Rekurzió

ELTE

Rekurzió

Klasszikus példák

- Faktoriális

$$n! = \begin{cases} n*(n-1)! & ha \;\; n > 0 \\ 1 & ha \;\; n = 0 \end{cases}$$

- Fibonacci-számok

$$Fib(n) = \begin{cases} 0 & ha \;\; n = 0 \\ 1 & ha \;\; n = 1 \\ Fib(n-1) + Fib(n-2) & ha \;\; n > 1 \end{cases}$$

A rekurzió lényege: önhivatkozás

ELTE

Rekurzió

Klasszikus példák

- Binomiális számok:

$$B(n,k) = \begin{cases} 1 & ha \;\; k = 0 \\ B(n-1,k) + B(n-1,k-1) & ha \;\; 0 < k < n \\ 1 & ha \;\; k = n \end{cases}$$

$$B(n,k) = \begin{cases} 1 & ha \;\; k = 0 \\ B(n,k-1) * \dfrac{n-k+1}{k} & ha \;\; 0 < k \le n \end{cases}$$

ELTE

Rekurzió

Gyorsrendezés (quick sort):

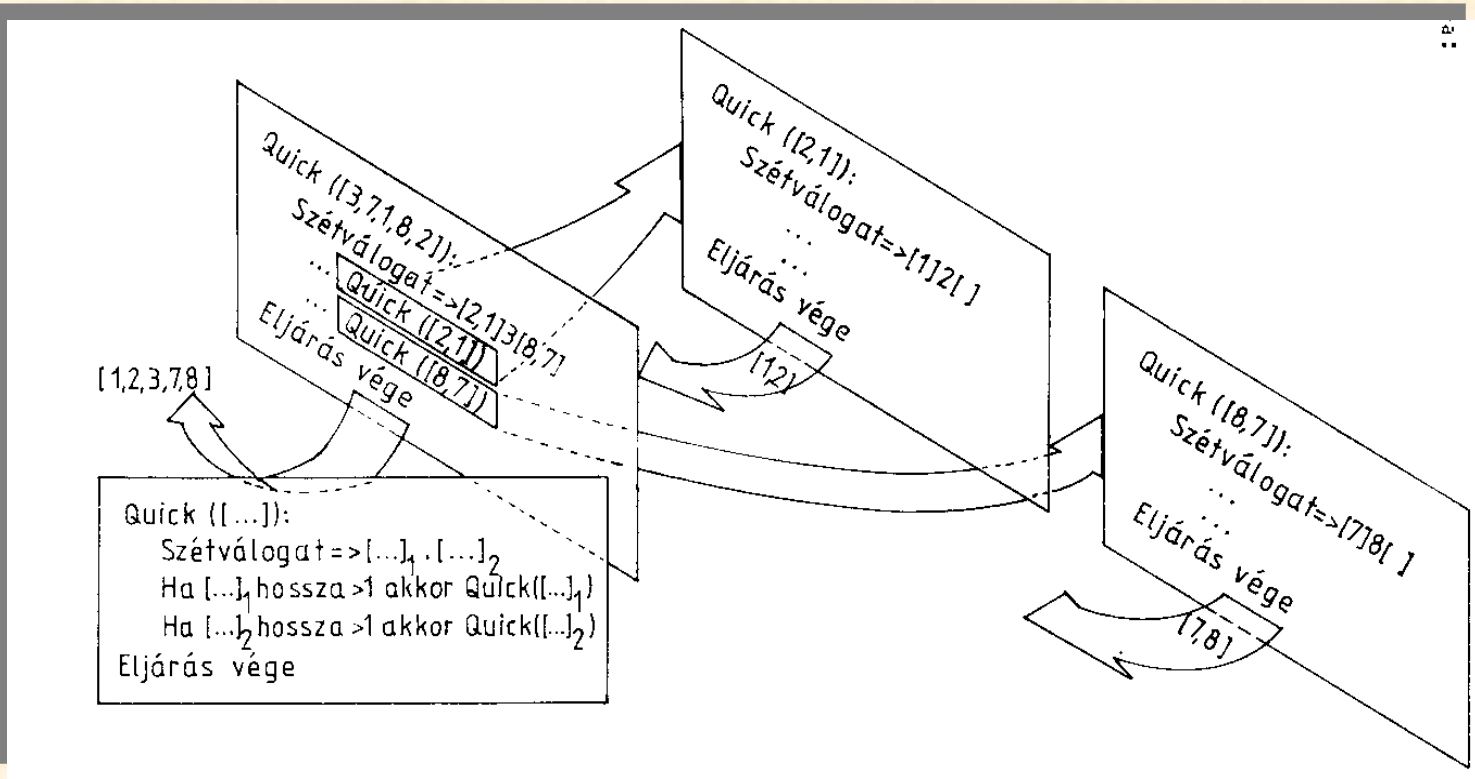
N elem rendezését a következőképpen végezhetjük:

