FORDÍTÓPROGRAMOK CÉLJA ÉS FELÉPÍTÉSE

FORMÁLIS NYELVEK ÉS FORDÍTÓPROGRAMOK ALAPJAI

Dévai Gergely
ELTE

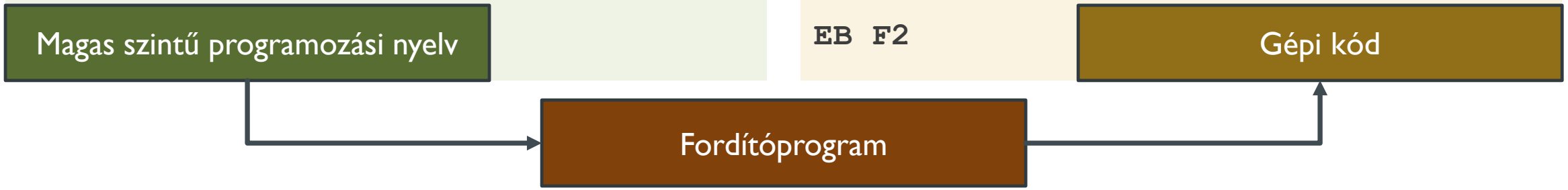
MIÉRT VAN SZÜKSÉG FORDÍTÓPROGRAMOKRA?

Így kényelmes programozni

```
int sum = 0;
for(int i=0; i<len; ++i)
    sum += t[i];
```

Ezt tudja végrehajtani a számítógép

```
B9 00 00 00 00
B8 00 00 00 00
81 F9 0A 00 00 00
7D 06
03 04 8B
41
EB F2
```



