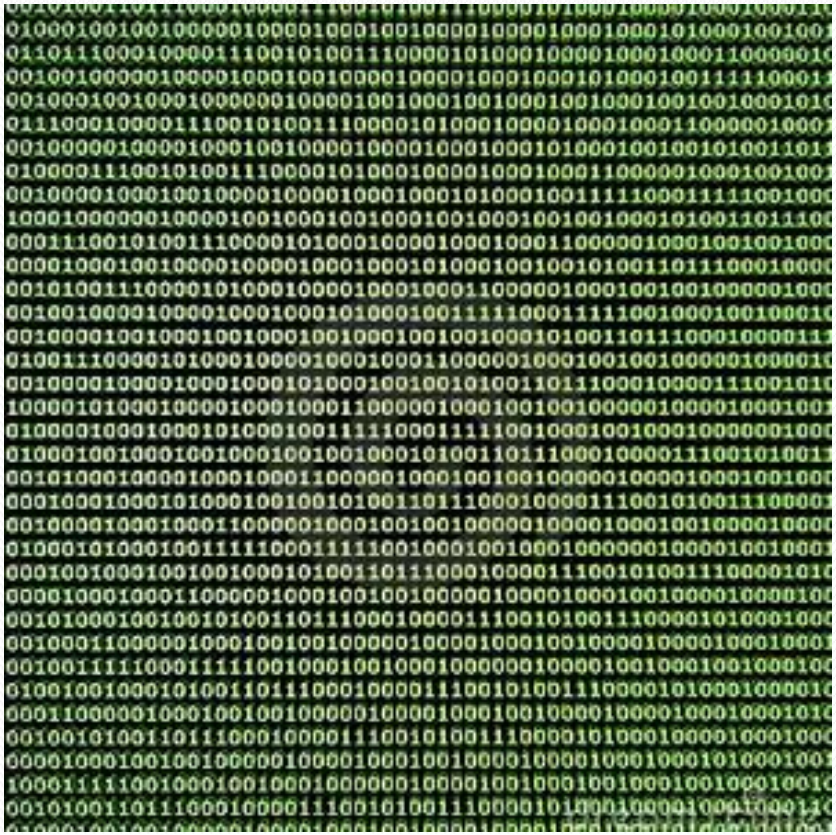
Bevezetés a programozásba

1. Előadás
Bevezetés

https://wiki.itk.ppke.hu/twiki/bin/view/PPKE/BevezetSAProgramozSba201516

Kommunikáció a számítógéppel

Emberi nyelvek



Számítógép nyelve

Kommunikáció a számítógéppel

Emberi nyelvek Számítógép nyelve

Programozási nyelvek

A programozás természete

- Hozzál krumplit!
- Hozzál egy kiló krumplit!
- Hozzál egy kiló krumplit a sarki közértből!
- Menj el a sarki közértbe, végy egy kosarat, tegyél bele egy kiló krumplit, adj annyi pénzt a pénztárosnak, amennyibe egy kiló krumpli kerül, tedd le a kosarat, gyere ki a közértből, és hozd haza a krumplit!

A programozás

- Recept készítés
  - Le kell írni lépésről lépésre a teendőket
  - Az egyes lépéseknek olyanoknak kell lenni, amit a számítógép megért
- Módszer keresés
  - Előfordulhat, hogy a gép által megértett parancsok között nincs olyan, amely megoldana egy részfeladatot
  - Ilyenkor a rendelkezésre álló parancsokat kombinálni kell

A programozás II.

- Matematika gyakorlatban
  - A számítógép alapvetően matematikai műveleteket ért meg közvetlenül
- Rendszer tervezés
  - A programok ritkán lesznek kész első nekifutásra, sok ember több napos munkája általában
  - Fontos, az előre tervezés, előre gondolkodás! (Második félévtől lesz jelentősége…)

Milyen a jó programozó?

- Absztrakciós készség
  - A modell helyes használata, az van a programban ami szükséges, és csak az
- Új fogalomrendszerekkel való gyors megismerkedés képessége
  - Programozási nyelvek érdemi elsajátítása
- Fogalmazási készség
  - A formalizmusok fogalmazása hasonlít az idegen vagy anyanyelvi fogalmazásra
- Algoritmikus gondolkodás
  - Meg kell tanulni a gép fejével gondolkodni

Mivel foglalkozunk a félévben?

- Alapvető fogalmakat tisztázunk
  - Kifejezések
  - Vezérlőszerkezetek
  - Változók, típusok, rekordok
  - Függvények
- Megtanuljuk a legegyszerűbb vezérlőszerkezeteket, két nyelven is
- Félév végére C++ nyelve függvényekkel, fájlkezeléssel, tömbökkel dolgozó programot fogunk írni.

