Recursivitate

Recursivitatea este o metodă pentru definirea şi calcularea funcțiilor ce se autoapelează.

Fie secvența $\{u_n\}$, n>=1. Dacă numărul $k \in N$ şi numerele u_i, i>1 sunt reale sau complexe şi pentru calculul tuturor numerelor se foloseşte relația:

$u_{n+k}=a_1u_{n+k-1}+a_2u_{n+k-2}+\ldots+a_ku_n$,

atunci secvența este o secvență recursivă de ordin k, iar relația este o relație recursivă de ordin k.

O secvență recursivă este caracterizată prin fiecare din termenii săi, prin valoarea de început (de start) şi prin relația recursivă ce foloseşte la determinarea unui termen oarecare.

De exemplu, pentru o progresie geometrică cu $u_1=a$, $u_2=aq$, …,$u_n=a*q^{n-1}$, relația recursivă se scrie $u_{n+1}=q*u_n$, unde k=1, $u_1=a$. Este o relație recursivă de ordin 1.

Fie progresia aritmetică definita prin $u_1=a$, $u_2=a+r$,…,$u_n=a+(n-1)r$,…

Doi termeni consecutivi se găsesc în relația :

$u_{n+1}=u_n+r$ (1)

care nu e recursivă. Pentru a obține o relație recursivă se consideră :

$u_{n+2}=u_{n+1}+r$ (2)

Din (1)-(2) rezultă $u_{n+2}=2u_{n+1}-u_n$, care este o relație recursivă de ordinul 2 cu $u_1=a$, $u_2=a+r$.

Tipuri de funcții recursive

Fie definiția recursivă de calcul a factorialului:

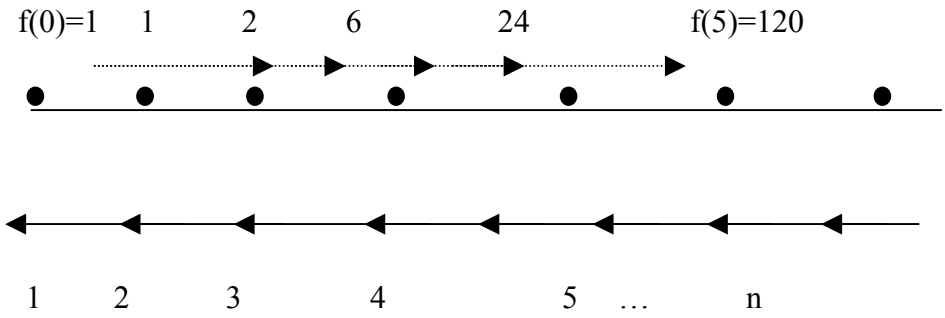
$$f(n) = \begin{cases} 1, & n = 0 \\ n * f(n-1), & n >= 1 \end{cases}$$

Calculul pentru n=5 decurge astfel:

f(5)=5*f(4)=5*(4*f(3))=5*(4*(3*f(2)))=5*(4*(3*(2*f(1))))= 5*(4*(3*(2*1*f(0)))))= 5*(4*(3*(2*(1*1))))=120.

După cum se observă, are loc aplicarea succesivă a definiției, până la identificarea argumentului 0, pentru care f(0)=1, apoi valoarea lui f(0) se înmulțeşte succesiv cu 1,2,3,4,5, adică cu argumentele la care s-au efectuat referiri, în procesul aplicării definiției.

Acest calcul e ilustrat grafic prin:



După ce valoarea lui f(0) a fost calculată, ea va participa la un şir de calcule ce se încheie cu f(5).

Pentru calculul valorilor unor funcţii definite recursiv este necesară, în general, folosirea stivelor şi/sau reprezentarea cu ajutorul arborilor a structurilor de calcul ce permit obţinerea valorilor căutate.