MAGAS SZINTŰ PROGRAMOZÁSI NYELV VS. GÉPI KÓD

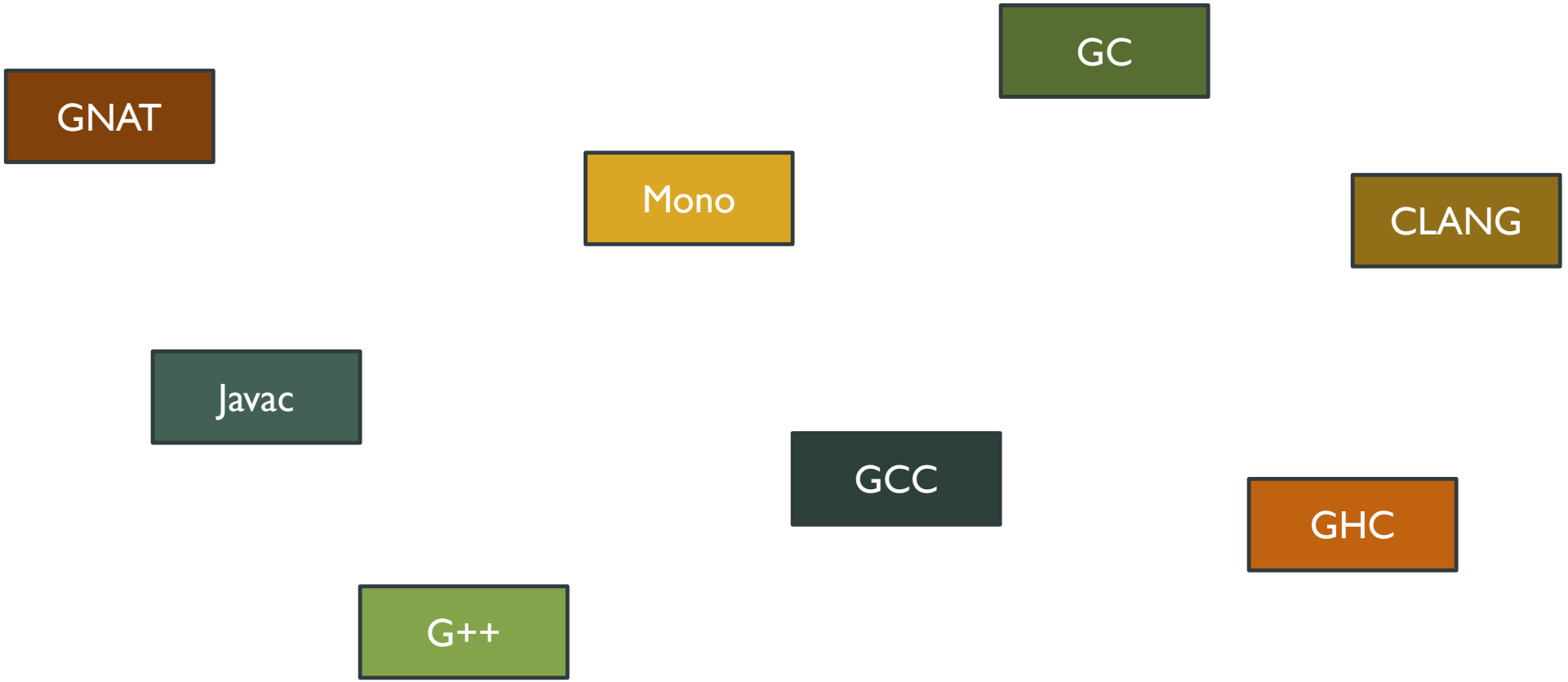
Magas szintű programozási nyelv

- könnyebb programozni
- közelebb a megoldandó problémához
- platform-független

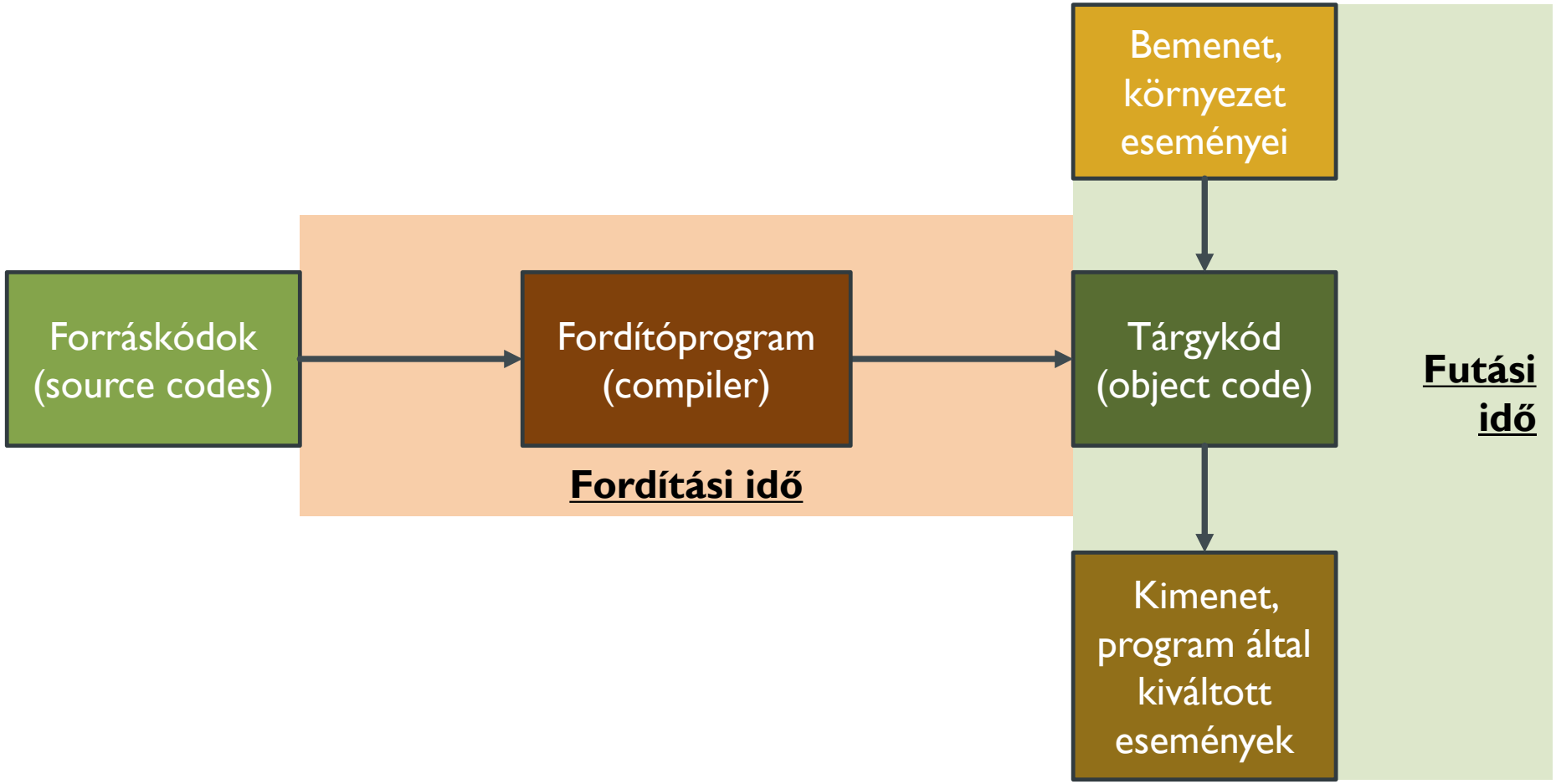
Gépi kód

- gépközeli (numerikus utasításkódok, regiszterek, memóriahivatkozások, ...)
- erősen platformfüggő
- optimalizált

