ATESTAT LA INFORMATICĂ

|  |  |
| --- | --- |
| Profesor îndrumător: | Elevi: |
| Stan Mihaela-Veronica | Bîncă Andreea Cristina  Dudău Claudia Maria |

OPERAȚII CU ARBORI BINARI DE CĂUTARE

Cuprins

[Introducere 4](#_Toc8249250)

[Descriere proiect 6](#_Toc8249251)

[Mod de utilizare 13](#_Toc8249252)

[Anexă 21](#_Toc8249253)

[Bibliografie 31](#_Toc8249254)

# Introducere

În teoria grafurilor, un arbore este un graf neorientat, conex și fără cicluri. Arborii reprezintă grafurile cele mai simple ca structură din clasa grafurilor conexe, ei fiind și cei mai frecvent utilizați în practică.

Termenul de „arbore” din teoria grafurilor a fost folosit pentru prima dată de Arthur Cayley în anul 1857. El a plecat de la o analogie cu noțiunea de „arbore” din botanică.

Arborii au fost studiați intensiv de numeroși matematicieni și fizicieni, precum matematicianul britanic Arthur Cayley, pe care l-au interesat aplicațiile lor în chimia organică, de exemplu grafurile chimice, sau fizicianul german G. R. Kirchhoff, care a studiat această categorie pornind de la studiul rețelelor electrice.

Structurile arborescente se folosesc în programare deoarece:

- reprezintă un model natural pentru o ierarhie de obiecte (entități, operații etc.);

- sunt structuri de căutare cu performante foarte bune, permițând și menținerea în ordine a unei colecții de date dinamice (cu multe adăugări și ștergeri).

De asemenea, studiul arborilor ajută la o mai bună înțelegere a modului în care funcționează structurile dinamice de tip listă/coadă, dar și la îmbunătățirea gândirii logice a elevilor și a studenților, puși în fața unor noi tipuri de probleme.

**Ce este mai exact un arbore?**

Un arbore cu rădăcină ("rooted tree") este o structură neliniară, în care fiecare nod poate avea mai mulți succesori, dar un singur predecesor, cu excepția unui nod special, numit rădăcină, care nu are niciun predecesor.

Structura de arbore este o structură ierarhică, cu noduri așezate pe diferite niveluri, cu relații de tip părinte - fiu între noduri.

Nodurile sunt de două feluri:

- nodurile terminale, fără succesori, se numesc și "frunze"

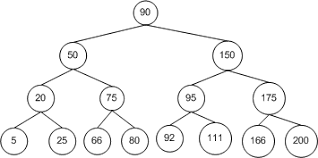
- noduri interne (interioare), cu unul sau mai mulți succesori

Un caz particular important de arbori îl constituie arborii binari, în care un nod poate avea cel mult doi succesori: un succesor la stânga și un succesor la dreapta.

**Arbore binar de căutare**

Un arbore binar de căutare (BST=Binary Search Tree), numit și arbore de sortare sau arbore ordonat, este un arbore binar cu proprietatea că orice nod interior are valoarea mai mare decât orice nod din subarborele stâng și mai mică decât orice nod din subarborele drept.

Exemplu de arbore binar de căutare:



# Descriere proiect

Proiectul constă în executarea următoarelor operații asupra arborilor binari de căutare cu valori pozitive:

- parcurgere

- căutare

- adăugare

- ștergere

Algoritmul a fost proiectat în Code Blocks, în limbajul C++, folosind interfața grafică (biblioteca „graphics.h”).

**Inițializarea arborelui**

Algoritmul construiește arborele (*r*) utilizând implementarea dinamică (prin folosirea unei structuri):

struct nod

{

int nr,poz;

nod \*st, \*dr;

}\*r;

-> *nr* = valoarea pe care o are nodul

-> *poz* = poziția nodului în vectorul inițial de noduri

-> *st* = pointer care trimite la succesorul stâng

-> *dr* = pointer care trimite succesorul drept

În continuare, algoritmul inițializează arborele, atât prin memorarea valorilor nodurilor, citite dintr-un fișier, într-un vector (ce va facilita o ulterioară resetare a arborelui), cât și prin subprogramul de creare.

Subprogramul *creare* verifică existența primului nod, pe care îl creează in mod dinamic, dar îl și memorează într-o matrice de coordonate (*coord*), pentru a ușura afișarea în interfața grafică:

r=new nod;

r->nr=copnoduri[j];

r->poz=j;

r->st=0;

r->dr=0;

coord[j][1]=r->nr;

coord[j][2]=725;

coord[j][3]=10;

-> *copnoduri* = o copie a vectorului creat inițial, pentru a nu se pierde informația

Arborele este construit folosind subprogramul *inserare*, care introduce recursiv nodurile în arbore, pentru fiecare nod verificând în care subarbore trebuie introdus:

//*inserarea în arborele stâng*

if(r->nr>k)

{

x0=coord[r->poz][2]-200;

y0=coord[r->poz][3]+80;

p++;

v=1;

inainte=r->poz;

inserare(r->st,k,e);

}

//*inserarea în aborele drept*

else

{

x0=coord[r->poz][2]+200;

y0=coord[r->poz][3]+80;

p++;

v=2;

inainte=r->poz;

inserare(r->dr,k,e);

}

Atunci când s-a găsit locul de introdus, algoritmul verifică mai întâi dacă nodul se încadrează pe ecran (fereastra grafică are anumite dimensiuni, ceea ce face ca nodurile să apară în fereastră doar până la un anumit rând), îl memorează dinamic și în matricea de coordonate, apoi afișează nodul fie în cadrul arborelui, fie pe un rând separat, unde se vor afla toate nodurile care nu se încadrează pe ecran.

//afișarea în grafică a nodurilor

char \*s;

int j;

s=new char;

itoa(k,s,10);

j=ad[p];

if(v==1)

{

outtextxy(x0+j,y0,s);

circle(x0+j+9,y0+11,20);

x2=x0+j;

}

else

{

outtextxy(x0-j,y0,s);

circle(x0-j+9,y0+11,20);

x2=x0-j;

}

-> *j* (aici) = valoare ajutătoare care modifică distanța dintre noduri în funcție de rândul pe care va fi afișat – valorile sunt luate din vectorul ad care conține aceste distanțe

-> *itoa* (int to array) = funcție care transformă numerele, cu ajutorul căreia putem folosi funcția *outtextxy* pentru afișarea grafică

După afișarea arborelui inițial, cu valorile citite dintr-un fișier, pe ecran apare *meniul*, ce conține toate operațiile efectuabile asupra arborelui.

**Parcurgerea arborelui**

Arborele binar de căutare este parcurs în inordine (stânga – rădăcina – dreapta) pentru a afișa numerele în ordine crescătoare și, pe măsura ce sunt parcurse, valorile din arbore sunt colorate.

//*alegerea parcurgerii*

if(alg==1)

{

outtextxy(10,700,"PARCURGEREA ESTE: ");

srd(r);

}

//*colorarea valorilor arborelui*

if(ramase[r->poz]==0)

{

setcolor(9);

outtextxy(coord[r->poz][2],coord[r->poz][3],s);

}

-> *ramase* = vector care arată ce noduri nu se încadrează pe ecran (pe poziția fiecărui nod se afla fie 0, dacă nodul încape pe ecran, fie 1, dacă nodul nu se încadrează).

**Căutarea unei valori în arbore**

Subprogramul *căutare* este unul recursiv și afișează „GASIT”/„NEGASIT”, iar atunci când valoarea se află în arbore, algoritmul o colorează.

//*alegerea căutării*

if(alg==2)

{

outtextxy(10,700,"INTRODUCETI VALOAREA CE TREBUIE CAUTATA!");

cout<<"VALOARE CE TREBUIE CAUTATA: ";

int caut;

cin>>caut;

cautare(r,caut);

}

//*condiția recursivă*

if(r->nr>k) cautare(r->st,k);

else cautare(r->dr,k);

**Adăugarea unui nou nod**

Adăugarea se face cu ajutorul subprogramului *inserare,* folosit și la inițializarea arborelui.

//*alegerea adăugării*

if(alg==3)

{

outtextxy(10,700,"INTRODUCETI VALOAREA CE TREBUIE ADAUGATA!");

cout<<"VALOARE CE TREBUIE ADAUGATA: ";

int adaug;

cin>>adaug;

n++;

copnoduri[n]=adaug;

p=0;

v=0;

inserare(r,adaug,n);

if(ramase[n]==1)

{

char \*s;

s=new char;

itoa(copnoduri[n],s,10);

outtextxy(480+var2,550,s);

var2=var2+40;

}

}

-> *if*-ul de după apelarea inserării verifică dacă numărul adăugat se încadrează sau nu pe ecran, iar în caz contrat îl afișează separat

**Ștergerea unui nod din arbore**

Ștergerea folosește subprogramele recursive *cmmnd* (care produce ștergerea nodurilor care au atât subarbore stâng cat ți drept) și *ștergere* (care șterge nodurile frunze sau care au un singur subarbore).

//*verificarea poziției*

if(r->nr==k)

{

if(r->st==0 && r->dr==0)

{

delete r;

r=0;

}

else

{

if(r->dr==0)

{

g=r->st;

delete r;

r=g;

}

else

{

if(r->st==0)

{

g=r->dr;

delete r;

r=g;

}

else cmmnd(r,r->st);

}

}

}

//*ștergerea nodului cu cmmnd*

r->nr=g->nr;

man=g->st;

delete g;

g=man;

**Resetarea arborelui**

Dacă după operațiile făcute asupra arborelui se dorește întoarcerea la arborele inițial, atunci trebuie aleasă opțiunea de *resetare*, care practic reafișează arborele memorat în vectorul de la început.

//*alegerea resetării*

if(alg==5)

{

cleardevice();

for(int i=1;i<=n;i++)

copnoduri[i]=noduri[i];

n=cn;

creare();

}

**Reinițializarea ecranului**

După execuția oricărei operații, ecranul primește un „refresh”, pentru a se putea continua lucrul. Acest „refresh” realizează o parcurgere în postordine (stânga – dreapta – rădăcină), pentru reîntoarcerea la rădăcină, și reinițializează coordonatele ca și când ar avea loc prima operație, dar de data aceasta asupra arborelui modificat:

sdr(r);

for(int i=1;i<=n;i++)

{

coord[i][1]=0;

coord[i][2]=0;

coord[i][3]=0;

ramase[i]=0;

}

var=0;

vardelay++;

cleardevice();

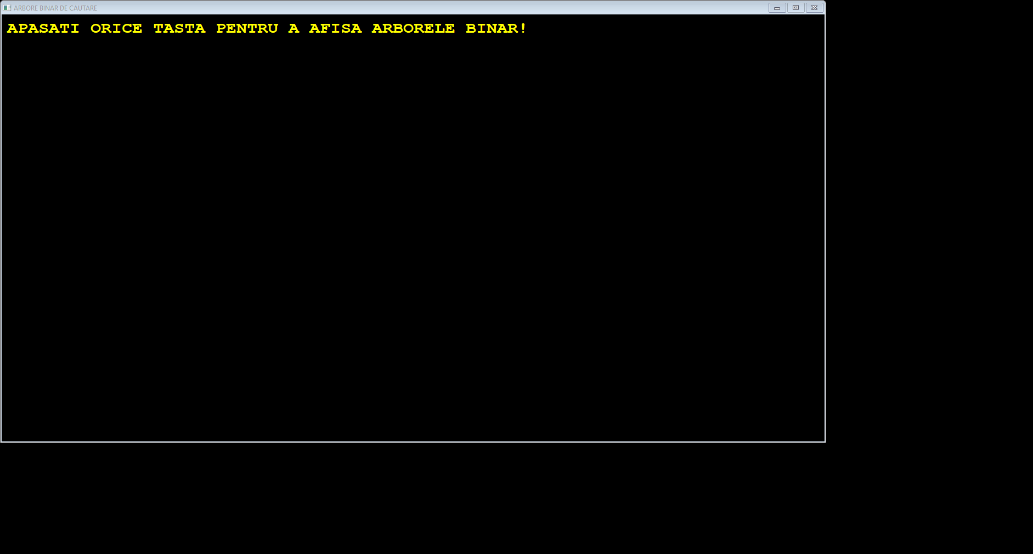
**Ieșirea din proiect**

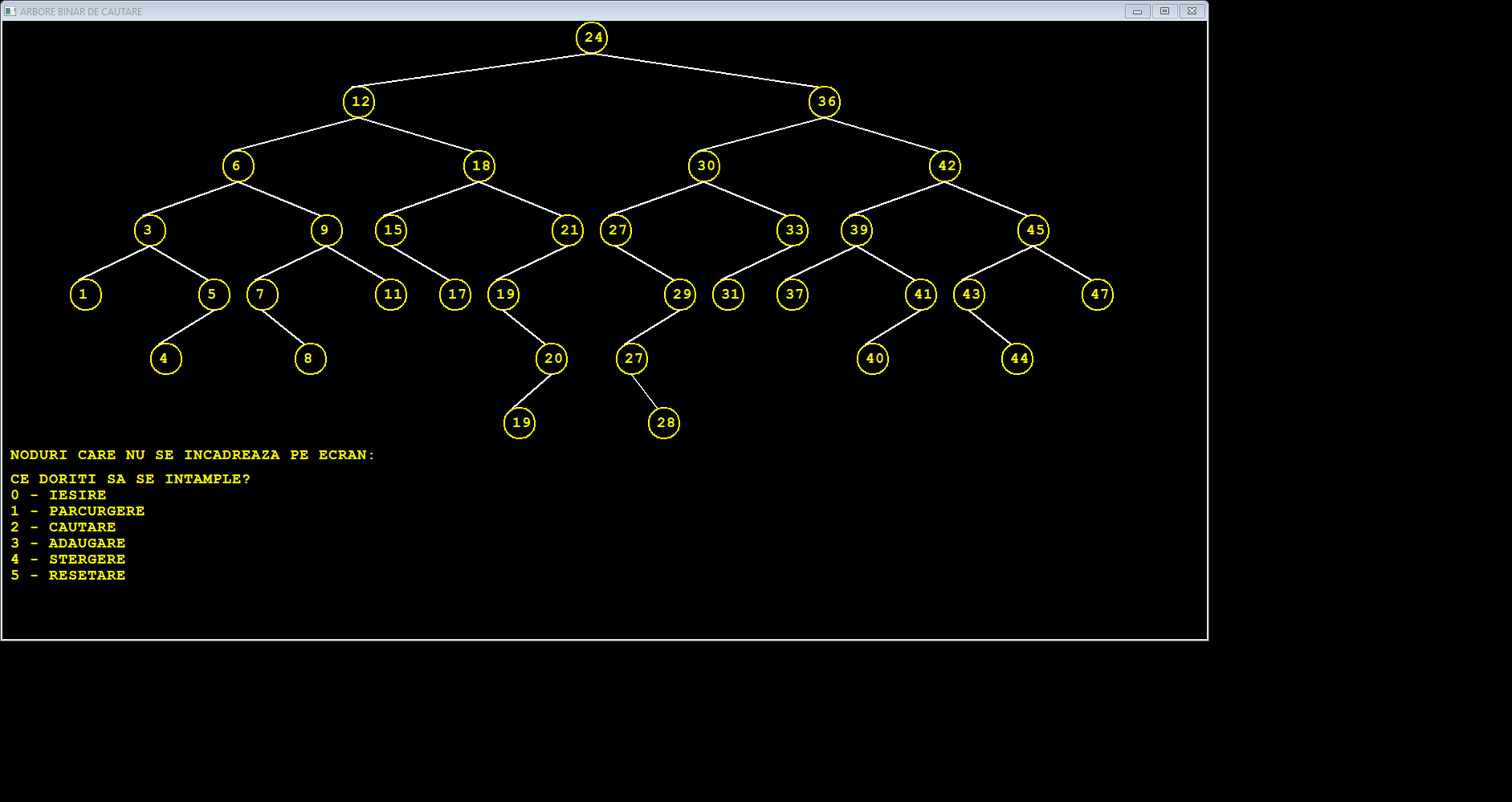
Ieșirea din proiect se face prin opțiunea 0 din meniu. În acest caz programul va ieși instant din ecranul grafic, fără sa mai efectueze vreo operație.

