha_2, \ldots, \alpha_l$ şi $\beta_1, \beta_2, \ldots, \beta_l$ adresele memoriei operative în care se păstrează vectorii N_1 şi G_1. Cu $\alpha_1^1, \alpha_2^1, \ldots, \alpha_{l_1}^1$ şi $\beta_1^1, \beta_2^1, \ldots, \beta_l^1$ notăm adresele în care se vor păstra vectorii N_1^i, G_1^i precum şi descompunerile lor, iar cu $\alpha_1^2, \alpha_2^2, \ldots, \alpha_l^2$ şi $\beta_1^2, \beta_2^2, \ldots, \beta_l^2$ notăm adresele în care se vor păstra vectorii $'N_1$ şi $'G_1$. În afară de acestea, mai avem adresele $\omega, \omega_1, \omega_2, \rho, \varepsilon_0, \varepsilon_1$ al căror conţinut este următorul :

$$'\omega = n\,;\ '\omega_2 = n\,;\ '\rho = l\,;\ '\varepsilon_0 = 0\,;\ '\varepsilon_1 = 1,$$

unde, după cum am mai spus, n reprezintă numărul nivelelor arborelui a iar l reprezintă numărul nodurilor arborelui a.

Schema logică a algoritmului adresă a interpretorului se compune din trei blocuri. Blocul $\mathcal{J}$, reprezentat în figura 5, are rolul de a verifica dacă este necesară descompunerea vectorilor N_1 şi G_1. El serveşte oarecum drept bloc de conducere a lucrului realizat de celelalte blocuri.

Blocul $\mathcal{D}$ care are schema prezentată în figura 6, are rolul de a executa descompunerea vectorilor N_1 şi G_1.

Blocul $\mathcal{C}$ care are schema prezentată în figura 7, are rolul de a executa calculele respective.

În afara acestor blocuri, deoarece se presupune că şirurile $\{\alpha_i^1, \beta_i^1\}$ şi $\{\alpha_i^2, \beta_i^2\}$ pentru $i = l, 2, \ldots, 1$ se pot păstra în memoria externă a

