Gestionarea preciziei în evaluarea de expresii cu numere foarte mari

Ec. Claudiu Marian ARITON
S.C. Tronic S.R.L., Bucuresti

Obiectivul lucrarii este realizarea de functii pentru efectuarea operatiilor aritmetice cu operanzi având lungimi mai mari decât definirile standard. Se trateaza distinct calcule cu numere foarte mari pentru reprezentarile binara, virgula mobila, zecimal împachetata, zecimal despachetata si pe caractere. Se analizeaza erorile si precizia pe care le determina o anumita lungime a operanzilor în evaluarea de expresii si în efectuarea de calcule complexe specifice algoritmilor analizei numerice. Se structureaza functiile de prelucrare ale calculatorului, se efectueaza teste si se analizeaza performanta acestuia. Se lucreaza si se fac referiri la limbajele: Pascal, C/C++, FORTRAN, COBOL.
Cuvinte cheie: *numere foarte mari, expresii, precizie, algoritmi.*

Algortimi pentru calcule cu numere foarte mari

Reprezentarea numerelor mari. Operanzii se definesc sub forma de liste dublu înlantuite, unde d_1, d_2, …, d_n semnifica cifrele zecimale ale numarului mare (figura 1). Pentru efectuarea calculelor listele se traverseaza de la dreapta la stânga, iar pentru afisare se traverseaza de la stânga la dreapta. Pentru efectuarea operatiilor, listele se traverseaza în ambele sensuri, deoarece lungimea operanzilor si a rezultatelor este necunoscuta.

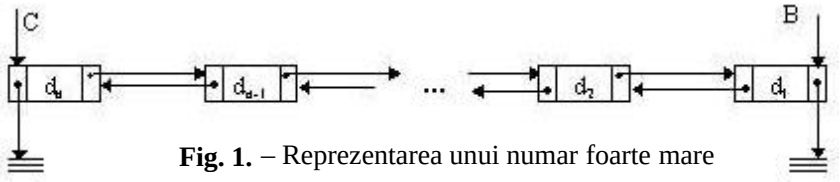

Fig. 1. – Reprezentarea unui numar foarte mare

Algoritmul de adunare consta în adunarea cifrelor de acelasi rang cu transportul de la adunarea de rang mai mic si se obtine o noua cifra a sumei, precum si transportul pentru urmatoarea adunare de rang mai mare. De remarcat ca la prima adunare (cea corespunzatoare rangului unitatilor) transportul anterior (care nu exista, fiind prima adunare) trebuie initializat cu 0 pentru omogenitatea buclei **for**. Merita atentie speciala cazul în care ultimul transport este diferit de 0. În acest caz, apare fenomenul numit uzual *depasire* sau *overflow*.

Algoritmul de scadere este asemanator celui de adunare. Cel mai dificil caz este cel în care are loc scaderea dintr-o cifra mai mica a unei cifre mai mari, pentru un anumit rang. Cazul trebuie tratat asemanator cu *transportul* de cifra de la adunare (*împrumut de la cifra de rang mai mare)*. Exista si aici posibilitatea unei *subdepasiri* sau *underflow*. În cazul utilizarii listelor dublu înlantuite nu mai apar operatiile de depasire si subdepasire, lungimea reprezentarii operanzilor si a rezultatului fiind dinamica în timp (nu se retin cifrele nule din fata numarului).

Algoritmul de înmultire este bazat pe înmultiri polinomiale. Astfel, fiecarui numar i se ataseaza un polinom ai carui coeficienti sunt cifrele numarului. Deci, dupa o înmultire (care comporta o complexitate patratica) si o normalizare a coeficientilor (un fel de "transport" care are o complexitate liniara) se poate obtine polinomul atasat rezultatului, de unde se poate obtine rezultatul.

Algoritmul de împartire se realizeaza prin scaderi succesive. Daca în cazul operatiilor prezentate mai sus se parcurg numerele mari de la dreapta la stânga, în acest caz listele se traverseaza de la stânga la dre-

apta. Din acest motiv, precizia cu care se lucreaza trebuie sa fie fixata (conditia de oprire a algoritmului de împartire). Semnul rezultatului este dat de regula semnelor.

