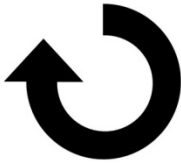




Bevezetés a programozásba

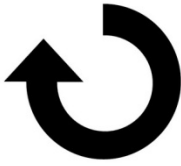
2. Előadás Program konstrukciók: Elágazás, Ciklus

Ismétlés I.



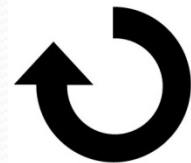
- **Kifejezés:** Olyan műveleti jeleket és értékeket tartalmaz, amelyeknek van együtt jelentése
- **Típus:** Értékhalmoz és művelet halmoz együttese
- **Bemenet (Input):** a felhasználó adatokat/értékeket közöl a programmal
- **Kimenet (Output):** a program megörökíti/közli az eredményeket/adatokat a külvilággal

Ismétlés II.

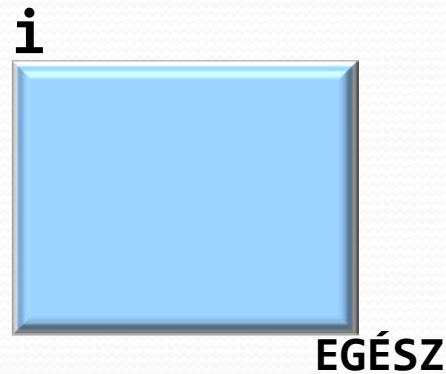


- **Változó:** Lefoglalt memóriaterület, amelyben értéket lehet tárolni. (egész, valós, logikai, memóriacím, karakter ...)
A memóriaterülethez egy változónevet rendelünk – ezzel a névvel hivatkozunk rá.
- „Sima változó” esetén a memória lefoglalása és címkézése egy lépésben megtörténik.
- Deklaráció
 - Típus és név meghatározása
 - i: **EGÉSZ**;
- Értékkadás
 - $i := 4$;

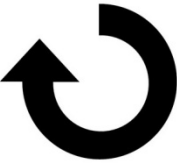
Változók I.



→ **i: EGÉSZ** //deklaráció

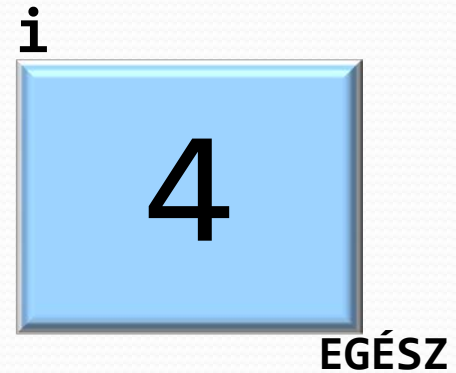


Változók II.



i: EGÉSZ //deklaráció

 **i := 4** //értékadás



Egyszerű Program

PROGRAM kifejezés

KI: „Helló világ”

PROGRAM_VÉGE

Szekvencia

PROGRAM szekvencia

VÁLTOZÓK:

a: EGÉSZ

BE: a

a := a + 1

a := a * 2

KI: a

PROGRAM_VÉGE

Krumplis példa

- Menj el a sarki közértbe és vedyél 1 kiló krumplit és ha sikerült gyere haza.
- Ha nincs nyitva a bolt vagy ha nincs krumpli, akkor menj el a piacra.
- Vedyél a piacon 1 kiló krumplit és gyere haza, ha sikerült.
- Ha nincs krumpli a piacon gyere haza.



Elágazás I.

- A programunkat nem mindig fogalmazhatjuk meg egyértelműen egymást követő utasítások sorozataként.
- Előfordulhat, hogy egyes utasításokat csak bizonyos esetben kell elvégeznie a programnak, valamilyen feltételtől függenek.
- Ezek lekezelésére teszünk elágazásokat a programba, ennek egyik ágát fogja végrehajtani a program a feltétel teljesülése esetén, míg egy másik ágát a feltétel nem teljesülése esetén.
- Az elágazás szerkezete:

HA *<feltétel>*

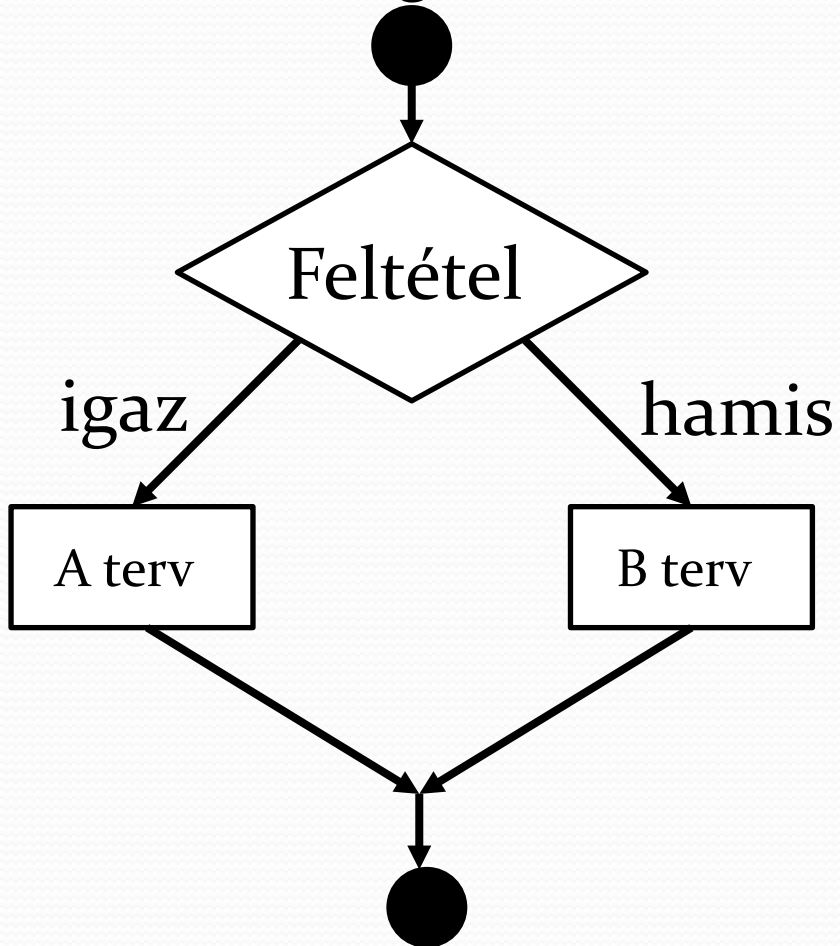
AKKOR *<utasítások>*

KÜLÖNBEN *<utasítások>*

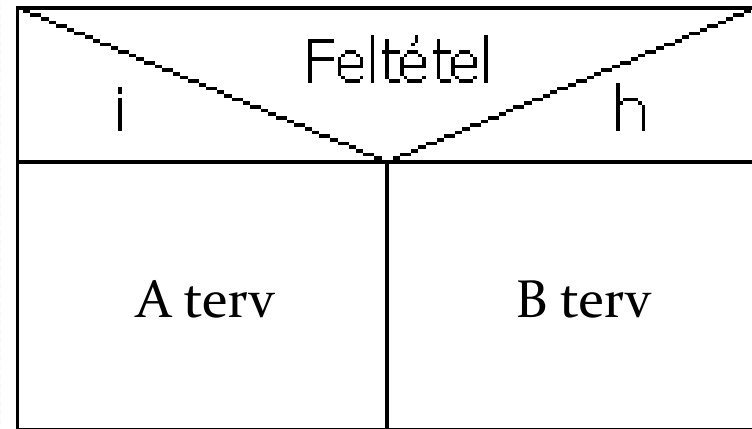
HA_VÉGE

Elágazás II.

- Blokkdiagramm



- Stuktrogramm



Elágazás III.

- A feltétel egy logikai kifejezés (pl.: $a < 5$), amelyben lehet kötőszavakat használni (**VAGY**, **ÉS**), így összetettebb kifejezéseket is megadhatunk feltételként.
- Ha a feltétel teljesül, az **AKKOR** ág utasításai hajtódnak végre, itt annyi utasítást teszünk egymást követően, amennyit csak szeretnénk.
- Ha a feltétel nem teljesül, a **KÜLÖNBEN** ág utasításai hajtódnak végre.
- A **KÜLÖNBEN** ág kihagyható, ha nincs szükségünk rá, ekkor csak annyit írunk:
 HA *<feltétel>*
 AKKOR *<utasítások>*
 HA_VÉGE
- Mindig figyeljünk arra, hogy minden **HA**-t megfelelő helyen lezárjunk egy **HA_VÉGE**-vel

Elágazás általánosan

PROGRAM

...

