Bináris fák
1. előadás

(Horváth Gyula anyagai felhasználásával)

Bináris fa

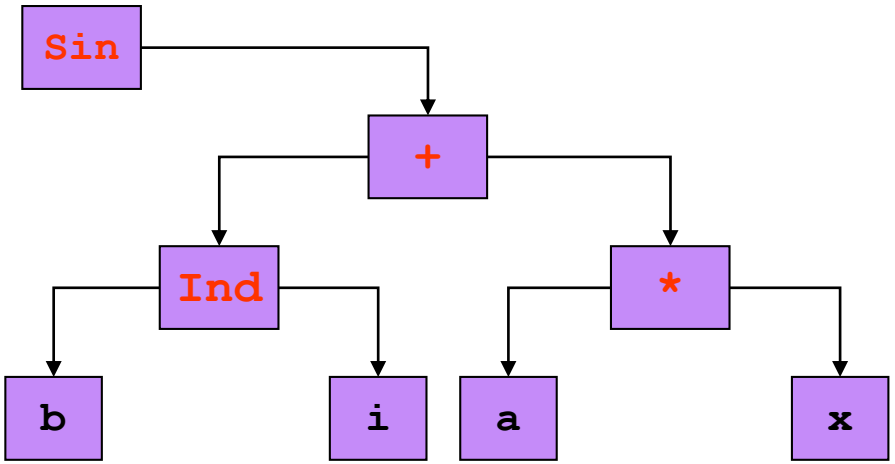
Az adatkezelés szintjei:

1. Probléma szintje.
2. Modell szintje.
3. Absztrakt adattípus szintje.
4. Absztrakt adatszerkezet szintje.
5. Adatszerkezet szintje.
6. Gépi szint.

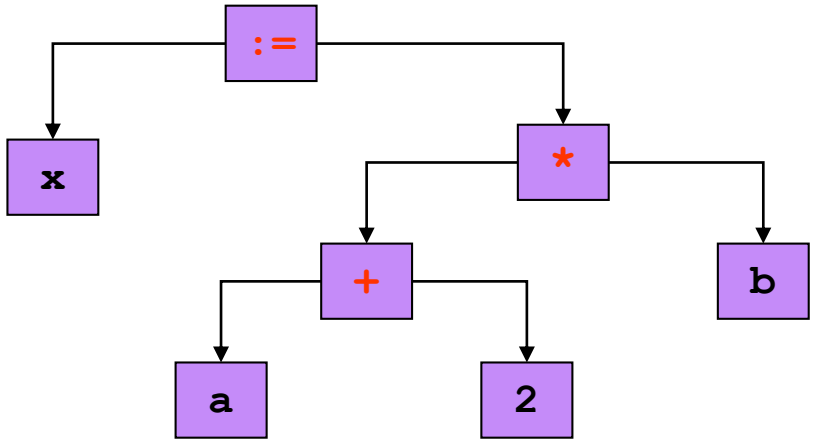
Bináris fa: példák

- `sin(b(i)+a*x)`

A modell mindkét esetben egy bináris fa.