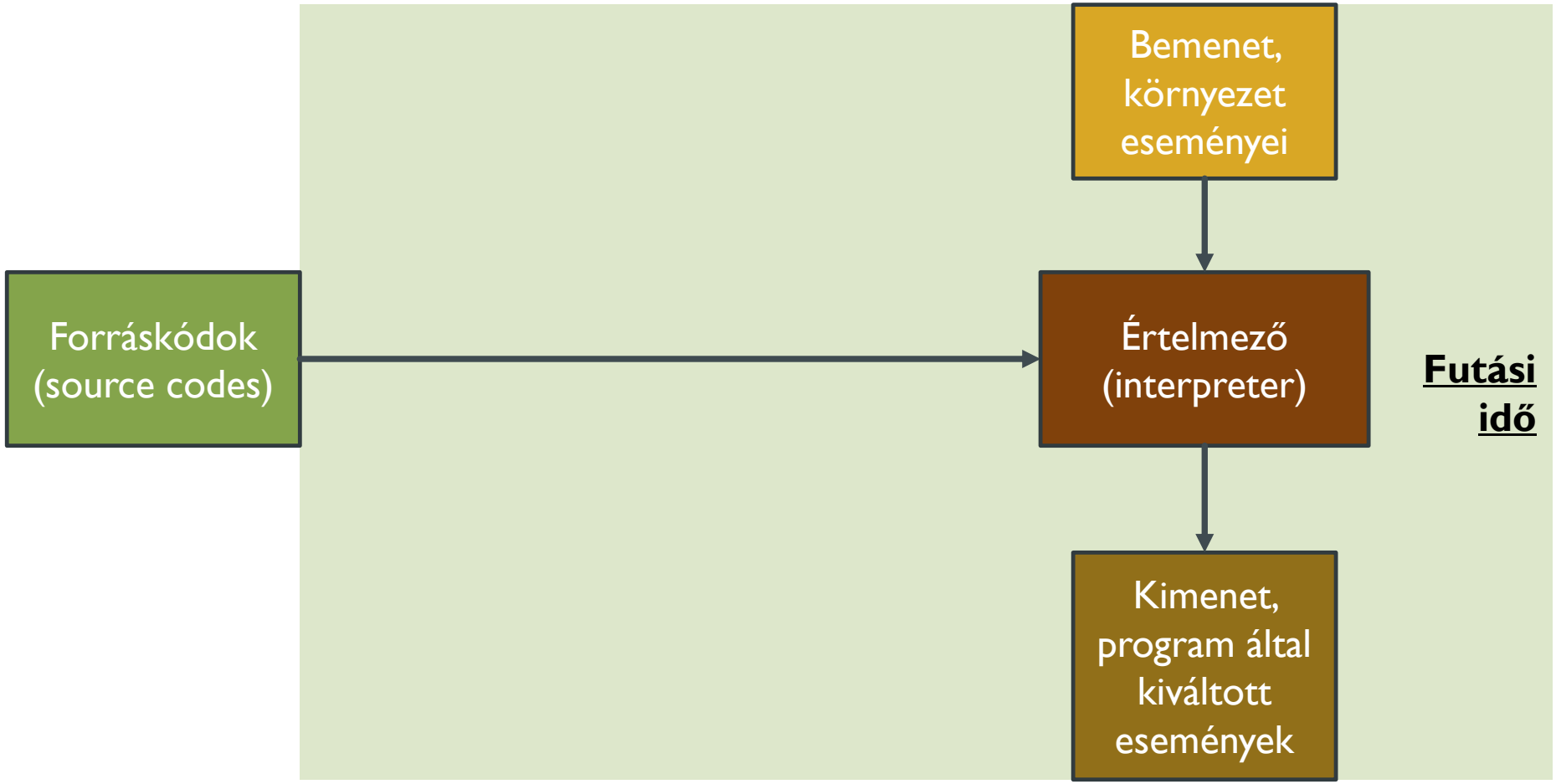
PÉLDÁK



FORDÍTÁS ÉS VÉGREHAJTÁS



ÉRTELMEZÉS


Bemenet, környezet eseményei
Forráskódok (source codes)
Értelmező (interpreter)
Futási idő
Kimenet, program által kiváltott események

FORDÍTÁS VS. ÉRTELMEZÉS

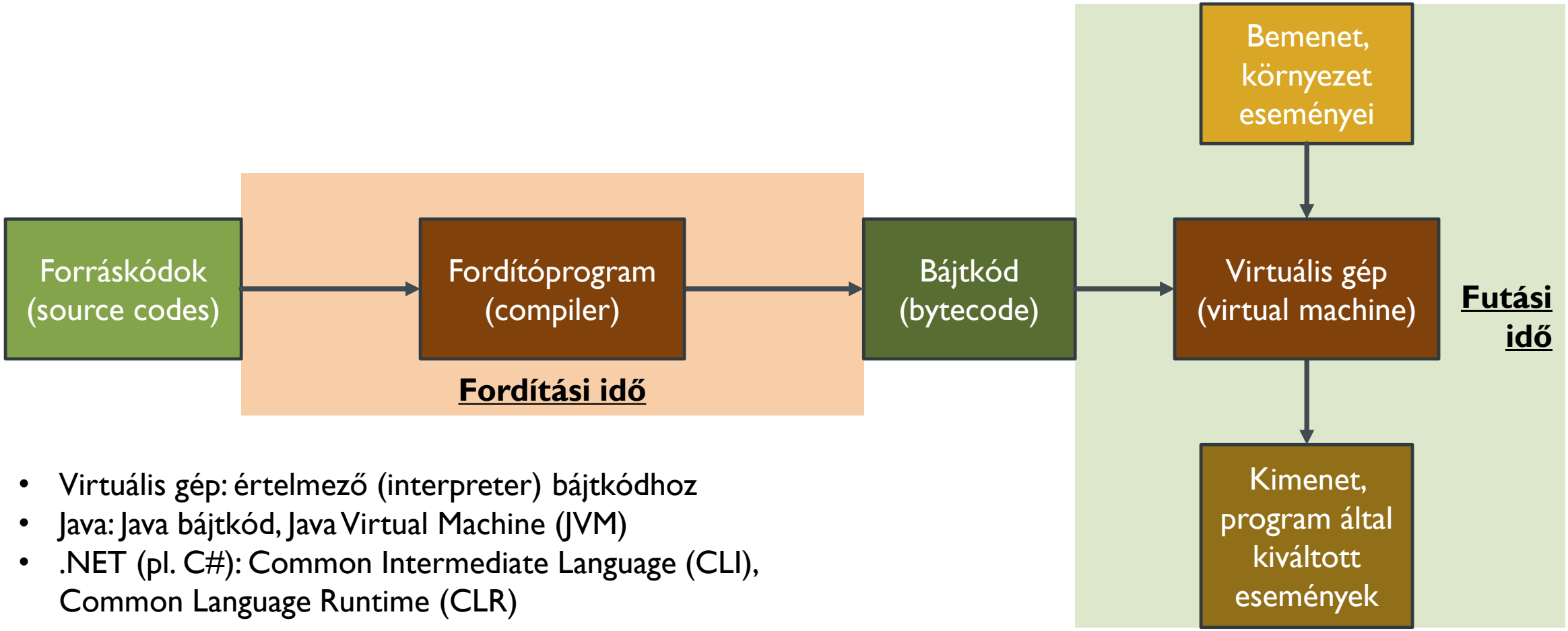
Fordítás

- Gyorsabb végrehajtás
- A forrás alaposabb ellenőrzése
- A tárgykód optimalizálása
- Minden platformra külön-külön le kell fordítani
- Elkülönül a fordítási és futási idő
- *C, C++, Haskell, Ada, ...*

Értelmezés

- Rugalmasabb
  (pl. utasítások fordítási időben történő összeállítása)
- Jellemzően jelentősen lassabb a végrehajtás
- Minden platformon azonnal futtatható, ahol az interpreter rendelkezésre áll
- Csak futási idő van
- *Python, Perl, Php, Javascript, ...*

FORDÍTÁS ÉS ÉRTELMEZÉS EGYMÁSRA ÉPÍTVE



- Virtuális gép: értelmező (interpreter) bájtkódhoz
- Java: Java bájtkód, Java Virtual Machine (JVM)
- .NET (pl. C#): Common Intermediate Language (CLI), Common Language Runtime (CLR)