- Válogassuk szét az elemeket az első elemnél kisebbekre, illetve nagyobbakra!
- A két keletkezett részre (ha legalább 2 elemet tartalmaznak) alkalmazzuk ugyanezt az eljárást!

ELTE

Rekurzió

Gyorsrendezés (quick sort):

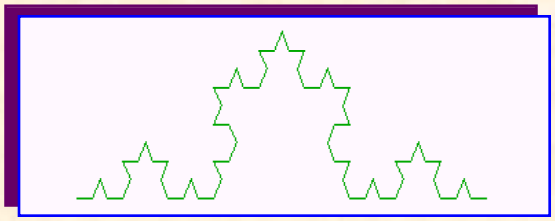


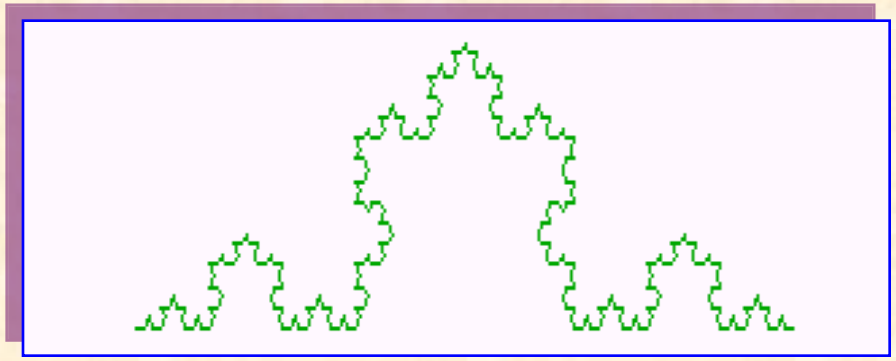
ELTE

Rekurzió

Koch fraktál:

- Vegyünk egy egységnyi szakaszt!
- Vágjuk ki a középső harmadát!
- Illesszük be a kivágott részt egy egyenlő oldalú háromszög száraiként!
- Alkalmazzuk ugyanezt az így kapott 4 szakaszra!





ELTE

Rekurzió

Hanoi tornyai:

Adott 3 rudacska. Az elsőn egyre csökkenő sugarú korongok vannak. Az a feladat, hogy tegyük át a harmadik rudacskára a korongokat egyenként úgy, hogy az átpakolás közben és természetesen a végén is minden egyes korongon csak nála kisebb lehet. Az átpakoláshoz lehet segítségül felhasználni a középső rudacskát.

ELTE

Rekurzió

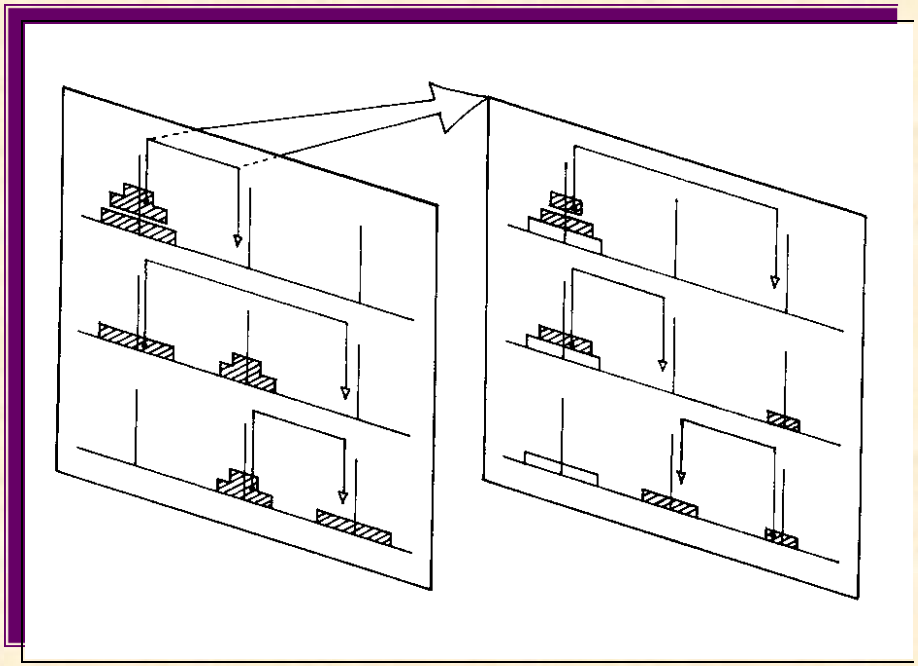
Hanoi tornyai:

Úgy tegyük át az N db korongot az A-ról a C pálcikára (B közvetítésével), hogy először a felső N–1 darabot a B-re tesszük (C segítségével), majd az így „felszabadult” legalsót a végleges helyére (C-re). Ezután nincs is már másra szükség, csak arra, hogy a B-n levő N–1 db-ot a C-re emelgessük. (Persze ezt ugyanazzal a szisztémával, mint amivel korábban az „1. lépést” megtettük.)

ELTE

Rekurzió

Hanoi tornyai:



ELTE

Rekurzió

A rekurzió helye a programkészítés folyamatában:

Rekurzív specifikáció → Nem rekurzív specifikáció

↓ ↓

Rekurzív algoritmus ↔ Nem rekurzív algoritmus

↓ ↓

Rekurzív programkód → Nem rekurzív programkód

↓ ↓

Számítógép Számítógép

ELTE

Rekurzív specifikáció és algoritmus

Faktoriális:

$$n! = \begin{cases} n*(n-1)! & ha \;\; n>0 \\ 1 & ha \;\; n=0 \end{cases}$$

```
Fakt(n):
  Ha n>0 akkor Fakt:=n*Fakt(n-1)
  különben Fakt:=1
 Függvény vége.
```

ELTE

Rekurzív specifikáció és algoritmus

Fibonacci számok:

$$Fib(n) = \begin{cases} 0 & ha\ \ n = 0 \\ 1 & ha\ \ n = 1 \\ Fib(n-1) + Fib(n-2) & ha\ \ n > 1 \end{cases}$$

```
Fib(n):
   Ha n=0 akkor Fib:=0
   különben ha n=1 akkor Fib:=1
   különben Fib:=Fib(n-1)+Fib(n-2)
Függvény vége.
```

ELTE

Rekurzív specifikáció és algoritmus

Binomiális számok:

$$B(n,k)=\begin{cases}1 & ha\ \ k=0\\ B(n-1,k)+B(n-1,k-1) & ha\ \ 0<k<n\\ 1 & ha\ \ k=n\end{cases}$$

```
Bin(n,k):
  Ha k=0 vagy k=n akkor Bin:=1
  különben Bin:=Bin(n-1,k)+Bin(n-1,k-1)
 Függvény vége.
```

$$B(n,k)=\begin{cases}1 & ha\ \ k=0\\ B(n,k-1)*\dfrac{n-k+1}{k} & ha\ \ 0<k\le n\end{cases}$$

```
Bin(n,k):
  Ha k=0 akkor Bin:=1
  különben Bin:=Bin(n,k-1)*(n-k+1)/k
 Függvény vége.
```

ELTE

Rekurzív algoritmus

Gyorsrendezés:

```
Quick(A,e,v):
   Szétválogat(A,e,v,k)
   Ha k-e>1 akkor Quick(A,e,k-1)
   Ha v-k>1 akkor Quick(A,k+1,v)
Eljárás vége.
```

Meggondolandó:

1. a paraméterek eltérő hozzáférési joga;
2. a „Szétválogat” eljárás;

ELTE

Rekurzív algoritmus

Szétválogatás helyben:

```
Szétválogat(A,e,v,k):
  i:=e; j:=v; y:=A(e)
  Ciklus amíg i<j
    Ciklus amíg i<j és A(j)≥y
      j:=j-1
    Ciklus vége
    Ha i<j akkor A(i):=A(j); i:=i+1
              Ciklus amíg i<j és A(i)<y
                i:=i+1
              Ciklus vége
              Ha i<j akkor A(j):=A(i); j:=j-1

  Ciklus vége
  A(i):=y; k:=i
Eljárás vége.
```

ELTE

Rekurzív algoritmus

Gyorsrendezés:

```
Quick(A,e,v):
   Szétválogat(A,e,v,k)
   Rendezés(A,e,k-1)
   Rendezés(A,k+1,v)
Eljárás vége.

Rendezés(A,e,v):
   Ha v-e>0 akkor Quick(A,e,v)
Eljárás vége.
```

Közvetlen rekurzió: Az A eljárás saját magát hívja.