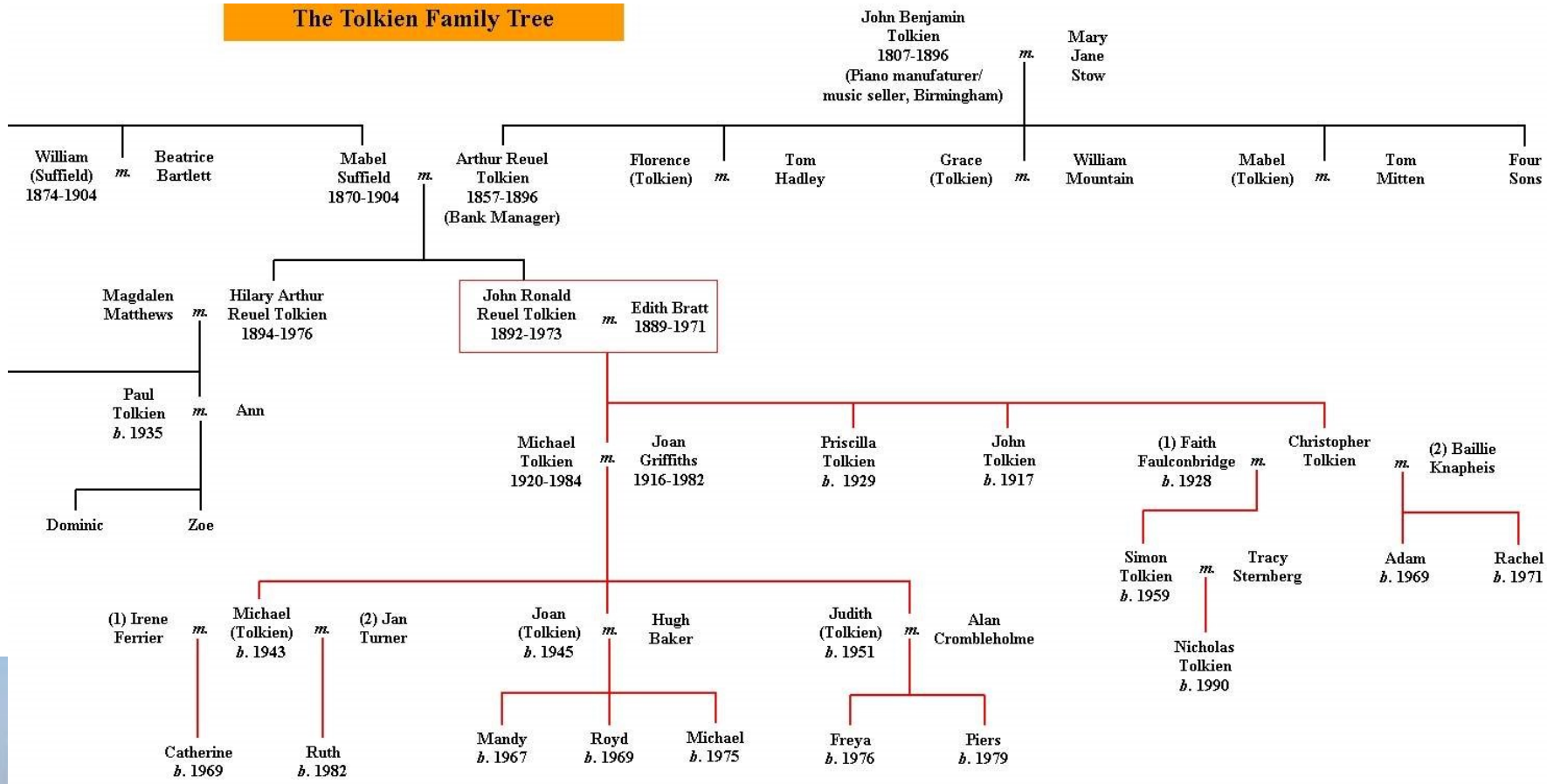


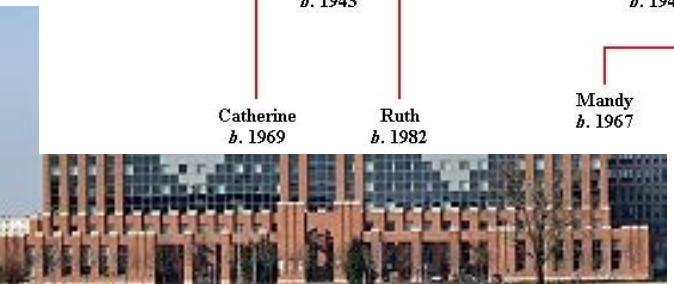
- `x:=(a+2)*b`



Nem bináris fa: példák

John Ronald Reuel Tolkien családfája





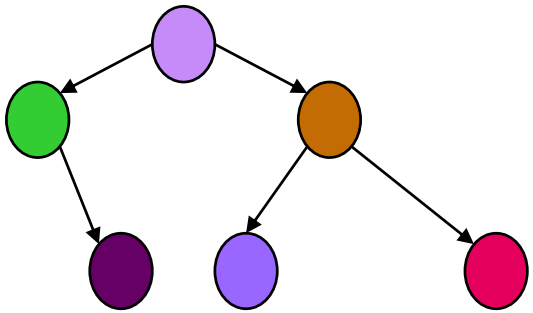
Bináris fa

A bináris fa (fa) absztrakt adattípus:

$$\text{BinFa:=} \begin{cases} \textit{Üres}\text{Fa} \\ \textit{Rekord}(\text{Elem},\text{BinFa},\text{BinFa}) \end{cases}$$

$$\text{Fa:=} \begin{cases} \textit{Üres}\text{Fa} \\ \textit{Rekord}(\text{Elem},\text{Fák}) \end{cases}$$

A Fák egy sokaság (halmaz, sorozat, …)



Bináris fa absztrakt adattípus

A bináris fa modell/absztrakt adattípus jellemzői:

- sokaság: azonos típusú elemekből áll;
- akár 0 db elemet tartalmazhat;
- Üres: rekurzív „nullelem”, kitüntetett konstans;
- Fraktál (=önhasonlóság) tulajdonság: a részei ugyanolyan szerkezetűek, mint az egész;
- nem lineárisan rendezett (azaz nem sorozatféle): bármely elemének 0, 1, 2 (közvetlen) rákövetkezője lehet;

- minden elemnek legfeljebb egy (közvetlen) előzője van.

