PROIECT INDIVIDUAL LA INFORMATICĂ

TEMA: **Desparte și stăpânește**

REALIZAT: Proca Virgiliu

VERIFICAT: Gutu Maria

NOTIUNI INTRODUCTIVE

Metoda de programare ***desparte și stăpânește*** constă în împărțirea problemei inițiale de dimensiuni [n] în două sau mai multe probleme de dimensiuni reduse .în general se execută împărțirea în două subprobleme de dimensiuni aproximativ egale si anume [n/2] .Împărțirea în subprobleme are loc până când dimensiunea acestora devine suficient de mică pentru a fi rezolvate în mod direct(cazul de bază).După rezolvarea celor două subprobleme se execută faza de combinare a rezultatelor în vederea rezolvarii întregii probleme .

Metoda ***desparte și stăpânește*** se poate aplica în rezolvarea unei probleme care îndeplinește urmatoarele condiții :

- se poate descompune în (două sau mai multe) suprobleme ;
- aceste suprobleme sunt independente una față de alta (o subproblemă nu se rezolva pe baza alteia și nu folosește rezultate celeilalte);
- aceste subprobleme sunt similare cu problema întâi
- la rândul lor subproblemele se pot descompune (dacă este necesar) în alte subprobleme mai simple;
- aceste subprobleme simple se pot soluționa imediat prin algoritmul simplificat.

Deoarece puține probleme îndeplinesc condițiile de mai sus ,aplicarea metodei este destul de rară.După cum sugerează și numele "desparte și stăpânește " ***etapele*** rezolvării unei probleme (numită problema inițială) în ***desparte și stăpânește*** sunt :

1. descompunerea problemei inițiale în subprobleme independente, similare problemei de bază ,de dimensiuni mai mici ;
2. descompunerea treptată a subproblemelor în alte subprobleme din ce în ce mai simple ,până cand se pot rezolva imediat ,prin algoritmul simplificat ;
3. rezolvarea subproblemelor simple ;
4. combinarea soluțiilor găsite pentru construirea soluțiilor subproblemelor de dimensiuni din ce în ce mai mari ;
5. combinarea ultimelor soluții determină obținerea soluției problemei inițiale .

AVANTAJE

- -Programele cu această metodă sunt simple de înțeles
- -Timpul execuției relativ mic
- -Această tehnică admite o implementare recursivă
- -Această metodă stă la baza a mai multe metode de sortarea rapidă

DEZAVANTAJE

- -Metoda poate fi aplicată numai când prelucrarea cerută admite divizarea problemei curente în subprobleme
- -Metodă complicată de folosit pentru mulți
- -Metodă ce necesită mai multe condiții, astfel este folosit destul de rar
- -Necesar competență de programator și logică bună

➢ Problema maxim

Dacă i=j, valoarea maximă va fi v[i];contrar vom împarti vectorul în doi **vectori:** primul vector va conține componentele de la i la (i+j) div 2, al doilea vector va conține componentele de la (I+J) div 2 +1 la j; rezolvăm problemele (aflăm maximul pentru fiecare din ele) iar soluția problemei va fi dată de valoarea maximă dintre rezultatele celor două 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.
```

➢ Sortarea rapidă QuickSort

Un tablou V se completează cu n elemente numere reale . Să se ordoneze crescător folosind metoda de sortare rapidă .

Soluția problemei se bazează pe urmatoarele etape implementate în programul principal:

1. se apelează procedura “quick” cu limita inferioara li=1 si limita superioară ls=n;
2. funcția ”poz” realizează mutarea elementului v[i] exact pe poziția ce o va ocupa acesta în vectorul final ordonat; funcția ”poz” întoarce (in k) poziția ocupată de acest element;
3. în acest fel ,vectorul V se împarte în doua parți : li …k-1 si k+1…ls;
4. pentru fiecare din aceste părți se reapelează procedura“quick”,cu limitele modificate corespunzător ;

5. în acest fel primul element din fiecare parte va fi pozitionat exact pe pozitia finală ce o va ocupa în vectorul final ordonat (funcția “poz”);
6. fiecare din cele două parți va fi astfel împărțită în alte două parți, procesul continuă până când limitele părților ajung să se suprapună, ceea ce indică că toate elementele vectorului au fost mutate exact pe pozițiile ce le vor ocupa in vectorul final; deci vectorul este ordonat ;
7. În acest moment se produc întoarcerile din apelurile recursive și programul își termină execuția .

```
program quicksort;
type vector= array [1..50] of real;
var v:vector; i,n,k:integer;
function 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,ls);
quick(li,k-1);
quick(k+1,ls);
end;
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.
```

- **Sortare prin interclasare(mergesort)**

Tabloul unidimensional V se completează cu n numere reale. Să se ordoneze crescător folosind sortare prin interclasare. Sortarea prin interclasare se bazează pe urmatoarea logica : vectorul V se împarte prin înjumătățiri succesive în vectori din ce în ce mai mici când se ating vectorii de maxim doua elemente, fiecare dintre aceștia se ordonează printr-o simplă comparare a elementelor; câte doi astfel de mini- vectori ordonați se interclasează succesiv până se ajunge iar la vectorul V.