HA *feltétel* AKKOR

igaz ág, „if ág”

[KÜLÖNBEN
hamis ág, „else ág”]

HA_VÉGE

...

PROGRAM_VÉGE

Elágazás példa I.

PROGRAM elágazás

VÁLTOZÓK:

a: EGÉSZ

BE: a

HA a > 0 AKKOR

KI: „pozitív”

HA_VÉGE

PROGRAM_VÉGE

Elágazás példa II.

PROGRAM elágazás

VÁLTOZÓK:

a: EGÉSZ

BE: a

HA a > 0 AKKOR

KI: „pozitív”

KÜLÖNBEN

KI: „nem pozitív”

HA_VÉGE

PROGRAM_VÉGE

Elágazás példa III.

```
PROGRAM elágazás
  VÁLTOZÓK:
    a: EGÉSZ

  BE: a
  HA a > 0 AKKOR
    KI: „pozitív”
  KÜLÖNBEN
    HA a < 0 AKKOR
      KI: „negatív”
    KÜLÖNBEN
      KI: „nulla”
    HA_VÉGE
  HA_VÉGE

PROGRAM_VÉGE
```

Specifikáció I.

- A feladat: **„Adjuk meg egy valós szám gyökét!”**
- De: Mi történik, ha a szám negatív?
 - a program futásidejű hibával leáll
 - a program nem ad semmilyen eredményt
 - a program kiírja, hogy érvénytelen a bemenő adat
 - a program áttér komplexszám tartományra és megoldja a feladatot
- Pontosításra szorul a feladat

Specifikáció II.

- A specifikáció lényege, hogy a feladatot a lehető legprecízebben megfogalmazzuk
- **Előfeltétel:** Azaz milyen bemeneteket kell minden esetben lekezelnie a programnak, milyen körülmények között követelünk helyes működést.
- **Utófeltétel:** A meghatározott bemenetekre milyen válaszokat/eredményeket kell adnia a programnak, mit várunk a kimenettől, mi a kapcsolat a bemenet és kimenet között.

Specifikáció III.

- Ha a megadott **feltételek** teljesülnek akkor a program megoldja a feladatot.
- A specifikáció nem utasításokból áll, hanem feltételekből, mert a feladatot írja le, nem a programot.
(*A specifikáció a MIT, a program a HOGYAN*)

Példa Specifikációra I.

- BE: a : nem negatív valós
KI: b : nem negatív valós, ahol $b*b = a$

PROGRAM elágazás

VÁLTOZÓK:

a, b : VALÓS

BE: a

$b := a ^ 0.5$

KI: b

PROGRAM_VÉGE

Példa Specifikációra II.

- BE: a: valós

KI: ha $a \geq 0$: b nem negatív valós, ahol $b^2 = a$

PROGRAM elágazás

VÁLTOZÓK:

a, b: VALÓS

BE: a

HA $a \geq 0$ AKKOR

b := $a^{0.5}$

KI: b

HA_VÉGE

PROGRAM_VÉGE

Példa Specifikációra III.

- BE: a: valós
- KI: ha $a \geq 0$: b nem negatív valós, ahol $b^2 = a$
ha $a < 0$: „érvénytelen bemenet”

```
PROGRAM elágazás
  VÁLTOZÓK:
    a, b: VALÓS
  BE: a
  HA a >= 0 AKKOR
    b := a ^ 0.5
  KI: b
  KÜLÖNBEN
    KI: „érvénytelen adat”
  HA_VÉGE
PROGRAM_VÉGE
```

Összeadó program v1.0

PROGRAM összeadó1

VÁLTOZÓK:

a, b: EGÉSZ

BE: a, b

KI: $a + b$

PROGRAM_VÉGE

Összeadó program v1.1

PROGRAM összeadó2

VÁLTOZÓK:

a, b, c: EGÉSZ

BE: a, b, c

KI: $a + b + c$

PROGRAM_VÉGE

Összeadó program v1.2

PROGRAM összeadó3

VÁLTOZÓK:

a,b,c,d,e,f,g: EGÉSZ

BE: a, b, c, d, e, f, g

KI: a + b + c + d + e + f + g

PROGRAM_VÉGE

Sok számot hogyan adunk össze?