Adminisztratív tudnivalók

- **wiki.itk.ppke.hu**
- Tantárgyak / 1. félév / Bevezetés a programozásba I.
  - Előadásra, gyakorlatra járás kötelező (max 3 hiányzás)
  - Pluszmínusz: félév végére legalább 0 pont kell
  - Papíros ZH: 30 pont, minimum: 5 pont
    - Program vázlat írás, hiba keresés, elmélet
  - Géptermi ZH: 60 pont, minimum: 10 pont
    - 3 óra alatt működő C++ kódok készítése
  - Csak gyakorlati jegy lesz a tárgyból, vizsga nincs
  - Pót pluszmínusz, PótZH-k

Alapfogalmak

- **Gépi kód:** a számítógép számára közvetlenül értelmezhető utasítások sorozata *(pl. futtatható állományok, exe-k)*
- **Programozási nyelv:** Egy magasabb szintű eszköz, amely segítségével ember közelebbi módon tudunk programot írni. *(Pl. Pascal, Basic, C++, Java, PLanG, stb.)*
- **Forrás kód:** az adott programozási nyelven megírt program. Általában ember által írt.
- **Fordítás:** az a folyamat, amikor a forrásprogramból gépi kódot állítunk elő, a ***fordító*** feladata.
- A ***fordítóprogram*** ellenőrzi a forráskód szintaktikáját, közben ***fordítási hibákat*** kaphatunk.

Kifejezések

1+1

Ezt mindenki érti. Tudjuk, hogy ezek számok és hogy a számokat össze lehet adni.

Kifejezések

1 + 

???

Értelmezési hiba.

Kifejezés

- A **kifejezés** olyan műveleti jeleket és értékeket tartalmaz, amelyeknek együtt van értelme.
- Magának a kifejezésnek is van **típusa.**
  - az „1+1” egy szám típusú kifejezés
  - az „igaz VAGY hamis” egy logikai típusú kifejezés
- A típus meghatározza, hogy *milyen adatot* tárol illetve, hogy *milyen műveletet* végezhetünk rajta.
- Összetett kifejezéseket is meg lehet fogalmazni, de ilyenkor ügyelni kell arra, hogy helyes részkifejezésekből álljon
  - „8+3*(3+7)”

Típusok

	Típusértékhalmaz	Műveletek
Egész	Egész számok	aritmetika, hasonlítások
Valós	számok	aritmetika, hasonlítások
Karakter	betűk, számjegyek,	konverziók, hasonlítások
Szöveg	karaktersorozatok	összefűzés, részek, hason.
Logikai	igaz, hamis	ÉS, VAGY, NEM ...

- Ezen kívül még sokféle típus létezik, de ezek szinte minden nyelvben megtalálhatóak.

