Rekurzió

A rekurzív gondolat egy olyan feladat-megfogalmazás, melyben egy metódust átparaméterezve addig hívunk újra meg újra, amíg a kilépési feltételt el nem érjük.

A rekurzióban a METHOD() metódus meghívásának helye a METHOD() metódus!

A rekurzív metódusok ezek alapján **saját magukat hívják meg**. A hívásnál új paramétereket adnak át. Ha nincs új paraméter, akkor globális paramétereket kell használni (ez a Python-tól – úgyszólván -, nem áll távol…).

A rekurzió a **ciklust** váltja ki. A legtöbb rekurzióban ezért nem található ciklus! (Létezik kivétel!)

Előnye, hogy rövid, elegáns kódokat tudunk létrehozni. Hátránya, hogy - sok hívás esetén -, megzabálja a memóriát. Amennyiben számokat keresünk rekurzívan, figyelni kell a tárolási méretekre is, mert hamar ki tudunk lépni egy-egy változó maximális méretéből.

A rekurzió több modellel is leírható. Amennyiben a rekurzív metódus egy függvény, a pince modell a legjobb leírója: elindulunk lefele egy pince lépcsőjén, és minden lépcsőn hagyunk egy kannát. Amikor elérjük a kilépési feltételt (pl. a pincét), a legalsó kannában lévő folyadékot beletöltjük az előző lépcsőn lévő kannába. Mire felérünk a pincéből, minden kanna tartalma valamilyen mértékben megváltozott. Ebben az esetben viszont, csak a legfelső kanna tartalma a számunkra érdekes.

Amennyiben a rekurzív metódus egy eljárás, akkor nincs visszafelel lépegetés. Ilyenkor a kilépési feltétel bekövetkezésekor érvényben lévő változó-tartalmakat keressük.

Programozási taktika:

1. Az eredeti problémát az eredetivel megegyező (nagyon hasonló) részproblémákra bontjuk, mindaddig, amíg az alapesetet el nem érjük.
2. A részproblémák megoldásait *összekapcsoljuk*
   1. minden rekurzív definíciónak van egy, vagy több alapesete, melyeknek triviális megoldása van.
      1. pl. rekurzív hatványozás:  
         nk – ha k = 0 return 1
      2. pl. rekurzív faktoriálisnál:  
         n! – ha n = 0 return 1
   2. minden rekurzív definíciónak van egy, vagy több rekurzív esete, mikor a függvény *új paraméterekkel* meghívja **saját magát**.

A rekurzió típusai:

1. Hívások száma (a kódon belül) alapján:
   1. egyszeres rekurzió, mely csak egy hívást tartalmaz;
   2. többszörös, mikor önmagát többféleképpen paraméterezve hívhatja meg.
2. A hívott függvény szerint
   1. közvetlen rekurzió, mikor a fv. saját magát hívja;
   2. közvetett rekurzió, mikor f => g, majd a g => f módon valósul meg a hívás.
3. Nevesítés szerint:
   1. Névvel ellátott rekurzió;
   2. Névtelen (lambda) rekurziók;

REKURZÍV ADATÍPUSOK

Amikor egy program előre nem látható mennyiségű adatot produkál, a programozó rekurzív módon definiálhatja az adatszerkezetet. Ezt kétféleképpen lehet megtenni, induktív, vagy koinduktív módon.

Tipikus induktív adathalmaz pl. a **N** – természetes számok halmaza: Definíciója:

Egy természetes szám vagy 0, vagy n+1 alakú, ahol n egy természetes szám.

Hasonló a láncolt lista:

***Class Csomópont:***

***KövetkezőCsomópont = null***

*#mit tárolunk ezen a csomóponton:*

***objektum***

A fenti példából kihagytam a sallangokat 😊.

Példa egy koinduktív adathalmaz definícióra:

„Egy string-lánc egy objektum, ahol:

ELSO\_betu – egy darab (egyelmű) string

TOBBI\_betu – egy string-lánc”

Hasonlít az induktívra, de ez pontosan meghatározza, hogy miként megy a feltöltése a string-láncnak.