***METODA DESPARTE SI STAPINESTE (DIVIDE ET IMPERA)***

SLIDE1: Metoda de programare DIVIDE ET IMPERA consta in impartirea problemei initiale de dimensiuni [n] in doua sau mai multe probleme de dimensiuni reduse .In general se executa impartirea in doua subprobleme de dimensiuni aproximativ egale si anume [n/2] . Impartirea in subprobleme are loc pana cand dimensiunea acestora devine suficient de mica pentru a fi rezolvate in mod direct(cazul de baza).Dupa rezolvarea celor doua subprobleme se executa faza de combinare a rezultatelor in vederea rezolvarii intregii probleme .

SLIDE2: *Metoda DIVIDE ET IMPERA se poate aplica in rezolvarea unei probleme care indeplineste urmatoarele conditii*

* se poate descompune in ( doua sau mai multe) suprobleme
* aceste suprobleme sunt independente una fata de alta (o subproblema nu se rezolva pe baza alteia si nu se foloseste rezultate celeilalte);
* aceste subprobleme sunt similare cu problema initiala;
* la randul lor subproblemele se pot descompune (daca este necesar) in alte subprobleme mai simple;
* aceste subprobleme simple se pot solutiona imediat prin algoritmul simplificat.

SLIDE3: Deoarece putine probleme indeplinesc conditiile de mai sus ,aplicarea metodei este destul de rara.

Dupa cum sugereaza si numele "desparte si stapaneste "etapele rezolvarii unei probleme (numita problema initiala) in DIVIDE ET IMPERA sunt :

* descompunerea problemei initiale in subprobleme independente ,smilare problemei de baza ,de dimensiuni mai mici ;
* descompunerea treptata a subproblemelor in alte subprobleme din ce in ce mai simple ,pana cand se pot rezolva imediata ,prin algoritmul simplificat ;
* rezolvarea subproblemelor simple ;
* combinarea solutiilor gasite pentru construirea solutiilor subproblemelor de dimensiuni din ce in ce mai mari ;
* combinarea ultimelor solutii determina obtinerea solutiei problemei initiale .

SLIDE4: Metoda DIVIDE ET IMPERA admite o implementare recursiva ,deorece subproblemele sunt similare problemei initiale, dar de dimensiuni mai mici .

Principiul fundamental al recursivitatii este autoapelarea unui subprogram cand acesta este activ;ceea ce se intampla la un nivel ,se intampla la orice nivel ,avand grija sa asiguram conditia de terminare ale apelurilor repetate .Asemanator se intampla si in cazul metodei DIVITE ET IMPERA ; la un anumit nivel sunt doua posibilitati :

s-a ajuns la o (sub)problema simpla ce admite o rezolvare imediata caz in care se rezolva (sub)problema si se revine din apel (la subproblema anterioara,de dimensiuni mai mari);

s-a ajuns la o (sub)problema care nu admite o rezolvare imediata ,caz in care o descompunem in doua sau mai multe subprobleme si pentru fiecare din ele se continua apelurile recursive(ale procedurii sau functiei).

In etapa finala a metodei DIVIDE ET IMPERA se produce combinarea subproblemelor (rezolvate deja) prin secventele de revenire din apelurile recursive.

SLIDE5:PROBLEME

1. PROBLEMA TURNURILOR DIN HANOI

Fie trei tije verticale notate A,B,C .Pe tija A se gasesc asezate n discuri de diametre diferite ,in ordinea crescatoare a diametrelor,privind de sus in jos . Initial ,tijele B si C sunt goale .Sa se afiseze toate mutarile prin care discurile de pe tija A se muta pe tija B , in aceeasi ordine ,folosind ca tija de manevra C si resspectand urmatoarele reguli:

* la fiecare pas se muta un singur disc;
* un disc se poate aseza numai peste un disc cu diametrul mai mare .

program P1;

var n:byte;

procedure hanoi(n:byte;a,b,c:char);

begin

if n=1 then writeln(a,'à',b)

else begin

hanoi(n-1,a,c,b);

writeln(a,'à',b);

hanoi(n-1,c,b,a);

end;

end;

begin

write('nr discuri pe tija A =');readln(n);

writeln('mutarile sunt urmatoarele :');

hanoi(n,'A','B','C');

readln;

end.

2. Sortare rapida(quicksort)

Un tablou V se completeaza cu n elemente numere reale .Sa se ordoneze crescator folosind metoda de sortare rapida .

program quicksort;

type vector= array [1..50] of real ;

var v:vector;

i,n,k:integer;

functia poz(li,ls:integer):integer;

var i,j,modi,modj,m:integer;

man:real;

begin

i:=li; j:=ls;

modi:=0; modj:=-1;

while i<j do

begin

if v[i]>v[j] then

begin

man:=v[i];

v[i]:=v[j];

v[j]:=man;

m:=modi ;

modi:=-modj;

modj:=-m;

end;

i:=i+modi;

j:=j+modj;

end;

poz:=i;

end;

procedure quick(li,ls:integer);

begin

if li<ls then begin

k::=poz(li,

quick(li,k-1);

quick(k+1,ls);

end;

begin

write('cate elemente are vectorul ?=');readln(n);

for i:=1 to n do

begin

write('tastati elementul ',i,'=');

readln(v[i]);

end;

quick(1,n);

writeln('vectorul ordonat este :');

for i:=1 to n do writeln(v[i]);

readln;

end.