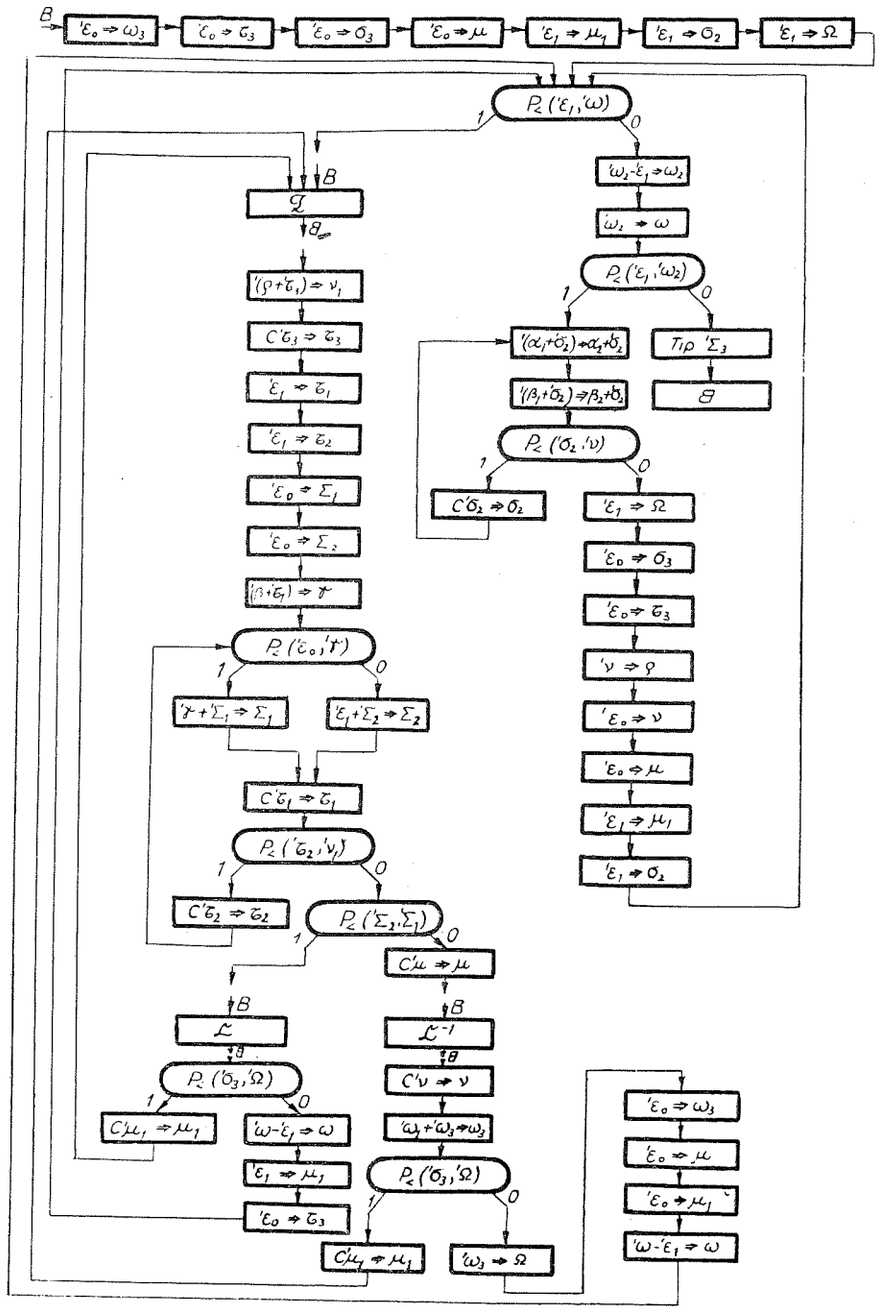


Fig. 5

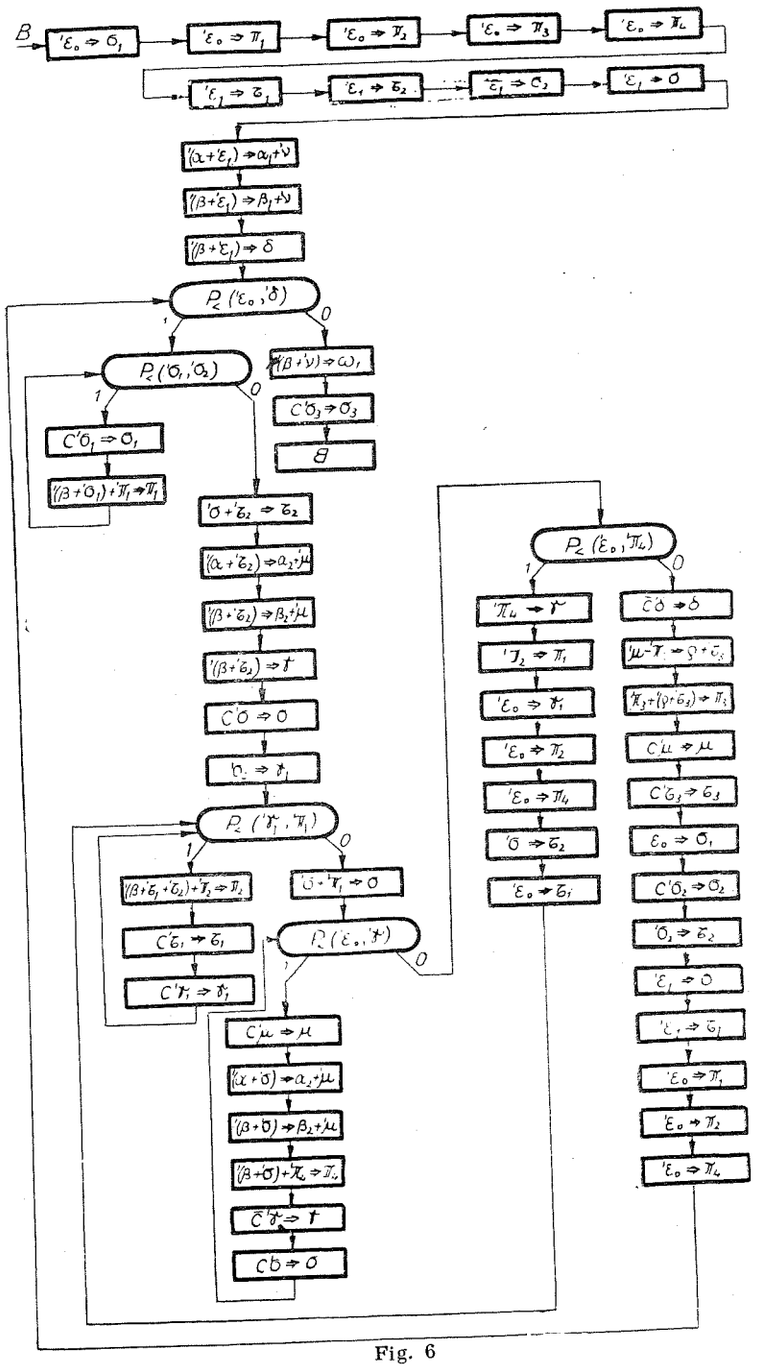


Fig. 6

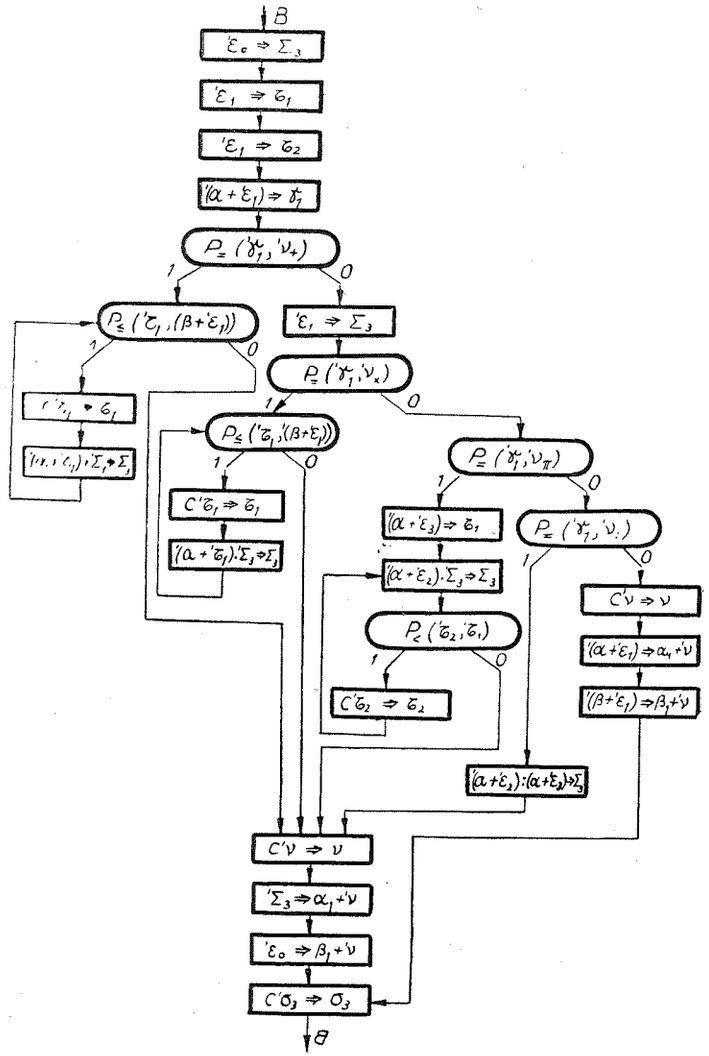


Fig. 7

maşinii (după cum se procedează la DACICC—1), în figura 8 este prezentată schema logică a algoritmului adresă care realizează transporturile necesare.

Pentru uşurarea descifrării acestor scheme adăugăm că am notat mai sus prin $\alpha_j + i$, $\beta_j + i$, $j = 1, 2$; $i = 1, 2, \ldots, 1$, adresele şirurilor $\{\alpha_i^1, \beta_i^1\}$ şi $\{\alpha_i^2, \beta_i^2\}$ iar prin $\alpha + i$ şi $\beta + i$ am notat adresele şirului $\{\alpha_i, \beta_i\}$.

