

# Bevezetés a programozásba

## 1. Előadás Bevezetés

# A programozás természete

- Hozzá! krumplit!
- Hozzá! egy kiló krumplit!
- Hozzá! 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!



Egyre precízebb

# 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

- Előadásra, gyakorlatra járás kötelező (max 3 hiányzás)
- Pluszmínusz: félév végére legalább 0 pont kell
- Házifeladatokat vizsgaidőszakig meg kell oldani
- Papíros ZH: 30 pont, minimum: 5 pont
  - Program vázlat írás, hiba keresés, elmélet
- Géptermi vizsga: 60 pont, beugró feladattal
  - 3 óra alatt működő C++ kódok készítése
- Pót pluszmínusz, PótZH-k

# Néhány fogalom

- **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. Python, C++, Java, PLanG, stb.*)
- **Forráskód:** az adott programozási nyelven megírt program. Általában ember által írt.
- **Fordítás:** az a folyamat, amikor a forrásprogramból gépi kódot állítunk elő, a ***fordító*** feladata.



# Kifejezések

$$1+1$$

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

# Kifejezések



???

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

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

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

# Értelmezés – Fordítás – Végrehajtás

- A **fordítóprogram** ellenőrzi a forráskód szintaktikáját, legyártja a gépi kódot, közben **fordítási hibákat** kaphatunk.
- A program soronként, lépésről lépésre hajtódik végre. 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!)