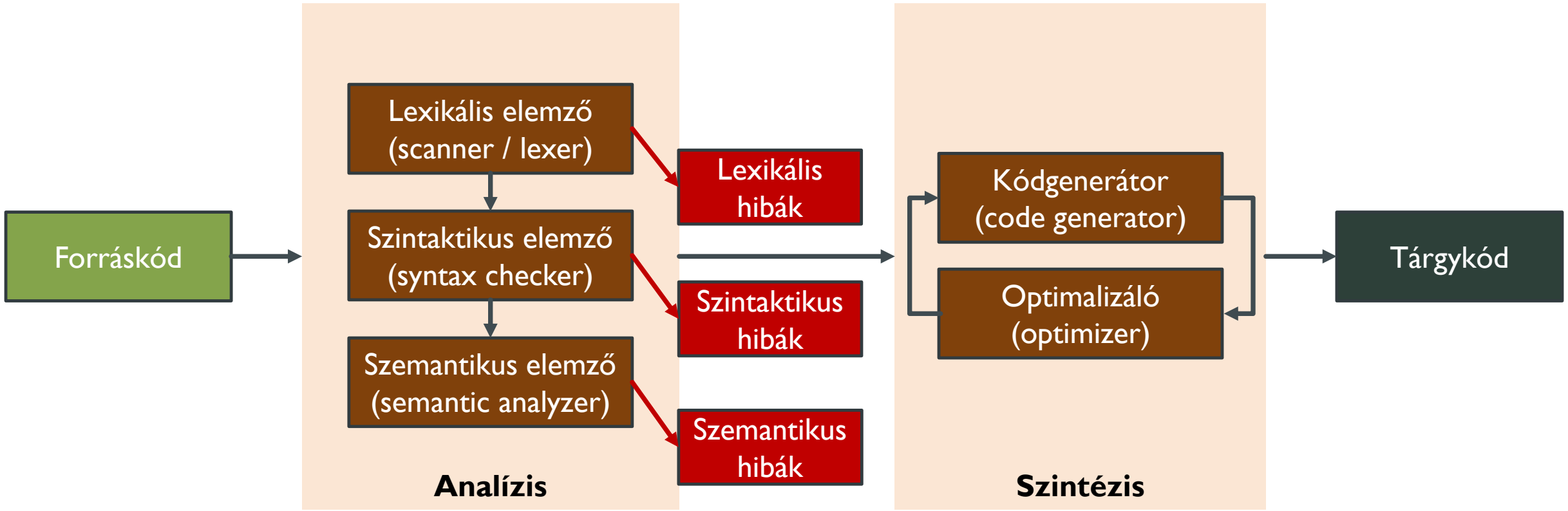
FORDÍTÁS VÉGREHAJTÁS KÖZBEN: JUST IN TIME COMPILATION (JIT)

- Probléma:
  Az értelmezés lassabb a gépi kód végrehajtásánál, és ez igaz a bájtkód értelmezésére (a virtuális gépekre) is.
- Ötlet: A bájtkód fordítása gépi kódra futási időben.
- Teljes fordítás a végrehajtás kezdetén: túl nagy kezdeti lassulás.
- Megoldás:
  - Kezdetben értelmezés (interpretálás)
  - Statisztikák gyűjtése a leggyakrabban lefutó kódrészletekről („hot spots”)
  - Ezek fordítása gépi kódra
  - A következő alkalommal a lefordított kódrészlet fut az értelmezés helyett
- Bónusz: A JIT fordító futási időben gyűjtött információkat is figyelembe vehet a kódoptimalizálásnál. Ilyenekhez a klasszikus fordítóprogram nem fér hozzá!
- A bájtkódok végrehajtása jellemzően még így is lassabb a gépi kódhoz képest, de ez sepciális alkalmazási területeket leszámítva nem baj.

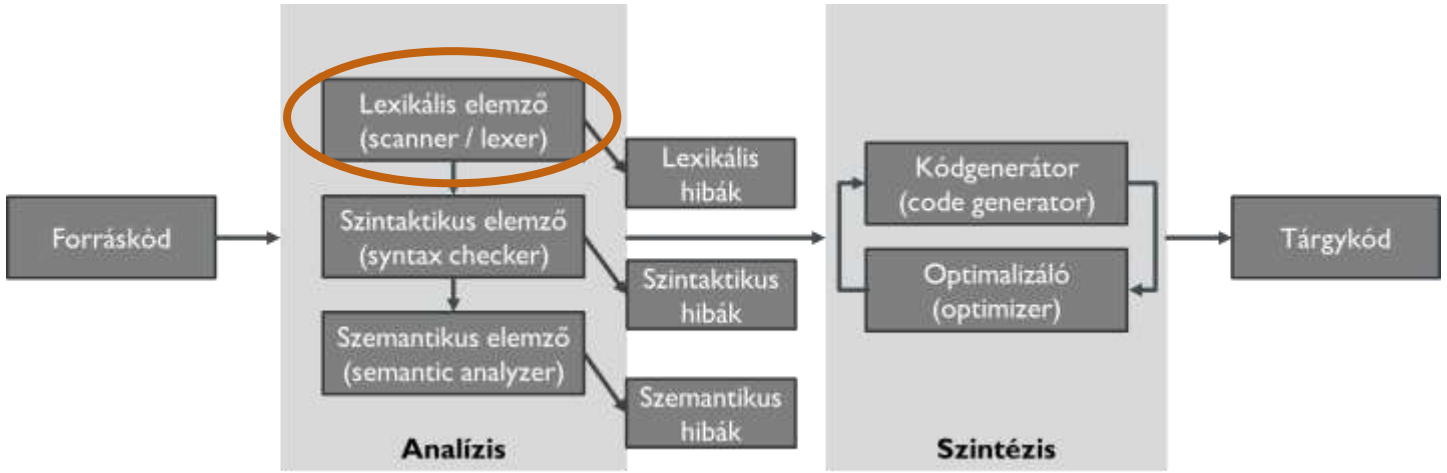
FORDÍTÓPROGRAMOK FEJLŐDÉSE

- 1957: Első Fortran compiler – 18 emberévnyi munka
- Azóta fejlődött a formális nyelvek és automaták elmélete.
- Ma: A fordítóprogramok létrehozásának egy része automatizálható elemzőgenerátorokkal.
  - A programszöveg elemi egységekre (tokenekre) bontása
  - A programszöveg formai helyességének vizsgálata
- A további ellenőrzések és a kódgenerálás nem automatizálható, de az implemetációt keretrendszerek segíthetik.
- A kódoptimalizálás (és a hozzá szükséges elemzések) komoly kihívás.