Közvetett rekurzió: A hívja B-t, B hívja A-t.

ELTE

Emlékeztető: grafika és rekurzió

Rekurzív festés pontonként:

```
RekPont(x,y):
   Pont(x,y)
   Ha Üres(x-1,y) akkor RekPont(x-1,y)
   Ha Üres(x,y-1) akkor RekPont(x,y-1)
   Ha Üres(x+1,y) akkor RekPont(x+1,y)
   Ha Üres(x,y+1) akkor RekPont(x,y+1)
Eljárás vége.
```

ELTE

Előreutalás: rekurzió Logo nyelven

Koch fraktál, Logo megoldás:

```
Tanuld koch :n :h
  Ha :n=0 [előre :h]
   [koch :n-1 :h/3 balra 60
    koch :n-1 :h/3 jobbra 120
    koch :n-1 :h/3 balra 60
    koch :n-1 :h/3]
Vége
```

koch 1 100 koch 2 100

ELTE

Rekurzív algoritmus

Hanoi tornyai:

```
Hanoi(n,A,B,C):
  Ha n>0 akkor  Hanoi(n-1,A,C,B)
                Ki: n,A,B
                Hanoi(n-1,C,B,A)
  Elágazás vége
Eljárás vége.
```

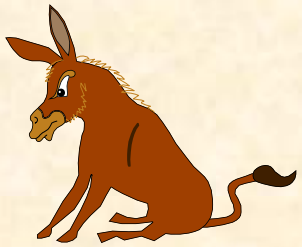
```
Hanoi(n,A,B,C):
  Ha n>1 akkor  Hanoi(n-1,A,C,B)
                Ki: n,A,B
                Hanoi(n-1,C,B,A)
       különben Ki: n,A,B
Eljárás vége.
```

ELTE

A megvalósítás problémái

Problémák

„Állatorvosi ló (-: csacsi :-)” – a faktoriális függvény



```
Fakt(n):
   Ha n=0 akkor  f:=1
          különben f:=n*Fakt(n-1)
  Fakt:=f
Függvény vége.
```

Tételezzük föl, hogy a „környezet” képes az eljárások *többszörös* (értsd: egymásba ágyazott) *hívás*ának lekezelésére. (*Visszatérési cím a verembe; aktuális és formális paraméterek összekapcsolása.*)

ELTE

A megvalósítás problémái

Milyen problémákkal szembesül a *fordítóprogram*, hogy a függvény végrehajtható legyen bármely n-re.

```
Pl. ?Fakt(2)?
 Fakt(2)=  2⇒n
           [Ha n=0 akkor … különben]
           f:=n* ?Fakt(n-1)?
           Fakt:=f
```

ELTE

A megvalósítás problémái

Összefoglalva a problémákat:

1. A *bemenő paraméter* problematikája.

Hogyan kerül a bemenő érték ugyanazzal a kóddal megvalósított eljáráshoz, azaz a rekurzívan újból a hívotthoz?

2. A *függvényeljárások értékvisszaadása*.

Hogyan kerül a (rekurzívan) hívott függvény értéke a hívó eljárásban „felszínre”?

3. A *lokális változók egyedisége*.

ELTE

A megvalósítás problémái

A megoldás vázlata:

1. és 3. probléma – belépéskor *verme*lés, kilépéskor *verem*ből kivétel.

2. probléma (nem csak rekurziónál) – az érték *verem*be tétele közvetlenül a visszatérés előtt.