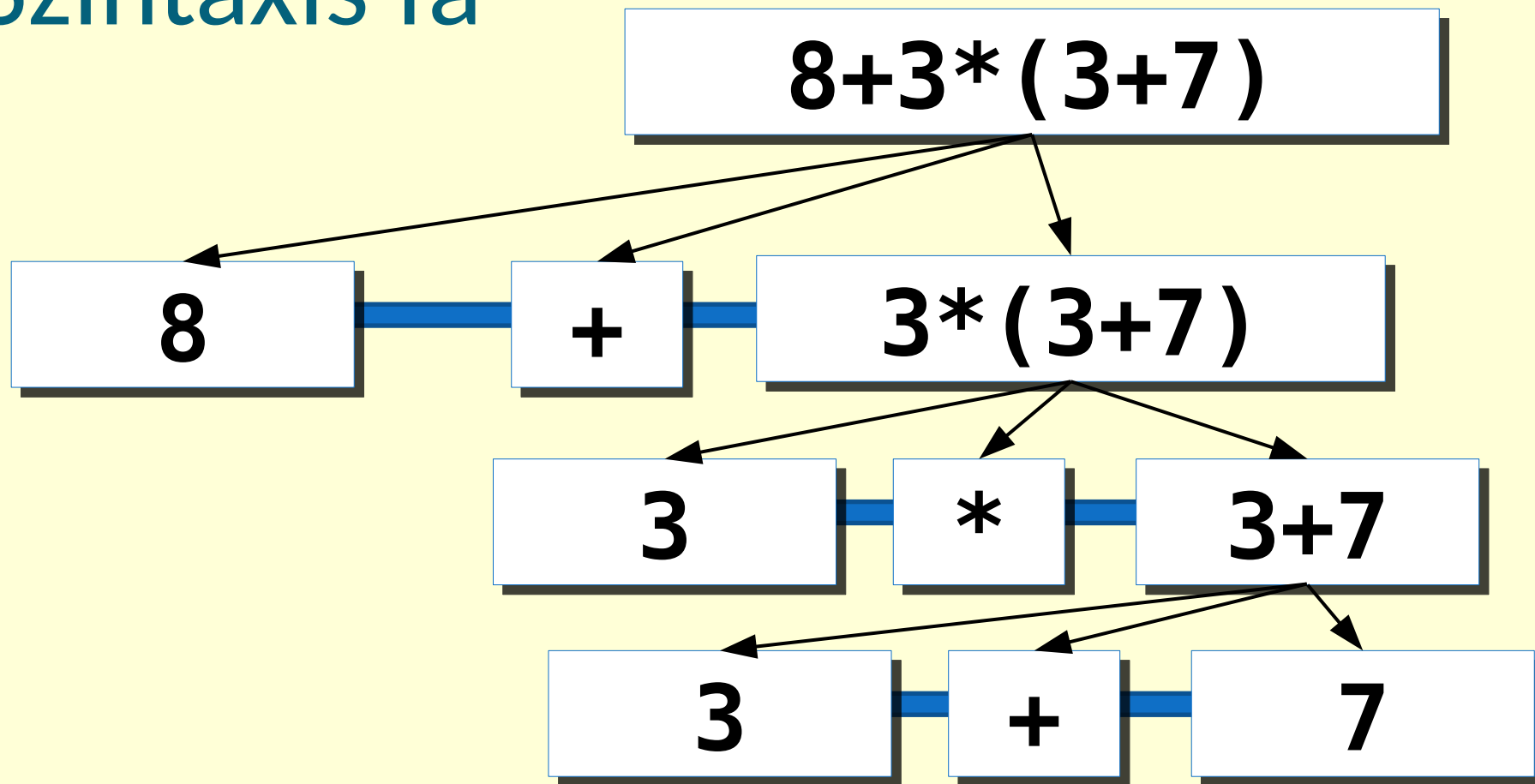
# Példa Program

**PROGRAM kifejezés**

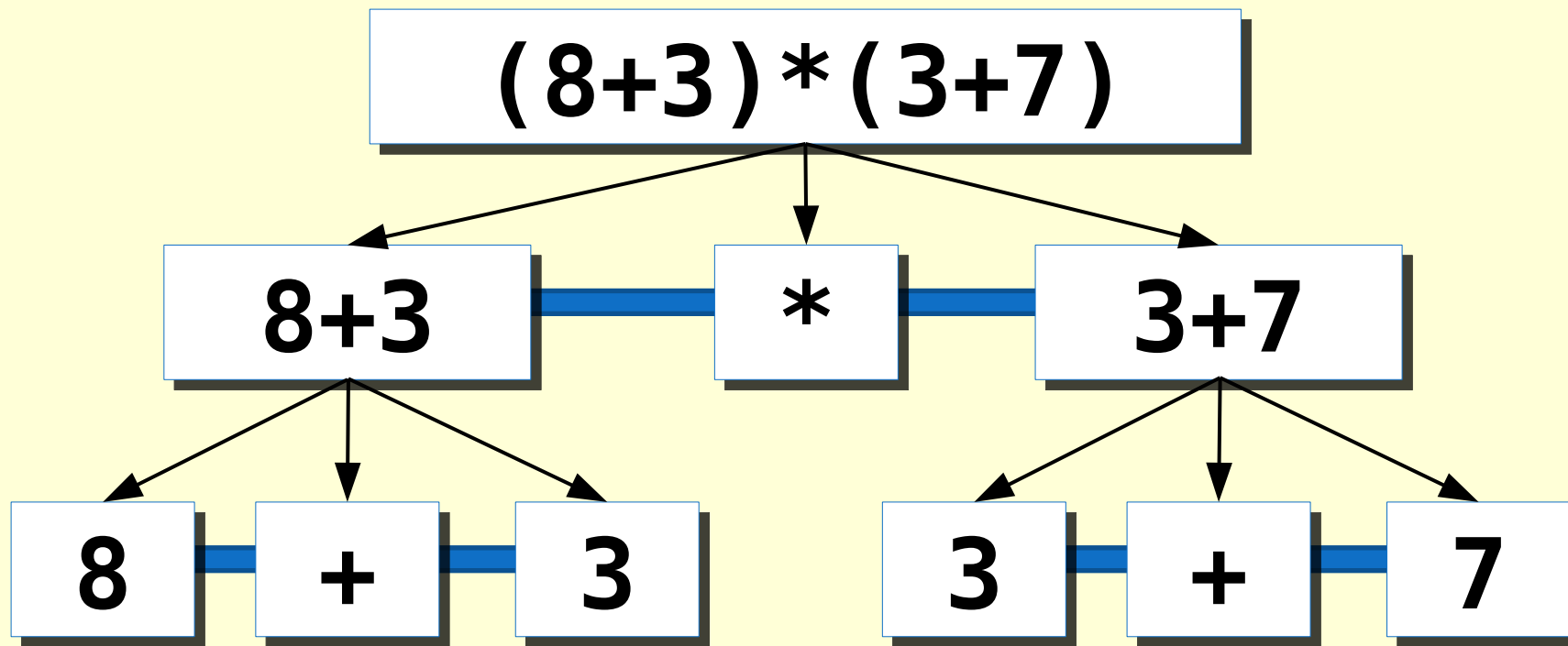
**KI:  $8+3*(3+7)$**

**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.

# Változó létrehozása

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

# 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.
- Az „ $1+X$ ” többféleképpen lehet értelmetlen
  - $X$  nem volt deklarálva, tehát ismeretlen azonosító
  - $X$  deklarálva volt, de olyan típussal amelyikben nincs  $+$  művelet számmal