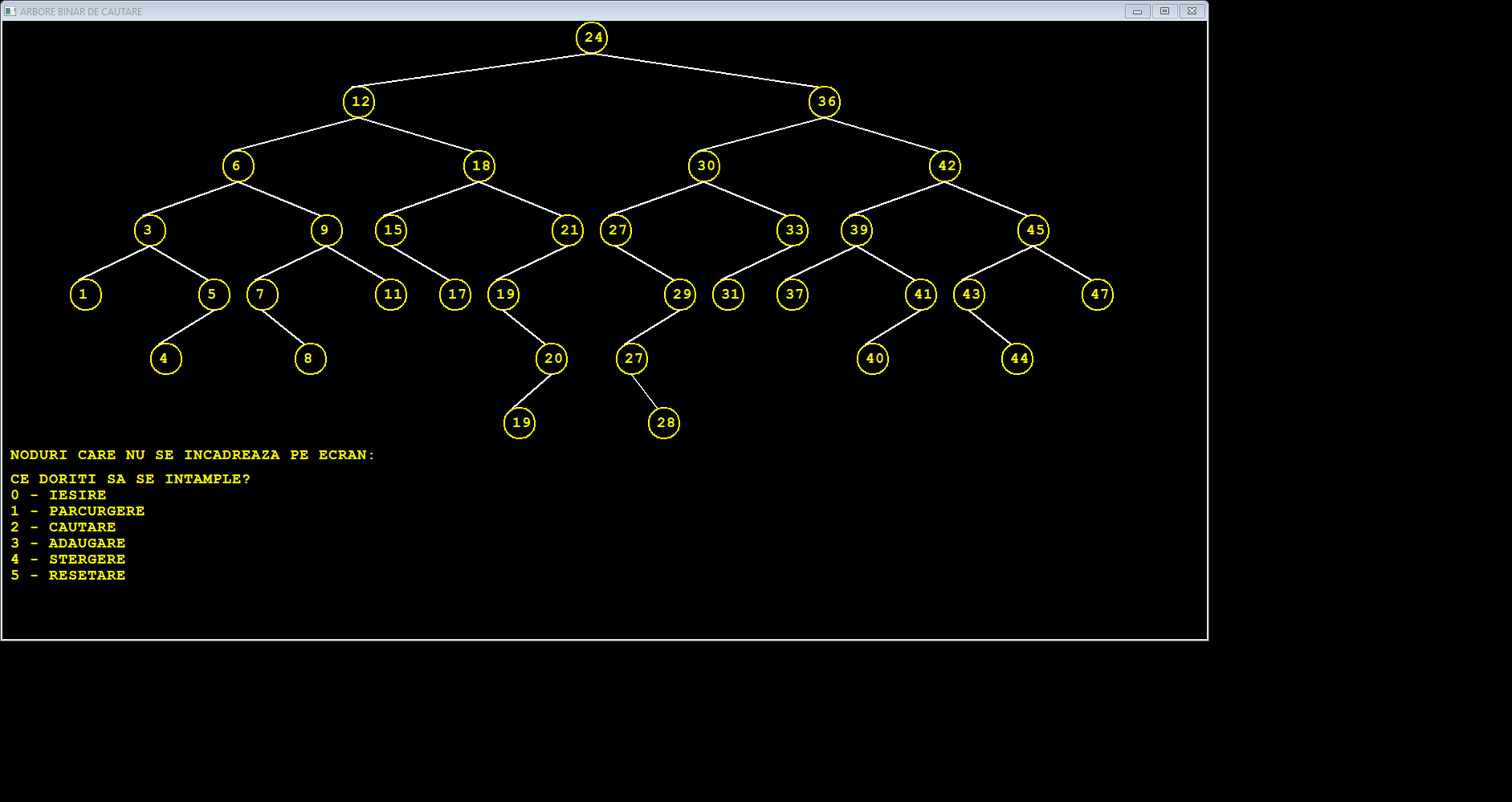
# Mod de utilizare

**Rularea programului**

Programul începe prin apariția unui ecran negru pe care apare mesajul: „APĂSAȚI ORICE TASTĂ PENTRU A AFIȘA ARBORELE BINAR!”. După apăsarea oricărei taste se va afișa pe ecran arborele creat, urmat de o listă a nodurilor care nu se încadrează în dimensiunile ecranului si de meniu.





Nodurile ce nu se încadrează pe ecran sunt cele aflate pe rândul al 8-lea sau mai jos, acestea neîncăpând în dimensiunile ecranului. Aceste noduri se vor afișa între arbore și meniu, în dreptul mesajului „NODURI CARE NU SE ÎNCADREAZĂ PE ECRAN”.

Meniul are 6 opțiuni: - ieșirea din program (tasta 0)

- parcurgerea arborelui (tasta 1)

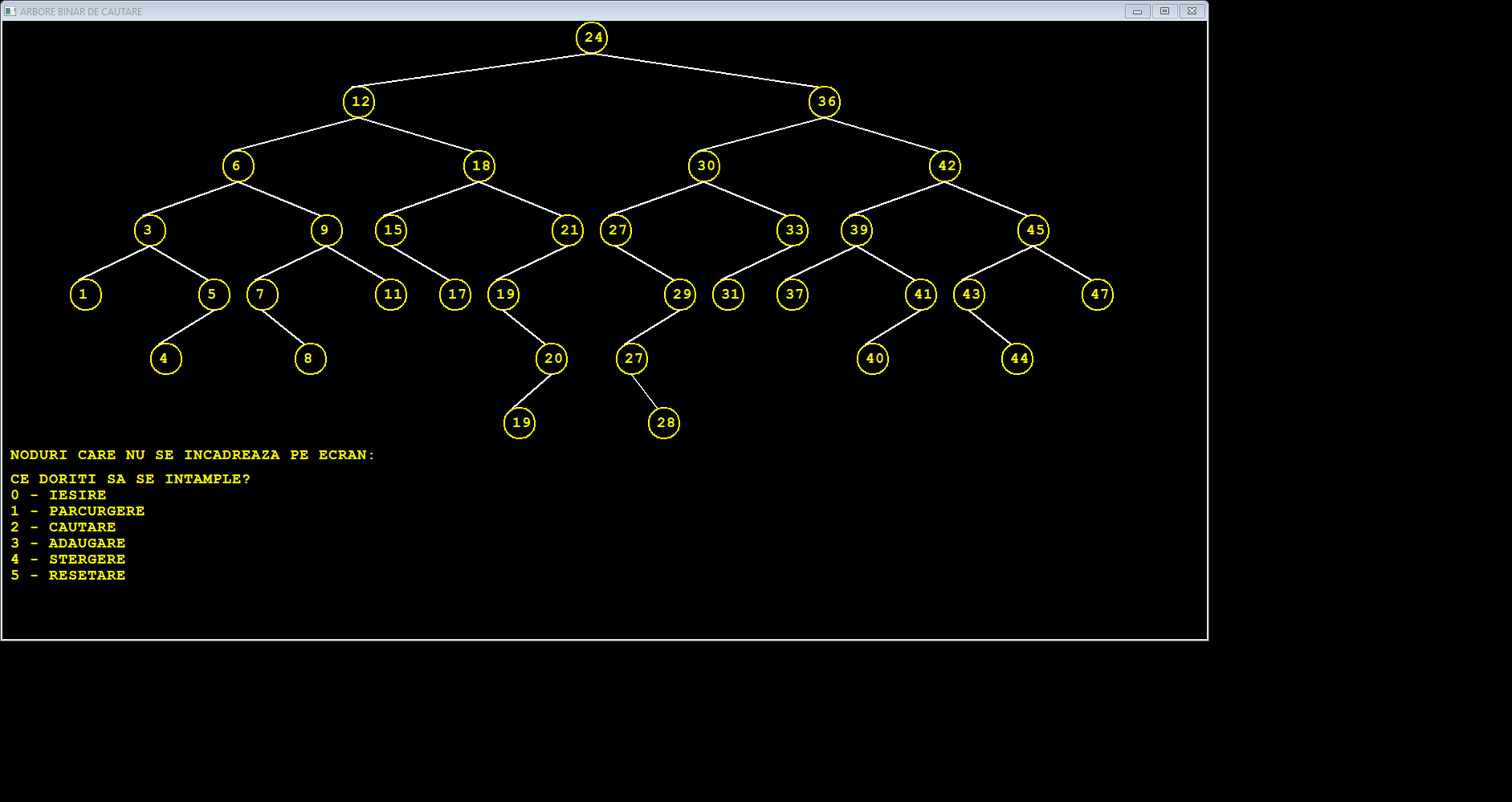
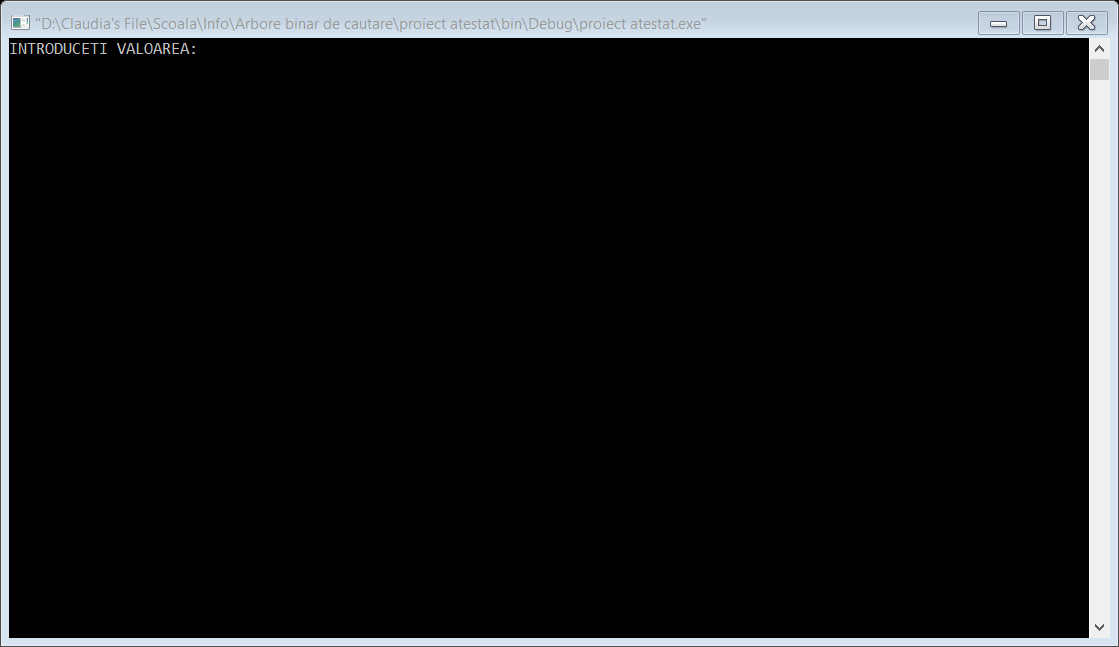
- căutarea unei valori în arbore (tasta 2)

- adăugarea unei valori în arbore (tasta 3)

- ștergerea unei valori din arbore (tasta 4)

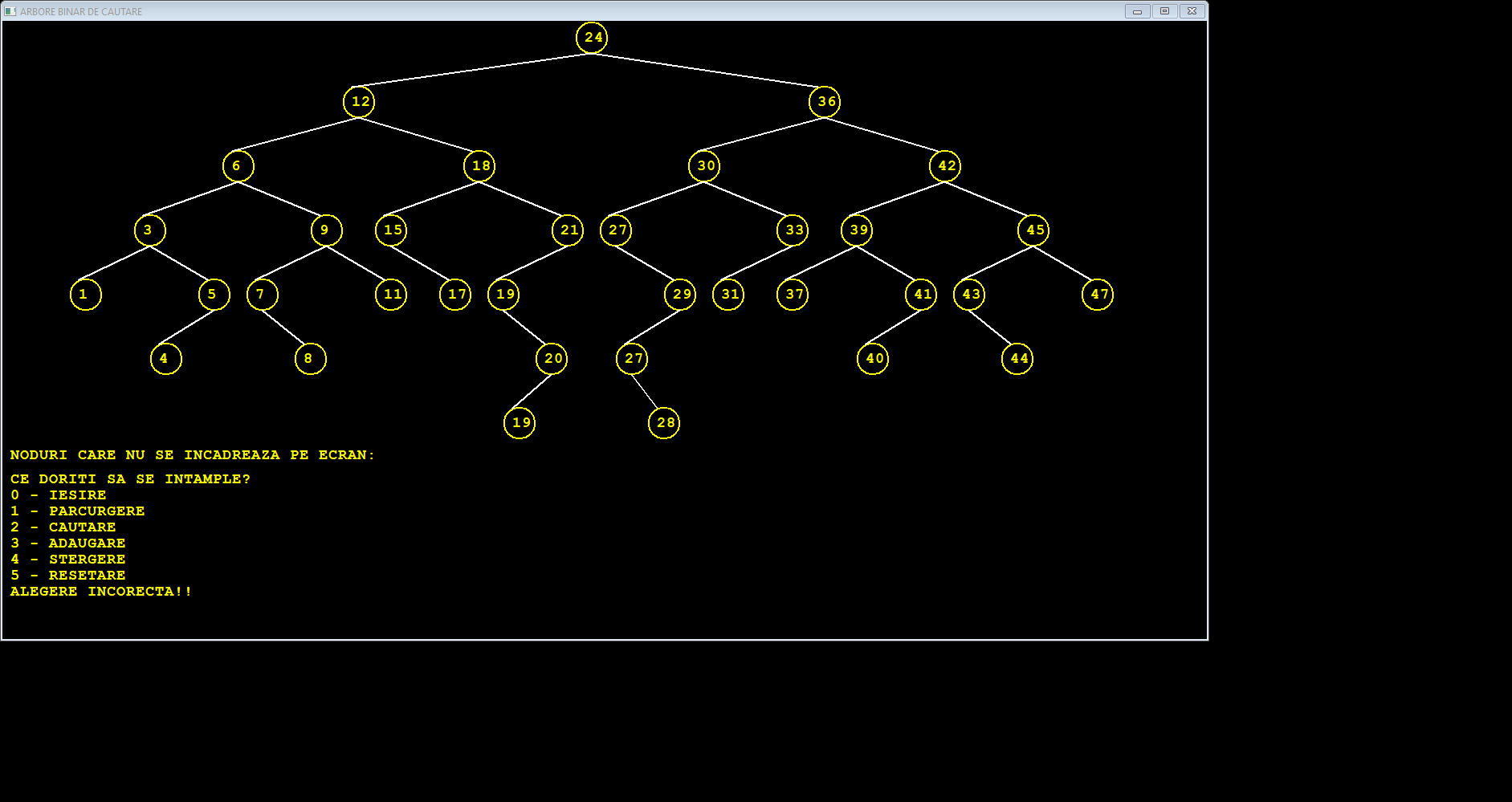
- revenirea la arborele inițial / resetare (tasta 5)

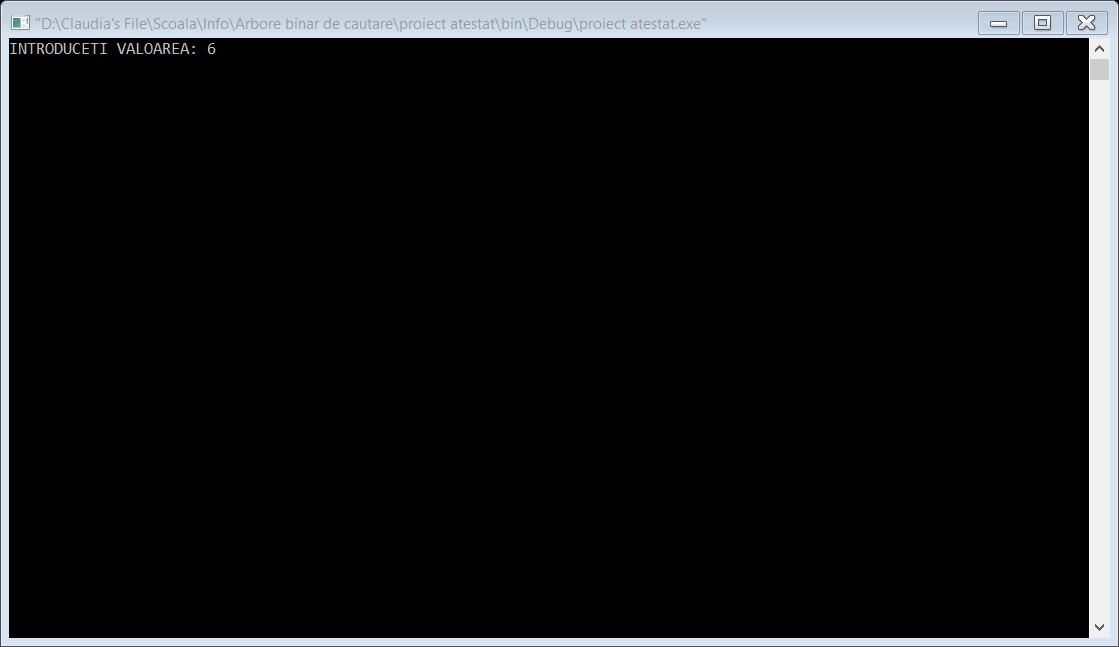
Odată cu apariția meniului apare și ecranul de control unde vor fi introduse valorile corespunzătoare acțiunilor dorite.