Bináris fa absztrakt adattípus

A Bináris fa absztrakt adattípus műveletei:

```
Üres:Binfa
üres?(Binfa):Logikai
egyeleműfa(Elem):Binfa
```

Bináris fa absztrakt adattípus

A Bináris fa absztrakt adattípus műveletei:

`Balrailleszt(Binfa,Binfa):Binfa ∪ {NemDef}`

előfeltétel: az első paraméter Binfa nem üres, a bal része üres

`Jobbrailleszt(Binfa,Binfa):Binfa ∪ {NemDef}`

előfeltétel: az első paraméter Binfa nem üres, a jobb része üres

A balra és a jobbra illesztés legtöbbször egyetlen elemet (vagy egyelemű fát) vesz a fához. Lehetséges művelet pl.:

`Balrailleszt(Binfa,Elem):Binfa ∪ {NemDef}`

előfeltétel: az első paraméter Binfa nem üres, a bal része üres

Bináris fa absztrakt adattípus

A Bináris fa absztrakt adattípus műveletei:

`balgyerek(Binfa):Binfa ∪ {Nemdef}`

előfeltétel: Binfa nem üres

`jobbgyerek(Binfa):Binfa ∪ {Nemdef}`

előfeltétel: Binfa nem üres

`gyökérelem(Binfa):Elem ∪ {NemDef}`

előfeltétel: Binfa nem üres

`gyökérmódosít(Binfa,Elem):Binfa ∪ {Nemdef}`

előfeltétel: Binfa nem üres

Bináris fa absztrakt adatszerkezet

A bináris fa (fa) absztrakt adatszerkezet ábrázolása:

```
Típus TBinfa=Rekord
          (elem: TElem
           bal,jobb: TBinfa)

Típus TFa=Rekord
          (elem: TElem
           ágak: Sorozat(TFa))
```

Általában ez a közvetlen rekurzív ábrázolás nem működik, így konkrét adatszerkezetnél közvetett megoldás kell!

Bináris fa – dinamikus láncolás

A bináris fa (fa) adatszerkezet ábrázolása dinamikus láncolással:

```
Típus TBinfa=Rekord
          (elem: TElem
           bal,jobb: Tbinfa címe)

Típus TFa=Rekord
          (elem: TElem
           ágak: Sorozat(Tfa címe))
```

Minden fa, illetve bináris fa pedig egy TFa, TBinfa címe.

Bináris fa – dinamikus láncolás

Koncepció:

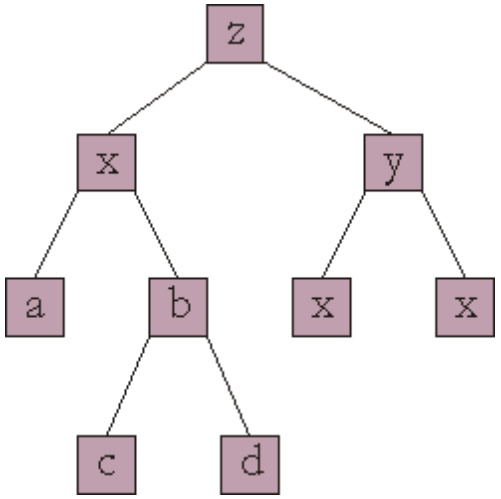
- a bináris fa egy rekurzív adatszerkezet, ha bármely részét megváltoztatom, az egész meg fog változni;
- megvalósítás dinamikus láncolással;
- a műveletek értékmegosztást feltételeznek.

Megjegyzés: az előadásban feltesszük, hogy az előfeltételeket mindenhol betartjuk, nincs ellenőrzés.

Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

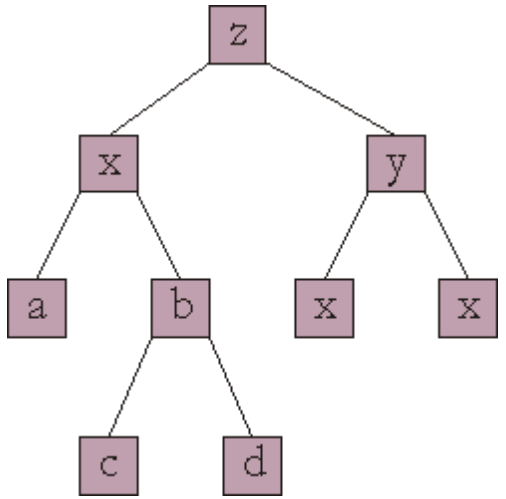
```
Üres(bf):
  bf:=sehova
Eljárás vége.
üres?(bf):
  üres?:=bf=sehova
Függvény vége.
egyeleműfa(Elem):
  Lefoglal(bf,(Elem,sehova,sehova))
  egyeleműfa:=bf
Függvény vége.
```



Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

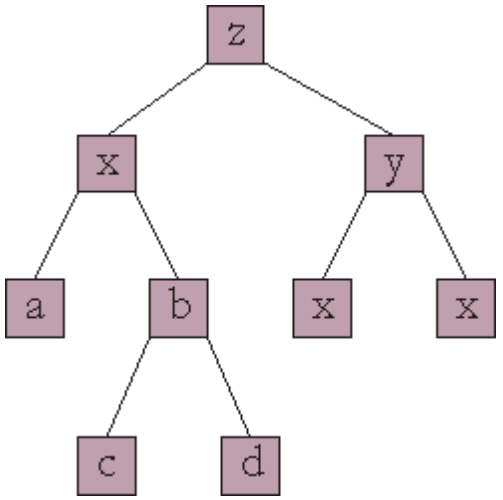
```
Balrailleszt(bf,rf):
  Tartalom(bf).bal:=rf
Eljárás vége.
Jobbrailleszt(bf,rf):
  Tartalom(bf).jobb:=rf
Eljárás vége.
gyökérelem(bf):
  gyökérelem:=Tartalom(bf).elem
Függvény vége.
```



Bináris fa – dinamikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
balgyerek(bf):
  balgyerek:=Tartalom(bf).bal
Függvény vége.
jobbgyerek(bf):
  jobbgyerek:=Tartalom(bf).jobb
Függvény vége.
Gyökérmódosít(bf,Elem):
  Tartalom(bf).elem:=Elem
Eljárás vége.
```



Bináris fa – statikus láncolás

Koncepció:

- a bináris fa egy sokaság, amelyben a sorrend speciális;
- megvalósítás statikus láncolással;
- a műveletek a struktúrában való mozgást feltételeznek, kell egy aktuális elem.

```
Típus TBinfa=Rekord
   (gyökér,akt,szabad: 0..Max,
    t: tömb(1..Max,TBinfaelem))

TBinfaelem=Rekord
(elem: TElem, bal,jobb: 0..Max)
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

`Üres(Binfa)`

`üres?(Binfa):Logikai`

`Egyeleműfa(Elem,Binfa)`

`Balrailleszt(Binfa,Elem)`

előfeltétel: Binfa aktuális eleme nem üres, a bal része üres

`Jobbrailleszt(Binfa,Elem)`

előfeltétel: Binfa aktuális eleme nem üres, a jobb része üres

`Gyökérre(Binfa)`

A különbség:

beilleszteni csak elemet tudunk.

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

`érték(Binfa):Elem ∪ {NemDef}`

előfeltétel: Binfa aktuális eleme nem üres

`Balra(Binfa)`

előfeltétel: Binfa aktuális eleme nem üres

`Jobbra(Binfa)`

előfeltétel: Binfa aktuális eleme nem üres

`Módosít(Binfa,Elem)`

előfeltétel: Binfa aktuális eleme nem üres

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Üres(bf):
  bf.gyökér:=0; bf.akt:=0
  bf.szabad:=1; Szabadlista(bf)
Eljárás vége.
üres?(bf):
  üres?:=bf.gyökér=0
Függvény vége.
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Egyeleműfa(bf,Elem):
  bf.gyökér:=1; bf.akt:=1; bf.szabad:=2
  bf.t(1):=(Elem,0,0)
Eljárás vége.
Gyökérre(bf):
  bf.akt:=bf.gyökér
Eljárás vége.
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Balrailleszt(bf,Elem):
   Szabadlistából(x); bf.t(bf.akt).bal:=x
   bf.t(x):=(Elem,0,0); bf.akt:=x
Eljárás vége.
```