A FORDÍTÓPROGRAMOK LOGIKAI FELÉPÍTÉSE



LEXIKÁLIS ELEMZŐ

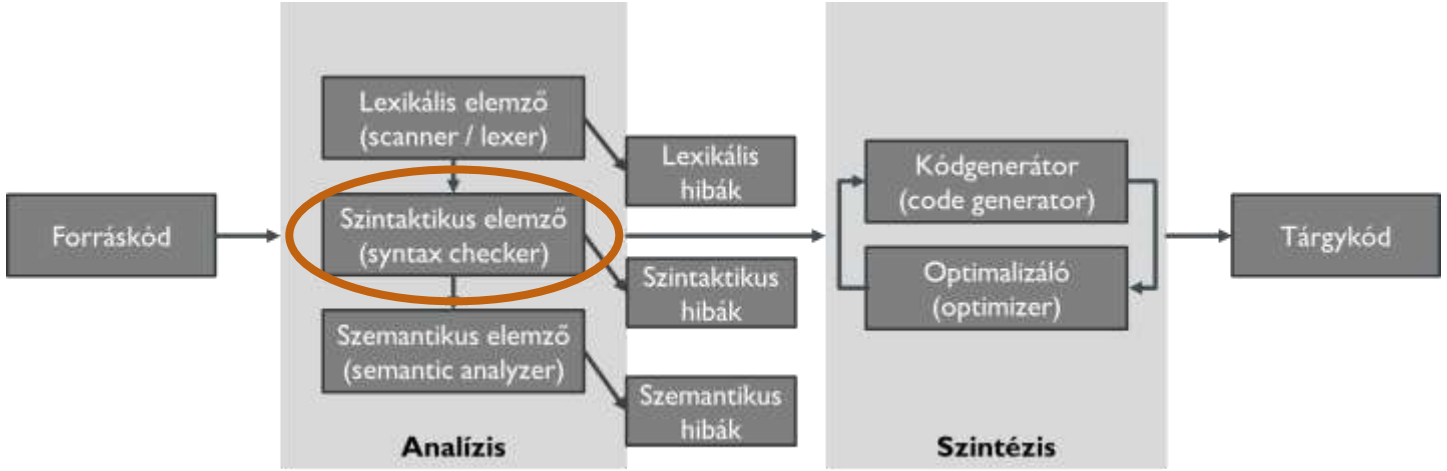


- *Feladat:*
  A forrásszöveg elemi egységekre tagolása
- *Bemenet:*
  Karaktersorozat
- *Kimenet:*
  Lexikális elemek (tokenek) sorozata + lexikális hibák
- *Eszközök:*
  Reguláris kifejezések, véges determinisztikus automaták

```
x = x + 1;
```

Változó ('x'), Operátor ('='), Változó ('x'), Operátor ('+'), Literál (1), Utasításvég

SZINTAKTIKUS ELEMZŐ



- *Feladat:*
  A forrásszöveg szerkezetének felderítése, formai ellenőrzése
- *Bemenet:*
  Lexikális elemek (tokenek) sorozata
- *Kimenet:*
  Szintaxisfa + szintaktikus hibák
- *Eszközök:*
  Környezetfüggetlen nyelvtanok, veremautomaták