- A program tervezésekor alapvető szempont, hogy a programszöveg nem függhet az adatoktól.
- Az nem megoldás, hogy legyen annyi változóm ($a+b+c+d+e+f+g+h+i+j+k+l\dots$) amennyi kell. Más módszerre van szükség.
- Ha kielemezzük a problémát akkor észrevehetjük, hogy lényegében ugyan azt az utasítást (műveletet) kell megismételgetni, amíg a kívánt eredményt nem kapjuk.
- Ehhez **ismétlődő szakaszokat** kell létrehozni a programban

Sok számot hogyan adunk össze?

- Hogyan oldjuk meg sok szám beolvasását kevés változóval?
- Tételezzük fel, hogy ismerjük a beolvasandó számok számát. Azaz legyen az első adat a beolvasandó sorozat hossza (n).
- Legyen egy „tároló” változónk, mely az aktuális részeredményt tartalmazza (**összeg**).
- Szükség lesz, még egy változóra, mely a soron következő beolvasott számot tartalmazza (a).
- A módszer: **BE**: beolvasandó számok száma (n)
Ismételd meg az **összeg** növelését a következő sorozat elemmel (a) n alkalommal.

Összeadó program v2.0

PROGRAM összeadó3

VÁLTOZÓK:

a, b, összeg: EGÉSZ

BE: a, b

összeg := a + b

KI: összeg

PROGRAM_VÉGE

Összeadó program v2.1

PROGRAM összeadó4

VÁLTOZÓK:

a, b, összeg: EGÉSZ

BE: a

összeg := a

BE: b

összeg := összeg + b

KI: összeg

PROGRAM_VÉGE

Összeadó program v2.3

PROGRAM összeadó6

VÁLTOZÓK:

a, összeg: EGÉSZ

összeg := 0

BE: a

összeg := összeg + a

BE: a

összeg := összeg + a

KI: összeg

PROGRAM_VÉGE

Összeadó program v2.3

PROGRAM összeadó6

VÁLTOZÓK:

a, összeg: EGÉSZ

összeg := 0

BE: a

összeg := összeg + a

BE: a

összeg := összeg + a

KI: összeg

PROGRAM_VÉGE

Ismétlődés

Ismétlődő
szakasz, amit
annyiszor kell
végrehajtani
ahány adat van

Összeadó program v3.0

PROGRAM sorozatösszeadó

VÁLTOZÓK:

n, a, összeg, i: EGÉSZ

BE: n

i := 0

összeg := 0

CIKLUS AMÍG i < n

BE: a

összeg := összeg + a

i := i + 1

CIKLUS_VÉGE

KI: összeg

PROGRAM_VÉGE

Összeadó program összevetése

```
PROGRAM összeadó
VÁLTOZÓK:
    a, összeg: EGÉSZ

    összeg := 0
    BE: a
    összeg := összeg + a
    BE: a
    összeg := összeg + a
    KI: összeg
PROGRAM_VÉGE
```

```
PROGRAM sorozatösszeadó
VÁLTOZÓK:
    n, a, összeg, i: EGÉSZ

    BE: n
    i := 0
    összeg := 0
    CIKLUS AMÍG i < n
        BE: a
        összeg := összeg + a
        i := i + 1
    CIKLUS_VÉGE
    KI: összeg
PROGRAM_VÉGE
```


Ciklus

- Gyakran előfordul, hogy valamilyen utasítást többször (akár nagyon sokszor) is végre akarunk hajtani.
- A ciklus az ismétlődés lehetősége a programozásban
- Egy programrészletet addig ismételünk meg, **amíg** (*while*) egy adott feltétel (ciklusfeltétel) teljesül, és befejezzük, ha ez a feltétel hamissá válik
 - Egyes programozási nyelvek az **amíg nem** (*until*) feltételt használják

Ciklus általánosan (fogalmak)

PROGRAM

...

Ciklusfeltétel

CIKLUS AMÍG *Logikai kifejezés*

Ciklusmag

programrészlet

CIKLUS_VÉGE

...