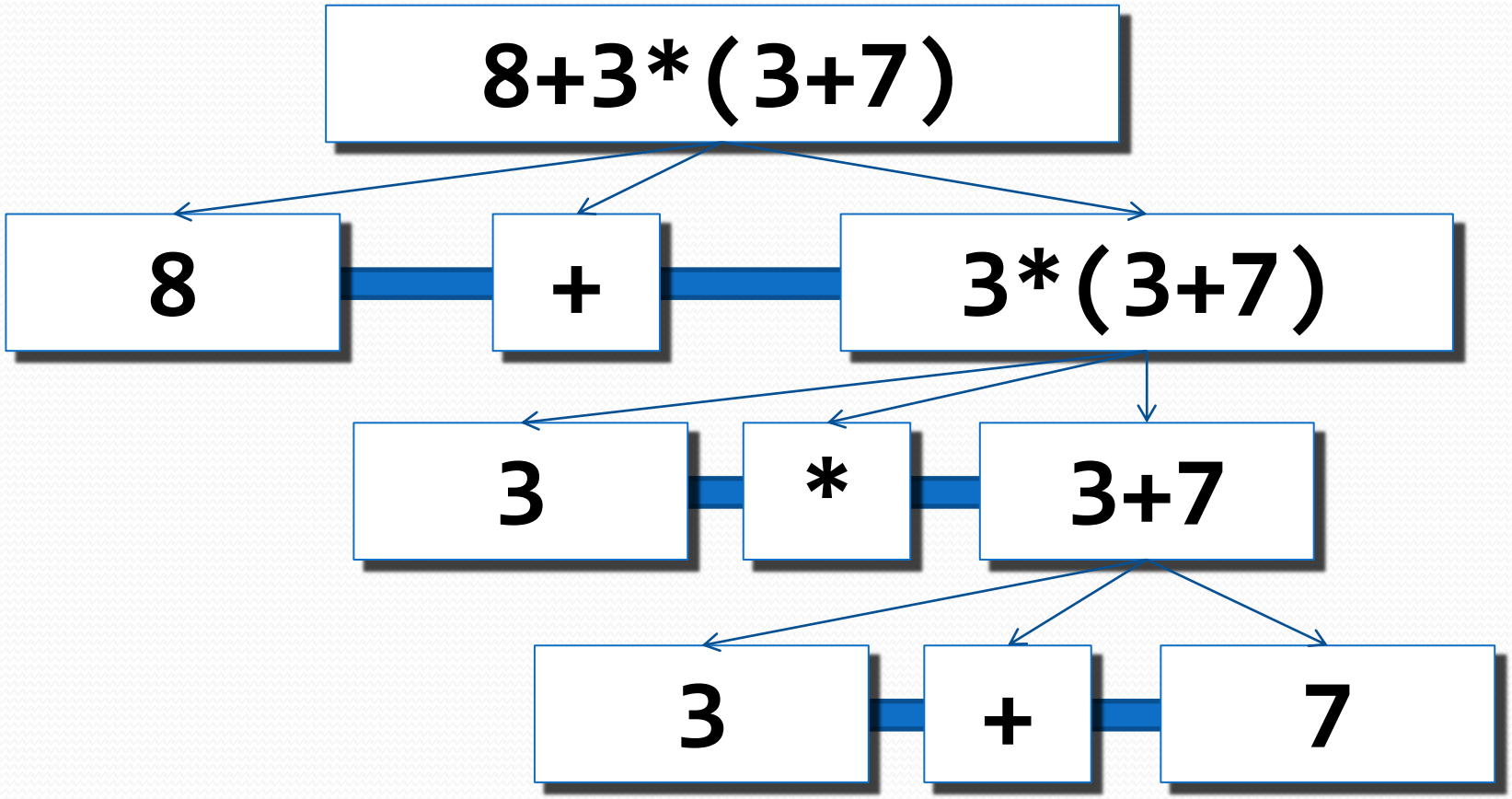
Végrehajtás

- A program (miután lefordult) soronként, lépésről lépésre hajtódik végre (gyakorlaton látni fogjuk). Az eközben keletkező hibákat ***futási hibának*** nevezzük. (Pl. Elfogyott a memória, a program érvénytelen memóriaterületet akar elérni, stb.)
- **A program csak azt hajtja végre, amit megparancsoltunk neki. Se többet, se kevesebbet!**
- **(A program csak arra emlékszik, amit eltároltattunk vele! Se többre, se kevesebbre!)**
- **Specifikáció:** megadjuk, hogy melyek a be- illetve kimeneti feltételek (később)

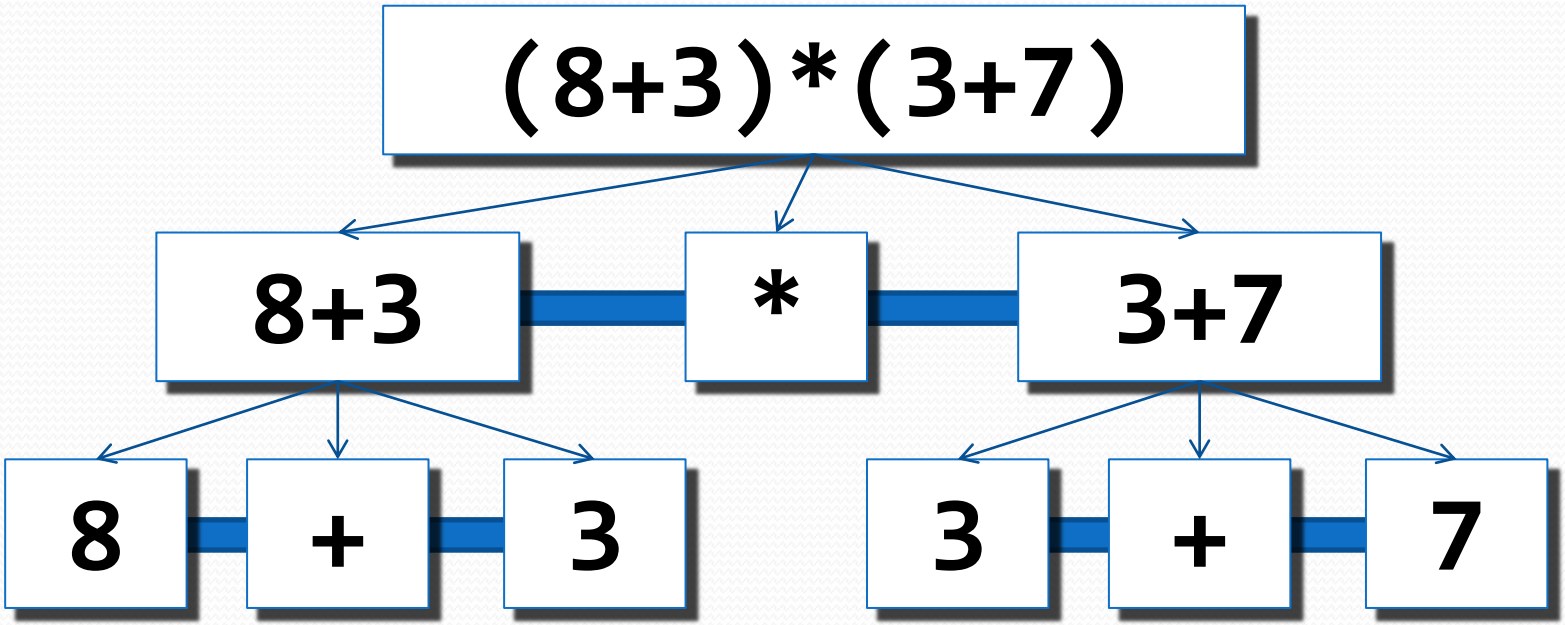
Példa Program

```
PROGRAM kifejezés

    KI: 8+3*(3+7)

PROGRAM_VÉGE
```