```
Jobbrailleszt(bf,Elem):
   Szabadlistából(x); bf.t(bf.akt).jobb:= x
   bf.t(x):=(Elem,0,0); bf.akt:=x
Eljárás vége.
```

```
érték(bf):
   elem:=bf.t(bf.akt).elem
Függvény vége.
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei (ha nincs elem törlés a fából, nem kell szabadlista):

```
Balrailleszt(bf,Elem):
  bf.t(bf.akt).bal:=bf.szabad; bf.akt:=bf.szabad
  bf.t(x):=(Elem,0,0); bf.szabad:=bf.szabad+1
Eljárás vége.
Jobbrailleszt(bf,Elem):
  bf.t(bf.akt).jobb:=bf.szabad; bf.akt:=bf.szabad
  bf.t(x):=(Elem,0,0); bf.szabad:=bf.szabad+1
Eljárás vége.
```

Bináris fa – statikus láncolás

A Bináris fa rekurzív adatszerkezet műveletei:

```
Balra(bf):
  bf.akt:=bf.t(bf.akt).bal
Függvény vége.
Jobbra(bf):
   bf.akt:=bf.t(bf.akt).jobb
Függvény vége.
Módosít(bf,Elem):
  bf.t(bf.akt).elem:=Elem
Eljárás vége.
```

Bináris fa – statikus láncolás

Megjegyzések:

- A statikus láncolással megvalósított bináris fa (aktuális elemmel és gyökérelemmel) dinamikus láncolással is menne.
- A dinamikus láncolással megvalósított bináris fa az értékmegosztás miatt statikus láncolással nem működne.

Bináris fa – statikus láncolás

Módosítás: a bináris fa egyes elemei tartalmazhatják a fölöttük levő elem azonosítóját is – statikus láncolás.

```
Típus TBinfa=Rekord
   (gyökér,akt: 0..Max,
    t: tömb(1..Max,TBinfaelem))

Tbinfaelem=Rekord
   (elem: TElem,
    bal,jobb,szülő: 0..Max)
```

Bináris fa – dinamikus láncolás

Módosítás: a bináris fa egyes elemei tartalmazhatják a fölöttük levő elem címét is – dinamikus láncolással, egy példa eljárással.

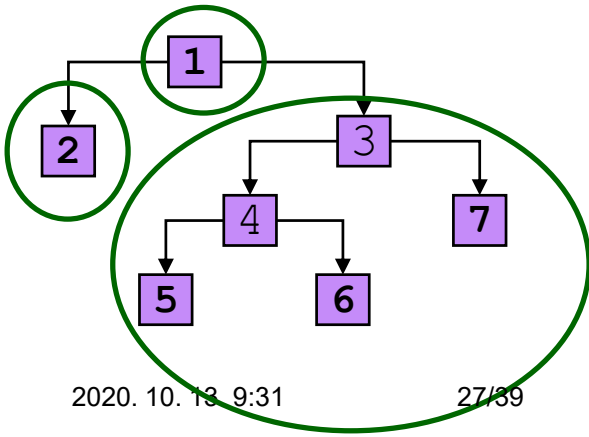
```
Típus TBinfa=Rekord
   (gyökér: TBinfaelem'Mutató)

TBinfaelem=Rekord
   (elem: TElem,
    bal,jobb,szülő: TBinfaelem'Mutató)
Balrailleszt(bf,rf):
  Tartalom(bf).bal:=rf; Tartalom(rf).szülő:=bf
Eljárás vége.
```

Bináris fa – új műveletek

Fa bejárások: Bal-közép-jobb

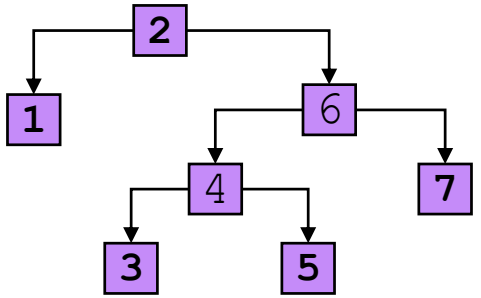
```
BKJ(bf):
    Ha nem üres?(bf) akkor
                BKJ(balgyerek(bf))
                Ki: gyökérelem(bf)
                BKJ(jobbgyerek(bf))
  Elágazás vége
Eljárás vége.
```