A függvényt *eljárás*sá alakítva:

```
Fakt(n):
  Verembe(n)
  Ha n=0 akkor f:=1
  különben Fakt(n-1); Veremből(f)
           n:=Veremtető; f:=n*f
  Veremből(n); Verembe(f)
 Függvény vége.
```

ELTE

A megvalósítás problémái

Megállapodások:

- Az *eljárásnév* egyben egy *lokális változó* azonosítója, „elvárható” típussal.
- A veremműveleteknek *nem* tüntetjük föl a *veremparaméter*ét.
- Igazából azzal sem törődünk, hogy a verem garantáltan *azonos típusú* adatok gyüjtője legyen. (Most nem ez a lényeg, bár véletlenül teljesül.)

Rekurzió és iteráció

Bajok a rekurzióval

Hely: nagyra dagadt veremméret.

Idő: a vermelés adminisztrációs többletterhe, a többszörösen ismétlődő hívások.

ELTE

Rekurzió és iteráció

Példa: Fibonacci-számok

$$Fib(n) = \begin{cases} 0 & ha \ \ n = 0 \\ 1 & ha \ \ n = 1 \\ Fib(n-1) + Fib(n-2) & ha \ \ n > 1 \end{cases}$$

```
Fib(n):
  Ha n=0 akkor Fib:=0
  különben ha n=1 akkor Fib:=1
  különben Fib:=Fib(n-1)+Fib(n-2)
 Függvény vége.
Fib(n):
  F(0):=0; F(1):=1
  Ciklus i=2-től n-ig
    F(i):=F(i-1)+F(i-2)
  Ciklus vége
  Fib:=F(n)
Függvény vége.
```

ELTE

Rekurzió és iteráció

Jobbrekurzió

A rekurzió problematikáját – mint láttuk – a lokális adatok, ill. paraméterek kezelése jelenti. Ha az eljárás ***utolsó*** utasításaként szerepel a rekurzív hívás, akkor ***nincs szükség*** a lokális adatok és paraméterek visszaállítására, azaz ***vermelésére***.

Ezt az esetet nevezzük jobbrekurziónak (vagy hosszabban jobboldali rekurziónak).

ELTE

Rekurzió és iteráció

Jobbrekurzió alapesetei

```
R(x):
   S(x)
   Ha p(x) akkor R(x)
Eljárás vége.
```

```
R(x):
   Ciklus
      S(x)
   amíg p(x)
   Ciklus vége
Eljárás vége.
```

Azaz az átírás egy egyszerű hátultesztelős ciklus.

ELTE

Rekurzió és iteráció

Jobbrekurzió alapesetei

```
R(x):
   Ha p(x) akkor T(x); R(x)
Eljárás vége.
```

```
R(x):
   Ciklus amíg p(x)
      T(x)
   Ciklus vége
Eljárás vége.
```

Azaz az átírás egy egyszerű elöltesztelős ciklus.

ELTE

Rekurzió és iteráció

Jobbrekurzió alapesetei

```
R(x):
   S(x)
   Ha p(x) akkor T(x); R(x)
         különben U(x)
Eljárás vége.
```

```
R(x):
   S(x)
   Ciklus amíg p(x)
      T(x); S(x)
   Ciklus vége
   U(x)
Eljárás vége.
```

ELTE

Rekurzió és iteráció

Jobbrekurzió példa

Egy szó kiírása:

```
Kiír(szó):
  Ha nem üres?(szó) akkor
    Ki: első(szó)
    Kiír(elsőutániak(szó))
Eljárás vége.
```

```
Kiír(szó):
  Ciklus amíg nem üres?(szó)
    Ki: első(szó)
    szó:=elsőutániak(szó)
  Ciklus vége
Eljárás vége.
```

ELTE

Rekurzió és iteráció

Jobbrekurzió példa

Logaritmikus kiválasztás (az A vektor E. és V. eleme között az X elem biztosan megtalálható):

```
Kiválaszt(A,X,K,E,V):
  K:=(E+V) div 2
  Elágazás
    A(K)<X esetén E:=K+1
    A(K)>X esetén V:=K-1
  Elágazás vége
  Ha A(K)≠X akkor Kiválaszt(A,X,K,E,V)
Eljárás vége.
```

ELTE

Rekurzió és iteráció

Jobbrekurzió példa

Logaritmikus kiválasztás (az A vektor E. és V. eleme között az X elem biztosan megtalálható):

```
Kiválaszt(A,X,K,E,V):
  Ciklus
    K:=(E+V) div 2
    Elágazás
      A(K)<X esetén E:=K+1
      A(K)>X esetén V:=K-1
    Elágazás vége
  amíg A(K)≠X
  Ciklus vége
Eljárás vége.
```

ELTE

Rekurzió és iteráció

Balrekurzió

Ha az eljárás első utasításaként szerepel a rekurzív hívás, akkor a rekurzió lényegében az első nem rekurzívan számolható érték megkeresését szervezi. Majd a visszatérés(ek) után végzi el a transzformációt. Vagyis fordított sorrendű földolgozást végez.