Szintaxis fa



Szintaxis fa II



Változók

- *„Amit a matematikában ismeretlennek hívnak, csak itt ismert.”*
- A **változó** a memóriában tárolt adat, amelynek a futás során megváltoztatható az értéke.
- A legtöbb programozási nyelvben a változónak van **típusa**. A típus meghatározza, hogy *milyen adatot* tárolhatunk benne, illetve, hogy *milyen műveleteket* végezhetünk rajta.
- A változók elérésére neveket használunk (pl: a:=5). A változók neveit mi adhatjuk meg, érdemes beszédes neveket használni, amelyek utalnak a változó feladatára.
- Tehát **Névvel jelölt, adott típushoz tartozó elem**.

Példa változóra

- Tekintsük az „**1 + X**” kifejezést, ahol **X** egy változó
- Ennek csak akkor van értelme, ha **X** olyan változó, amely képes számot tárolni és értelmezve van rajta a „számmal összeadás művelet”. Például **X** egy szám típusú változó.

Példa változóra

- Ahhoz, hogy egy programban/kifejezésben változót használhassunk, először jelezni kell az igényünket a program számára.
- Ezt **deklarációnak** nevezzük:

```
PROGRAM változó
    VÁLTOZÓK:
        x: EGÉSZ          …
```

- Innentől a programnyelvet értelmező rendszer ismeri az **X** nevet és megfelelő módon fogja kezelni.
- **Ha ezt elmulasztjuk, akkor a program nem fogja tudni értelmezni az „X”-et a program kódban!**

Változók, értékadás

```
        VÁLTOZÓK:
              a: EGÉSZ
…     a := 1 …
```

- Az értékadás a változó tartalmát megváltoztatja egy új értékre/ egy kifejezés eredményére
  - **a := 8+3*(3+7)**
  - **a := a + 1**
  - **a := b + c + d**
    - A fentebbi kifejezés csak akkor értelmes, ha b, c és d már deklarált,megfelelő (EGÉSZ) típusú változók
- Fontos! Egy változónak MINDIG értéket kell adni mielőtt bármi másra használnánk a programban!

Kimenet, bemenet

```
VÁLTOZÓK:
    a: EGÉSZ
BE: a
KI: a+1 …
```

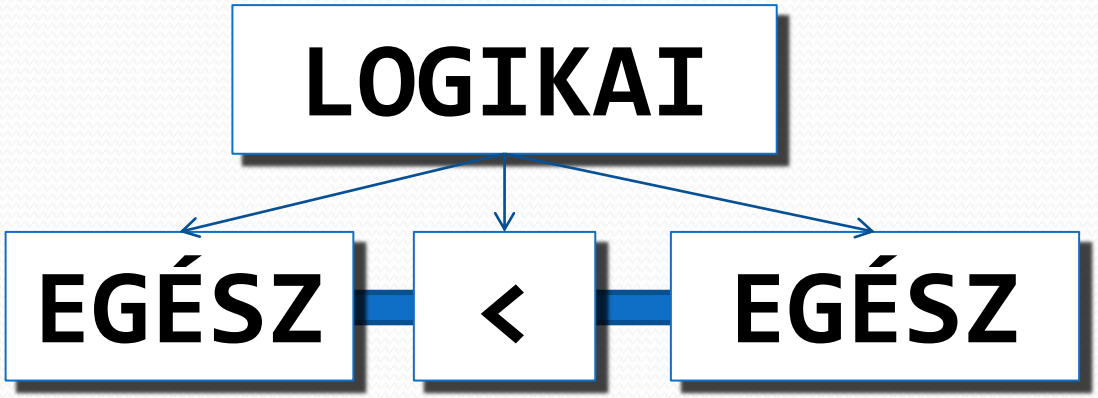
- A változók a külvilággal való kapcsolattartást is segítik.
- A bemenet (input) jelentése (egyelőre) az, hogy a program felhasználója futás közben tud értéket adni egy változónak.
- A kimenet (output) jelentése (egyelőre) az, hogy egy változó/kifejezés eredményét megörökítjük a külvilágnak például a képernyőre írással.