O definiţie recursivă poate fi interpretată ca definirea unei ecuaţii cu diferenţe pentru care valorile direct calculabile reprezintă condiţiile iniţiale sau valorile pe frontieră. De exemplu:

- valoarea factorialului este interpretată astfel :

F(n)=nF(n-1), cu F(0)=1, n e N.

- funcţia Ackermann este definită prin :

$$Ac(m,n)=\begin{cases} n+1 & , m=0 \\ Ac(m-1,1) & , n=0 \\ Ac(m-1,Ac(m,n-1)) & , \text{în rest} \end{cases}$$

- funcţia Fibonacci este dată de :

$$Fib(n)=\begin{cases} 1 & , n=0,1 \\ Fib(n-1)+Fib(n-2) & , n>=2 \end{cases}$$

Exemple de programe rezolvate recursiv: FIB.CPP, FACT.CPP, CMMDC.CPP, SUMACIFR.CPP, BAZA10K.CPP (conţin şi varianta iterativă)

Divide et Impera

Descrierea metodei

Dându-se o funcție care lucrează asupra a n date, tehnica Divide et Impera (Divide şi Stăpâneşte) presupune o împărțire a celor n date în k submulțimi distincte, 1<=k<=n care produc k subprobleme, elementare sau nu. Subproblemele elementare se rezolvă simplu, printr-un algoritm optim, iar cele neelementare se descompun la rândul lor, în subprobleme, până când toate descompunerile au ajuns la probleme elementare care se pot rezolva imediat. Apoi trebuie găsită o metodă de a recombina soluțiile problemelor elementare pentru a obține soluția problemei. Această metodă foloseşte tehnica recursivă, o procedură sau o funcție ce se autoapelează până când se poate rezolva problema.

Fie cele n date memorate în tabloul A(1,n). Procedura generală o vom numi DI şi va fi apelată prin DI(1,n). Funcția DI(p,q) rezolvă subproblema cu intrările p şi q. Procedura, scrisă în limbaj algoritmic, este următoarea:

```
Procedure DI(p,q)
Array a[1,n];
Integer n, p, q, m; 1<=p; q<=n
If  MIC(p,q) then return G(p,q)
                else m=DIVIDE(p,q); p<=q
                        Return (combină (DI(p,m), DI(m+1,q))
Endif
EndDI
```

S-a folosit MIC(p,q) o funcție booleană care determină dacă submulțimile sunt suficient de mici pentru a mai fi sau nu divizate. Dacă da, atunci se apelează funcția G(p,q) care rezolvă subproblema, altfel se apelează funcția DIVIDE care furnizează m=DIVIDE(p,q). Pentru determinarea lui m problema inițială se împarte în două subprobleme A(p,m) şi A(m+1,q) ale căror soluții se găsesc aplicând recursiv DI. Funcția combină soluțiile celor două subprobleme.

Exemplu ce utilizează tehnica Divide et Impera

Găsirea recursivă a maximului dintr-un vector

Se dă un tablou A(1,n) de numere reale şi se cere să se determine maximul elementelor sale folosind tehnica Divide et Impera.

Rezolvare

O problemă I=(a_1, a_2, …,a_n) se desface în două subprobleme I_1= (a_1, a_2, …,$a_{[n/2]}$) şi I_2=($a_{[n/2]+1}$, $a_{[n/2]+2}$, …,a_n).

Notând MAX(I) maximul elementelor pentru problema I, atunci acesta este cel mai mare număr dintre MAX(I_1) şi MAX(I_2). Procedura care va fi elaborată va determina recursiv maximul din mulțimea a_i, a_{i+1}, …,a_j. Cazuri ca i=j, adică mulțimea este formată dintr-un singur element şi i=j-1, adică mulțimea este formată din două elemente vor fi tratate separat, prezentând problema elementară. Dacă mulțimea are mai mult de două elemente, ea va fi divizată în alte două probleme după acelaşi procedeu.

Programul MAX-DI.CPP rezolvă problema.

Problema Turnurilor din Hanoi este implementată în programul HANOI.CPP.