Alkalmazás: rendezőfa



Bináris fa – új műveletek

```
BKJ(bf):
  Üres(v); Verembe(v,sehova); rf:=bf
  Ciklus amíg nem üres?(v)
    Ciklus amíg nem üres?(rf)
      Verembe(v,rf); rf:=balgyerek(rf)
    Ciklus vége
    Veremből(v,rf)
    Ha nem üres?(v) akkor
       Ki: Gyökér(rf); rf:=jobbgyerek(rf)
    Elágazás vége
  Ciklus vége
 Eljárás vége.
```

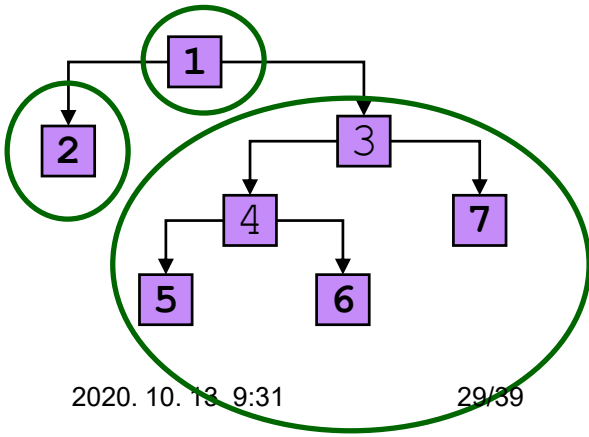


Bináris fa – új műveletek

Fa bejárások: Közép-bal-jobb

```
KBJ(bf):
  Ha nem üres?(bf) akkor
            Ki: gyökérelem(bf)
            KBJ(balgyerek(bf))
            KBJ(jobbgyerek(bf))
  Elágazás vége
Eljárás vége.
```

Alkalmazás: Fa elemszáma, magassága, ...

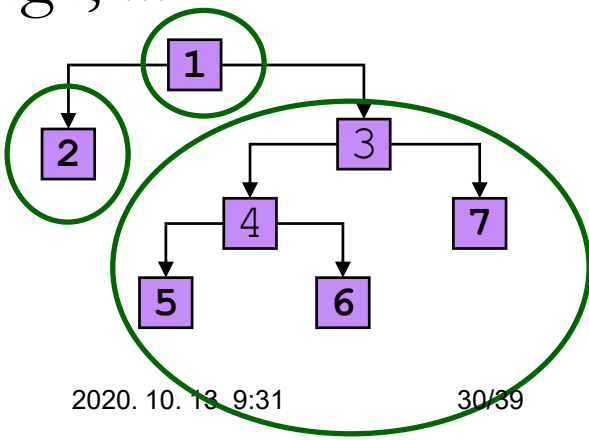


Bináris fa – új műveletek

Fa bejárások: Bal-jobb-közép

```
BJK(bf):
   Ha nem üres?(bf) akkor
                BJK(balgyerek(bf))
                BJK(jobbgyerek(bf))
                Ki: gyökérelem(bf)
   Elágazás vége
Eljárás vége.
```

Alkalmazás: Fa törlése, elemszáma, magassága, ...

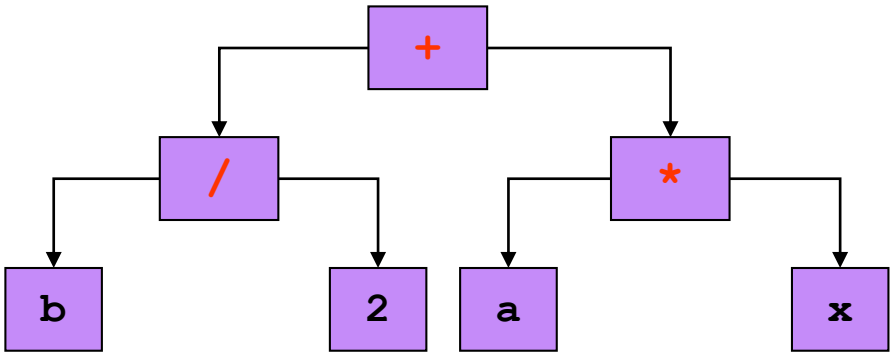


Bináris fa – új műveletek

Az egyszerű kifejezésfa egy bináris fa, amelynek leveleiben az adatok, nem levélelemeiben pedig a műveletek szerepelnek.

Minden műveletnek pontosan 2 paramétere van.