Komplett példa

```
PROGRAM input-output
      VÁLTOZÓK:
            a: EGÉSZ

      BE: a
      KI: a + 1

PROGRAM_VÉGE
```

Vegyes típusú kifejezés

- X < 5
- A „<” egy olyan művelet, amikét számot fogad és logikai eredményt ad: *igaz* vagy *hamis*
- … $\subseteq$ **R x R x L** , reláció
- Szintaxisfában:



Vegyes típusú példa

```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ

      BE: a
      KI: 2 < a

PROGRAM_VÉGE
```

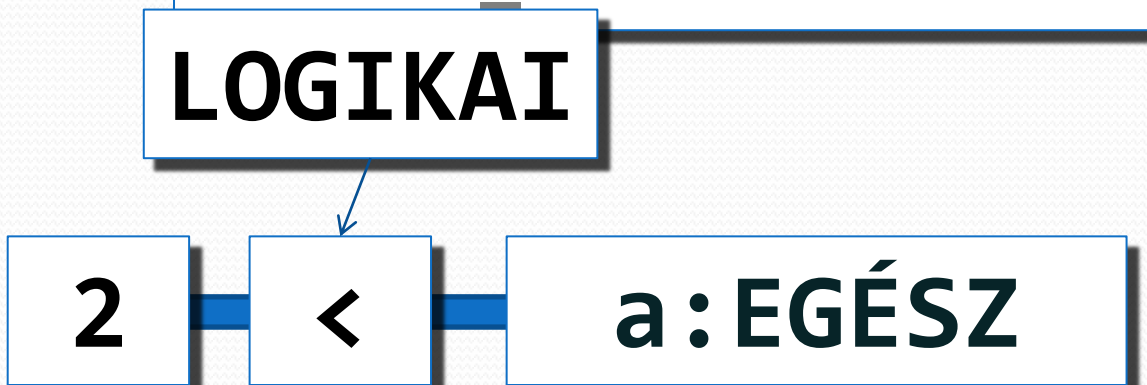
Példa HIBÁS programra

```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ

      BE: a
      KI: 2 < a < 5

PROGRAM_VÉGE
```

Példa HIBÁS programra

```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ

      BE: a
      KI: LOGIKAI < 5

PROGRAM_VÉGE
```

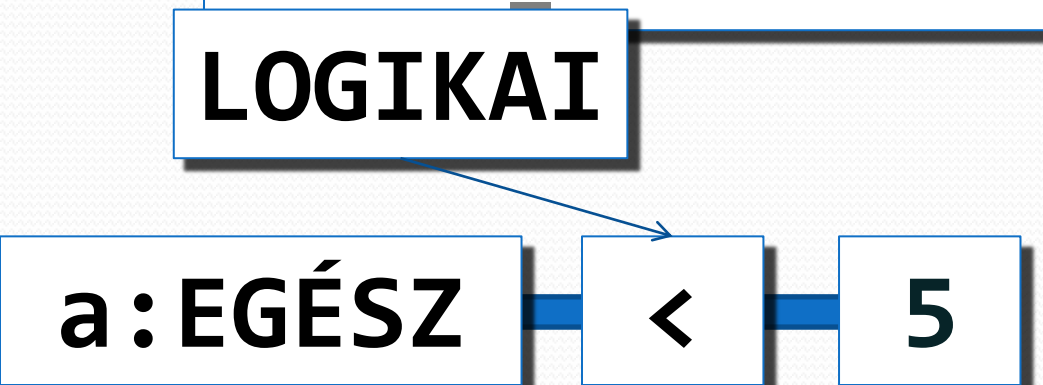
Nincs ilyen művelet!