Algoritmul de comparare consta în compararea cifrelor de acelasi rang de la stânga spre dreapta. În caz de egalitate, se trece la compararea cifrelor de rang imediat mai mic. Daca pentru un anumit rang am gasit ca cifra curenta a unuia dintre numere este mai mare decât cifra curenta a celuilalt numar, putem spune ca si între numere exista aceeasi relatie. În cazul în care, pâna la epuizarea cifrelor, nu apare neegalitate, atunci putem spune ca cele doua numere sunt egale. În cazul utilizarii listelor, algoritmul de comparare este identic cu cel dinainte, cu mentiunea ca apare la început un test relativ la numarul de cifre. Daca unul din numere are mai multe cifre decât celalalt, atunci acel numar este cu siguranta mai mare.

Celelalte operatii implementate, *calculul functiei factorial, ridicarea la putere, extragerea radacinii de ordin **n**, calculul constantei lui Euler cu **p** zecimale exacte,* nu folosesc structura interna a datelor, ele apelând la operatiile primitive definite anterior.

Solutia informatica

Numerele mari se vor memora în liste dublu înlantuite (structuri cu alocare dinamica) sub forma unui sir (fiecare cifra zecimala din numar va corespunde unui element al listei).

Avantajele utilizarii structurilor dinamice constau în:

- utilizarea eficienta a memoriei (alocare dinamica);
- timp constant de efectuare a operatiilor fundamentale;
- operatii de inserare si stergere facile prin gestionarea legaturilor.

Dezavantajele utilizarii structurilor dinamice constau în:

- ocuparea unei zone necontigue de memorie;
- operatii suplimentare de alocarare/eliberare a memoriei;
- manipularea legaturilor dintre elemente;
- aparitia fragmentarii memoriei;
- traversarea într-un singur sens (dezavantaj care dispare la listele dublu înlantuite).

Dezavantajele pot fi evitate printr-o manipulare si gestiune optima a spatiului de memorie dinamica.

O alternativa de reprezentare consta în utilizarea structurilor statice (vectori) pentru memorarea numerelor mari, dar care comporta o serie de dezavantaje:

- utilizarea ineficienta a memoriei (alocare statica);
- operatiile de inserare si stergere se realizeaza dificil (presupun operatia de deplasare a elementelor din masiv);
- limita maxima impusa.

Normalizarea operanzilor este operatia de aliniere la dreapta a cifrelor de la partea zecimala a unui operand în functie de ordinul de marime al acestuia si de precizia utilizata. Fie **p** precizia folosita si **op** un operand de lungime **l**, având **n** cifre la partea întreaga si **m** cifre la partea zecimala, unde **l = n+m** (figura 2).

a_n	a_{n-1}	…	a_1	z_1	z_2	…	z_m

Fig. 2. – Reprezentarea unui operand

Algoritmul de normalizare consta în compararea parametrilor **m** si **p**, rezultând cazuri posibile:

1) m<p, caz în care se adauga la dreapta operandului **op** (la partea zecimala) **p-m** zerouri, iar m=p, l' = n+m, deci creste lungimea lui **op** cu **p-m** unitati (l'>l).

Exemplu: Fie p=5 si op=25,678 (n=2, m=3, l=2+3=5). În urma operatiei de normalizare **op** va deveni op=25,67800 (n=2, m=5, l'=2+5=7).

2) *m=p*, caz în care operandul **op** ramâne neschimbat.
3) *m>p*, caz în care operandul **op** se rotunjeste si apoi se trunchiaza

Daca cifra de pe pozitia n+p+1=5, atunci se adauga o unitate la cifra de pe pozitia **n+p** (rotunjire, poate aparea si transport) si apoi se sterg de la dreapta operandului **op** (la partea zecimala) **m-p** cifre (trunchiere), iar m=p, l'=n+m, deci scade lungimea lui **op** cu **m-p** unitati (l'<l).

Exemplu: Fie p=5 si op=27,67899678 (n=2, m=8, l=2+8=10). În urma operatiei de normalizare:

- deoarece cifra de pe pozitia n+p+1=2+5+1=8 este 6 (6>5), rezulta ca cifra de pe pozitia n+p=2+5=7 va creste cu o unitate (9+1=10 avem transport s.a.m.d.), deci **op** va deveni 27,67900678;
- se trunchiaza **op** cu m-p=8-5=3 cifre, deci op=27,67900 (n=2, m=5, l'=2+5=7).

Daca cifra de pe pozitia n+p+1<5, atunci nu se mai efectueaza operatia de rotunjire, ci doar operatia de trunchiere.