Programul interpretorului a fost realizat în limba maşinii DACICC—1 şi ocupă un loc de 400 cuvinte adresă în memoria operativă. Experienţele făcute cu acesta au arătat că în cazurile în care se cere să se calculeze valorile unor expresii aritmetice complicate se obţine economie de timp şi de muncă destul de mare. Spre exemplu valoarea expresiei de mai sus se calculează în 6 secunde. S-a putut observa de asemenea că un astfel de program devine foarte util dacă se foloseşte ca subprogram al unei biblioteci.

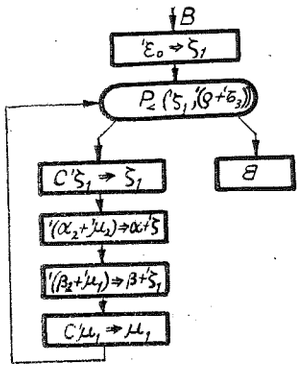


Fig. 8

§4. EXTINDERI ALE ALGORITMULUI INTERPRETORULUI

După cum deja am observat, interpretorul operatorilor aritmetici devine util şi comod de folosit ca şi un subprogram al unei biblioteci. În cele ce urmează vom nota prin $\mathcal{J}$ acest interpretor, şi vom construi schemele logice ale unor procese ciclice în care se poate folosi cu succes acest program.

Considerăm pentru început următoarea problemă: să se calculeze valoarea funcţiei $f(x_1, x_2, \ldots, x_k)$ pentru valorile $(x_1^j, x_2^j, \ldots, x_k^j)$, $j = 1, 2, \ldots, n$, în ipoteza că funcţia f se poate reprezenta sub forma unui arbore.

Să notăm cu N_1 şi G_1 vectorul nod şi vectorul grad al reprezentării matriciale a arborelui care reprezintă funcţia f. De asemenea, notăm prin $C_{x_1}, C_{x_2}, \ldots, C_{x_k}$ nişte coduri pe care le ataşăm variabilelor $x_1, x_2, \ldots, x_k$. Presupunem că N_1 şi G_1 se păstrează în memoria externă într-un loc standard, de exemplu în adresele $\alpha_0 + i$, $\beta_0 + i$. Pentru descrierea algoritmului adresă care rezolvă problema de mai sus cu ajutorul interpretorului vom mai folosi următoarele adrese: $'\varepsilon_1 = 1$, $'\varepsilon_0 = 0$, $'\nu_1 = n$, $'\nu_2 = l$, $'\nu_3 = k$, $'\eta_1 = i$, $'\eta_2 = j$, $'\eta_3 = k$, $'c_i = C_{x_i}$, $i = 1, 2, \ldots, k$. Cu aceste notaţii, schema logică a algoritmului adresă care rezolvă problema pusă mai sus este prezentată în figura 9.

Programul acestui algoritm în limba maşinii DACICC—1 nu ocupă decît un număr de 40 de cuvinte adresă.

O altă problemă, în a cărei rezolvare se poate aplica cu succes interpretorul $\mathcal{J}$, este următoarea: să se rezolve ecuaţia $F(X) = 0$, cu ajutorul metodei iteraţiilor. Pentru aceasta se transformă ecuaţia $F(X) = 0$ în ecuaţia echivalentă cu ea $X - \varphi(X) = 0$. Dacă $\varphi(X)$ îndeplineşte anumite condiţii privind convergenţa procesului iterativ, atunci pornind

de la o valoare oarecare X_0 se poate rezolva problema de mai sus cu ajutorul schemei

$$X_{n1} = \varphi(X_n).$$

Procesul continuă pînă cînd diferența a două iterații succesive satisface condiția $|X_{n+1} - X_n| < \varepsilon$, ε fiind un număr determinat de precizia