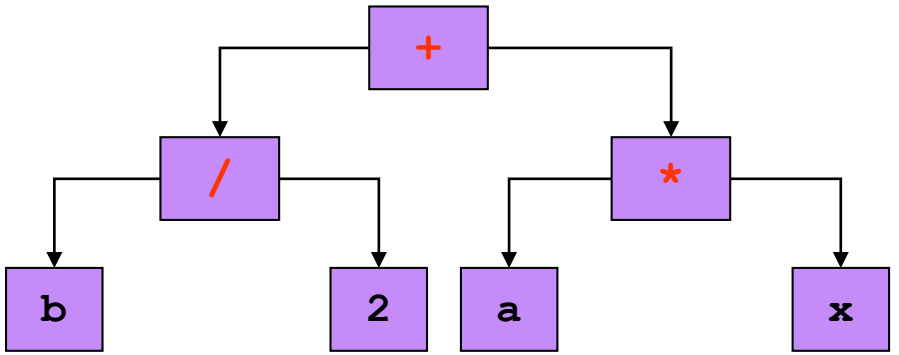
Példa: `b/2+a*x`



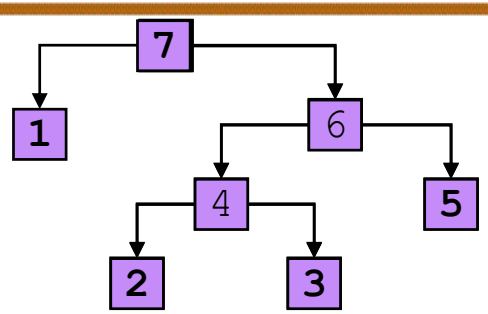
Bináris fa – új műveletek

Bal-jobb-közép bejárás alkalmazása – kifejezésfa kiértékelése:

```
BJK(bf):
  Ha nem üres?(balgyerek(bf)) akkor
            a:=BJK(balgyerek(bf))
            b:=BJK(jobbgyerek(bf))
            BJK:=művelet(gyökérelem(bf),a,b)
  különben BJK:=gyökérelem(bf)
Függvény vége.
```



Bináris fa – új műveletek

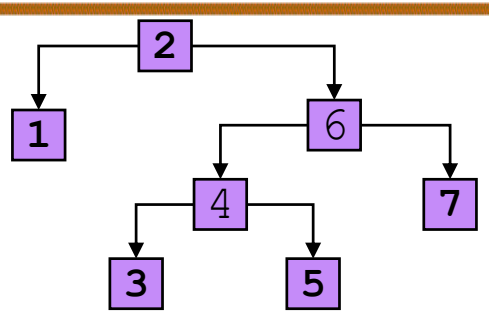


Fa törlése (dinamikus láncolással)

```
Törlés(bf):
  Ha nem üres?(bf) akkor
            Törlés(balgyerek(bf))
            Törlés(jobbgyerek(bf))
            Gyökértörlés(bf)
  Elágazás vége
Eljárás vége.
```

```
Gyökértörlés(bf):
  Felszabadít(bf); bf:=sehova
Eljárás vége.
```

Bináris fa – új műveletek



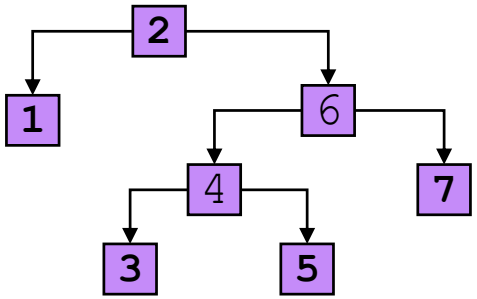
Néhány speciális fa művelet:

```
elemszám(bf):
   Ha üres?(bf) akkor elemszám:=0
   különben elemszám:=1+elemszám(balgyerek(bf))+
                       elemszám(jobbgyerek(bf))
Függvény vége.

magas(bf):
   Ha üres?(bf) akkor magas:=0
   különben magas:=1+max(magas(balgyerek(bf)),
                         magas(jobbgyerek(bf))
Függvény vége.
```

Bináris fa – új műveletek

Néhány speciális fa művelet:

```
Szélesség(bf,i):
  Ha nem üres?(bf) akkor
    szél(i):=szél(i)+1
    szélesség(balgyerek(bf),i+1),
    szélesség(jobbgyerek(bf),i+1)
Eljárás vége.
```



Globális `szél` tömb használata.