Exemplu: Fie p=5 si op=27,678993872 (n=2, m=9, l=2+9=11). În urma operatiei de normalizare:

- deoarece cifra de pe pozitia n+p+1=2+5+1=8 este 3 (3<5), nu se efectueaza operatia de rotunjire;
- se trunchiaza **op** cu m-p=9-5=4 cifre, deci op=27,67899 (n=2, m=5, l'=2+5=7).

Operatia de normalizare afecteaza doar partea zecimala a operandului **op**. În toate cazurile, în urma operatiei de normalizare numarul de cifre de la partea zecimala a operandului **op** va fi **p**, iar lungimea va fi l=n+p.

Calculul constantei lui Euler *e* utilizând numere mari

Fie dezvolatarea în serie Taylor a lui e^x în jurul valorii 0:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots$$

pentru x=1 obtinem dezvoltarea în serie Taylor în jurul valorii 0 lui e:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{n!} + \ldots$$

Pentru a calcula e cu p zecimale exacte vom considera:

$$x_0 = 1;$$

$$x_i = 1 + \sum_{k=1}^{i} \frac{1}{k!} \qquad i = 1,2,3,\ldots$$

$$x_{i+1} = 1 + \sum_{k=1}^{i+1} \frac{1}{k!} \qquad i = 1,2,3,\ldots$$

Pentru a minimiza erorile (realizarea unei singure împartiri), vom aduce la acelasi numitor:

$$x_i = 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{i!}$$

$$x_i = 1 + \frac{i(i-1)\ldots 2 + i(i-1)\ldots 3 + i(i-1)\ldots 4 + \ldots + i(i-1) + i + 1}{i!}$$

$$x_{i+1} = 1 + \frac{(i+1)i\ldots 2 + (i+1)i\ldots 3 + (i+1)i\ldots 4 + \ldots + (i+1)i + (i+1) + 1}{(i+1)!}$$

Se va calcula x_i si x_{i+1} cu **p** zecimale exacte folosind operatiile elementare definite cu numere mari. Oprirea este data de conditia ca cele **p** zecimale exacte ale lui x_i sa fie egale cu cele **p** zecimale exacte ale lui x_{i+1}, caz în care solutia este x_i.

Exemplu: Vom calcula **e** cu 40 zecimale exacte (p=40). Initial e=x_0=1.

i=35, e = 2.71828182845904523536028747135266249775 72
i=36, e = 2.71828182845904523536028747135266249775 72

Precizia e atinsa pentru i= 5. Constanta **e** calculata cu 40 zecimale exacte este:
e=2.7182818284590452353602874713526624977572

Se observa ca pentru i = 35 si i = 36 cele 40 de zecimale coincid si deci **e** va fi:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{35!}$$

Începând cu i =36, raporturile 1/i! nu mai au relevanta pentru cele 40 de zecimale.

Calculatorul GPEX

GPEX (**G**estuinea **P**reciziei în **E**valuarea de e**X**presii) este un software construit în jurul unei biblioteci de functii si proceduri caree lucreaza cu numere foarte mari, realizat în Turbo Pascal 7.0. Biblioteca poate fi reutilizata în alte aplicatii care opereaza cu numere mari. Operatiile aritmetice se executa cu precizia definita de utilizator. Magnitudinea operanzilor este limitata de memoria sistemelor de calcul. Programul are o interfata utilizator asemanatoare programelor Windows (figura 3).

Ecranul calculatorului are trei ferestre principale:

- o fereastra pentru afisarea numerelor mari introduse si a rezultatelor;
- o fereastra pentru afisarea mesajelor de eroare (‘Împartire la zero’, ‘Depasire’, ‘Depasire de memorie’, ‘Radical din numar negativ’ etc);
- o fereastra care contine o paleta de comenzi pentru introducerea numerelor mari si realizarea operatiilor cu acestea.

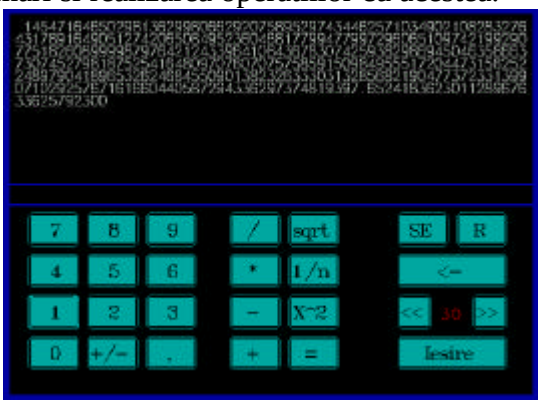

Fig. 3. – Interfata GPEX

Performanta procedurilor

Se studiaza dependenta dintre ordinul de marime al operanzilor si durata de efectuare a operatiilor elementare. Rezultatele sunt prezentate în tabelul 1.