3. Codare Sir

Se considera un sir cu n elemente, initial toate egale cu n. Se imparte sirul pe jumatate, elementele din jumatatea stanga incrementandu-se, iar cele din jumatatea dreapta decrementandu-se cu o unitate. Daca exista element 'nepereche' exact la mijloc acesta ramane neschimbat. Celor doua jumatati li se aplica acelasi 'tratament' si jumatatilor jumatatilor la fel etc. pana cand se obtin secvente de cate un element.

program codare;

var a:array[1..100] of integer;

n,i:integer;

procedure codare(p,q):integer;

var m,i:integer;

begin

if q-p=2 then begin a[q]:=a[q]-1;

a[p]:=a[p]+1;

end

else if q-p=1 then begin a[q]:=a[q]-1;

a[p]:=a[p]+1;

end

else if (q-p) mod 2=0 then

begin m:=(p+q) div 2;

for i:=p to m-1do a[i]:=a[i]+1;

for i:=m+1 to q do a[i]:=a[i]-1;

codare(p,m-1);

codare(m+1,q);

end

else

begin m:=(p+q) div 2;

for i:=p to m do a[i]:=a[i]+1;

for i:=m+1 to q do a[i]:=a[i]-1;

codare(p,m);

codare (m+1,q);

end;

end;

begin

readln(n);

for i:=1 to n do a[i]:=n;

codare(1,n);

for i:=1 to n do write(a[i]:4);

writeln;

end.

4. Cautarea binara intr-un sir

Sa se verifice daca o valoare data x exista intr-un sir de numere intregi ordonate crescator. Se va folosi un algoritm de cautare bazat pe metoda Divide Et Impera.

Se descompune problema in doua subprobleme de acelasi tip. Se imparte vectorul in doi subvectori: primul subvector va contine elementele pana la pozitia de mijloc, iar al doilea va fi alcatuit din elementele de dupa pozitia din mijloc. Verificam daca valoarea cautata se gaseste chiar pe pozitia din mijloc si in caz afirmativ cautarea se opreste. Daca valoarea cautata este mai mica decat elementul situat pe pozitia din mijloc, cautarea trebuie continuata in subvectorul stang, iar daca este mai mare, cautarea continua in subvectorul drept.

program cautare;

type vector=array[1..20] of integer;

var v:vector;n,x,i:integer;

function caut(p,q:integer):integer;

var mij:integer;

begin

if q<p then caut:=-1

else

begin

mij:=(p+q) div 2;

if v[mij]=x then caut:=mij

else if x<v[mij] then caut:=(p,mij-1)

else caut:=caut(mij+1,q);

end;

end;

begin

write(’n=’);

readln(n);

write(’v[1]=’);

readln(v[1]);

for i:=2 to n do

repeat

write(’v[’,i,’]=’);

readln(v[i]);

until v[i]>v[i-1];

write(’x=’);

readln(x);

writeln(caut(1,n));

end.

5. Partitionarea unui sir

Se considera un sir de n numere intregi, ordonat crescator si un numar intreg x. Sa se partitioneze sirul dat in doua subssiruri, astfel incat toate elementele primului sir sa fie mai mici decat x, iar toate elementele celui de-al doilea sir sa fie mai mari decat x. Se va folosi un algoritm Divide Et Impera.

program partitionare;

type vector=array[1..20] of integer;

var v:vector;n,x,i,ref:integer;

function part(p,q):integer;

var mij:integer;

begin

if q<p then part:=p

else begin mij:=(p+q) div 2;

if x=v[mij] then part:=mij else

if x<v[mij] then part:=part(p,mij-1)

else part:=part(mij+1,q);

end;

end;

begin

write(’n=’);

readln(n);

write(’v[1]=’);

readln(v[1]);

for i:=2 to n do

repeat

write(’v[’,i,’]=’);

readln(v[i]);

until v[i]>=v[i-1];

write(’x=’);

readln(x);

ref:=part(1,n);

writeln(’primul vector’);

for i:=1 to ref-1 do write (v[i]:5);

writeln;

writeln(’al doilea vector’);

for i:=ref to n do write(v[i]:5);

readln;

end.

SLIDE6:BIBLIOGRAFIE

<http://www.creeaza.com/referate/informatica/Metoda-de-programare-DIVIDE-ET449.php>

<http://www.scritub.com/stiinta/informatica/METODA-DIVIDE-ET-IMPERA25186243.php>