```
program mergesort;
type vector=array[1..50] of real ;
var v:vector ;n,i:word;
procedure schimba(li,ls:word;var a:vector);
var man:real;
begin
if a[li]>a[ls] then begin
man:=a[li];
a[li]:=a[ls];
a[ls]:=man;
end;
end;
procedure interclas(li,m,ls:word;var a:vector);
var b:vector;
i,k,p,j:word;
begin
i:=li; j:=m+1; k:=0;
while (i<=m)and(j<=ls) do
begin
inc(k);
if a[i] <a[j] then begin
b[k]:=a[i];
inc(i);
end
else begin
b[k]:=a[j];
inc(j);
end;
end;
if i<=m then for p:=i to m do begin1
inc(k);b[k]:=a[p];
end;
if j<=ls then for p:=j to ls do begin
inc(k) ;b[k]:=a[p];
end;
k:=0;
for p:=li to ls do begin
inc(k);
a[p]:=b[k];
end;
end;
procedure divi(li,ls:word; var a:vector);
var m:word;
```

```
procedure divi(li,ls:word; var a:vector);
var m:word;
begin
if (ls-li)<=1 then schimba(li,ls,a);
else begin
m:=(li+ls)div 2;
divi(li,m,a);
divi(m+1,ls,a);
interclas(li,m,ls,a);
end;
end;
begin
write('cate elemente are vectorul?=');readln(n);
for i:=1 to n do
begin
write('tastati elementul',i,'=');
readln(v[i]);
end;
divi(1,n,v);
writeln('vectorul sortat este:');
for i:=1 to n do writeln(v[i]);
end.
```

- **Cel mai mare divizor comun**

Fie n valori numere naturale a1,a2,a3,..,an. Determinați cel mai mare divizor comun al lor prin metoda Divide Et Impera. Se împarte șirul a1,a2,a3,..,an în două subșiruri a1,a2,a3,..,am,respectiv am+1,am+2,.,an,unde m reprezintă poziția 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
beginr:=x mod y; x:=y;y:=r;
end;
begin
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.
```

- **Problema plierilor**

Se consideră un vector de lungime n. Definim plierea vectorului prin suparapunerea unei jumătăți, numită donoare, peste cealalta jumătate, numită receptoare, cu precizarea că dacă numărul de elemente este impar, elementul din mijloc este eliminat. Prin pliere, elementele subvectorului obținut vor avea numărătoarea jumătății recepetoare. Plierea se poate aplica în mod repetat, până când se ajunge la un subvector format dintr-un singur element, numit element final.Scrieți un program care să afișeze toate elementele finale posibile și să figureze pe ecran pentru fiecare element final o succesiune de plieri.

```
program plieri;
const nmax=50;
type element=1..nmax;
var n,i:element;
efinal:array[element] of boolean;
m:array[element] of string;
procedure pliaza(p,q:element);
begin
if p=q then efinal [p]:=true
else
begin
if (q-p+1) mod 2=1 then
begin
efinal[(p+q) div 2]:=false;
ls:=(p+q) div 2-1;
end
else
ls:=(p+q) div 2;
ld:=(p+q) div 2+1;
pliaza(p,ls);
str(ls,ss);
str(ld,sd);
for i:=p to ls do
m[i]:='d'+sd+' '+m[i];
end;
end;
begin
write('n=');
readln(n);
for i:=1 to n do m[i]:='  ';
pliaza(1,n);
writeln('elementele finale sunt: ');
for i:=1 to n do if efinal[i] then begin write (' ', ': ');
writeln(m[i]);
end;
writeln;
end.
```

CONCLUZIE

Metoda desparte și stăpânește este o temă predestinată doar persoanelor ce tind în a se duce în domeniul IT, astfel această temă fiind opțională în curriculum. Din opiniile a mai multor persoane, această temă este una complicată, dar totuși foarte folositoare în multe cazuri, precum realizarea sortării rapide, prin metoda precum quicksort sau bubblesort. Unele din aceste cazuri le-am enumerat mai sus prin exemple de programe.

BIBLIOGRAFIE

- http://www.creeaza.com/referate/informatica/Metoda-de-programare-DIVIDE-ET449.php
- MANUAL CLASA XI-a EDITURA Știința
- http://lectura.bibliotecadigitala.ro/cazacunina/Caietul%20profesorului_de_Informatica_Cazacu_Nin a.pdf https://www.scribd.com/document/130021652/Algoritmi-in-Pascal
- https://www.didactic.ro/materiale-didactice/test-divide-et-impera-pascal
- http://www.scritub.com/stiinta/informatica/METODA-DIVIDE-ET-IMPERA25186243.php