**Proiect la informatică**

**Tema: Desparte și stăpânește**

Realizat de: Nicolaescu Cătălina, 11C

Metoda de programare DIVIDE ET IMPERA **(DESPARTE ȘI STĂPÂNEȘTE)**  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 .

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

4se poate descompune in ( doua sau mai multe) suprobleme ;

4aceste suprobleme sunt independente una fata de alta (o subproblema nu se rezolva pe baza alteia si nu se foloseste rezultate celeilalte);

4aceste subprobleme sunt similare cu problema initiala;

4 la randul lor subproblemele se pot descompune (daca este necesar) in alte subprobleme mai simple;

4aceste subprobleme simple se pot solutiona imediat prin algoritmul simplificat.

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 ;

4descompunerea treptata a subproblemelor in alte subprobleme din ce in ce mai simple ,pana cand se pot rezolva imediata ,prin algoritmul simplificat ;

4rezolvarea subproblemelor simple ;

4combinarea solutiilor gasite pentru construirea solutiilor subproblemelor de dimensiuni din ce in ce mai mari ;

4 combinarea ultimelor solutii determina obtinerea solutiei problemei initiale .

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.

Etapele metodei DIVIDE ET IMPERA (prezentate anterior)se pot reprezenta prin urmatorul subprogram general (procedura sau functie )recursiv exprimat in limbaj natural:

Subprogram DIVIMP (PROB);

Daca PROBLEMA PROB este simpla

Atunci se rezolva si se obtine solutia SOL

Altfel pentru i=1,k executa DIVIMP(PROB) si se obtine SOL1;

Se combina solutiile SOL 1,... ,SOL K si se obtine SOL;

Sfarsit \_subprogram;

Deci ,subprogramul DIVIMP se apeleaza pentru problema initiala PROB;aceasta admite descompunerea in K subprobleme simple ;pentru acestea se reapeleaza recursiv subprogramul ;in final se combina solutiile acestor K subprobleme.

De obicei problema initiala se descompune in doua subprobleme mai simple ; in acest caz etapele generale ale metodei DIVIDE ET IMPERA se pot reprezenta concret,in limbaj pseudocod ,printr-o procedura recursiva astfel :

Procedura DIVIMP(li,ls,sol);

Daca ((ls-li)<=eps)

18518g624s Atunci REZOLVA (li,ls,sol);

18518g624s Altfel

18518g624s DIVIDE (li,m,ls);

18518g624s DIVIMP(li,msol1);

18518g624s DIVIMP(m,ls,sol2);

18518g624s COMBINA(sol1,sol2,sol);

Sfarsit\_ procedura;

Procedura DIVIMP se apeleaza pentru problema initiala care are dimensiunea intre limita inferioara (li) si limita inferioara(ls);daca (sub)problema este simpla (ls-li<=eps),atunci procedura REZOLVA ii afla solutia imediat si se produce intoarcerea din apelul recursiv;daca (sub)problema este (inca) complexa ,atunci procedura DIVIDE o imparte in doua subprobleme ,alegand pozitia m intre limitele li si ls ;pentru fiecare din cele doua subprobleme se reapeleaza recursiv procedura DIVIMP; in final ,la intoarcerile din apeluri se produce combinarea celor doua soluitii sol1 si sol2 prin apelul procedurii COMBINA.

**Probleme**

1 PROBLEMA TURNURILOR DIN HANOI

Prezentarea algoritmului rezolvarii

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:

4la fiecare pas se muta un singur disc;

4un disc se poate aseza numai peste un disc cu diametrul mai mare .

Rezolvarea acestei probleme se bazeaza pe urmatoarele considerente logice :

-daca n=1 ,atunci mutarea este immediata AàB(mut discul de pe A pe B);

4daca n=2,atunci sirul mutarilor este : AàC,AàB,CàB;

4daca n>2 procedam astfel :

-mut (n-1)discuri AàC;

-mut un disc AàB ;

-mut cele (n-1)discuri CàB.

Observam ca problema initiala se descompune in trei subprobleme mai simple ,similare problemei initiale: mut (n-1)discuri AàC ,mut ultimul disc pe B ,mut cele (n-1)discuri C-->B.Dimensiunile acestor subprobleme sunt : n-1,1,n-1.

Aceste subprobleme sunt independente ,deoarece tijele initial (pe care sunt dispuse discurile ),tijele finale si tijele intermediare sunt diferite.Notam H(n,A,B,C)=sirul mutarilor a n discuri de pe A pe B, folosind C.