PROGRAM_VÉGE

Ciklus

- A ciklus magja egy tetszőleges önmagában teljes értékű programrészlet
- A ciklusfeltétel egy logikai típusú kifejezés
- A ciklusfeltétel kiértékelése újra és újra megtörténik lefutásonként
- Ha az eredmény hamis, a ciklus véget ér ,és a „ciklus_vége” sor után folytatódik
- Előfordulhat, hogy a kifejezés azonnal hamis, a program ilyenkor kihagyja a ciklusmag lépéseit

Ciklus változatai

- **Elöltesztelő:** a ciklusfeltétel a ciklusmag előtt van, már előre is csak akkor futnak le az utasítások, ha a feltétel teljesül
- **Hátultesztelő:** a ciklusfeltétel a ciklusmag után van, ezért az egyszer mindenképpen lefut, de többször csak akkor, ha teljesül a feltétel
- **Számláló:** mi akarjuk pontosan megadni, hányszor forduljon le a ciklusmag

Ciklus fajták

- Elöltesztelő ciklus szerkezete

CIKLUS AMÍG *ciklusfeltétel*

**** Csak akkor lép be, ha a feltétel már
kezdetben teljesült**

ciklusmag

CIKLUS_VÉGE

- Hátultesztelő ciklus szerkezete

CIKLUS

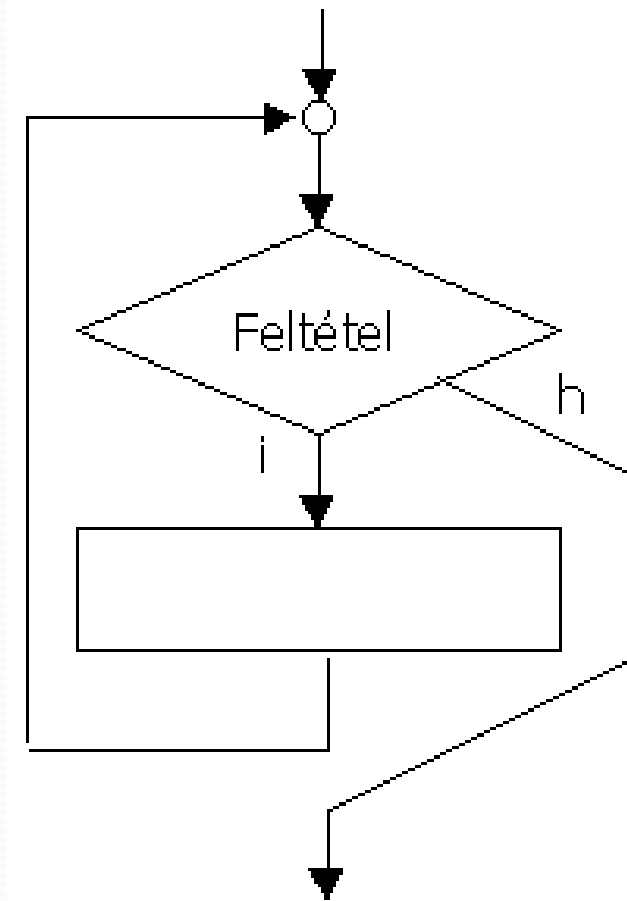
ciklusmag

**** Az utasítások egyszer mindenképpen
lefutnak**

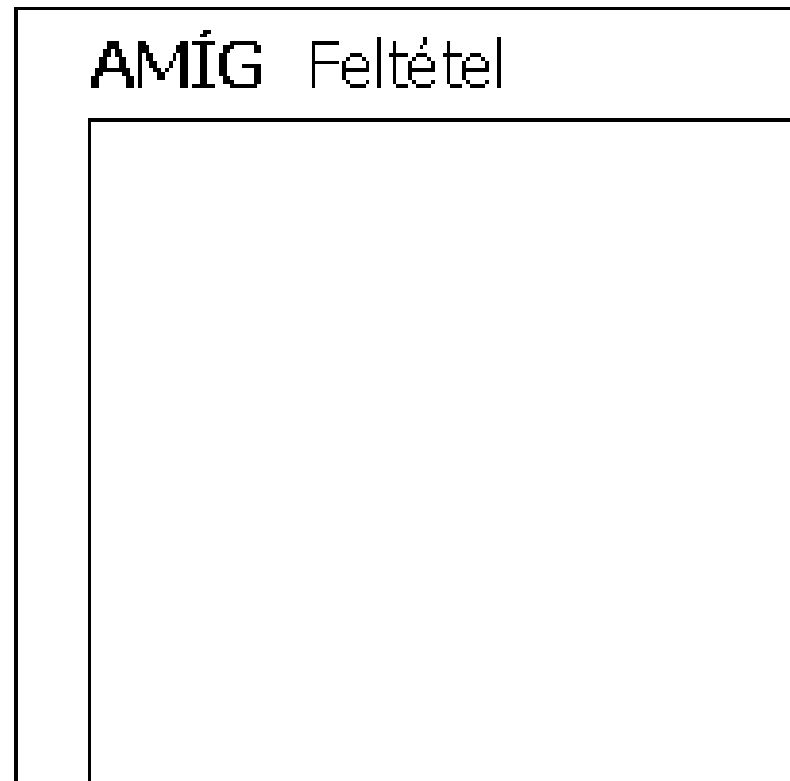
AMÍG *ciklusfeltétel*

Elöltesztelő ciklus

- Blokkdiagramm

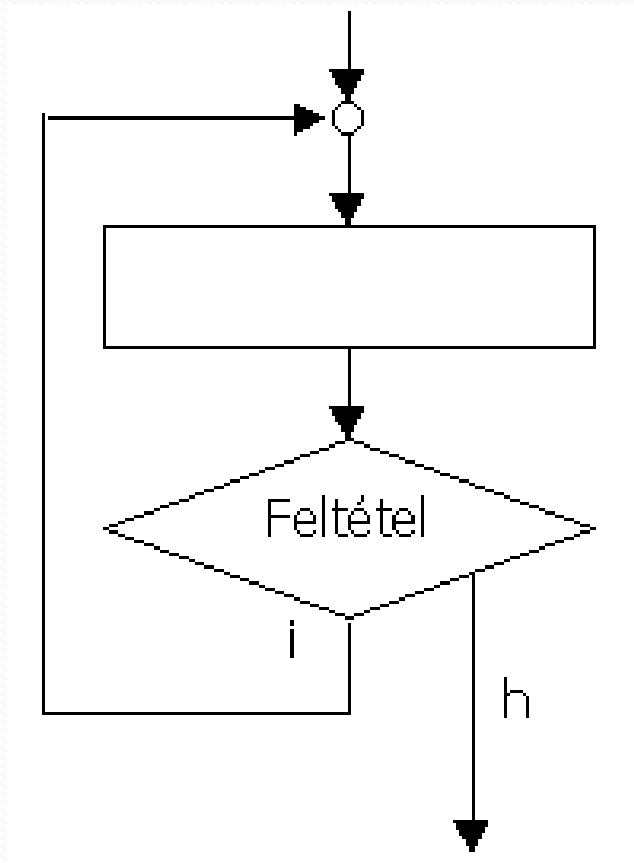


- Stuktrogramm