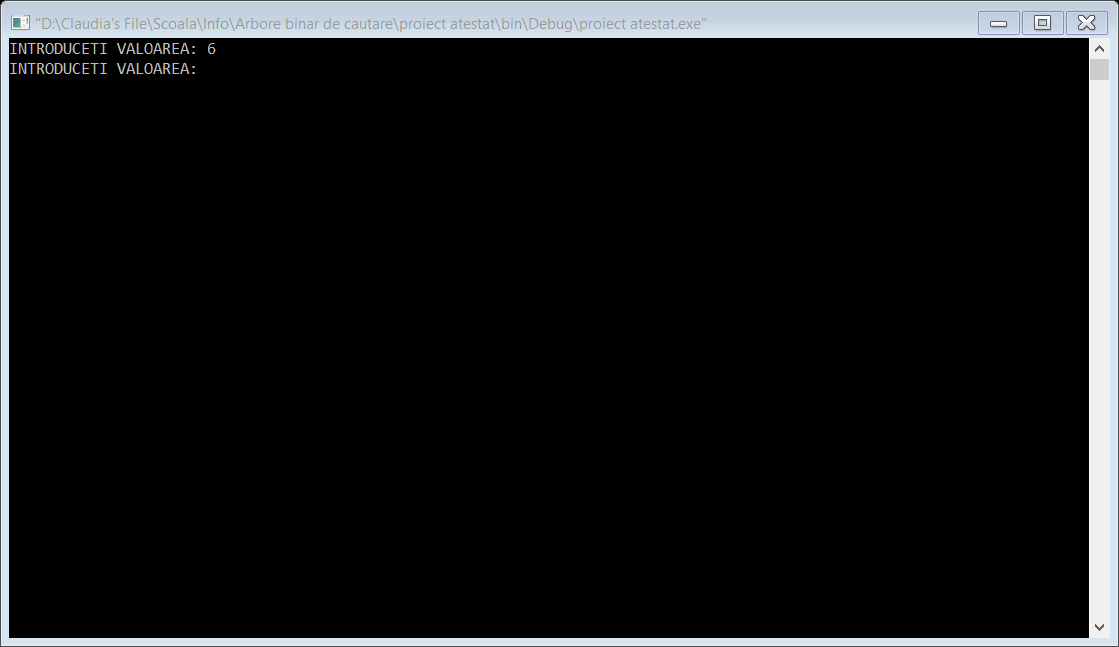


Meniu Ecran de control

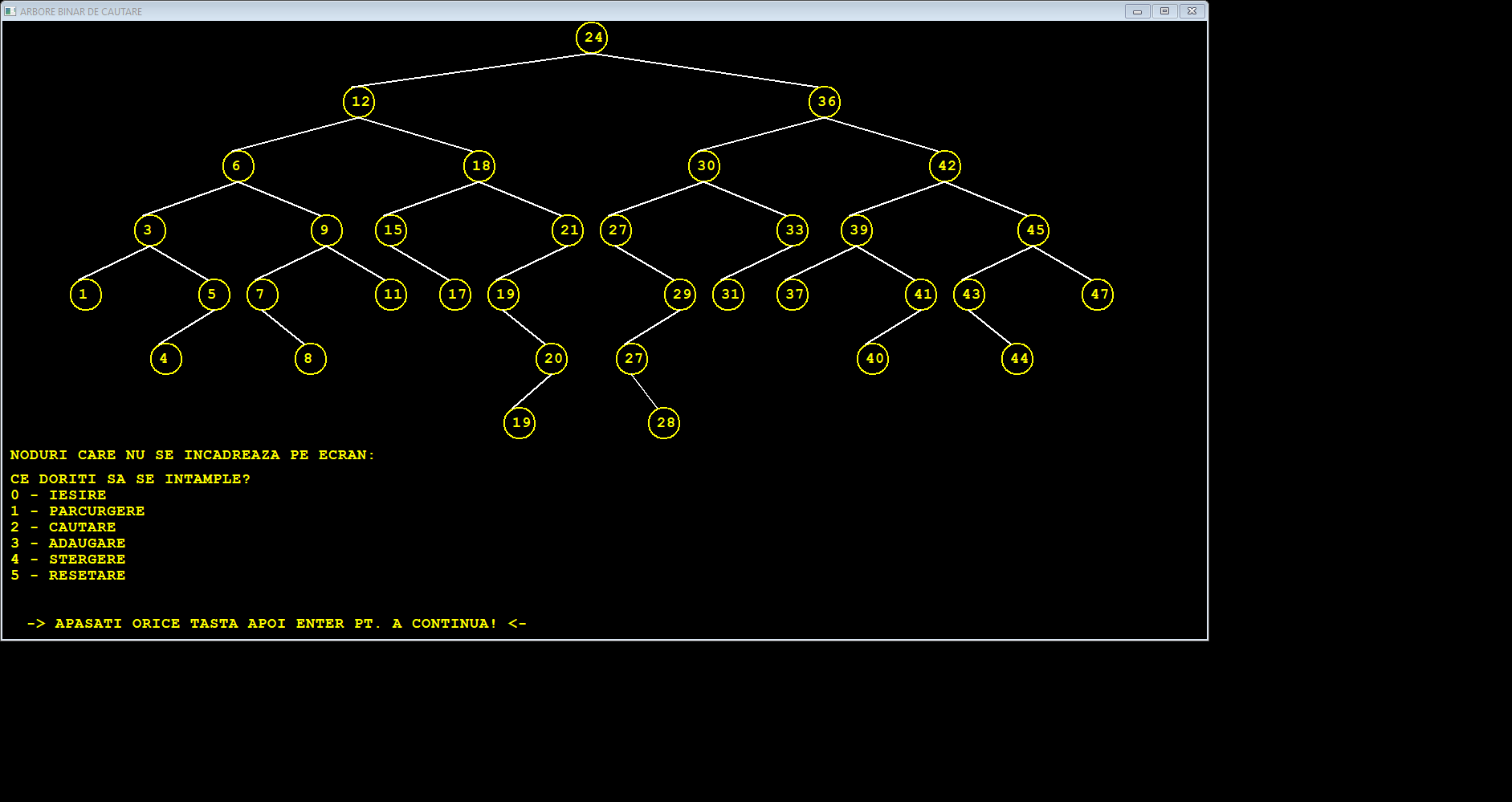
Dacă valoarea introdusă nu corespunde cu nici o valoare din meniu, atunci pe ecran va apărea mesajul „ALEGERE INCORECTĂ!!”, iar programul va aștepta o noua introducere a unei valori existente în meniu.



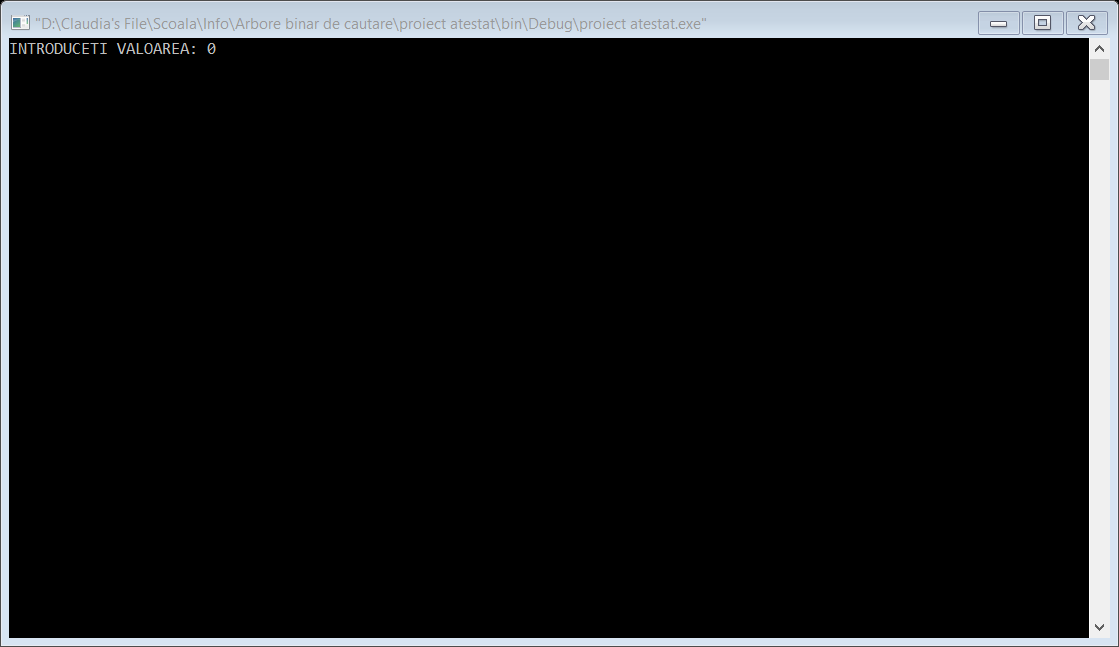
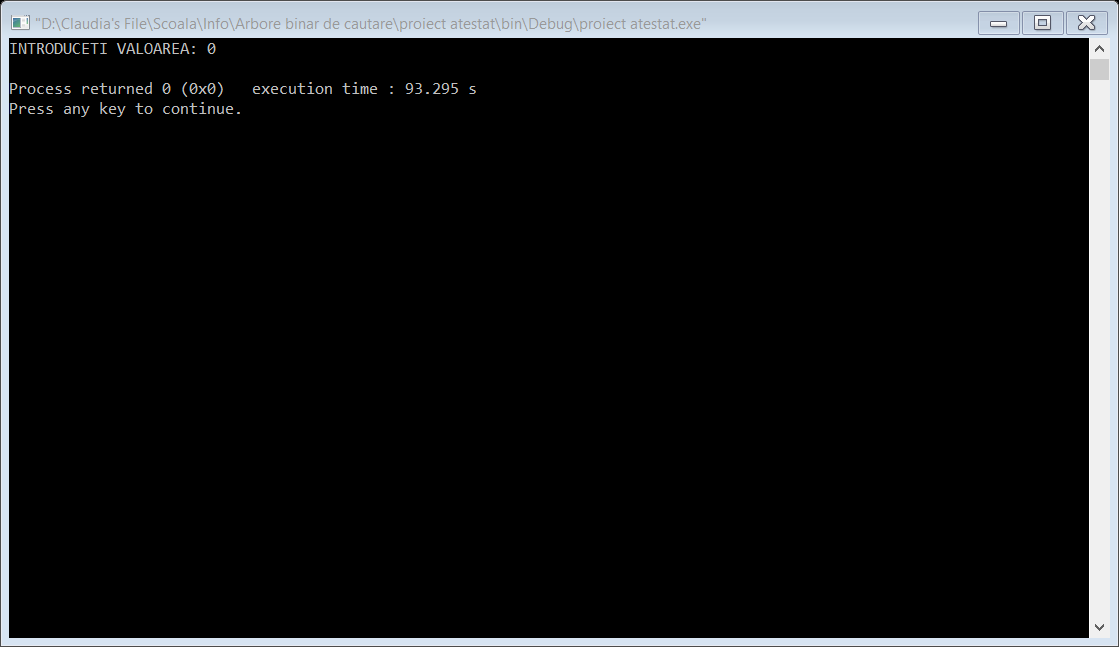




După ce o comandă este efectuată pe ecran apare mesajul: „APĂSAȚI ORICE TASTĂ APOI ENTER PT. A CONTINUA!” (cu excepția cazului când se dorește ieșirea din program), iar după efectuarea acestui pas, utilizatorul poate sa introducă o noua valoare în ecranul de control.



Programul continuă astfel până în momentul când se dorește ieșirea din acesta prin introducerea valorii 0 în ecranul de control.

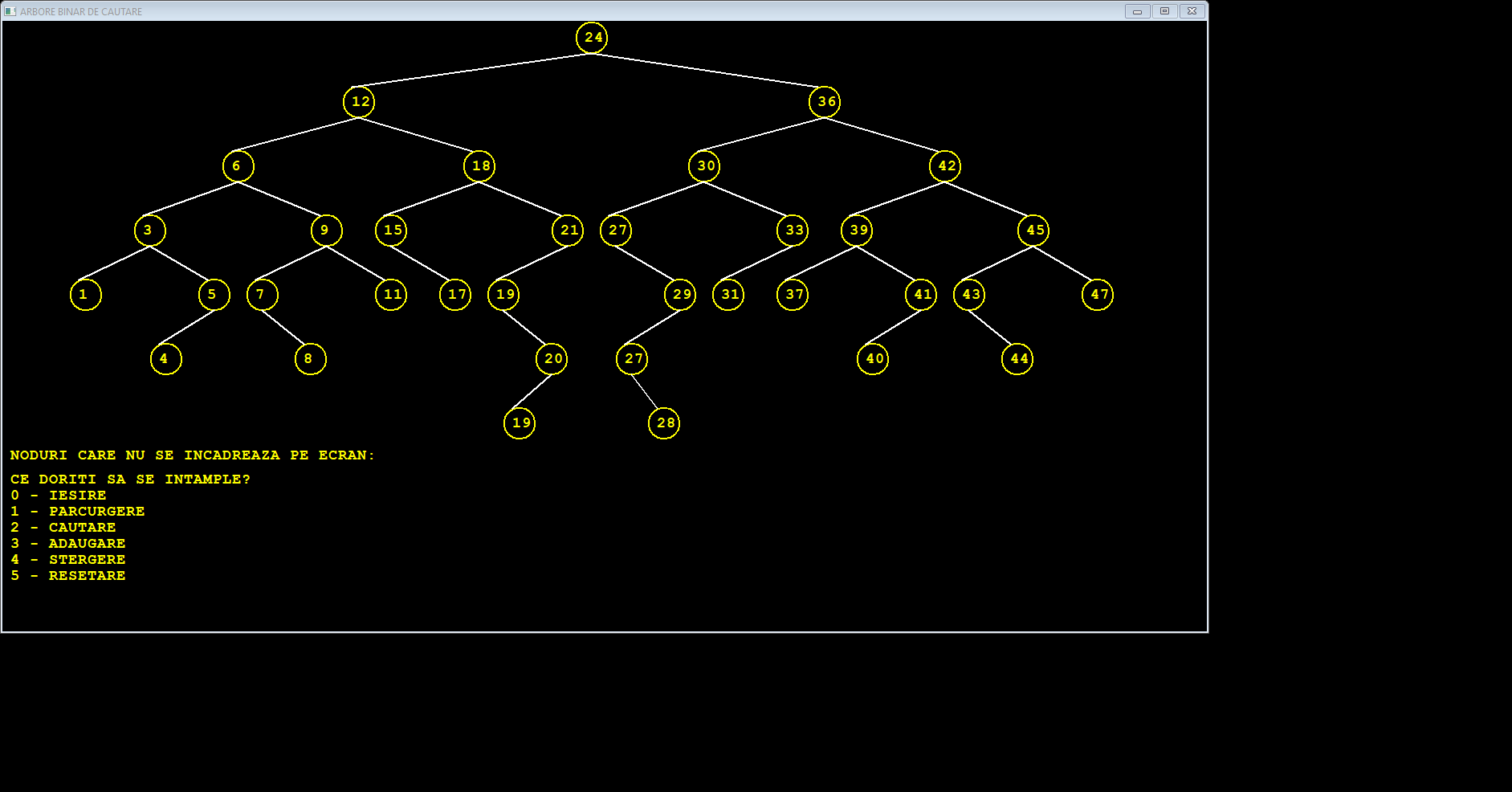


**Crearea unui arbore binar**

Programul creează dinamic un arbore binar cu valorile date de la tastatură, după următoarea regulă:

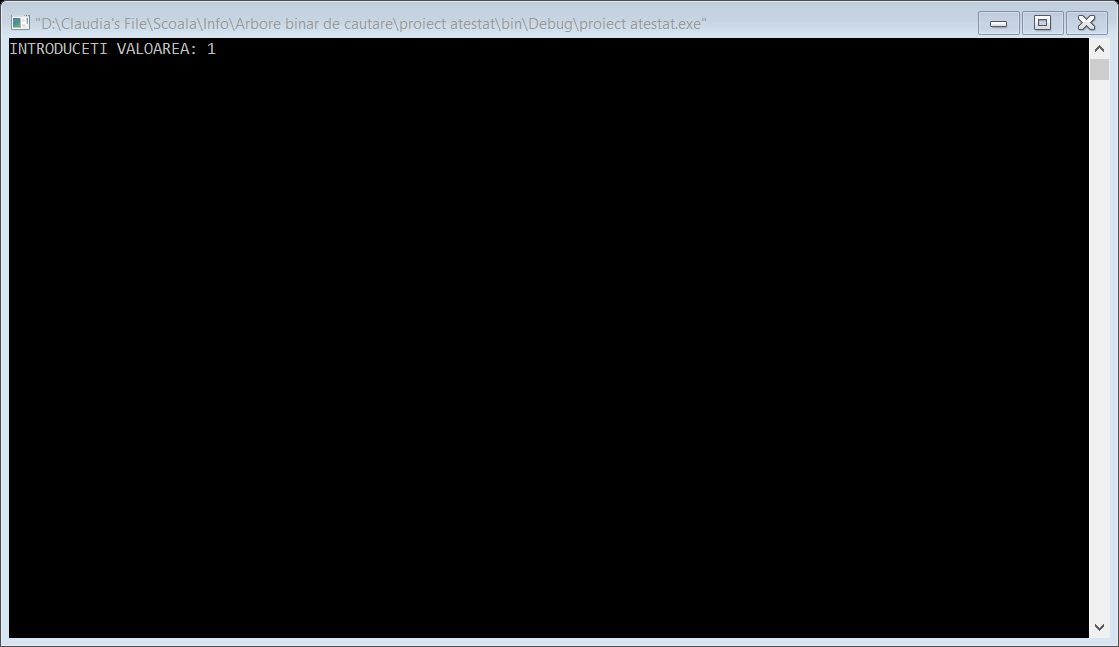
- informația descendentului stâng este mai mică decât aceea a ascendentului direct

- informația descendentului drept este mai mare decât aceea a ascendentului direct

Ex: pentru valorile 24 12 36 6 18 30 42 3 9 15 21 27 33 39 45 1 5 7 11 17 19 29 31 37 41 43 47 4 8 20 27 40 44 19 28 programul va crea următorul arbore:

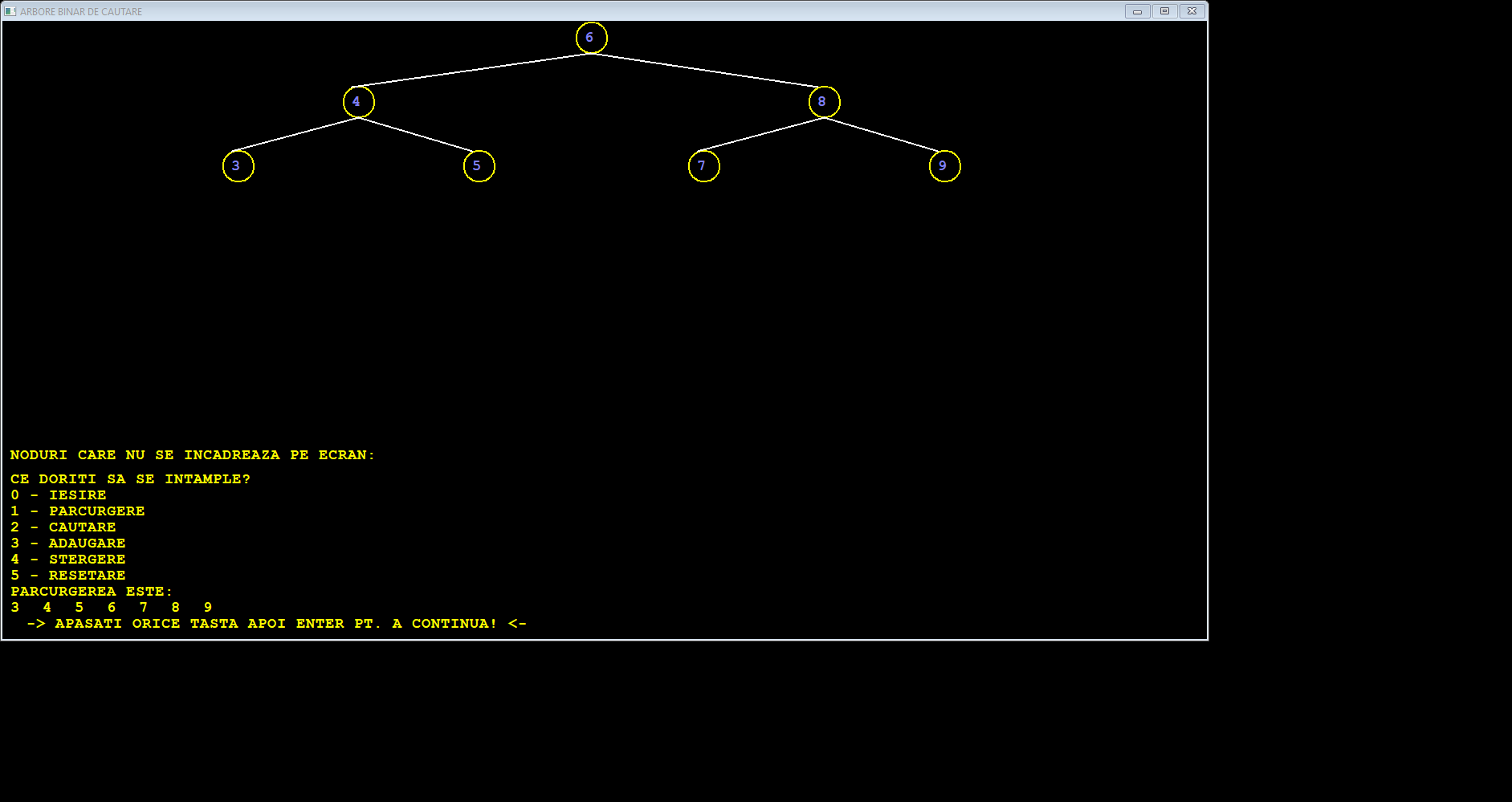
**Parcurgerea arborelui**

Pentru a se afișa valorile din arbore în ordine crescătoare se efectuează o parcurgere SRD (stânga-rădăcină-dreapta) a arborelui prin introducerea valorii 1 în ecranul de control.

