PROIECT IDIVIDUAL LA INFORMATICĂ

TEMA: TEHNICA RELUĂRII (BACKTRACKING)

A REALIZAT: Cernatinschi Nicolae, clasa a XI-a ”C”
A VERIFICAT: Maria Guțu
IPLT ‘Spiru Haret’

INFORMAȚIE

Backtrackingul este o tehnică de programare prin care o problemă se rezolvă prin generarea tuturor soluțiilor acesteia. Există 3 tipuri de folosire a metodei reluării:

1) Căutăm o soluție posibilă

2) Căutăm cea mai bună soluție

3) Căutam toate soluțiile posibile

AVANTAJE

-Elimină atribuiri care nu respectă condițiile

-Folosește puțină memorie

-Poate fi adaptată pentru orice problemă în care este necesară obținerea tuturor soluțiilor posibile

-Structura de memorare a datelor folosită de metoda reluării este STIVA

DEZAVANTAJE

-Metoda Backtracking are timpul necesar pentru execuție mare

-Metodă complicată pentru mulți

EXEMPLE DE PROGRAME

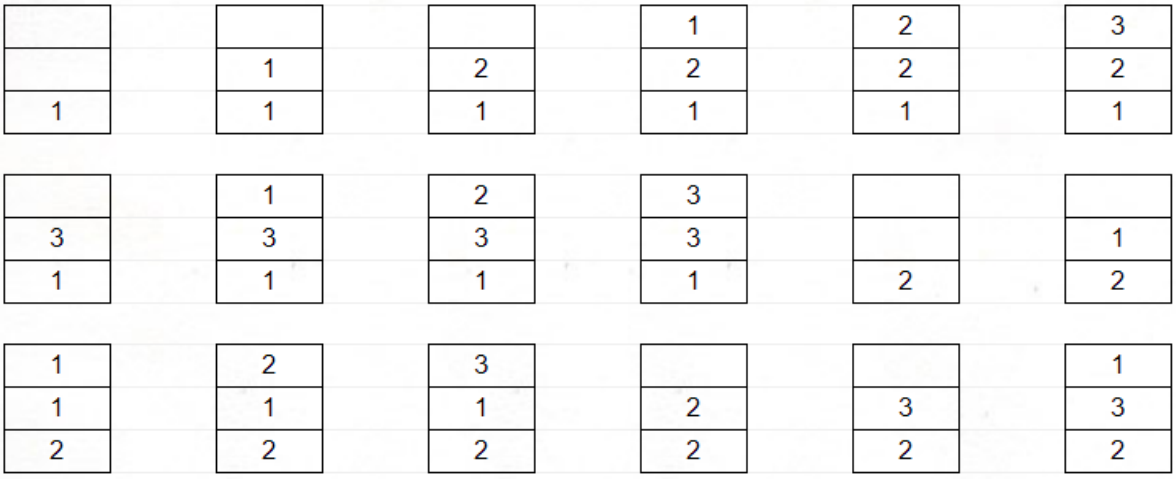
```
program permutari;
var st:array[1..25] of integer;i,n,p:integer;
procedure init;
var i:integer;
 begin
 write('N='); readln(n);for i:=1 to 25 do
st[i]:=0;end;
function valid(p:integer):boolean;
var i:integer;
```

```
 begin valid:=true;for i:=1 to p-1 do if
st[i]=st[p] then valid:=false;end;
procedure tipar(p:integer);
var i:integer; begin for i:=1 to p do
writeln(st[i],' ');end;
procedure back(p:integer);
 begin p:=1;
{plecam de la primul nivel }
st[p]:=0;
{initializam nivelul cu 0}
while p>0 do
{cat timp stiva nu este vida}
 begin if st[p]<n then
{mai exista valori neincercate pe nivelul p}
 begin st[p]:=st[p]+1;
{st[p]<-<o noua valoare din multimea valorilor
posibile>}
 if valid(p) then if p=n then tipar(p)
{solutia este finala}
 else begin p:=p+1;
{trecem la nivelul urmator}
st[p]:=0;
{initializam valoarea de pe nivel cu 0}
end;end else
p:=p-1; {pas inapoi}
 end;end;
begin
 init;back(1);end.
```

Generarea permutărilor. Se citeşte un număr natural n. Să se genereze toate permutările mulţimii {1, 2, 3, …,n}.