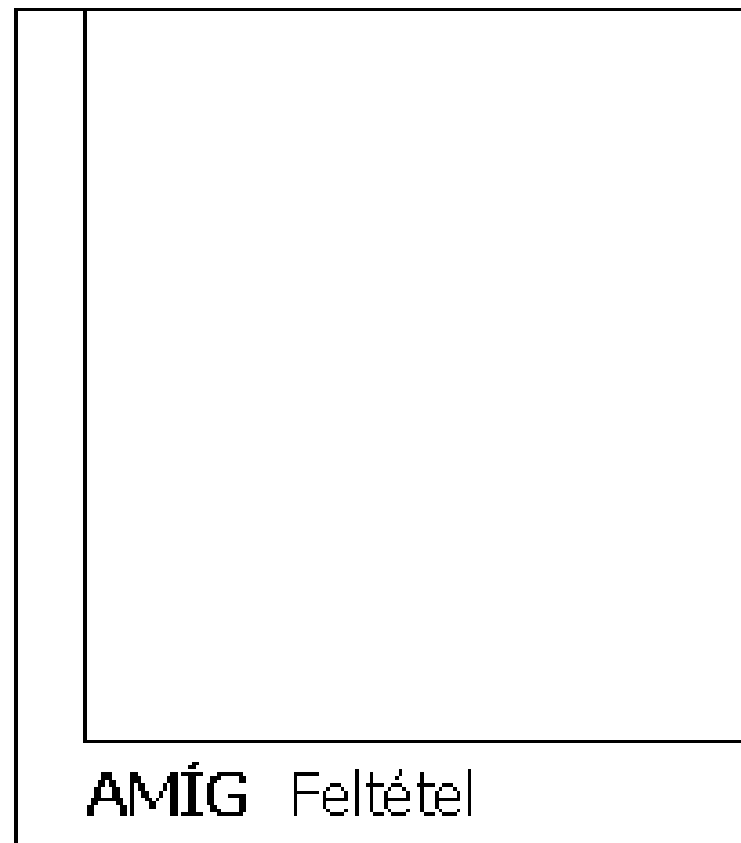


Hátultesztelő ciklus

- Blokkdiagramm



- Stuktrogramm



Összeadó program v3.0

PROGRAM sorozatösszeadó

VÁLTOZÓK:

n, a, összeg, i: EGÉSZ

BE: n

i := 0

összeg := 0

CIKLUS AMÍG i < n

BE: a

összeg := összeg + a

i := i + 1

CIKLUS_VÉGE

KI: összeg

PROGRAM_VÉGE

*n: ennyi elem van
a sorozatban*

*a: az aktuális
elem, amivel
éppen dolgozunk*

*i: az ennyiedik
elemnél tartunk*

*összeg: az eddig
feldolgozott
elemek összege*

Végtelen Ciklus *(Inkább Bug, mint Feature...)*

- Ha nem lenne ciklus (vagy ennek megfelelő más eszköz) a program csak annyi lépésből állhatna ahány sora van, ez nyilvánvalóan szűkös lenne.
- De túlzásokba is lehet esni.

```
...  
i := 0  
a := 1  
n := 9  
CIKLUS AMÍG i < n  
    a := a + 1  
CIKLUS_VÉGE  
...
```

```
...  
a := 1  
CIKLUS AMÍG a > 0  
    a := a + 1  
CIKLUS_VÉGE  
...
```

Végtelen Ciklus *(Inkább Bug, mint Feature...)*

- Ha nem lenne ciklus (vagy ennek megfelelő más eszköz) a program csak annyi lépésből állhatna ahány sora van, ez nyilvánvalóan szűkös lenne.
- De túlzásokba is lehet esni.

```
...  
i := 0  
a := 1  
n := 9  
CIKLUS AMÍG i < n  
    a := a + 1  
CIKLUS
```

Nincs
inkrementálva az
„i” ciklusváltozó!

```
...  
a := 1  
CIKLUS AMÍG a > 0  