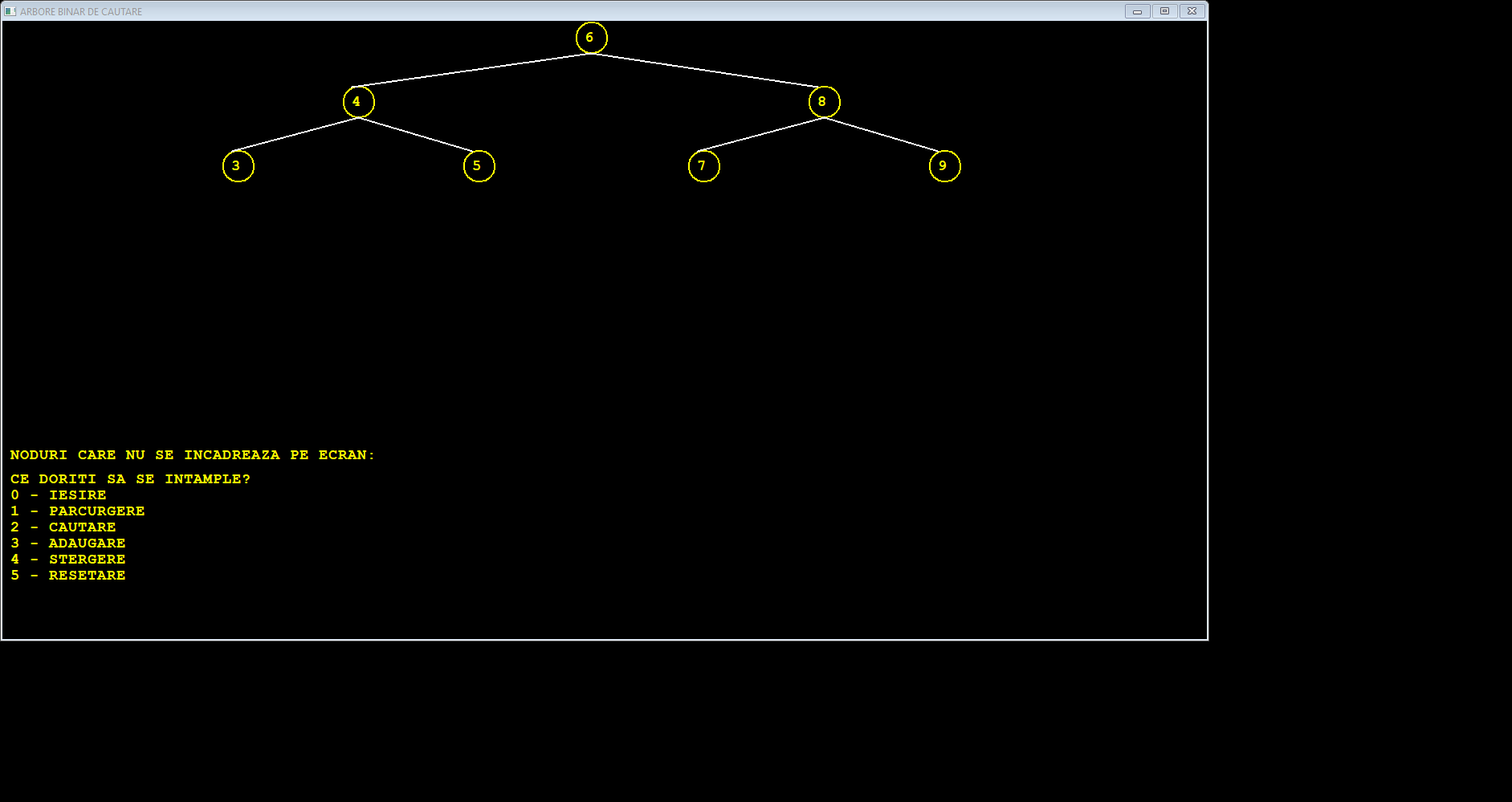
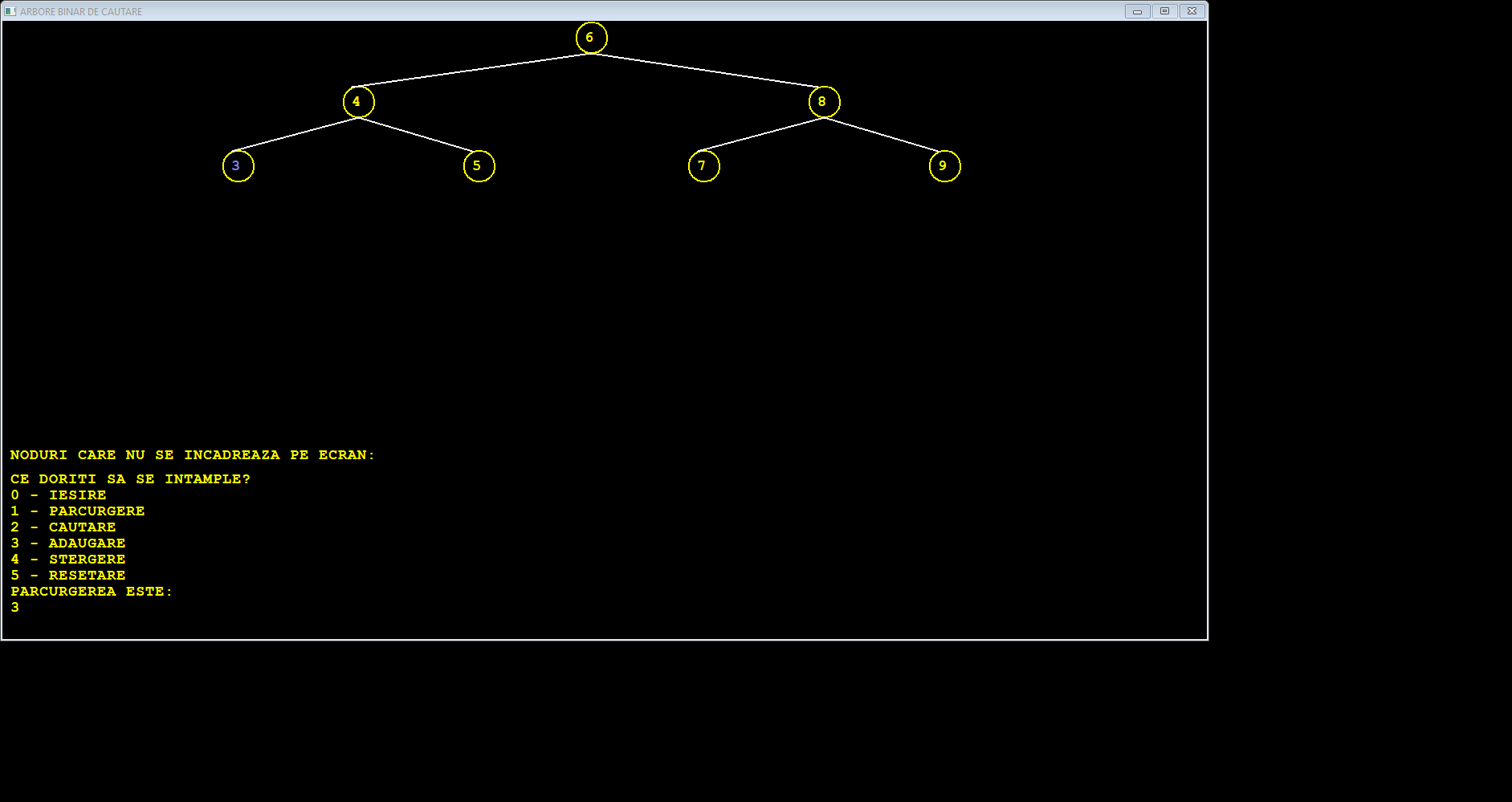


Pe măsură ce parcurge arborele, programul colorează în albastru nodurile parcurse si le afișează în ordine crescătoare în josul ecranului în dreptul mesajului „PARCURGEREA ESTE:”.

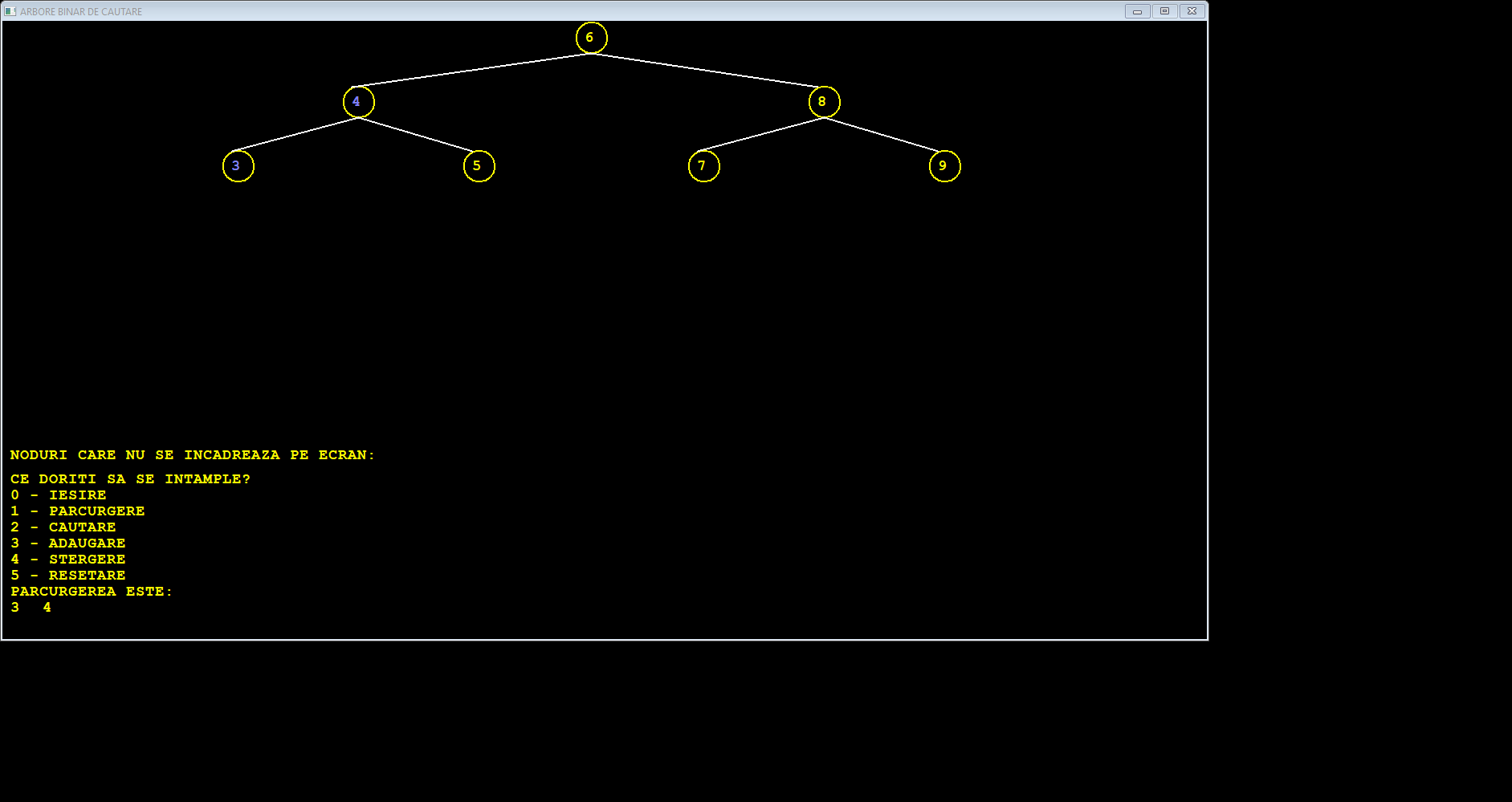
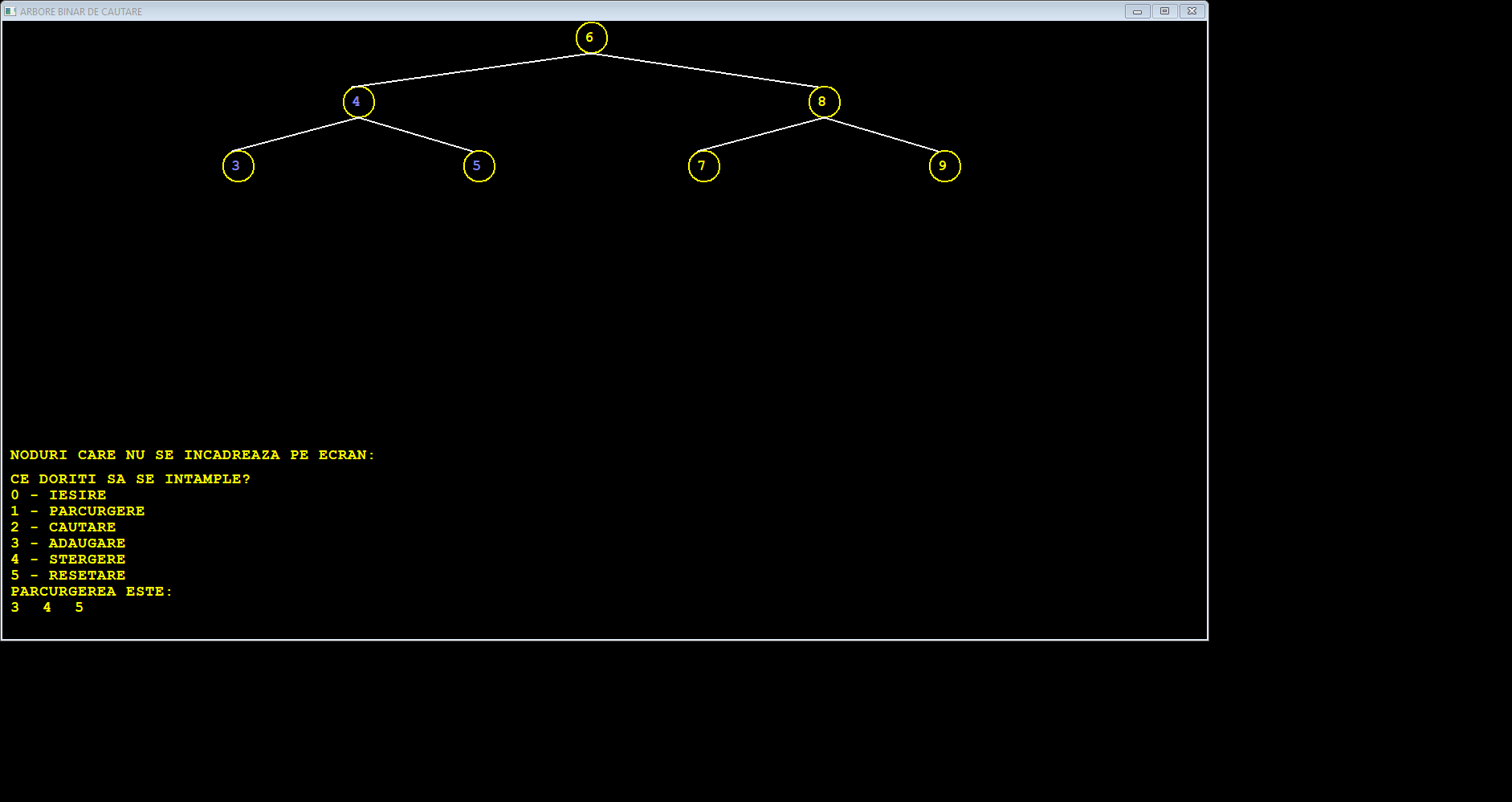
Ex: pentru valorile 6 4 8 3 5 7 9 programul va afișa:



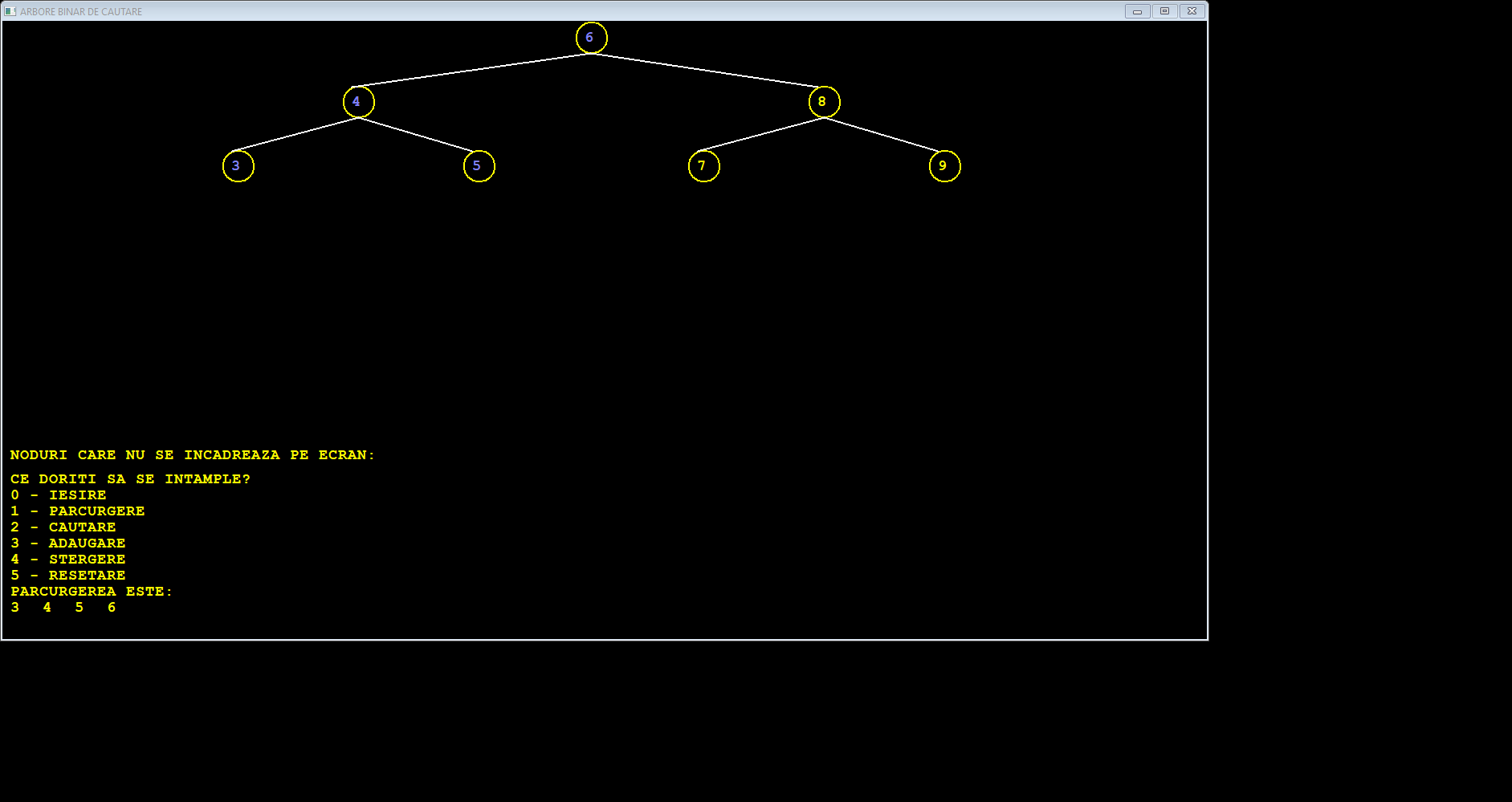
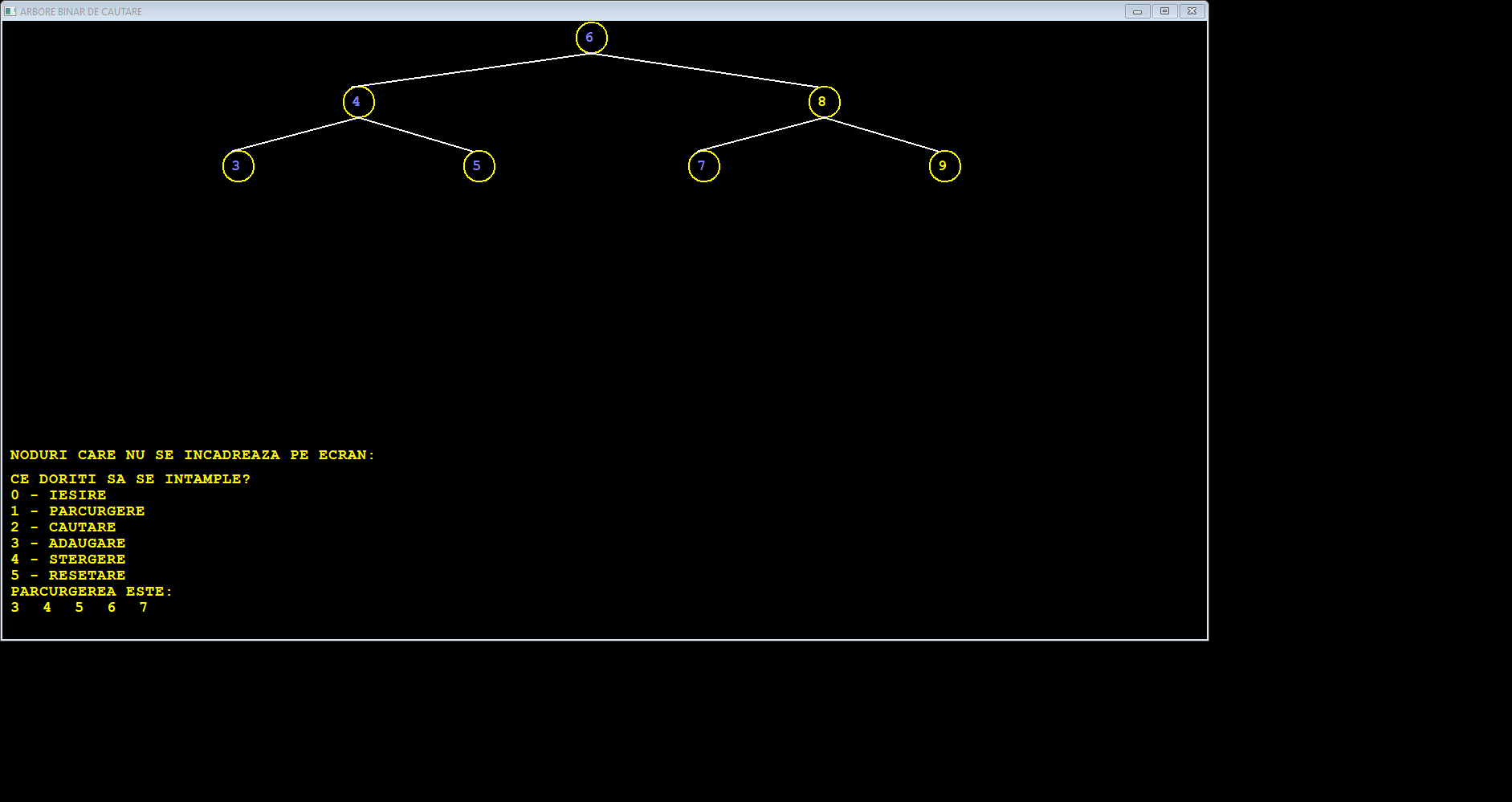
după cum urmează:



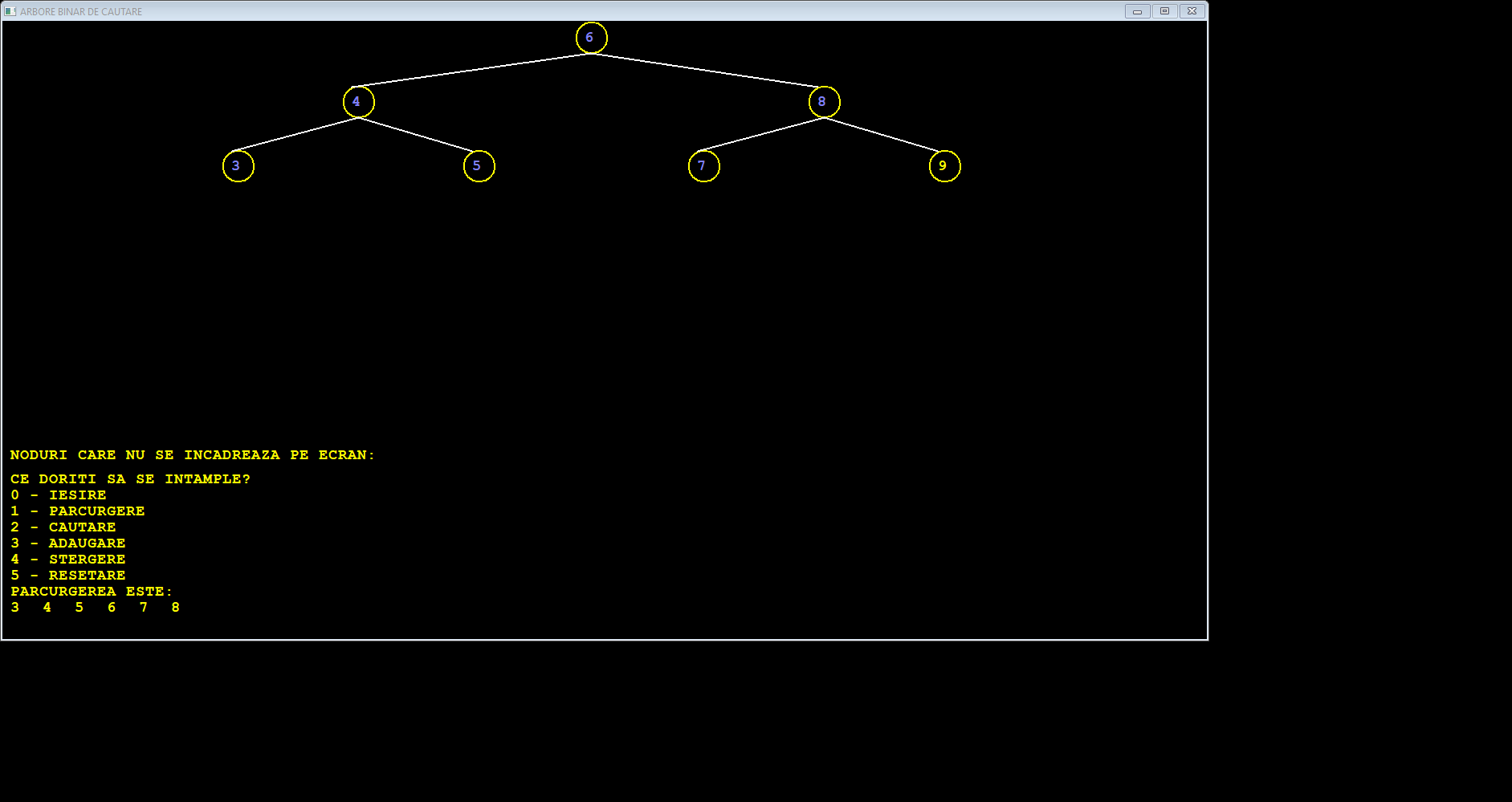
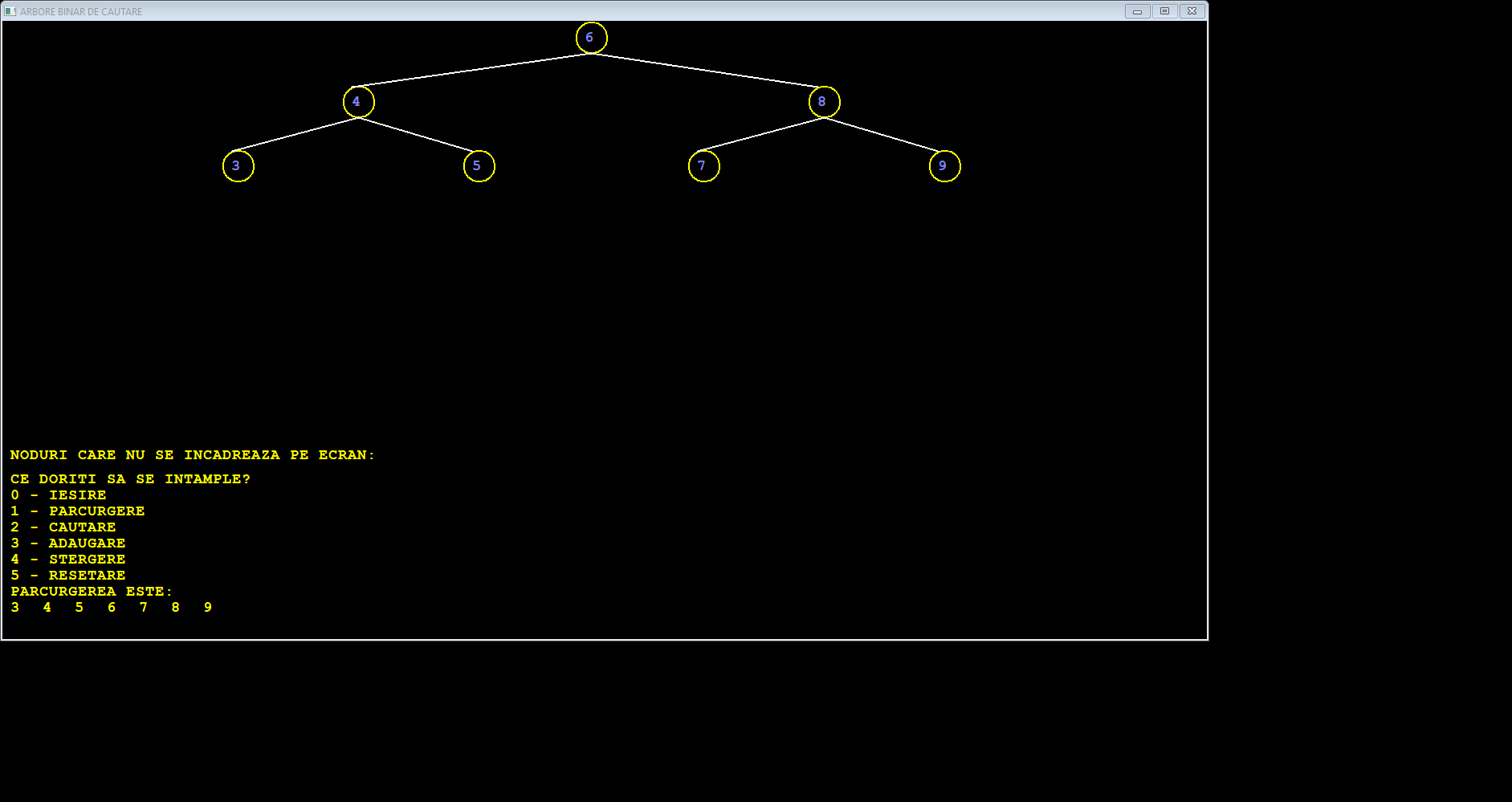
Scrie 3



Scrie 5 Scrie 4



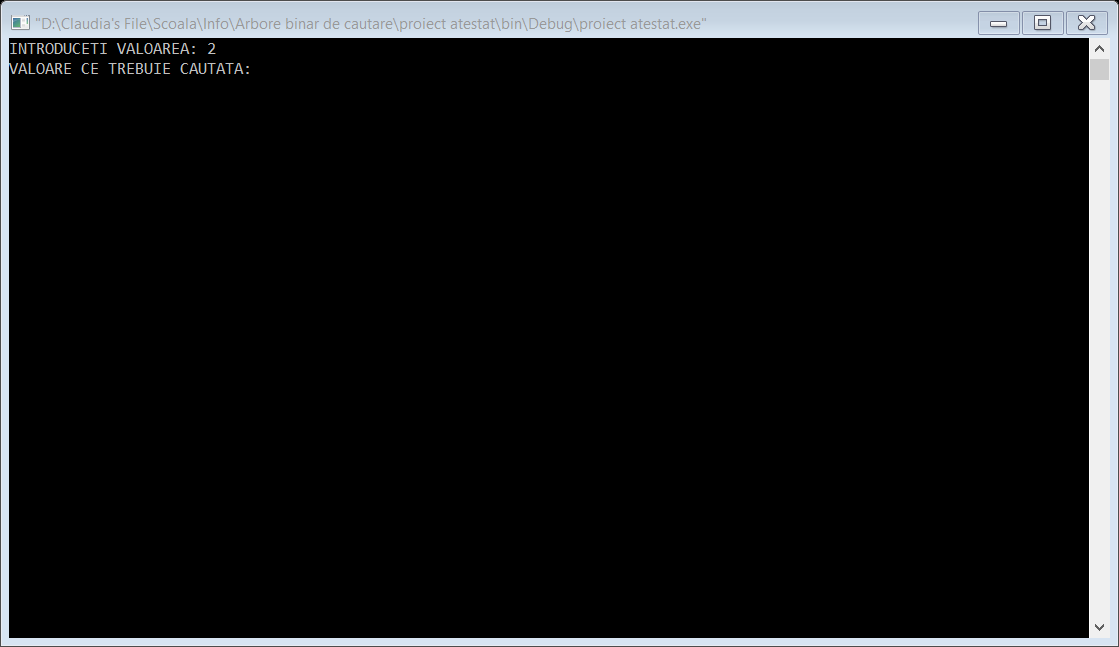
Scrie 6 Scrie 7



Scrie 9 Scrie 8

**Căutarea unei valori în arbore**

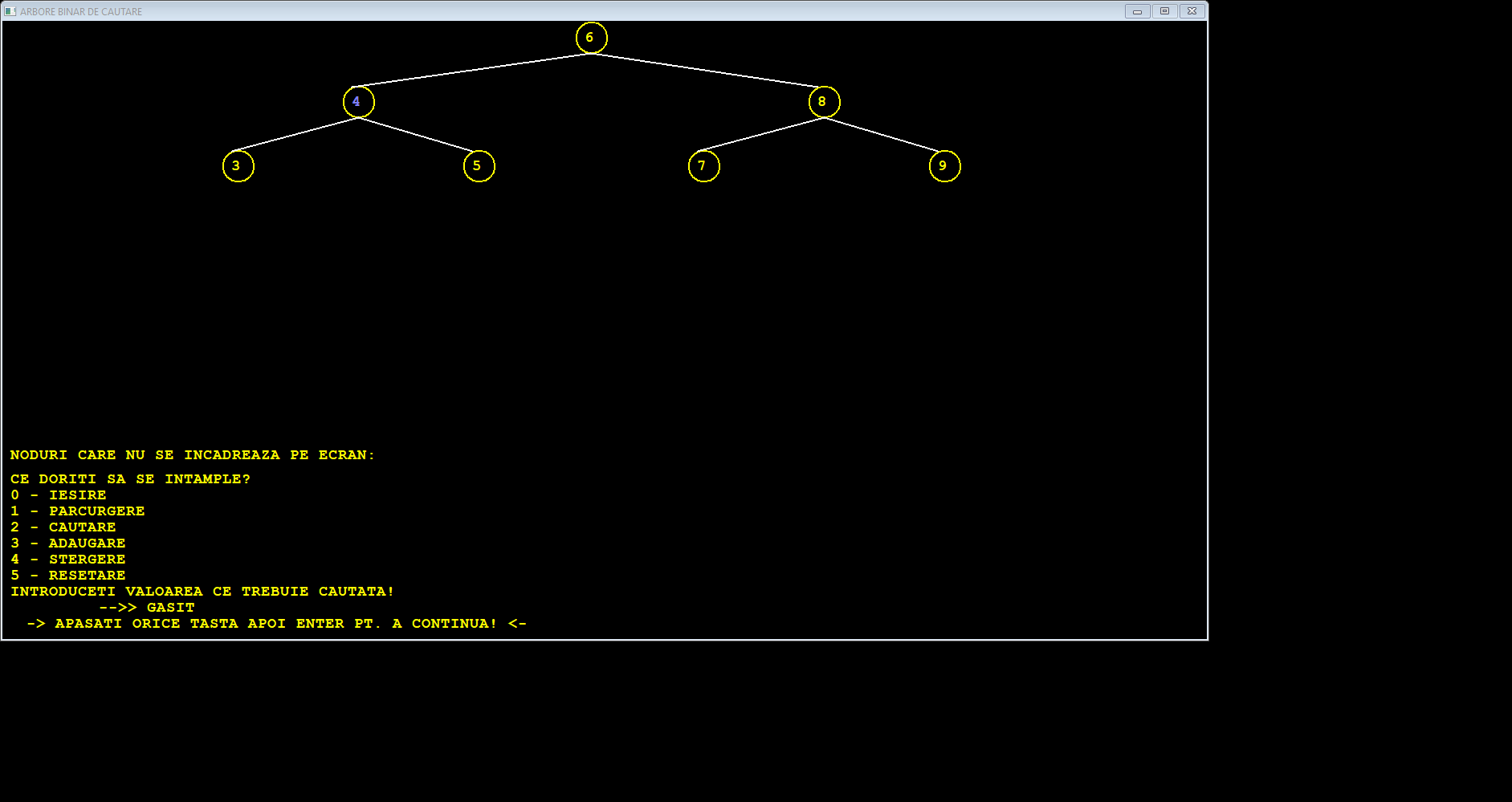
Prin introducerea valorii 2 în ecranul de control, programul va căuta în arbore o valoare introdusă de la tastatură.