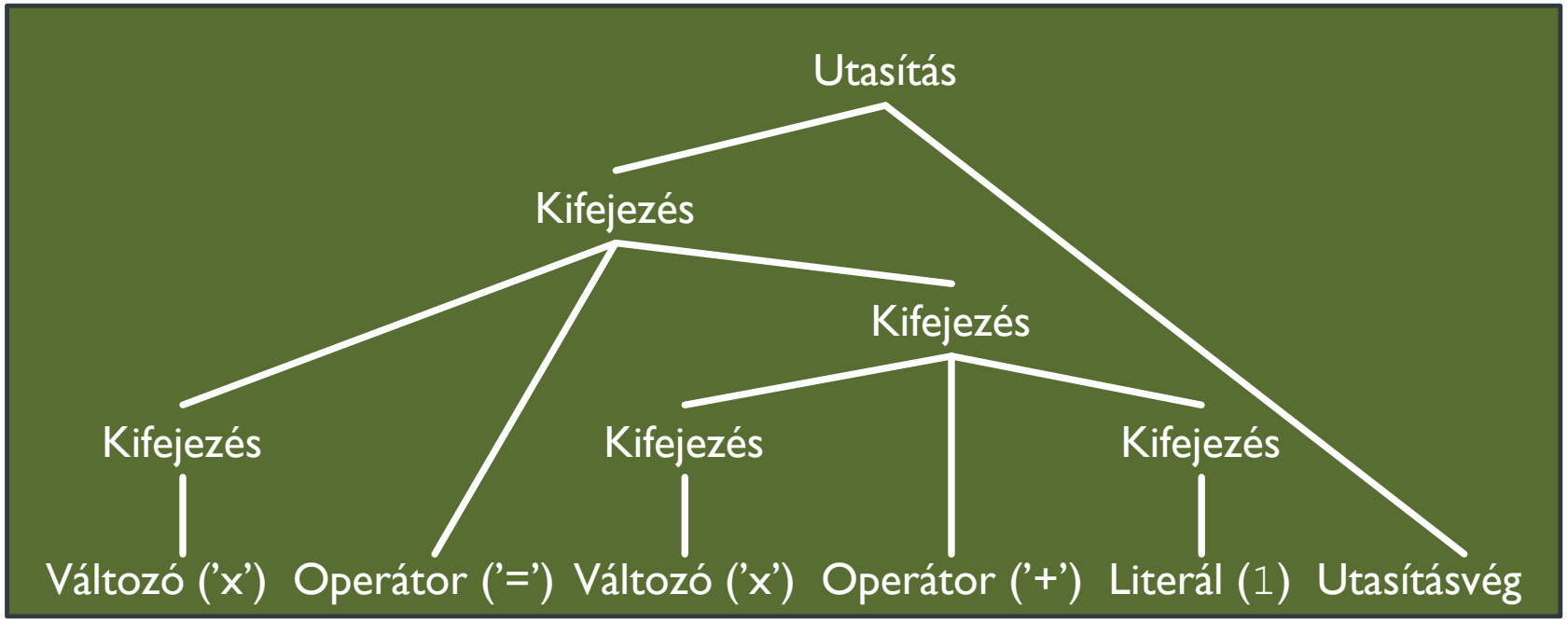
SZINTAKTIKUS ELEMZŐ - PÉLDA

Utasítás →
Kifejezés Utasításvég

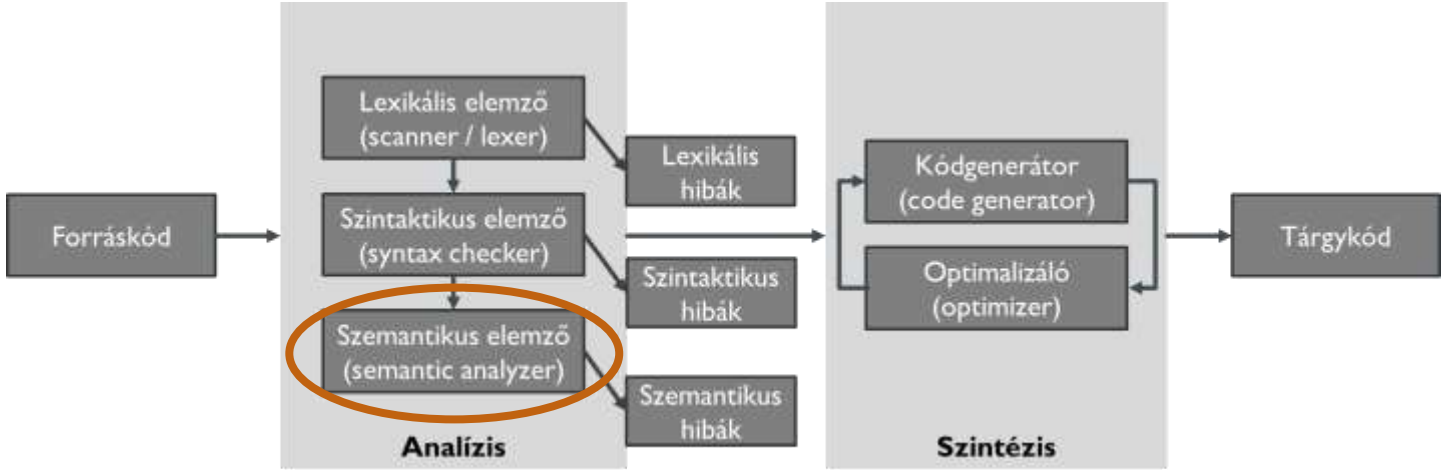
Kifejezés →
Változó | Literál |
Kifejezés Operátor Kifejezés

! A lexikális elemek (tokenek) a szintaktikus elemzés nyelvtanának terminális szimbólumai.

Változó (’x’), Operátor (’=’), Változó (’x’), Operátor (’+’), Literál (1), Utasításvég



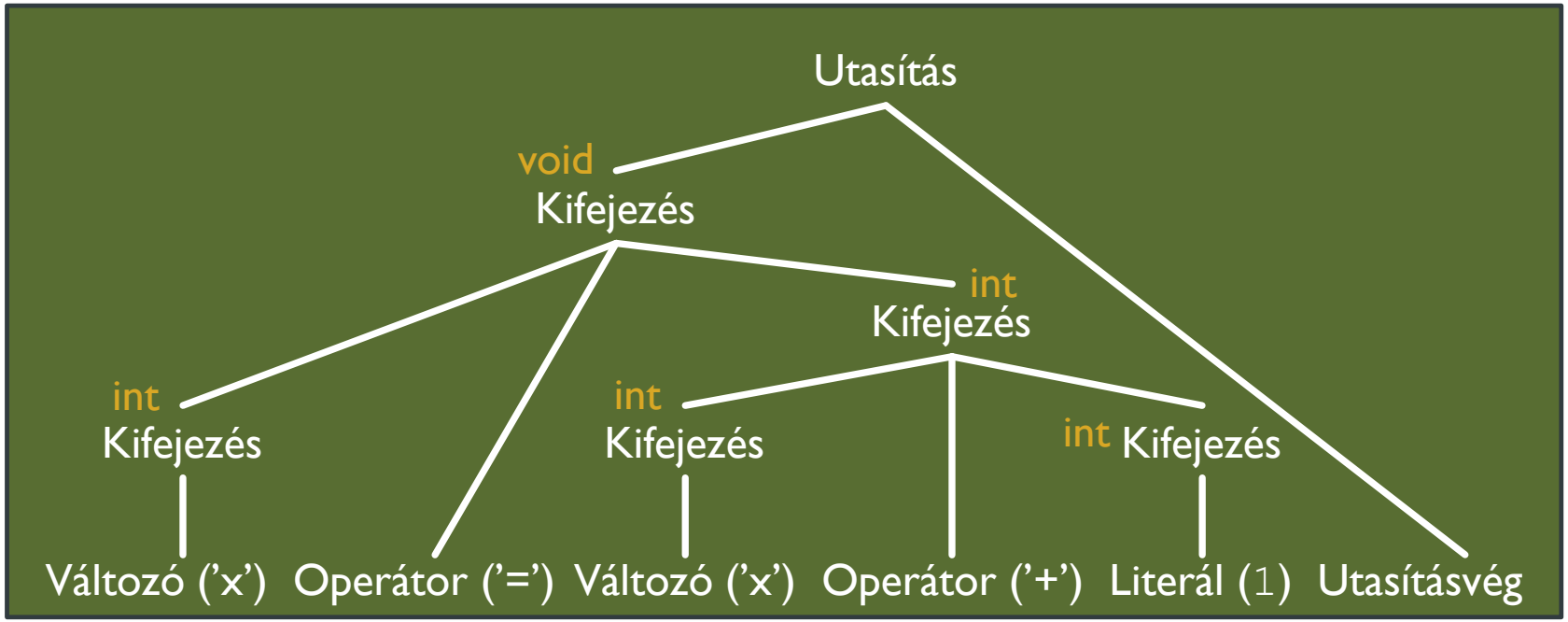
SZEMANTIKUS ELEMZŐ



- *Feladat:*
  A statikus szemantika (pl. változók deklaráltsága, típushelyesség stb.) ellenőrzése
- *Bemenet:*
  Szintaxisfa
- *Kimenet:*
  Szintaxisfa attribútumokkal, szimbólumtábla + szemantikus hibák
- *Eszközök:*
  Attribútumnyelvtanok