Példa HIBÁS programra

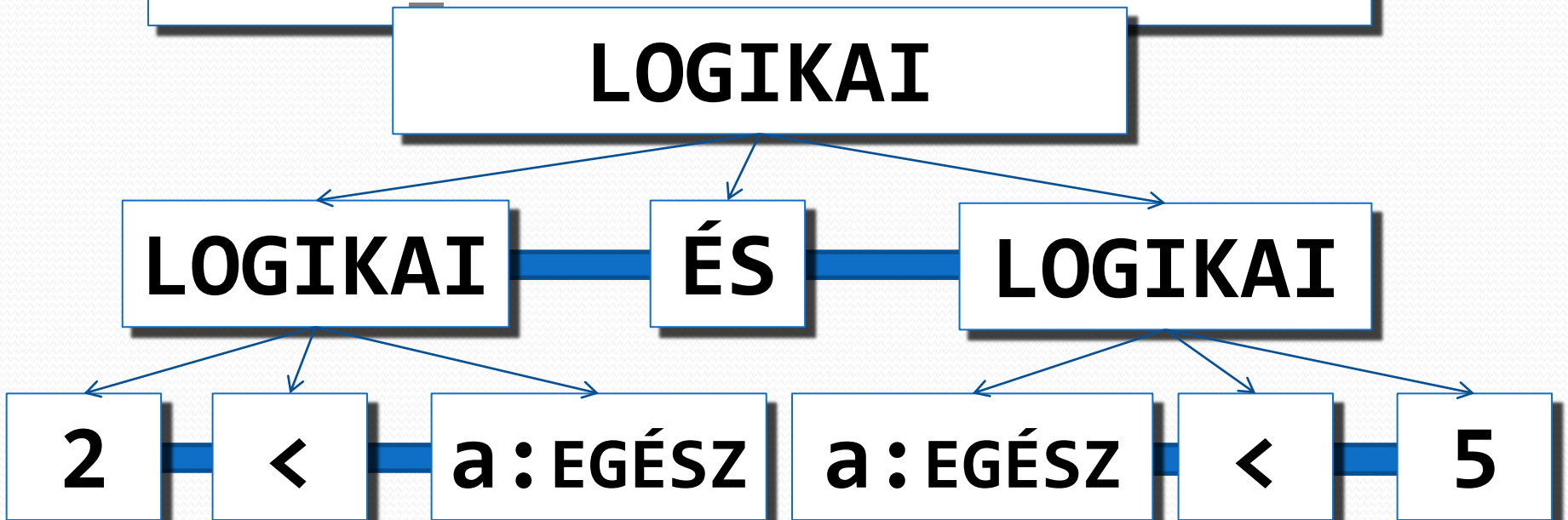
```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ

      BE: a
      KI: 2 < LOGIKAI

PROGRAM_VÉGE
```

Ilyen művelet sincs!



Előző példa helyesen

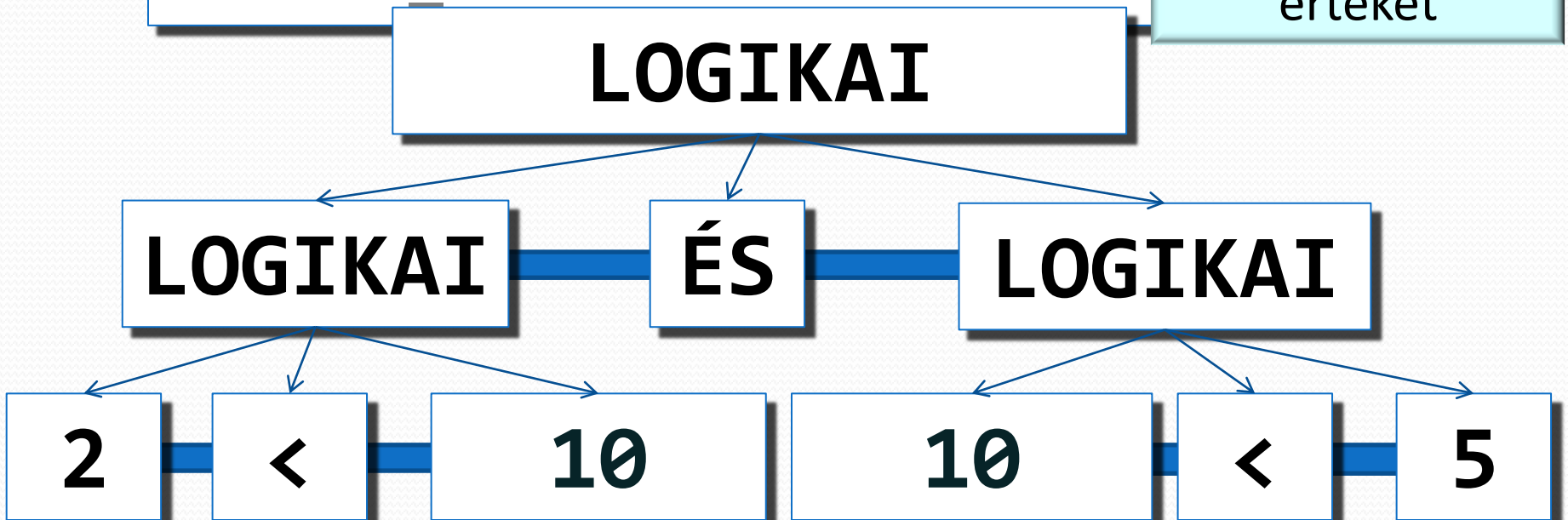
```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ
      BE: a
      KI: 2 < a ÉS a < 5
PROGRAM_VÉGE
```



