Implementarea AB:

a. static – vectori;
b. dinamic – pointeri.

a. static :

a.1. Cu 2 vectori st și dr:

st[i] – eticheta nodului succesor stâng al nodului i sau 0
dr[i] – eticheta nodului succesor drept al nodului i sau 0

Exemplu:
n=8
st: 2 4 0 0 7 0 0 0
dt: 3 5 6 0 0 8 0 0

Determinare:
Rădăcina :
Nodurile cu un singur fiu:
Nodurile cu doi fii:
Nodurile frunză:
Nodurile frați:

a.2. Cu 2 vectori t și fiu:

t[i] - tatăl lui i sau 0

fii[i] = –1 , 1 sau 0

Nodul rădăcină are t[rădăcină]=0

Exemplu:

n=9
t= 3 3 0 1 1 5 6 2 2
fii= -1 1 0 -1 1 -1 1 -1 1

Determinare:

Rădăcina :
Nodurile cu un singur fiu:
Nodurile cu doi fii:

Nodurile frunză:
Nodurile frați:

Obs*: Nodurile frați: f[i]*f[j]==-1 și t[i]==t[j]

Determinare rădăcină AB

```
int radAB()
{
    int v[101] = {0};
    for(int i = 1 ; i <= n ; i ++)
    {
        if(st[i] != 0)
            v[st[i]] = 1;
        if(dr[i] != 0)
            v[dr[i]] = 1;
    }
    for(int i = 1 ; i <= n ; i ++)
        if(v[i] == 0)
            return i;
    return 0;
}
```

Algoritmi pentru parcurgerea unui arbore binar

parcurgere în lăţime (la fel ca la grafuri);
parcurgere în adâncime (specifici arborilor binari):

a. RSD (traversarea în preordine);
b. SRD (traversarea în inordine);
c. SDR (traversarea în postordine).

```
void RSD(int x)
{
    if(x != 0)
    {
        cout << x << " ";
        RSD(st[x]);
        RSD(dr[x]);
    }
}
```

```
void SRD(int x)
{
    if(x != 0)
    {
        SRD(st[x]);
        cout << x << " ";
        SRD(dr[x]);
    }
}

void SDR(int x)
{
    if(x != 0)
    {
        SDR(st[x]);
        SDR(dr[x]);
        cout << x << " ";
    }
}
```

Transformarea reprezentării descendente în reprezentarea ascendentă și invers.

```
    int n,st[100],dr[100];
    cin>>n;/// 7
    for(int i=1;i<=n;i++)
        cin>>st[i]>>dr[i];
    /// 2 3 4 0 5 6 0 7 0 0 0 0 0 0
    int t[100]={0},f[100]={0};
    for(int i=1;i<=n;i++)
    {
        if(st[i]!=0)
            t[st[i]]=i,f[st[i]]=-1;
        if(dr[i]!=0)
            t[dr[i]]=i,f[dr[i]]=1;
    }
    for(int i=1;i<=n;i++)
        cout<<i<<"are "<<t[i]<<' '<<f[i]<<endl;
```

```
    int n,st[100]={0},dr[100]={0};
    int t[100]={0},f[100]={0};
```

```
    cin>>n;/// 7
    for(int i=1;i<=n;i++)
        cin>>t[i]>>f[i];
   /// 0 0 1 -1 1 1 2 -1 3 -1 3 1 4 1

    for(int i=1;i<=n;i++)
    {
      if(f[i]==-1) st[t[i]]=i;
      if(f[i]==1) dr[t[i]]=i;
    }

    for(int i=1;i<=7;i++)
        cout<<st[i]<<' ';
    cout<<endl;
    for(int i=1;i<=7;i++)
        cout<<dr[i]<<' ';
    return 0;
}
```

Înălțimea unui AB

```
int h(int i)
 {
    if (st[i]==0 && dr[i]==0) return 0;
    else
        return 1+max(h(st[i]),h(dr[i]));
}
```

Determinare dacă un nod este frunză

```
int frunza(int i)
{
  return st[i]+dr[i]==0;
}
```

Determinare numărul de frunze al unui AB

```
int frunze(int i)
{
 if (frunza(i)) return 1;
 else
    if (st[i]==0) return frunze(dr[i]);
    else
```

```
        if (dr[i]==0) return frunze(st[i]);
        else return frunze(st[i])+ frunze(dr[i]);
}
```

Citirea AB

```
#include <iostream>
using namespace std;
int n,st[100]={0},dr[100]={0};
    int t[100]={0},f[100]={0};
void creare(int x)
{
    int y;
    cout<<"fiul sting pentru "<<x<<":";cin>>y;
    if(y!=0) {st[x]=y;creare(y);}
    cout<<"fiul drept pentru "<<x<<":";cin>>y;
    if(y!=0) {dr[x]=y;creare(y);}
}
int main()
{
    cin>>n;
    creare(1);
    for(int i=1;i<=7;i++)
        cout<<st[i]<<' ';
    cout<<endl;
    for(int i=1;i<=7;i++)
        cout<<dr[i]<<' ';

    return 0;
}
```

b. dinamic – pointeri.

Declarație nod arbore binar:

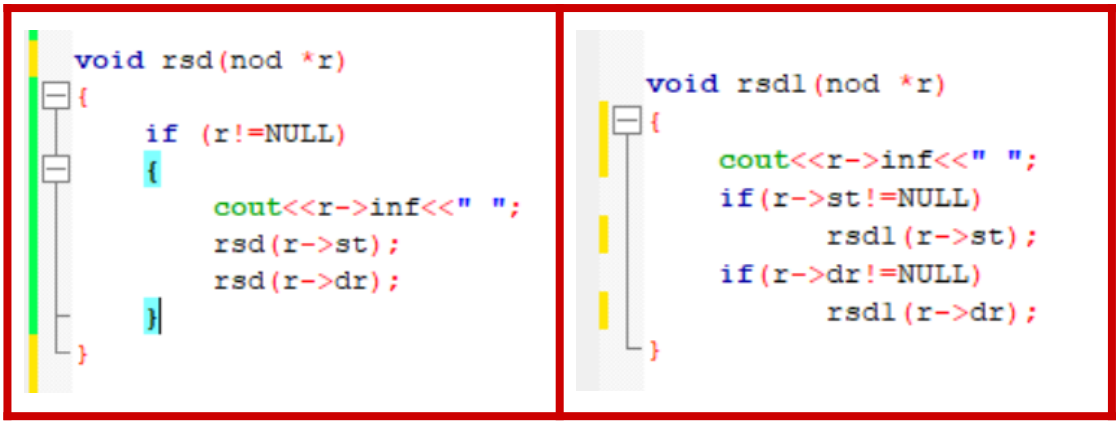
```
struct nod
{
    int inf;
    nod *st,*dr;
};
nod *r;
```

Procedură creare Arbore Binar:

```
void crearel(nod *&r)
{
    int x; cin>>x;
    if(x!=0)
    {   r=new nod;
        r->inf=x;
        cout<<"stinga lui "<<x<<" = ";
        crearel(r->st);
        cout<<"dreapta lui "<<x<<" = ";
        crearel(r->dr);
    }
   else r=NULL;

}
```

Parcurgerea AB în preordine:



```
void rsd(nod *r)
{
    if (r!=NULL)
    {
        cout<<r->inf<<" ";
        rsd(r->st);
        rsd(r->dr);
    }
}
```

```
void rsdl(nod *r)
{
    cout<<r->inf<<" ";
    if(r->st!=NULL)
        rsdl(r->st);
    if(r->dr!=NULL)
        rsdl(r->dr);
}
```