SZEMANTIKUS ELEMZŐ - PÉLDA

Név	Típus
”x”	int



HIBATÍPUSOK

```
std::cout << "hello" << std::endl;
```

Lexikális hiba

```
std::cout << "hello << std::endl;
```

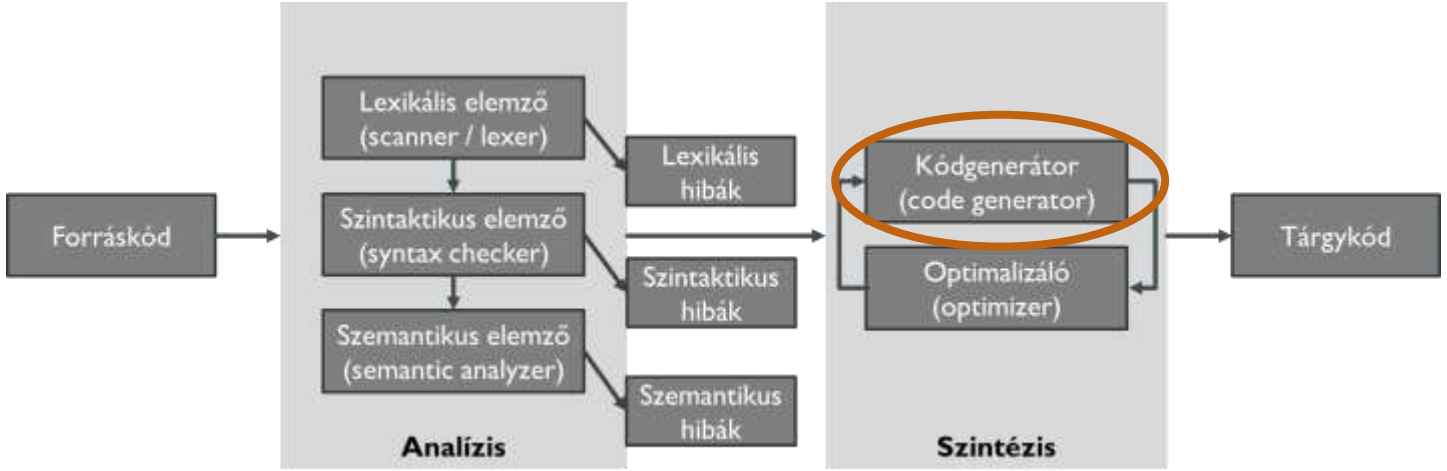
Szintaktikus hiba

```
std::cout   "hello" << std::endl;
```

Szemantikus hiba

```
std::cout >> "hello" << std::endl;
```

KÓDGENERÁTOR



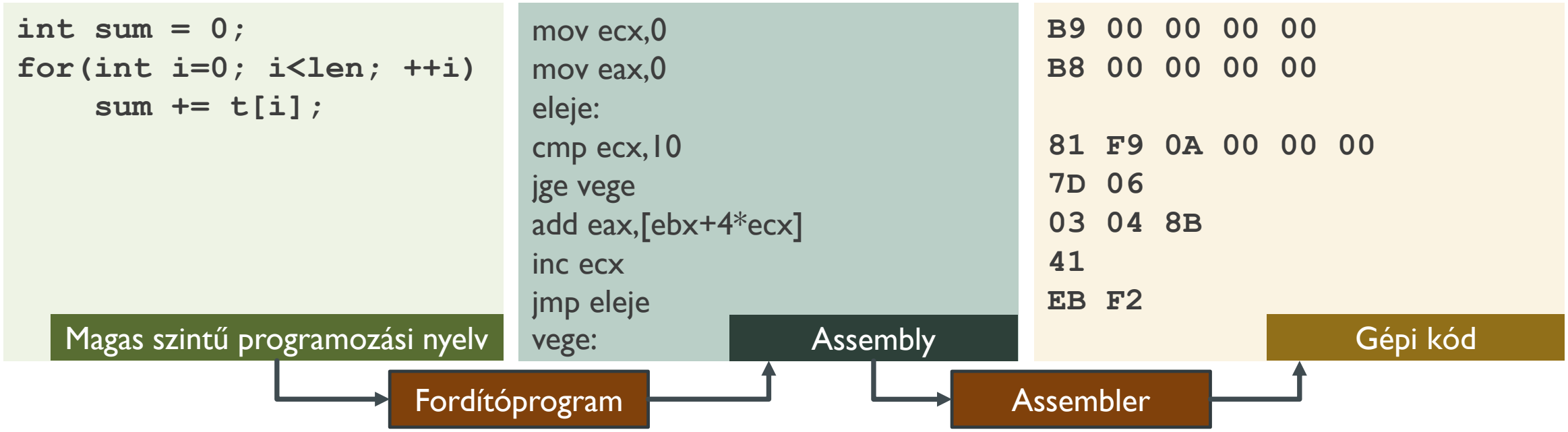
- *Feladat:*
  Alacsonyabb szintű belső reprezentációkra, végül tárgykóddá alakítja a programot
- *Bemenet:*
  Szintaxisfa attribútumokkal, szimbólumtábla
- *Kimenet (az utolsó menetben):*
  Tárgykód
- *Eszközök:*
  Kódgenerálási sémák