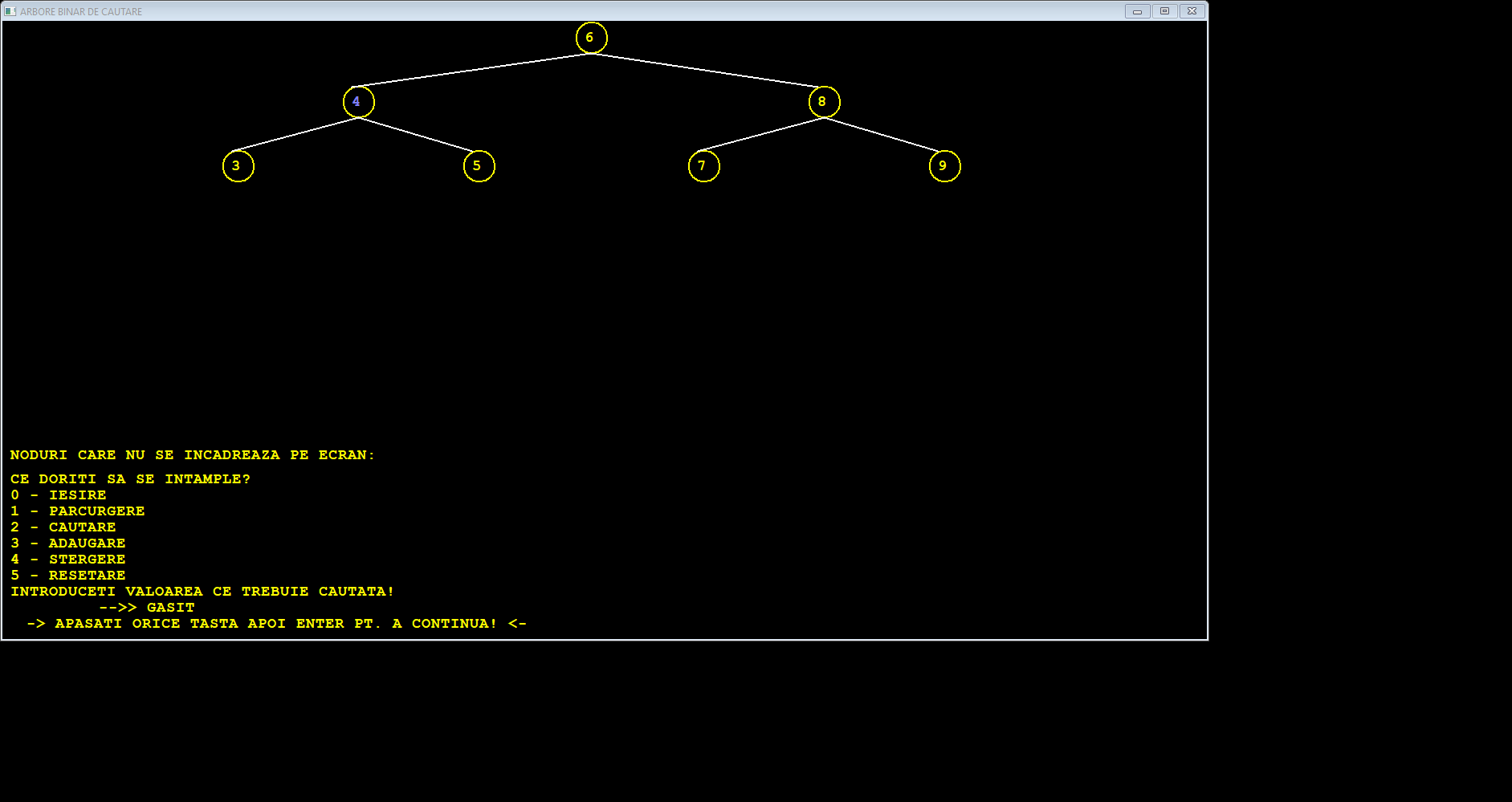


Dacă valoare căutată există în arbore, atunci programul va afișa mesajul „GASIT” și va colora nodul cu valoarea respectivă în albastru.

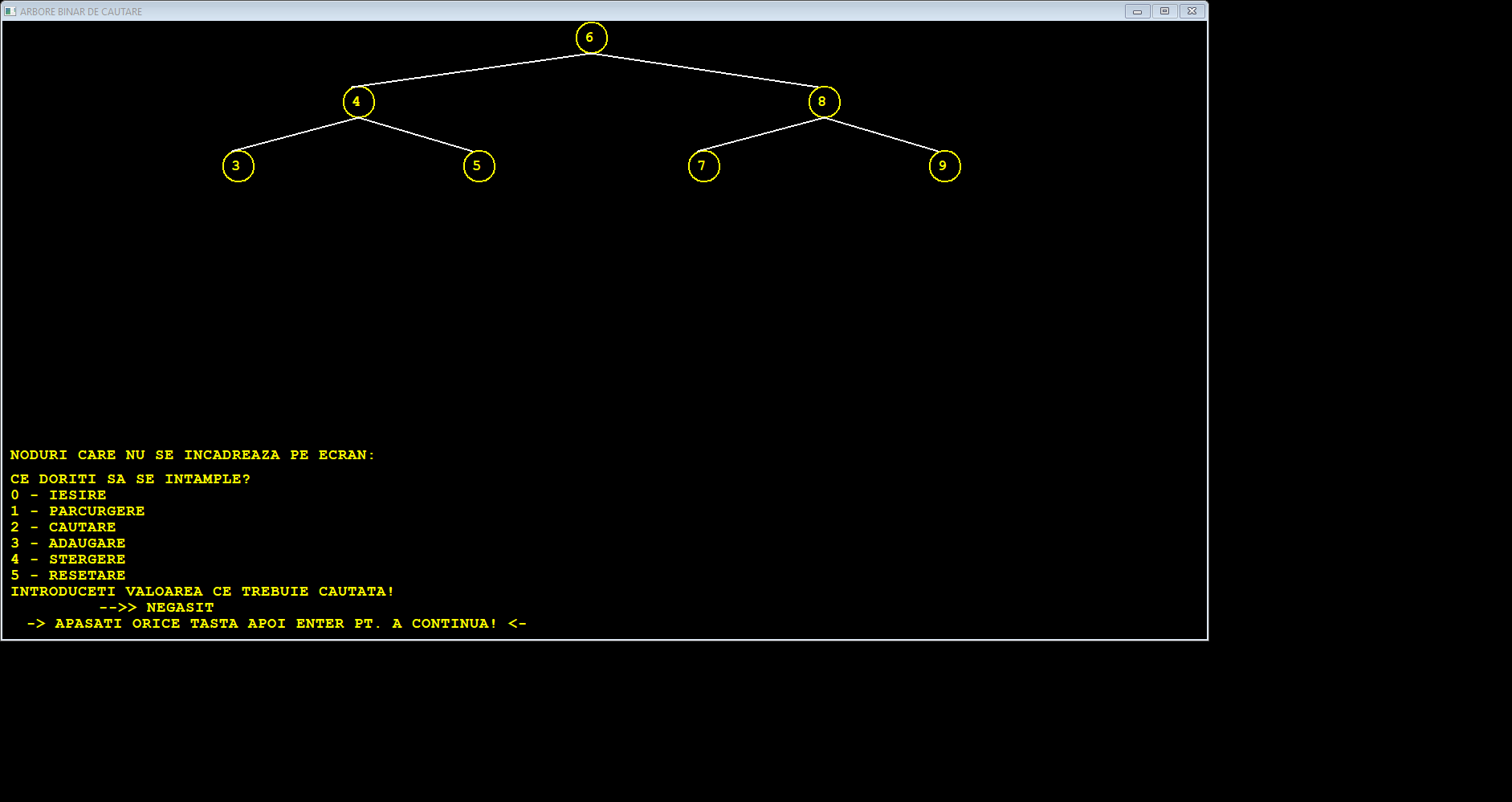
Dacă valoarea căutată nu exista în arbore, atunci se va afișa mesajul „NEGASIT”.

Ex: 1) pentru valorile 6 4 8 3 5 7 9 și valoarea căutată 4 se va afișa:



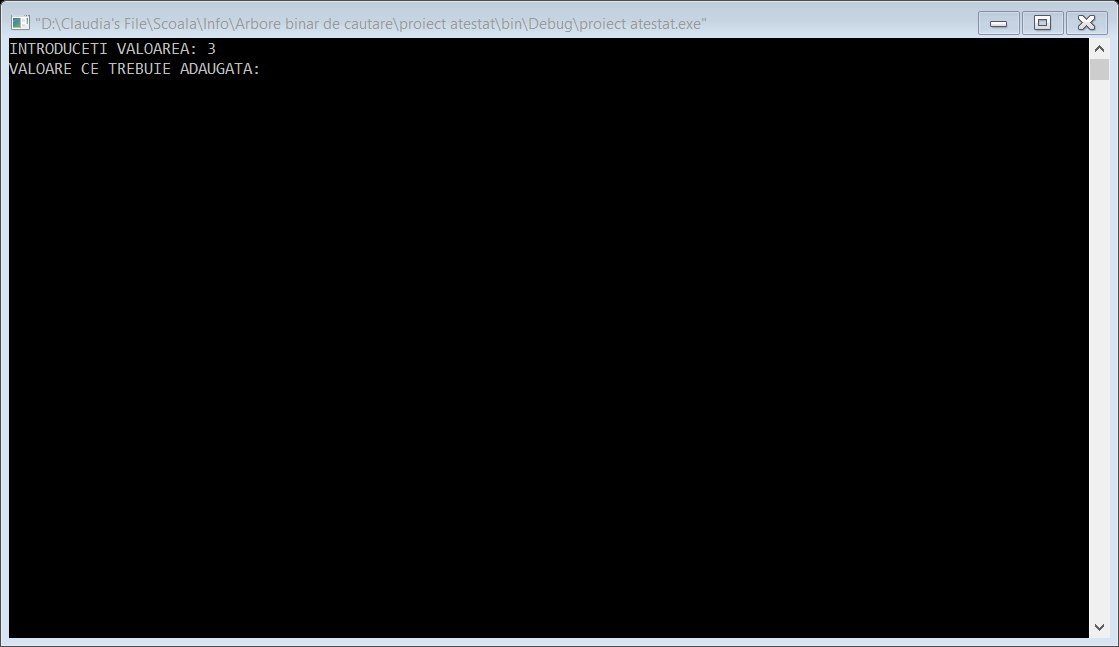


2) pentru valorile 6 4 8 3 5 7 9 și valoarea căutată 10 se va afișa:

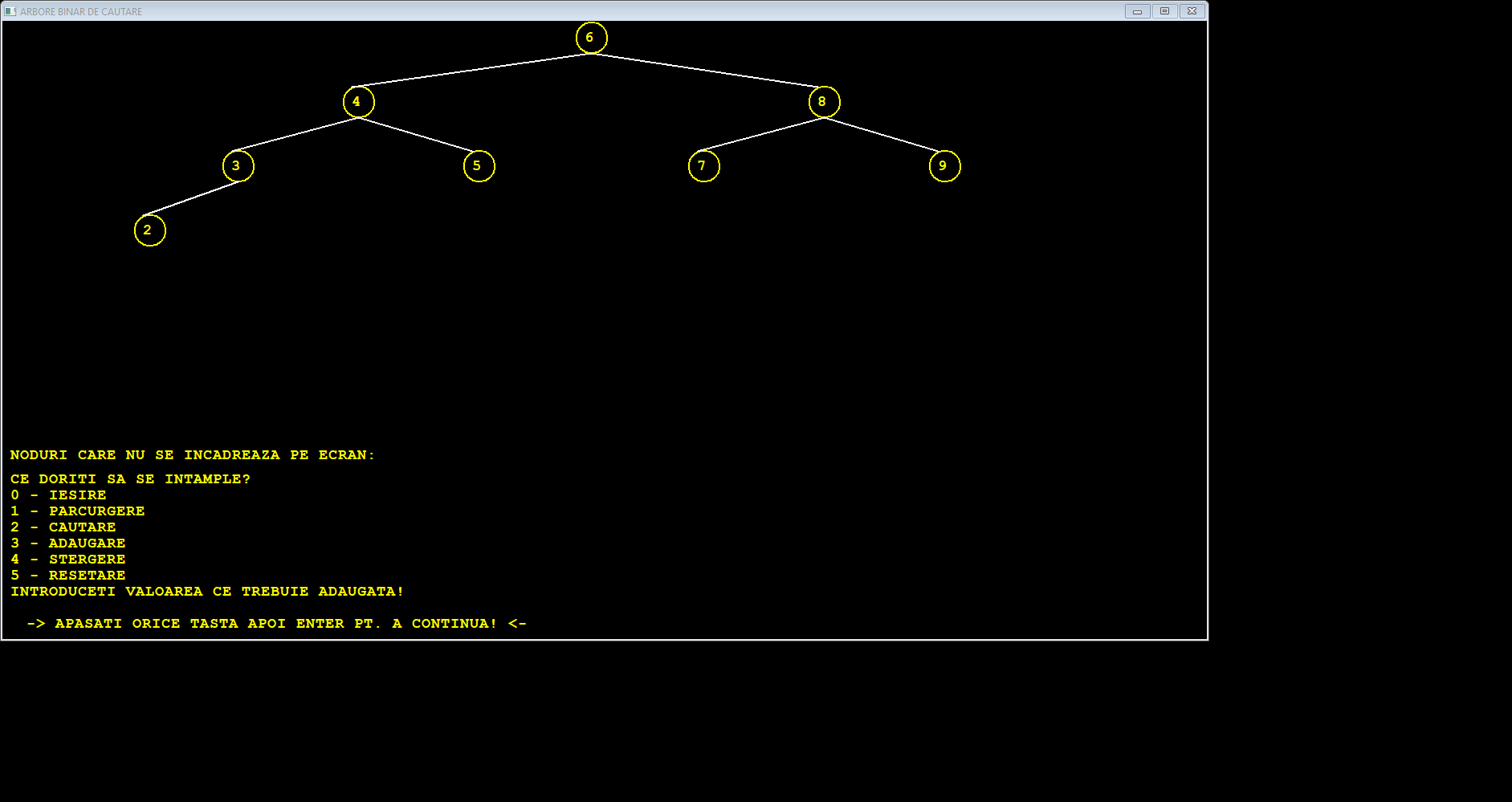
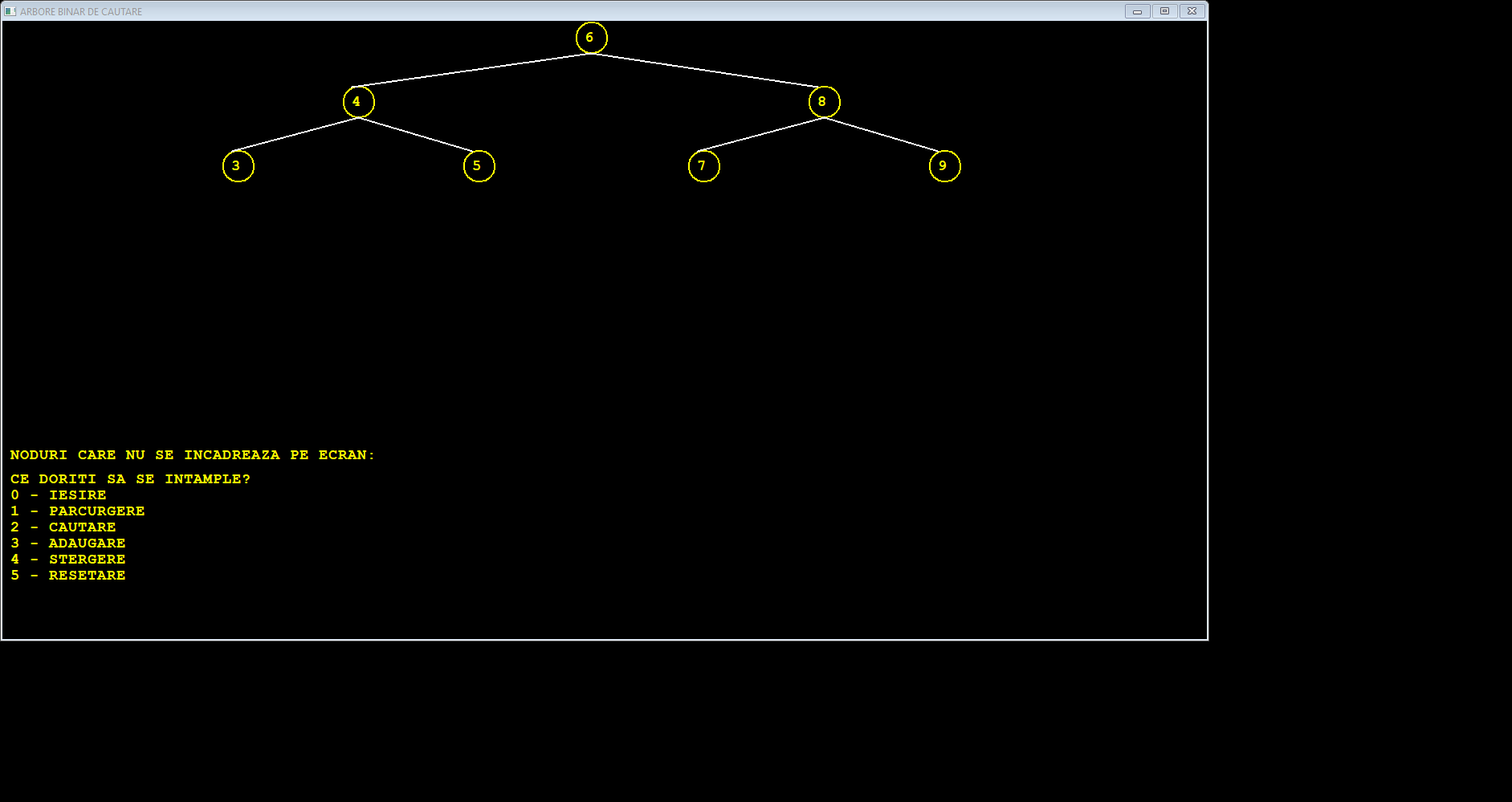


**Adăugarea unei valori în arbore**

Programul introduce în arbore o valoare dată de la tastatură în urma introducerii valorii 2 în ecranul de control împreună cu valoarea ce trebuie adăugată.