18518g624s

PENTRU 18518g624s 18518g624s 18518g624s

n=1 AàB

n>1 H(n,A,B,C)= H(n-1,A,C,B),AB, H(n-1,C,B,A) 18518g624s

18518g624s

program turnurile \_hanoi;

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);

18518g624s writeln(a,'à',b);

18518g624s hanoi(n-1,c,b,a);

18518g624s end;

end;

begin

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

writeln('mutarile sunt urmatoarele :');

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

readln;readln;

end.

2 Maxim intr-un vector

Se citeste un vector cu n componente, numere naturale. Se cere sa se tipareasca valoarea maxima.

> daca i=j, valoarea maxima va fi v[i];

> contrar vom imparti vectorul in doi vectori: primul vector va contine componentele de la i la (i+j) div 2, al doilea vector va contine componentele de la (I+J) div 2 +1 la j; rezolvam problemele (aflam maximul pentru fiecare din ele) iar solutia problemei va fi data de valoarea maxima dintre rezultatele celor doua subprobleme.

program maxim;

var v:array[1..10] of integer;

n,i:integer;

function max(i,j:integer):integer;

var a,b:integer;

begin

if i=j then max:=v[i]

else begin

a:=max(i, (i+j) div 2);

b:=max((i+j) div 2+1,j);

if a>b then max:=a

else max:=b;

end;

end;

begin

write(’n=’);

readln(n);

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

writeln(maximul este ’,max(1,n));

end.

3 Cel mai mare divizor comun

Fie n valori numere naturale a1,a2,a3,..,an. Determinati cel mai mare divizor comun al lor prin metoda Divide Et Impera. Se imparte sirul a1,a2,a3,..,an in doua subsiruri a1,a2,a3,..,am,respectiv am+1,am+2,.,an,unde m reprezinta pozitia de mijloc,m=(1+n) div 2.

Cmmdc(a1,a2,..,an)= Cmmdc(a1,a2,..,am), Cmmdc(am+1,am+2,.,an))

program cmmdc\_sir;

const nmax=20;

type indice=1..nmax;

var a:array[indice] of word;

n:indice;

procedure citire;

var i:indice;

begin

readln(n);

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

end;

function euclid(x,y:word):word;

var r:word;

begin

while y<>0 do

begin

r:=x mod y;

x:=y;

y:=r;

end;

euclid:=x;

end;

function cmmdc(p,q:indice):word;

var m:indice;

begin

if q-p<=1 then cmmdc:=euclid(a[p],a[q])

else

begin

m:=(p+q) div 2;

cmmdc:=euclid(cmmdc(p,m),cmmdc(m+1,q));

end;

end;

begin

citire;

writeln(’cmmdc=’,cmmdc(1,n));

readln; end.

4 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.

5 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.

6 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.

7 Pozitia k

Se citeste de la tastatura un sir de n elemente numere intregi.Sa se gaseasca elementul aflat pe o pozitie data k in sirul ordonat crescator, fara a efectua ordonarea.

program pozitia\_k;

const nmax=20;

type indice=1..nmax;

var a:array[indice] of integer;

n,k:indice;

procedure citire;

var i:indice;

begin

write(’n=’);

readln(n);

for i:=1 to n do begin write(’a[’,i,’]=’);

readln(a[i]);

end;

end;

procedure afisare;

var i:indice;

begin

writeln(’vectorul este’);

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

writeln;

end;

function divide(p,q:indice):indice;

var st,dr:indice;x:integer;

begin

st:=p;

dr:=q;

x:=a[p];

while st<dr do

begin

while (st<dr) and (a[dr]>=x) do dec(dr);a[st]:=a[dr];

while (st<dr) and (a[dr]<=x) do inc(st);a[dr]:=a[st];

end;

a[st]:=x;

divide:=st;

end;

function qselect(st,dr,k:indice):integer;

var p:indice;

begin

p:=divide(st,dr);

if k=p-st+1 then qselect:=qselect(st,p-1,k)

else qselect:=qselect(p+1,dr,k-(p-st+1));

end;

begin

citire;

write(’k=’)

;readln(k);

writeln(’pozitia ’,k,’ in vectorul sortat este ’,qselect(1,n,k));

afisare;

end.

**Bibliografie**

* [**http://www.creeaza.com/referate/informatica/Metoda-de-programare-DIVIDE-ET449.php**](http://www.creeaza.com/referate/informatica/Metoda-de-programare-DIVIDE-ET449.php)
* [**http://www.creeaza.com/referate/informatica/Metoda-de-programare-DIVIDE-ET449.php**](http://www.creeaza.com/referate/informatica/Metoda-de-programare-DIVIDE-ET449.php)