4. óra

# Tömbök

Nem tömb az összetartozó, de különböző típusú adatok: rekord (struct)
Pl.: ember.név ember.kor ember.magasság ember.nem

Ugyanolyan típusú adatok: tömb (array)
Pl.: jegyeim[9]=5 jegyeim[0]=5

## Statikus tömb

Előre tudjuk, hány db elemünk van (vagy maximum hány db) Ezt előre lefoglaljuk a memóriában. Pl.: A hét 7 napjának neve 7 elem. A csoport névsora 20 elem, de csak 17-en vagyunk itt, 3 üres marad.

### Konstans statikus tömb

Előre feltöltött adatokkal.

### Normál statikus tömb

Futás során töltjük fel adatokkal.

## Dinamikus tömb

A program futása során hozzá tudok venni még elemeket. Egyelőre nem fogjuk használni.

## Tömbök deklarálása

változó\_típusa változó\_neve[];

string jegyek[];

string het[7];

const string het[7]={"hetfo","kedd","szerda","csutortok","pentek","szombat","vasarnap"}

## Memóriában

A tömb változója egy memóriaterületre mutat. Az indexelés [0]-tól kezdődik, nem [1]-től. Analógia: "Hány elemet kell átugranom, hogy odaérjek."

## Magyar kártya mennyit ér

1 órán keresztül írtuk a specifikációt, én e helyett 5 perc alatt megírtam a programot specifikáció nélkül. (Ha ötöst akarok, nekem is szenvednem kell a specifikációval.)

Ilyesmik voltak a táblán:

Be: k-név szöveg, konstans k-nevek tömb(1..8 szöveg), kertek tömb(1..8 egész)

Ki: k-érték egész

Ef: ∃i (1≤i≤8): k-név = kertek[i]

Uf: ∃j (1≤j≤8): k-érték=kertek[j] ∧ k-név=k-nevek[j]

Algoritmus: struktogram

## Tábor napok

Be: szöveg: kezdőnap, egész: hossz, konstans szöveg tömb 1-7: napok (hét napjai)

Ki: szöveg: végnap = napok[j]

Ef: kezdőnap ∈ napok (∃i(1≤i≤7): kezdőnap = napok[i]), hossz>0

Uf: (i = kezdőnap sorszáma) ∃j(1≤j≤7): végnap=napok[j] ∧ ∃i(1≤i≤7): kezdőnap = napok[i] ∧ j = ((i + hossz-1) mod 7)+1

Algoritmus:
i=1;
ciklus amíg (kezdőnap!=napok[i]) {i++}
i=(i+hossz-1) mod 7
végnap=napok[i]

## HF

konst\_tomb.pdf-ből 4, 6, 7 + tábor napok implementálása