    a := a + 1  
CIKLUS_  
...
```

A ciklusfeltétel
sohasem lesz
hamis!

Végtelen Ciklus *(Inkább Bug, mint Feature...)*

- Ha a ciklusfeltétel kifejezésében nincs változó, vagy egyik változó sem szerepel a ciklusmagban értékadás baloldalán, az gyanús!
- Érdemes mindig végiggondolni a változók jelentését a ciklusokban!

Ciklus összefoglalás

- Akkor használjuk, ha valamit többször kell végrehajtani.
- Mindig ki kell találni, hogy milyen kezdeti értékekkel, milyen ciklusfeltétellel és milyen ciklusmaggal oldható meg a feladat.
 - Egy nagyon gyakori eset, hogy ciklusváltozót használunk és abban számoljuk, hogy hányszor futott le a ciklus.

Programkonstrukciók

- Elemi programok: értékadás, BE, KI, ...
- Minden elemi program helyes program.
- Minden helyes program kombinálható:
 - szekvencia
 - elágazás (ha más kódra van szükség egyes esetekben)
 - ciklus (ha ismételni kell lépéseket)
- Az eredmény: a program egy hierarchikus szerkezete a fenti programkonstrukcióknak, és végső soron elemi programok kombinációja.

Programkonstrukciók összefoglalás

- Okos megbízható matematikusok
bebizonyították: Minden algoritmikusan
megoldható probléma megoldható szekvencia,
elágazás és ciklus segítségével.
- Tehát vége is a féléves anyagnak...