Generarea permutărilor se va face ţinând cont că orice permutare va fi alcătuită din elemente distincte ale mulţimii A. Din acest motiv, la generarea unei permutări, vom urmări ca numerele să fie distincte.

Prezentăm algoritmul corespunzător cazului n=3:



- se încarcă în stivă pe nivelul 1 valoarea 1;
- încărcarea valorii 1 pe nivelul al 2-lea nu este posibilă, întrucât această valoare se găseşte şi pe nivelul 1 al stivei;
- încărcarea valorii 2 pe nivelul al 2-lea este posibilă, deoarece această valoare nu mai este întâlnită;
- valoarea 1 din nivelul al 3-lea se regăseşte pe nivelul 1;
- valoarea 2 din nivelul al 3-lea se regăseşte pe nivelul al 2-lea;
- valoarea 3 pe nivelul al 3-lea nu e întâlnită pe nivelurile anterioare; întrucât nivelul 3 este completat corect. Tipărim: 1 2 3

……

Algoritmul continuă până când stiva devine vidă.

Dintr-un nr. de 6 cursuri optionale un elev trebuie sa aleaga 3. Sa se afiseze toate posibilitatile de alegere precum si nr. lor.

```
program cursuri;
const n=6; p=3;
type stiva=array [1..10] of integer;
var st:stiva;
ev,ap:boolean;
k:integer;
procedure init(k:integer;var st:stiva);
```

```
begin
if k>1 then st[k]:=st[k-1]
else if k=1 then st[k]:=0;
end;
procedure succesor(var ap:boolean;var
st:stiva;k:integer);
begin
if st[k]<n-p+k then begin st[k]:=st[k]+1;
ap:=true;
end
else ap:=false;
end;
procedure valid(var ev:boolean;var
st:stiva;k:integer);
var i:integer;
begin
ev:=true;
for i:=1 to k-1 do if st[i]=st[k] then
ev:=false;
end;
function solutie(k:integer):boolean;
begin
solutie:=(k=p);
end;
procedure tipar;
var i:integer;
begin
for i:=1 to p do write (st[i]);
writeln;
end;
begin
k:=1;init(k,st);
while k>0 do
begin
repeat
succesor (ap,st,k);
if ap then valid(ev,st,k);
until (not ap) or (ap and ev);
```

```
if ap then
if solutie(k) then tipar
else begin
k:=k+1;
init(k,st)
end
else k:=k-1;
end;
end.
```