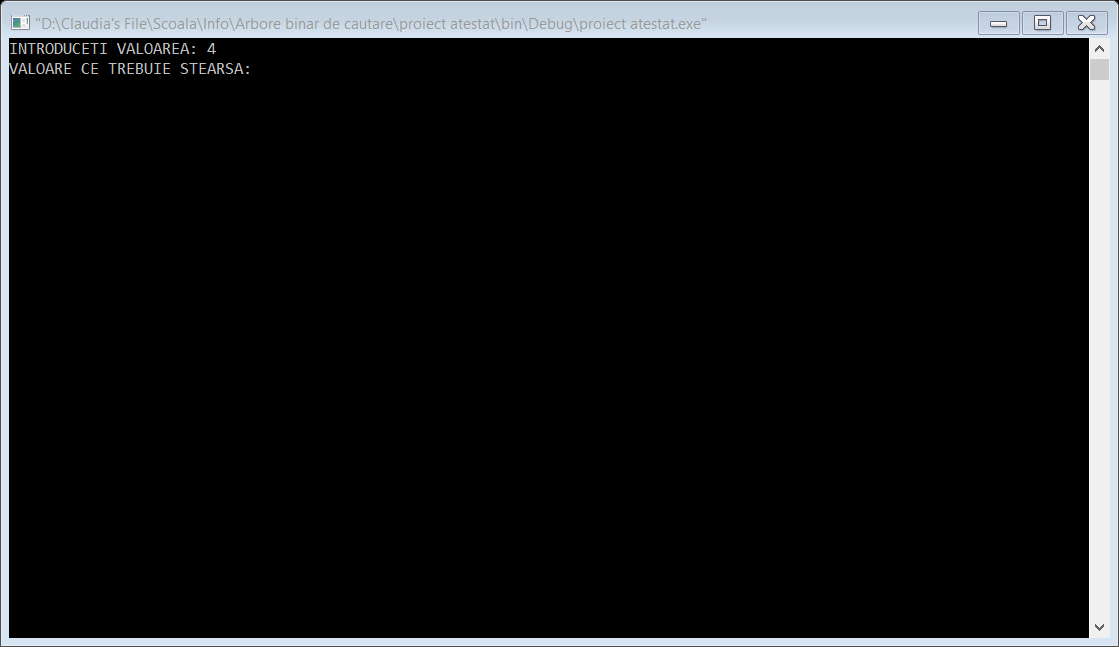


Ex: pentru valorile 6 4 8 3 5 7 9 și valoarea de adăugat 2 se va afișa:



**Ștergerea unei valori din arbore**

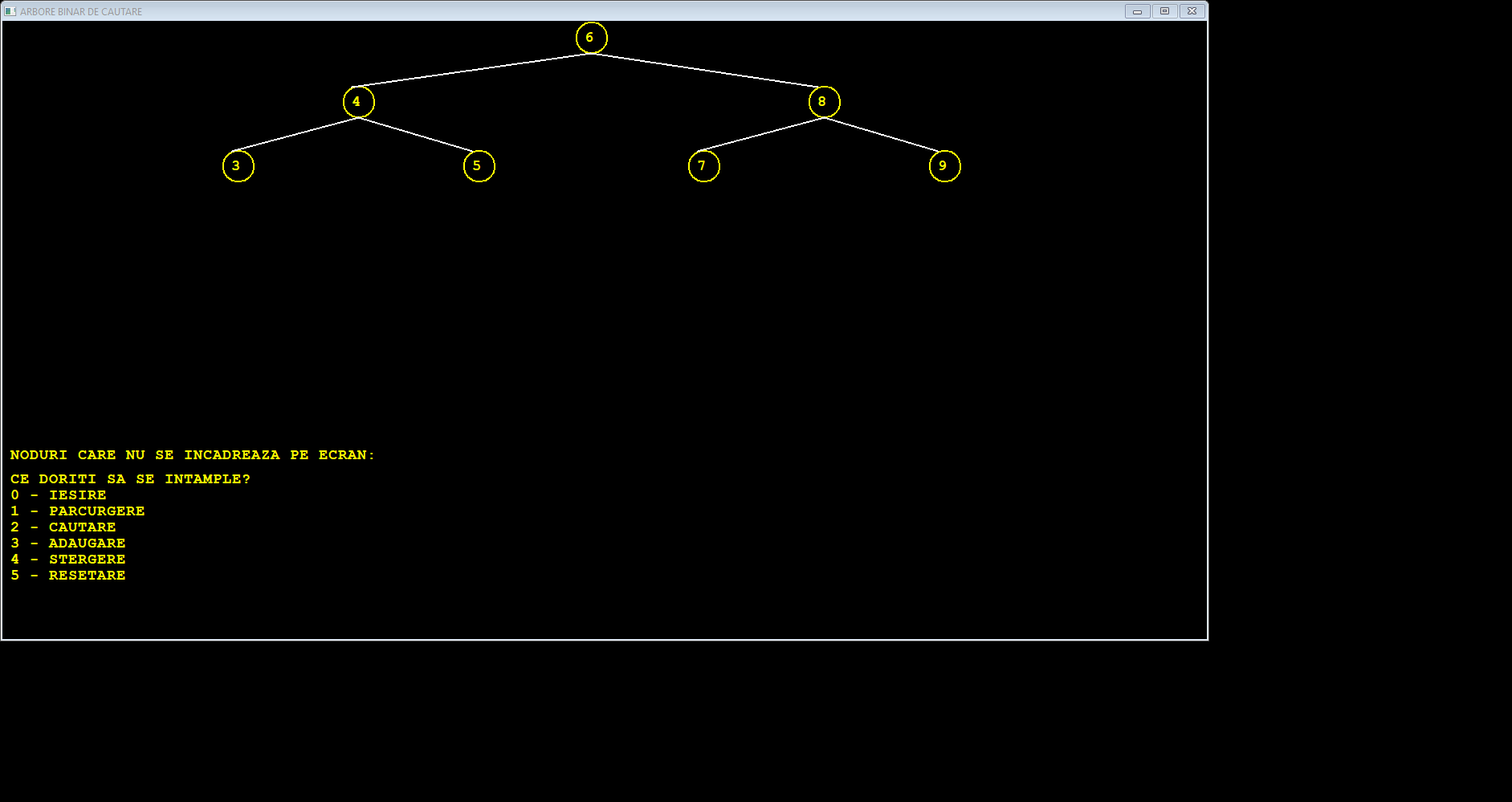
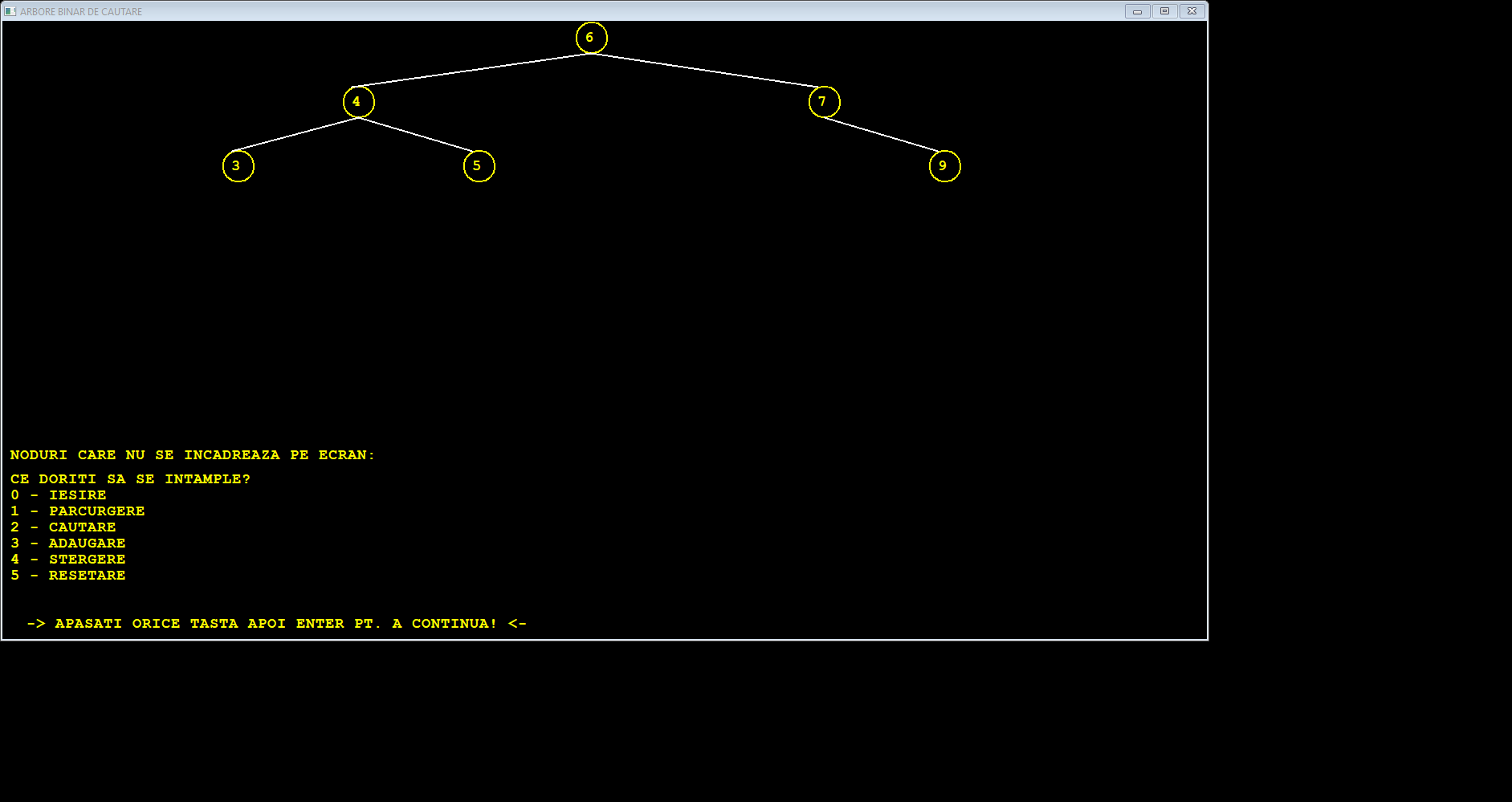
Eliminarea unui nod se realizează prin introducerea valorii 4 în ecranul de control și a valorii ce trebuie ștearsă.



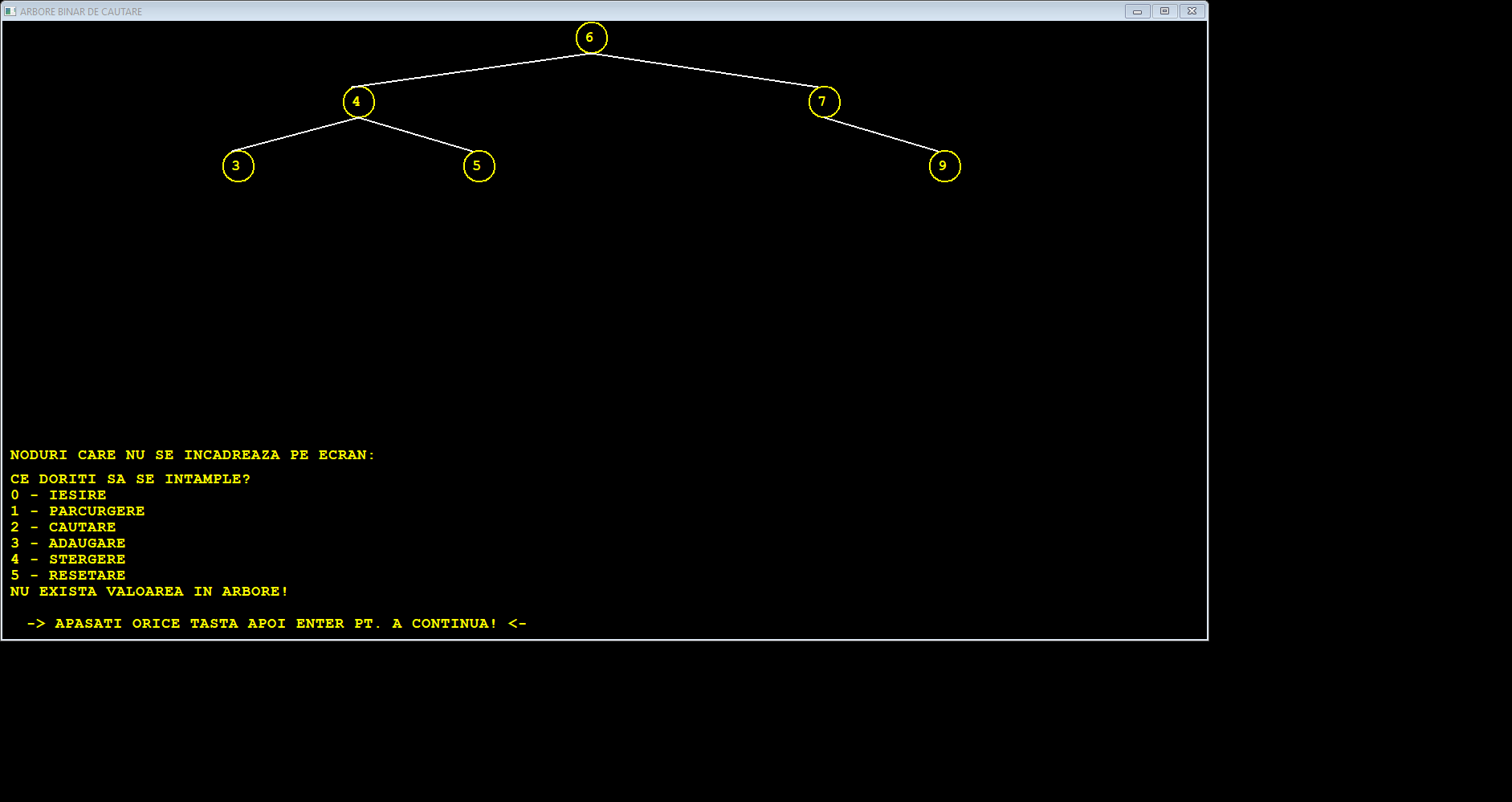
Dacă valoarea dată de la tastatură există în arbore atunci programul va șterge nodul cu valoarea respectivă din arbore și va afișa arborele modificat.