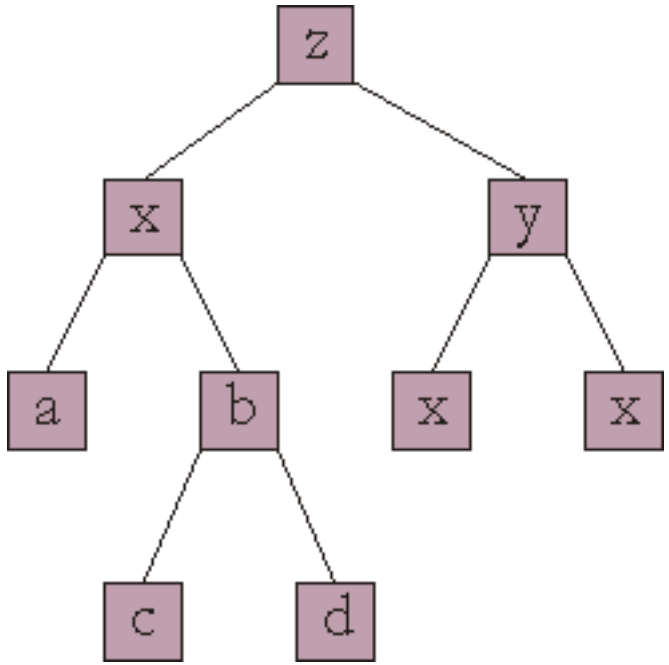
Bináris fa – felépítése

Bináris fák szövegesen is megadhatók az alábbi szabályok szerint:

1) Minden karakter, ami az angol ábécé eleme, faleírás.

2) Ha x karakter valamint f1 és f2 faleírás, akkor az x(f1,f2) szöveg is faleírás.

Pl.: z(x(a,b(c,d)),y(x,x))

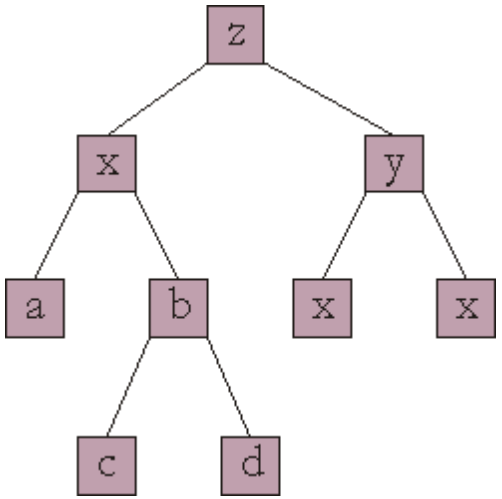


Bináris fa – felépítése

```
Felépít(bf,s):
  Lefoglal(bf,(s[i],sehova,sehova)); i:=i+1
  Ha i≤hossz(s) akkor
    Ha s[i]='(' akkor
       i:=i+1; Felépít(Tartalom(bf).bal,s)
       {',',}
       i:=i+1; Felépít(Tartalom(bf).jobb,s)
       {')'} i:=i+1
    Elágazás vége
Eljárás vége.
```

Globális `i` változó használata.

z(x(a,b(c,d)),y(x,x))

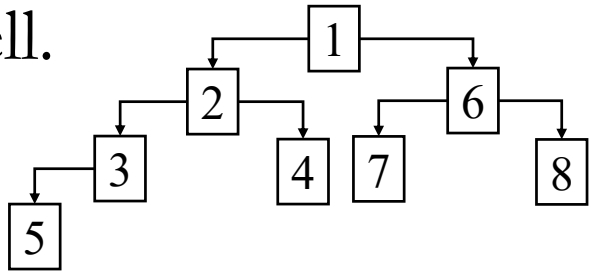


Bináris fa – elvi alkalmazás

A ***kupac*** olyan véges elemsokaság, amely rendelkezik az alábbi tulajdonságokkal:

1. Minden elemnek ***legfeljebb két rákövetkezője*** (leszármazottja) lehet. Azaz ***bináris fá***nak tekinthető.
2. Minden ***eleme kisebb*** (vagy egyenlő) a közvetlen leszármazottainál.
3. A kupac ***balról folytonos***, azaz ha nem teljes a fa, akkor csak a legutolsó szintből hiányozhatnak elemek, de azok is csak a szint jobb széléről.

Ezek következménye, hogy ***ábrázolható szintfolytonosan*** is, tömbben, azaz a bináris fa csak egy modell.



Bináris fák
1. előadás vége