Előző példa helyesen

```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ
      BE: a
      KI: 2 < a ÉS a < 5
PROGRAM_VÉGE
```

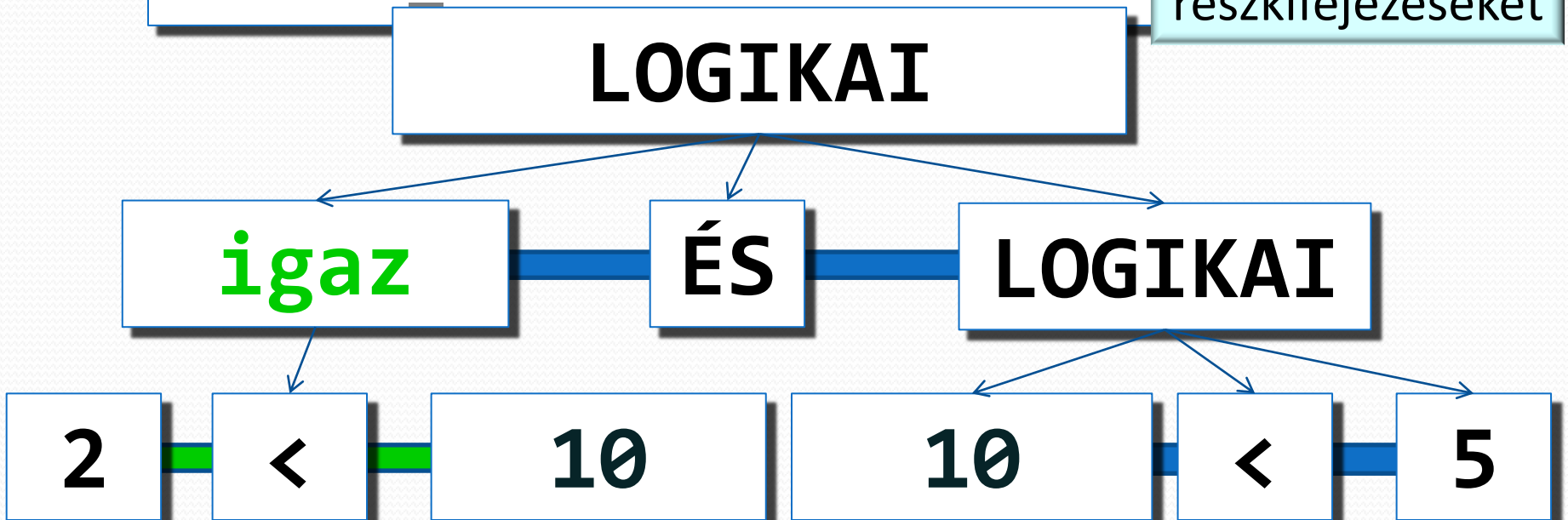
Adjunk a bemenetre 10-es értéket



Előző példa helyesen

```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ
      BE: a
      KI: 2 < a ÉS a < 5
PROGRAM_VÉGE
```

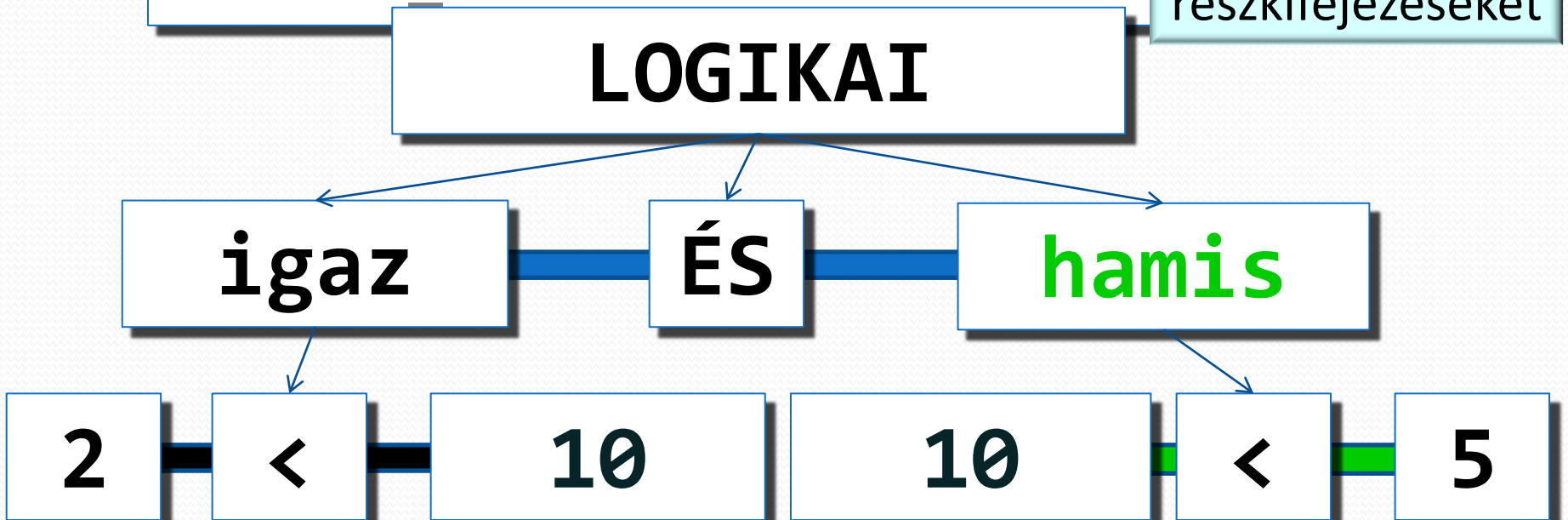
Balról jobbra kiértékeljük a részkifejezéseket