Tabelul 1 – Performanta procedurilor

Ordin de marime		**Repetari**	**Durata efectuarii operatiilor elementare (sec)**			
Operand 1	*Operand 2*		*Adunare*	*Scadere*	*Înmultire*	*Împartire*
10	30	1000	0,17	0,16	7,31	4,62
30	10	1000	0,17	0,16	4,20	8,08
40	30	1000	0,22	0,22	16,48	16,19
60	30	1000	0,28	0,27	22,63	28,62
80	100	1000	0,63	0,38	128,80	99,14

Graficele dependentei dintre ordinul de marime al operanzilor si durata de efectuare a operatiilor elementare sunt prezentate în figurile 4 si 5.

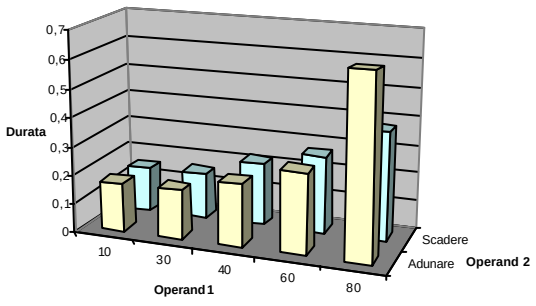

Fig. 4.

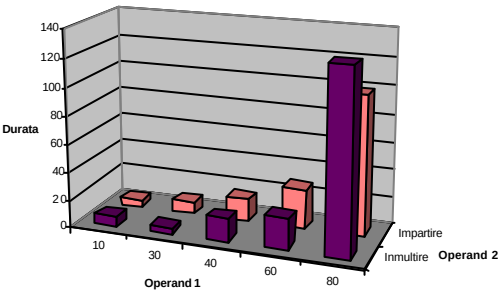

Fig. 5.

Concluzii:

GPEX (**G**estiunea **P**reciziei în **E**valuarea de e**X**presii) este un calculator util cu multe facilitati: precizie aritmetica arbitrara foarte mare; utilizarea algoritmilor

familiari; operatiile aritmetice se realizeaza cifra cu cifra; posibilitatea evaluarilor expresiilor complexe; functii matematice avansate (functia factorial, extragerea radacinii de ordin **n**, calculul constantei lui Euler); abilitati în a opera cu numere mari limitate totusi de memoria disponibila a calculatorului; obtinerea rezultatelor în timp real; optimizarea utilizarii spatiului de memorie disponibila; meniuri interactive usor de utilizat; prezentare generala si prezentarea algoritmilor utilizati; codul sursa complet pentru studiu si utilizare.

GPEX este usor si intuitiv de utilizat si se adreseaza, în special, utilizatorilor finali, pentru: calcule economice complexe; calcule tehnico-stiintifice care cer o precizie foarte mare; calculul functiilor matematice elementare, dar si complexe; biblioteca de functii si proceduri implementata poate fi reutilizata având un caracter mare de generalitate si eficienta. În viitor, în functie de cerintele problemei, se pot construi ierarhii de clase care sa porneasca de la aritmetica cu numere rationale si complexe.

Bibliografie

[Ivan92] Ion Ivan, Romica Adam – "Structuri de date", Editura Inforec, Bucuresti, 1992.

[Knut83] Donald E. Knuth, *Tratat de programarea calculatoarelor. Algoritmi fundamentali (volumul I). Algoritmi seminumerici (volumul III).*, Editura Tehnica, Bucuresti 1983.

[Soca94] Tiberiu Socaciu, *POISSON - Sistem de calcul simbolic*, Comunicare la Colocviul National de Informatica INFOIASI JUNIOR ?94, IASI 7-9 aprilie 1994

[Scal]** Robert-Michel di Scala, *SYPAC: Un system experimental de calcul formel en Pascal*, Thése de 3-iéme cycle, l'Université Scientifique et Médicale de Grenoble.