Lui IRINEL îi plac nr. formate numai din cifre pare cifre aflate în ordine crescatoare. Sa se determine si sa se afiseze pe ecran toate nr. de n cifre (0<n<10) care îi plac lui Gigel. Valoarea lui n este un nr. natural care se citeste de la tastatura.

```
program nr_lui_IRINEL;
type stiva=array[1..100] of integer;
var st:stiva;
i,n,k:integer;
ap,ev:boolean;
procedure init(k:integer;var st:stiva);
begin
st[k]:=-1;
end;
procedure succesor(var ap:boolean;var
st:stiva;k:integer);
begin
if st[k]<9 then begin st[k]:=st[k]+1;
as:=true;
end
else ap:=false;
end;
procedure valid(var
ev:boolean;st:stiva;k:integer);
var i:integer;
begin
```

```
ev:=true;
for i:=1 to k-1 do
if st[i] mod 2 <> 0 then ev:=false;
for i:=1 to k-1 do
if st[i]<st[i+1] then ev:=false;
end;
function solutie(k:integer):boolean;
begin
solutie:=(k=n);
end;
procedure tipar;
var i:integer;
begin
for i:=1 to n do write(st[i]);
writeln;
end;
begin
write('n= ');readln(n);
k:=1 ;init(k,st);
while k>0 do
begin
repeat
succesor(ap,st,k);
if ap then valid(ev,st,k);
until (not ap) or (ap and ev);
if ap then if solutie(k) then tipar
else begin
k:=k+1;
init(k,st);
end
else k:=k-1;
end;
readln;
 end.
```

```
program dame;
type stiva=array[1..100] of integer;
var st:stiva;
n,k:integer;
ap,ev:boolean;
procedure init(k:integer;var st:stiva);
begin
st[k]:=0;
end;
procedure succesor(var ap:boolean;var
st:stiva;k:integer);
begin
if st[k]<n then begin  st[k]:=st[k]+1;
ap:=true end
else ap:=false;
end;
procedure valid(var ev:boolean;var
st:stiva;k:integer);
var i:integer;
begin
ev:=true;
for i:=1 to k-1 do if (st[k]=st[i]) or
(abs(st[k]-st[i])=abs(k-i)) then ev:=false;
end;
function solutie(k:integer):boolean;
begin
solutie:=(k=n);
end;
procedure tipar;
var i:integer;
begin
for i:=1 to n do write(st[i]);
writeln;
end;
begin
write('n:');readln(n);
k:=1;init(k,st);
while k>0 do
```

```
begin
repeat
succesor(ap,st,k);
if ap then valid(ev,st,k);
until (not ap) or (ap and ev);
if ap then if solutie(k) then tipar
else begin
k:=k+1;
init(k,st); end
else k:=k-1;
end;
end.
```

```
program sortare;
type vector=array[1..10] of integer;
var a:vector;
n,i:integer;
procedure sort(p,q:integer;var a:vector);
var m:integer;
begin
if a[p]>a[q]  then
begin
m:=a[p];
a[p]:=a[q];
a[q]:=m
end;
end;
procedure interc(p,q,m:integer;var a:vector);
var b:vector;
i,j,k:integer;
begin
i:=p;
j:=m+1;
k:=1;
while (i<=m) and (j<=q) do
if a[i]<=a[j] then
begin
b[k]:=a[i];
i:=i+1;
```

```
k:=k+1
end
else begin
b[k]:=a[j];j:=j+1;k:=k+1
end;
if i<=m then
for j:=i to m do begin
b[k]:=a[j];
k:=k+1;
end
else
for i:=j to q do begin
b[k]:=a[j];
k:=k+1;
end;
k:=1;
for i:=p to q do begin
a[i]:=b[k];
k:=k+1;
end
end;
procedure divimp(p,q:integer; var a:vector);
var m:integer;
begin
if (q-p)<=1 then sort(p,q,a)
else begin
m:=(p+q) div 2;
divimp(p,m,a);
divimp(m+1,q,a);
interc(p,q,m,a);
end
end;
begin
write('n= ');read(n);
for i:=1 to n do begin
write('a[',i,']=');readln(a[i]);
end;
divimp(1,n,a);
```

```
for i:=1 to n do
writeln(a[i]);
end.
```

CONCLUZIE

Metoda reluării este o temă predestinată doar persoanelor ce tind în a se duce în domeniul IT. Din opiniile a mai multor oameni, această temă este una complicată, dar totuși folositoare în multe cazuri. Unele din aceste cazuri le-am enumerat mai sus prin exemple de programe.

BIBLIOGRAFIE

https://ro.wikipedia.org/wiki/Backtracking

MANUAL CLASA XI-a EDITURA Știința

https://www.scribd.com/document/13396582/Limbajul-Pascal-Metoda-Backtracking-Permutari

http://www.preferatele.com/docs/informatica/4/backtracking6.php

https://www.geeksforgeeks.org/backtracking-introduction/

https://prezi.com/kqddgev8wrku/metoda-backtracking-si-metoda-greedy/

http://www.scritub.com/stiinta/informatica/METODA-BACKTRACKING1055131414.php