Általános sémája:

```
R(x,y):
   Ha p(x,y) akkor R(f(x),y); S(x,y)
             különben T(x,y)
Eljárás vége.
```

ELTE

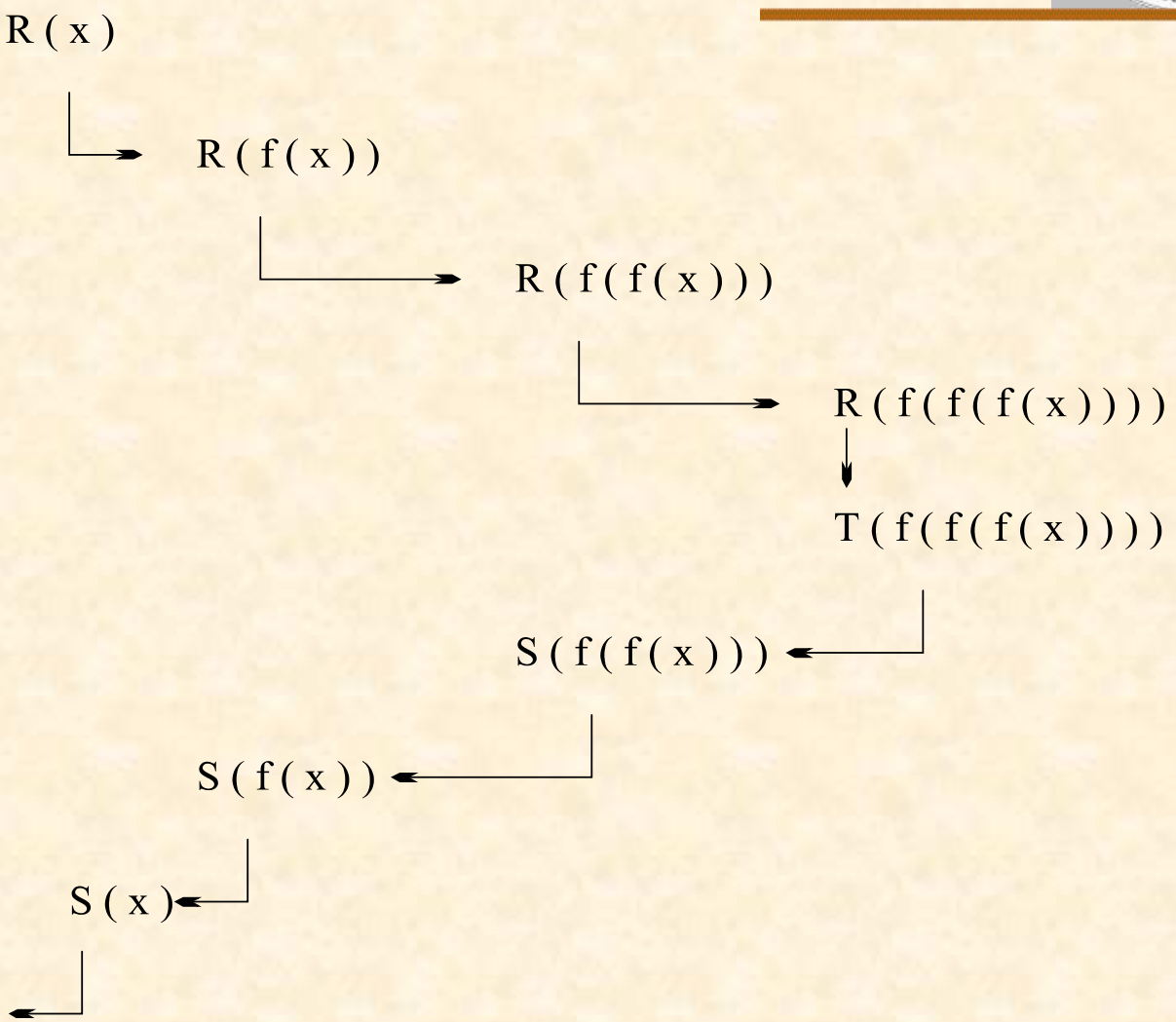
Rekurzió és iteráció

Balrekurzió – példák a feldolgozandóra

- a sorozat elemei egy soros állomány rekordjai,
- a sorozat elemeit láncolt ábrázolással tároljuk és az aktuális elemtől csak a következőt lehet elérni, az előzőt nem,
- a sorozat elemeit egy sor- vagy veremstruktúrában tároljuk,
- a sorozat elemeit mindig az előző elemből számítjuk, s a sorozat előre meg nem állapítható tagjától visszafelé kell kiírni az elemeket,
- nyelvünk csak olyan szövegkezelő függvényeket ismer, amelyek a szöveg első karakterét tudják megadni, illetve az első elhagyásával keletkezett részt.

ELTE

Rekurzió és iteráció



ELTE

Rekurzió és iteráció

Balrekurzió általános átírása

Ha feltehetjük, hogy az f függvénynek létezik inverze:

```
R(x,y):
   N:=0
   Ciklus amíg p(x,y)
      x:=f(x); N:=N+1
   Ciklus vége
   T(x,y)
   Ciklus I=1-től N-ig
      x:=f^-1(x); S(x,y)
   Ciklus vége
Eljárás vége.
```

ELTE

Rekurzió és iteráció

Balrekurzió általános átírása

Ha az f függvénynek nem létezik inverze:

```
R(x,y):
  N:=0
  Ciklus amíg p(x,y)
    Verembe(x); x:=f(x); N:=N+1
  Ciklus vége
  T(x,y)
  Ciklus i=1-től N-ig
    Veremből(x); S(x,y)
  Ciklus vége
Eljárás vége.
```

ELTE

Rekurzió és iteráció

Balrekurzió példa

Egy szó „tükrözve” kiírása, azaz betűi sorrendje megfordítása:

```
Tükröz(szó):
   Ha nem üres?(szó) akkor
      Tükröz(elsőutániak(szó))