# 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
- Érdemes a változó pályafutását azzal kezdeni hogy értéket adunk neki, értékadással vagy más eszközzel

# Komplett példa

**PROGRAM** input-output

**VÁLTOZÓK:**

**a: EGÉSZ**

**BE: a**

**KI: a + 1**

**PROGRAM** VÉGE

# Kimenet, bemenet

**VÁLTOZÓK:**

**a: EGÉSZ**

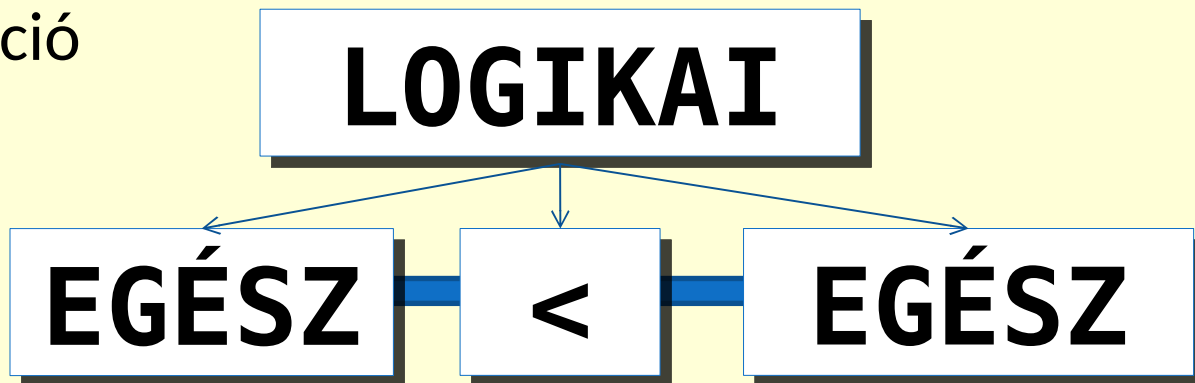
**BE: a**

**KI: a+1 ...**

- A változók a külvilággal való kapcsolattartást is segítik.
- A BE: hasonlít az értékadásra, a változó új értéket kap, de ebben az esetben az inputról.
- A kimenet célja az, hogy egy változót vagy kifejezés eredményét megosztjuk a külvilággal, például a képernyőre írással.

# Vegyes típusú kifejezés

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



# Vegyes típusú példa

```
PROGRAM vegyestípus
```

```
  VÁLTOZÓK:
```

```
    a: EGÉSZ
```

```
  BE: a
```

```
  KI: 2 < a
```

```
PROGRAM_VÉGE
```



# Példa HIBÁS programra

PROGRAM vegyestípus

VÁLTOZÓK:

a: EGÉSZ

BE: a

*KI:  $2 < a < 5$*

PROGRAM\_VÉGE

# Példa HIBÁS programra

PROGRAM vegyestípus

VÁLTOZÓK:

a: EGÉSZ

BE: a

*KI: LOGIKAI < 5*

**Nincs  
ilyen  
művelet!**

**LOGIKAI**

2

<

a: EGÉSZ

# Példa HIBÁS programra

PROGRAM vegyestípus

VÁLTOZÓK:

a: EGÉSZ

BE: a

*KI: 2 < LOGIKAI*

Ilyen  
művelet  
sincs!

**LOGIKAI**

**a: EGÉSZ**

**<**

**5**

# Előző példa helyesen

```
PROGRAM vegyestípus
```

```
  VÁLTOZÓK:
```

```
    a: EGÉSZ
```

```
  BE: a
```

```
  KI: 2 < a ÉS a < 5
```

```
PROGRAM_VÉGE
```

# Előző példa helyesen

PROGRAM vegyestípus

VÁLTOZÓK:

a: EGÉSZ

BE: a

KI:  $2 < a$  ÉS  $a < 5$

PROGRAM\_VÉGE

LOGIKAI

LOGIKAI

ÉS

LOGIKAI

2

<

a: EGÉSZ

a: EGÉSZ

<

5

# Előző példa helyesen

PROGRAM vegyestípus

VÁLTOZÓK:

a: EGÉSZ

BE: a

KI: 2 < a ÉS a < 5

PROGRAM\_VÉGE

LOGIKAI

ÉS

LOGIKAI

2

<

a: EGÉSZ

a: EGÉSZ

<

5

# 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 részeredményekeken keresztül történik.