Dacă valoarea introdusă nu există în arbore atunci programul va afișa mesajul „NU EXISTĂ VALOAREA ÎN ARBORE”

Ex: 1) pentru valorile 6 4 8 3 5 7 9 și valoarea de șters 8 se va afișa:

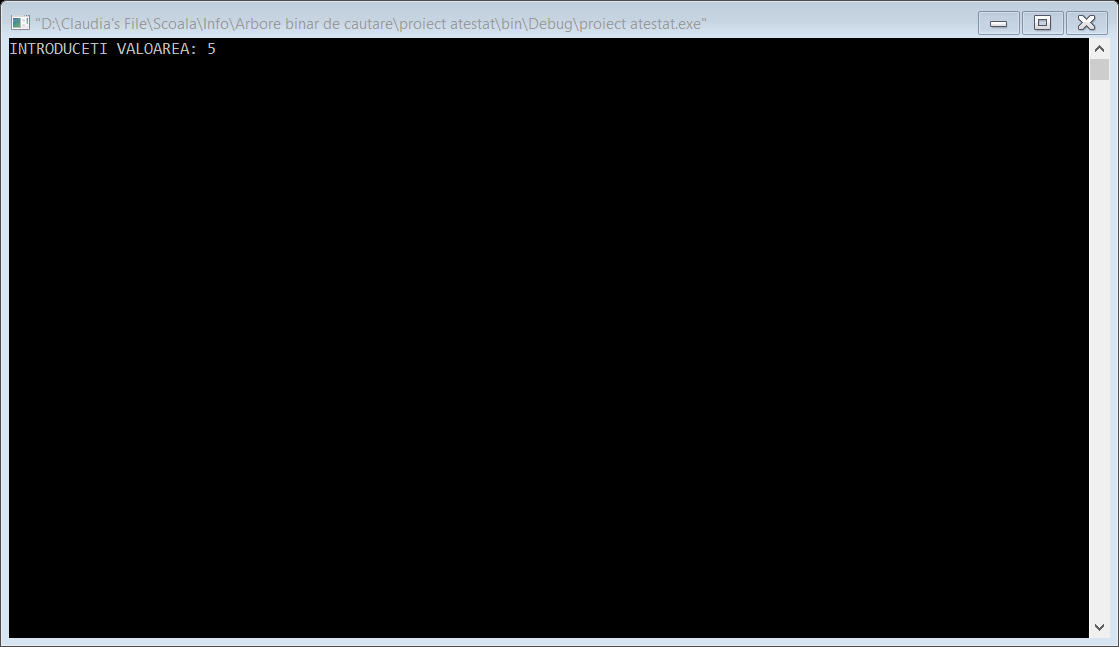


2) pentru valorile 6 4 8 3 5 7 9 și valoarea de șters 11 se va afișa:

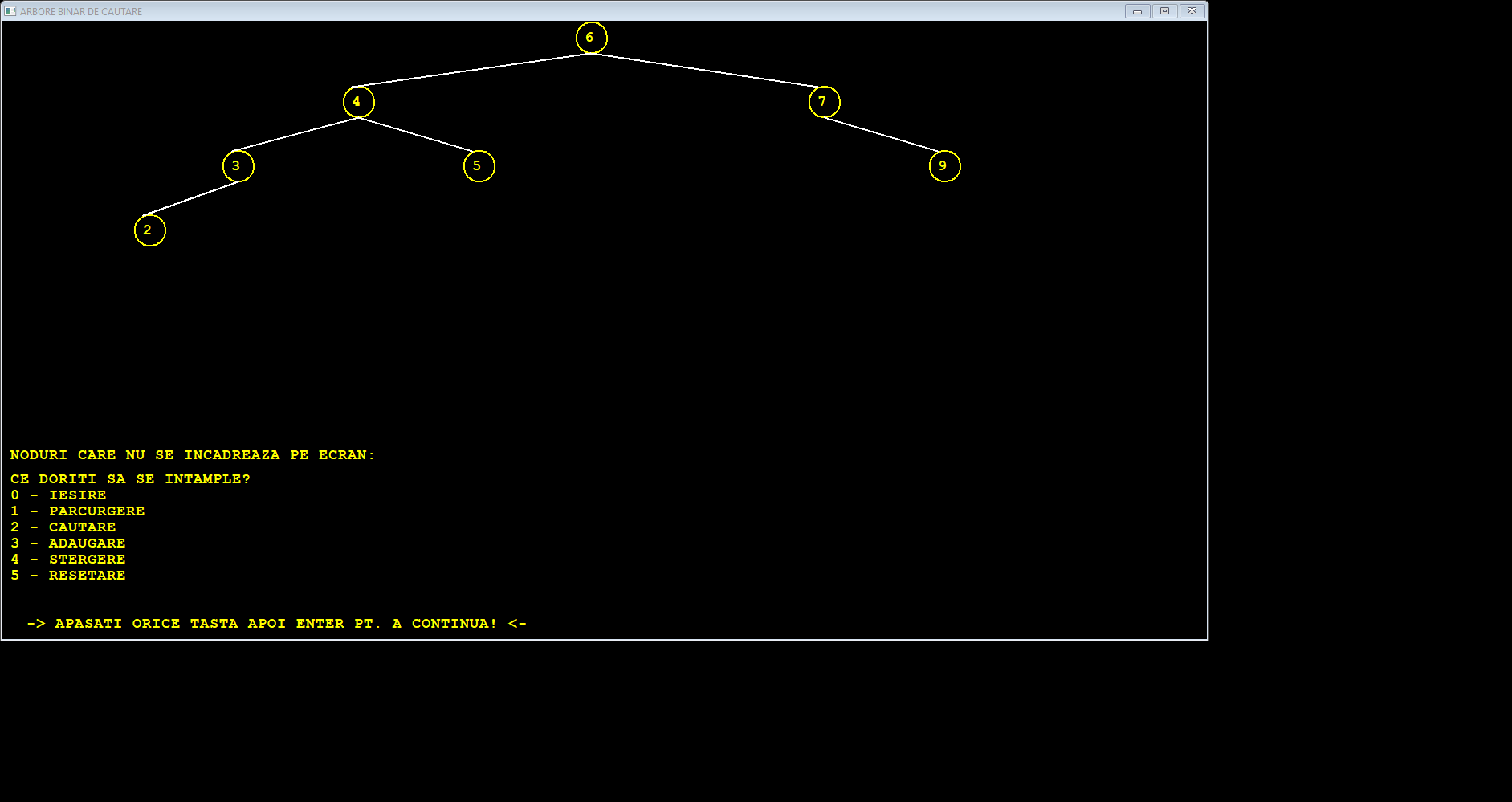
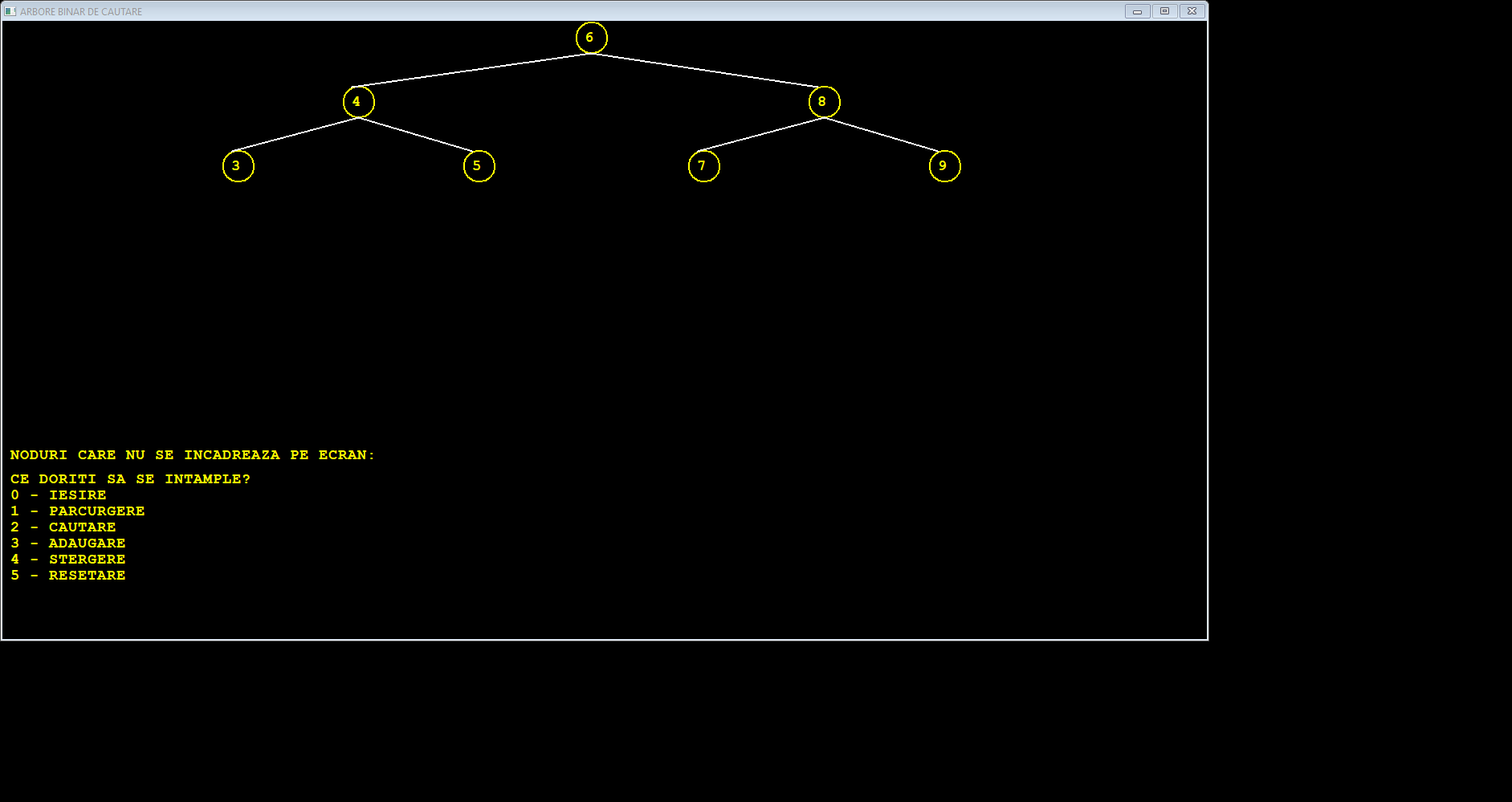


**Reinițializarea arborelui**

Resetarea arborelui presupune întoarcerea la arborele inițial prin introducerea valorii 5 în ecranul de control.



Ex: pentru valorile 6 4 8 3 5 7 9 și modificarea arborelui prin adăugarea valorii 2 și ștergerea valorii 8 se va executa:



# Anexă

**Codul folosit:**

*//inițializare*

#include <iostream>

#include <graphics.h>

#include <fstream>

using namespace std;

ifstream f("date.in");

*//structura de nod*

struct nod

{

int nr,poz;

nod \*st, \*dr;

}\*r;

*//declarare variabile*

int t,k,x0,y0,v,p,x2,y2,coord[100][4],inainte,iesire, noduri[100],copnoduri[100],n;

int ad[10]={0,-90,50,90,120,140,160,180,200},var,alg,vardelay, cn,ramase[100],l,var2;

*//inserare nod*

void inserare(nod \*&r, int k, int e)

{

if(r)

{

if(r->nr>k)

{

x0=coord[r->poz][2]-200;

y0=coord[r->poz][3]+80;

p++;

v=1;

inainte=r->poz;

inserare(r->st,k,e);

}

else

{

x0=coord[r->poz][2]+200;

y0=coord[r->poz][3]+80;

p++;

v=2;

inainte=r->poz;

inserare(r->dr,k,e);

}

}

else

{

if(p>6)

{

l++;

ramase[e]=1;

r=new nod;

r->nr=k;

r->poz=e;

r->dr=0;

r->st=0;

int j;

j=ad[p];

if(v==1)

x2=x0+j;

else

x2=x0-j;

y2=y0;

coord[e][1]=k;

coord[e][2]=x2;

coord[e][3]=y2;

}

else

{

r=new nod;

r->nr=k;

r->poz=e;

r->dr=0;

r->st=0;

t++;

char \*s;

int j;

s=new char;

itoa(k,s,10);

j=ad[p];

if(v==1)

{

outtextxy(x0+j,y0,s);

circle(x0+j+9,y0+11,20);

x2=x0+j;

}

else

{

outtextxy(x0-j,y0,s);

circle(x0-j+9,y0+11,20);

x2=x0-j;

}

y2=y0;

coord[e][1]=k;

coord[e][2]=x2;

coord[e][3]=y2;

if(t!=1)

{

setcolor(15);

line(coord[inainte][2]+8,coord[inainte][3]+30,

x2,y2-8);

}

}

}

}

*//creare arbore*

void creare()

{

//creare primul nod

if(n!=0)

{

int j=1;

while(copnoduri[j]<0)

j++;

r=new nod;

r->nr=copnoduri[j];

r->poz=j;

r->st=0;

r->dr=0;

coord[j][1]=r->nr;

coord[j][2]=725;

coord[j][3]=10;

char \*s;

s=new char;

itoa(r->nr,s,10);

outtextxy(725,10,s);

circle(725+9,10+11,20);

if(vardelay==1)

delay(500);

t=1;

l=0;

for(int i=j+1; i<=n; i++)

{

if(copnoduri[i]>=0)

{

p=0;

v=0;

inserare(r,copnoduri[i],i);

setcolor(14);

if(vardelay==1)

delay(500);

}

}

}

*//afișare noduri care nu intră în ecran*

var2=0;

outtextxy(10,530,"NODURI CARE NU SE INCADREAZA PE ECRAN: ");

if(l)

{

for(int i=1;i<=n;i++)

{

if(ramase[i]==1)

{

char \*s;

s=new char;

itoa(copnoduri[i],s,10);

outtextxy(480+var2,530,s);

var2=var2+40;

}

}

}

outtextxy(10,560,"CE DORITI SA SE INTAMPLE?");

outtextxy(10,580,"0 - IESIRE");

outtextxy(10,600,"1 - PARCURGERE");

outtextxy(10,620,"2 - CAUTARE");

outtextxy(10,640,"3 - ADAUGARE");

outtextxy(10,660,"4 - STERGERE");

outtextxy(10,680,"5 - RESETARE");

vardelay=2;

}

*//parcurgere arbore*

void srd(nod \*r)

{

if(r)

{

srd(r->st);

char \*s;

s=new char;

itoa(r->nr,s,10);

setcolor(14);

outtextxy(10+var,720,s);

var=var+40;

if(ramase[r->poz]==0)

{

setcolor(9);

outtextxy(coord[r->poz][2],coord[r->poz][3],s);

}

delay(600);

srd(r->dr);

}

}

*//căutare nod*

void cautare(nod \*r,int k)

{

if(r)

{

if(r->nr==k)

{

outtextxy(120,720,"-->> GASIT");

if(ramase[r->poz]==0)

{

setcolor(9);

char \*s;

s=new char;

itoa(r->nr,s,10);

outtextxy(coord[r->poz][2],coord[r->poz][3],s);

}

}

else

{

if(r->nr>k) cautare(r->st,k);

else cautare(r->dr,k);

}

}

else outtextxy(120,720,"-->> NEGASIT");

}

*//ștergere nod*

nod \*g;

void cmmnd( nod \*&r, nod \*&g)

{

nod \*man;

if(g->dr!=0) cmmnd(r,g->dr);

else

{

r->nr=g->nr;

man=g->st;

delete g;

g=man;

}

}

void sterge (nod \*&r, int k)

{

nod \*g;

if(r)

{

if(r->nr==k)

{

if(r->st==0 && r->dr==0)

{

delete r;

r=0;

}

else

{

if(r->dr==0)

{

g=r->st;

delete r;

r=g;

}

else

{

if(r->st==0)

{

g=r->dr;

delete r;

r=g;

}

else cmmnd(r,r->st);

}

}

}

}

else

{

if(r->nr<k) sterge(r->st,k);

else sterge (r->dr,k);

}

}

*//sdr*

void sdr(nod \*&r)

{

if(r)

{

sdr(r->st);

sdr(r->dr);

}

delete r;

}

*//main*

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "C:\\TC\\BGI");

initwindow(1500,770, "ARBORE BINAR DE CAUTARE");

setcolor(14);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 4);

outtextxy(10,10,"APASATI ORICE TASTA PENTRU A AFISA ARBORELE BINAR!");

getch();

cleardevice();

*//citire noduri*

n=0;

while(f>>k)

{

n++;

noduri[n]=k;

copnoduri[n]=k;

}

cn=n;

*//creare meniu*

vardelay=1;

alg=1;

while(alg)

{

settextstyle(BOLD\_FONT, HORIZ\_DIR, 2);

setlinestyle(SOLID\_LINE,0,2);

creare();

setbkcolor(0);

cout<<"INTRODUCETI VALOAREA: ";

cin>>alg;

while(alg!=0 && alg!=1 && alg!=2 && alg!=3 && alg!=4 && alg!=5)

{

outtextxy(10,700,"ALEGERE INCORECTA!!");

cout<<"INTRODUCETI VALOAREA: ";

cin>>alg;

}

if(alg==1) *//parcurgere*

{

outtextxy(10,700,"PARCURGEREA ESTE: ");

srd(r);

}

if(alg==2) *//căutare*

{

outtextxy(10,700,"INTRODUCETI VALOAREA CE TREBUIE CAUTATA!");

cout<<"VALOARE CE TREBUIE CAUTATA: ";

int caut;

cin>>caut;

cautare(r,caut);

}

if(alg==3) *//adăugare*

{

outtextxy(10,700,"INTRODUCETI VALOAREA CE TREBUIE ADAUGATA!");

cout<<"VALOARE CE TREBUIE ADAUGATA: ";

int adaug;

cin>>adaug;

n++;

copnoduri[n]=adaug;

p=0;

v=0;

inserare(r,adaug,n);

if(ramase[n]==1)

{

char \*s;

s=new char;

itoa(copnoduri[n],s,10);

outtextxy(480+var2,550,s);

var2=var2+40;

}

}

if(alg==4) *//ștergere*

{

outtextxy(10,700,"INTRODUCETI VALOAREA CE TREBUIE STEARSA!");

cout<<"VALOARE CE TREBUIE STEARSA: ";

int sterg;

cin>>sterg;

int i=1,pp=1;

while(i<=n && pp)

{

if(copnoduri[i]==sterg)

{

copnoduri[i]=-1;

sterge(r,sterg);

cleardevice();

vardelay++;

creare();

pp=0;

}

i++;

}

if(pp) outtextxy(10,700,"NU EXISTA VALOAREA IN ARBORE! ");

}

if(alg==5) *//resetare*

{

cleardevice();

for(int i=1;i<=n;i++)

copnoduri[i]=noduri[i];

n=cn;

creare();

}

*//reinițializare ecran*

setcolor(14);

if(alg!=0)

{

outtextxy(30,740,"-> APASATI ORICE TASTA APOI ENTER PT. A CONTINUA! <-");

char c;

cin>>c;

}

sdr(r);

for(int i=1;i<=n;i++)

{

coord[i][1]=0;

coord[i][2]=0;

coord[i][3]=0;

ramase[i]=0;

}

var=0;

vardelay++;

cleardevice();

}

closegraph();

return 0;

}

# Bibliografie

-> https://ro.wikipedia.org/wiki/Arbore\_(teoria\_grafurilor)

-> Cornelia Ivașc, Mona Prună, Luminița Condurache, Doina Logofătu-Hrinciuc – „Informatică - C++”, Ed. Petrion, 2002, ISBN 973-9470-33-5

-> Florian Moraru – „Structuri de date și algoritmi”