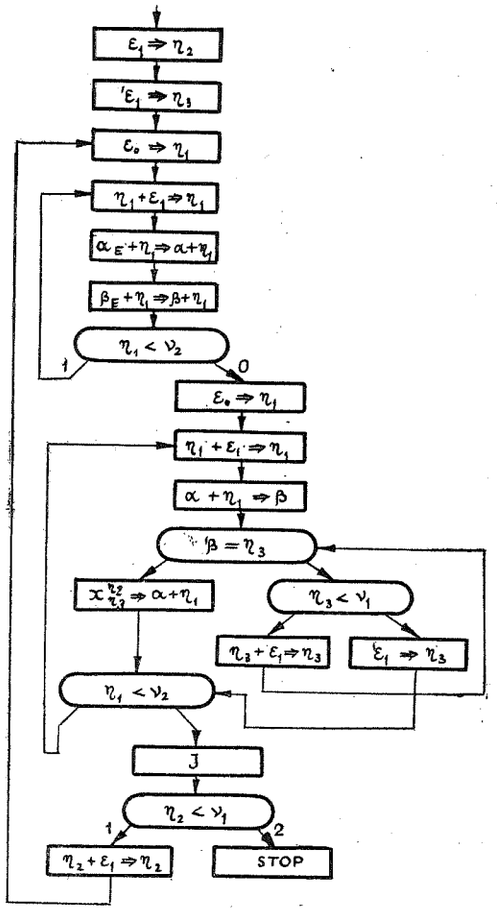


Fig. 9

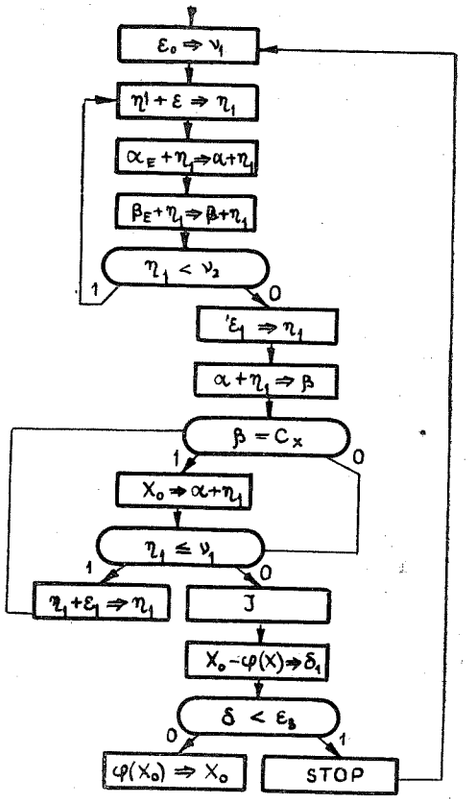


Fig. 10

soluției căutate. În ipoteza că φ se poate reprezenta cu ajutorul unui arbore, ca și la prima problemă, în rezolvarea ei putem folosi interpretorul $\mathcal{J}$. Algoritmul adresă care rezolvă această problemă are schema logică prezentată în figura 10.

În afară de codificările folosite și în celelalte scheme, aici am folosit și codificarea $'\varepsilon_3 = \varepsilon$. De asemenea, operatorul

$$'\beta = C_x$$

poate fi mai complicat sau mai simplu după cum X este o variabilă sau un vector. O aplicație imediată a schemei de mai sus se găsește la rezolvarea sistemelor de ecuații liniare cu ajutorul metodei lui Gauss-Seidel.

Primită la redacție la 20 noiembrie 1965

Institutul de calcul
al Academiei Republicii Socialiste România
— Filiala Cluj

BIBLIOGRAFIE

1. М. Р. Шура-Бура, А. П. Ершов, *Машинные языки и автоматическое программирование.* Тр. четверт. всес. мат. Съезда, Ленинград, 1961, **I**, 243—250 (1963).
2. А. А. Марков. *Теория логорифмов,* Тр. Матем. ин-та им. В. А. Стеклова, **XLII**, Москва, 1964.
3. T. Rus, *Asupra unui sistem formal I.* St. cerc. mat., **15**, *4*, 459—470 (1964).
4. T. Rus, *Asupra unui sistem formal II.* St. cerc. mat., **15**, *5*, 595—615 (1964).
5. T. Rus, *Algoritmi de prelucrare a expresiilor formale cu ajutorul mașinilor cifrice de calcul.* St. cerc. mat., **16**, *9*, 1129—1139 (1964).
6. T. Rus, *Asupra reprezentării matriciale a arborilor.* St. cerc. mat., **16**, *9*, 1179—1185 (1964).
7. Е. Л. Ющенко, *Адриное программирование.* Государственное издательство технической литературы, УССР, Киев, 1963.