      Ki: első(szó)
Eljárás vége.
```

ELTE

Rekurzió és iteráció

Balrekurzió példa

Egy szó „tükrözve” kiírása, azaz betűi sorrendje megfordítása:

```
Kiír(szó):
  N:=0
  Ciklus amíg nem üres?(szó)
    Verembe(első(szó)); N:=N+1
    szó:=elsőutániak(szó)
  Ciklus vége
  Ciklus i=1-től N-ig
    Veremből(B); Ki: B
  Ciklus vége
Eljárás vége.
```

ELTE

Iteráció és rekurzió

Problémák a ciklussal:

- Vannak programozási nyelvek, ahol nincs ciklus, csak rekurzió.

ELTE

Iteráció és rekurzió

Elöltesztelős ciklus átírása

```
R(x):
  Ciklus amíg p(x)
    T(x)
  Ciklus vége
Eljárás vége.
```

```
R(x):
  Ha p(x) akkor T(x); R(x)
Eljárás vége.
```

Azaz az átírás eredménye egy jobbrekurzió.

ELTE

Iteráció és rekurzió

Hátultesztelős ciklus átírása

```
R(x):
   Ciklus
      S(x)
   amíg p(x)
   Ciklus vége
Eljárás vége.
```

```
R(x):
   S(x)
   Ha p(x) akkor R(x)
Eljárás vége.
```

Azaz az átírás eredménye egy jobbrekurzió.

ELTE

Iteráció és rekurzió

Számlálós ciklus átírása

```
R(x):
  Ciklus i=1-től N-ig
    S(x,i)
  Ciklus vége
Eljárás vége.
```

Először írjuk át elöltesztelősre!

```
R(x):
  i:=1
  Ciklus amíg i≤N
    S(x,i); i:=i+1
  Ciklus vége
Eljárás vége.
```

ELTE

Iteráció és rekurzió

Számlálós ciklus átírása

Erre már van átírási szabályunk:

```
R(x):
  i:=1; RR(x,i)
Eljárás vége.
```

```
RR(x,i):
  Ha i≤N akkor S(x,i); i:=i+1; RR(x,i)
Eljárás vége.
```

Egyszerűbb az i növelése paraméterátadásnál:

```
RR(x,i):
  Ha i≤N akkor S(x,i); RR(x,i+1)
Eljárás vége.
```


Vége