Előző példa helyesen

```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ
      BE: a
      KI: 2 < a ÉS a < 5
PROGRAM_VÉGE
```

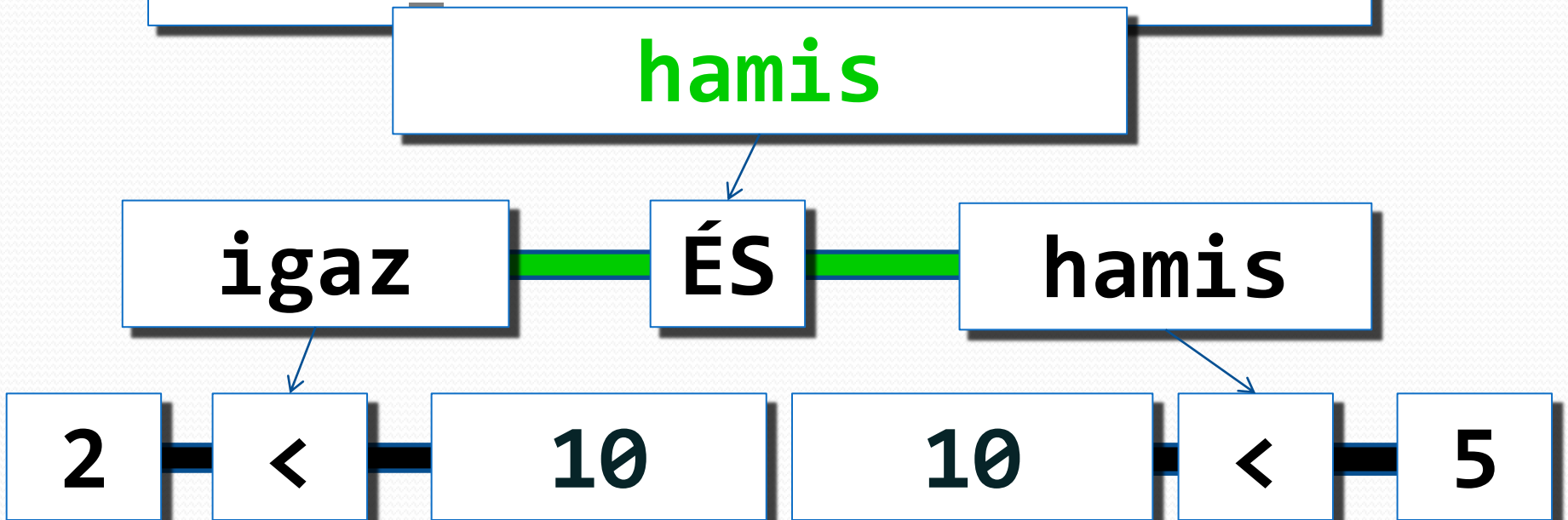
Balról jobbra kiértékeljük a részkifejezéseket



Előző példa helyesen

```
PROGRAM vegyestipus
      VÁLTOZÓK:
            a: EGÉSZ
      BE: a
      KI: 2 < a ÉS a < 5
PROGRAM_VÉGE
```

A végeredmény: hamis az állítás



Kifejezések összefoglalás

- Egy kifejezés akkor értelmes, ha szintaxisfába szervezhető
- Ismerni kell az adott programnyelv típusait, műveleteit (ezek komolyabb nyelveknél bővíthetőek lesznek), precedencia szabályait
- A kifejezés kiértékelése a szintaxisfa aljáról felfelé küldött részeredményeken keresztül történik.