KÓDGENERÁTOR KIMENETE

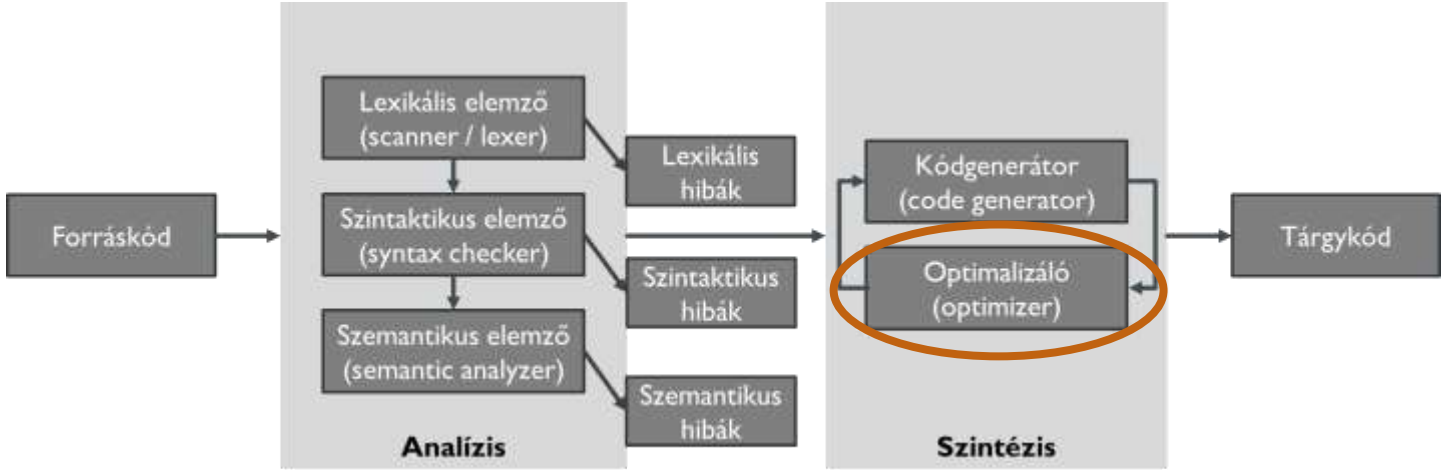
- Közvetlenül gépi kódot csak nagyon indokolt esetben érdemes generálni
- Helyette assembly kód (pl. valamely platform assembly nyelve vagy LLVM) generálható, amit assemblerekkel fordítunk tovább
- Transzláció: magas szintű nyelvek közötti fordítás
  - Ez lehet végcél, pl. projektek portolása egyik nyelvről a másikra
  - Elterjedt nyelvekre fordítás esetén használhatjuk azok fordítóit a gépi kód / bájtkód előállításához

ASSEMBLY, ASSEMBLER

- Assembly: Alacsony szintű (hardverközeli) nyelvek
  - Utasításnevek (mnemonikok)
  - Regiszernevek
  - Címkék a memóriacímek azonosítására
- Assembler: Fordítóprogram assemblyről gépi kódra

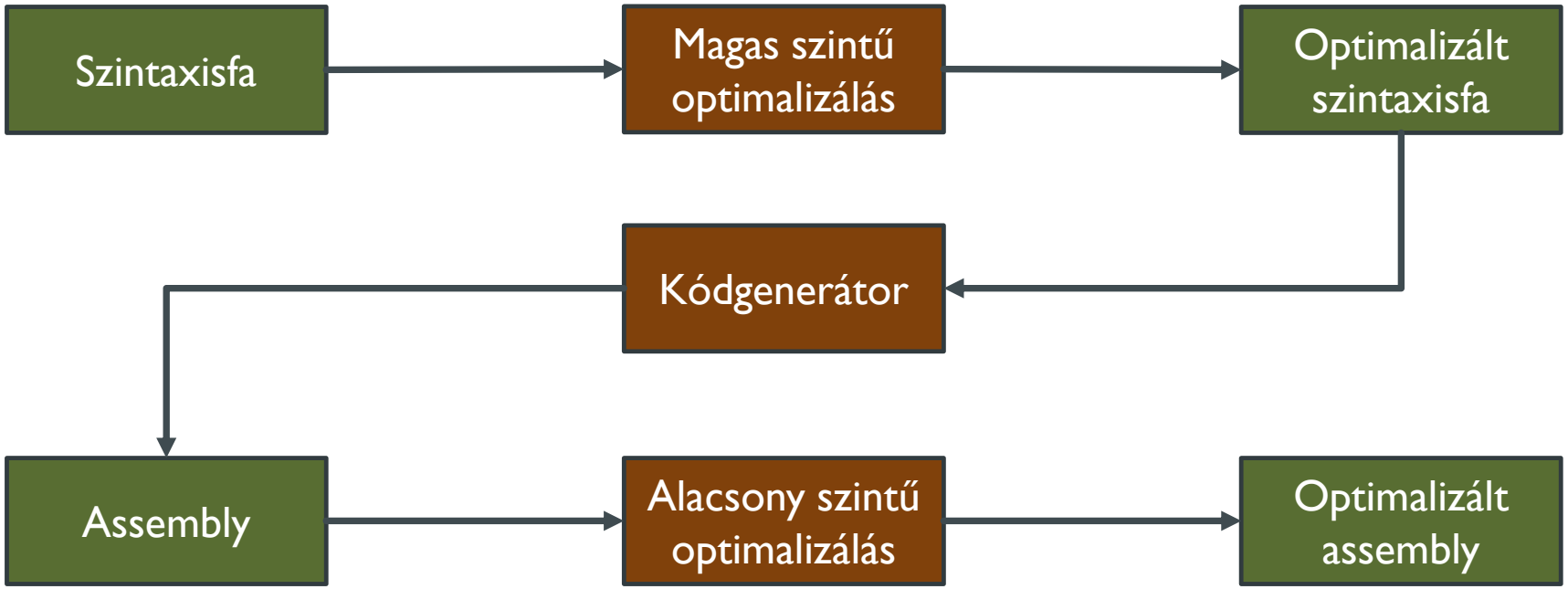


OPTIMALIZÁLÓ

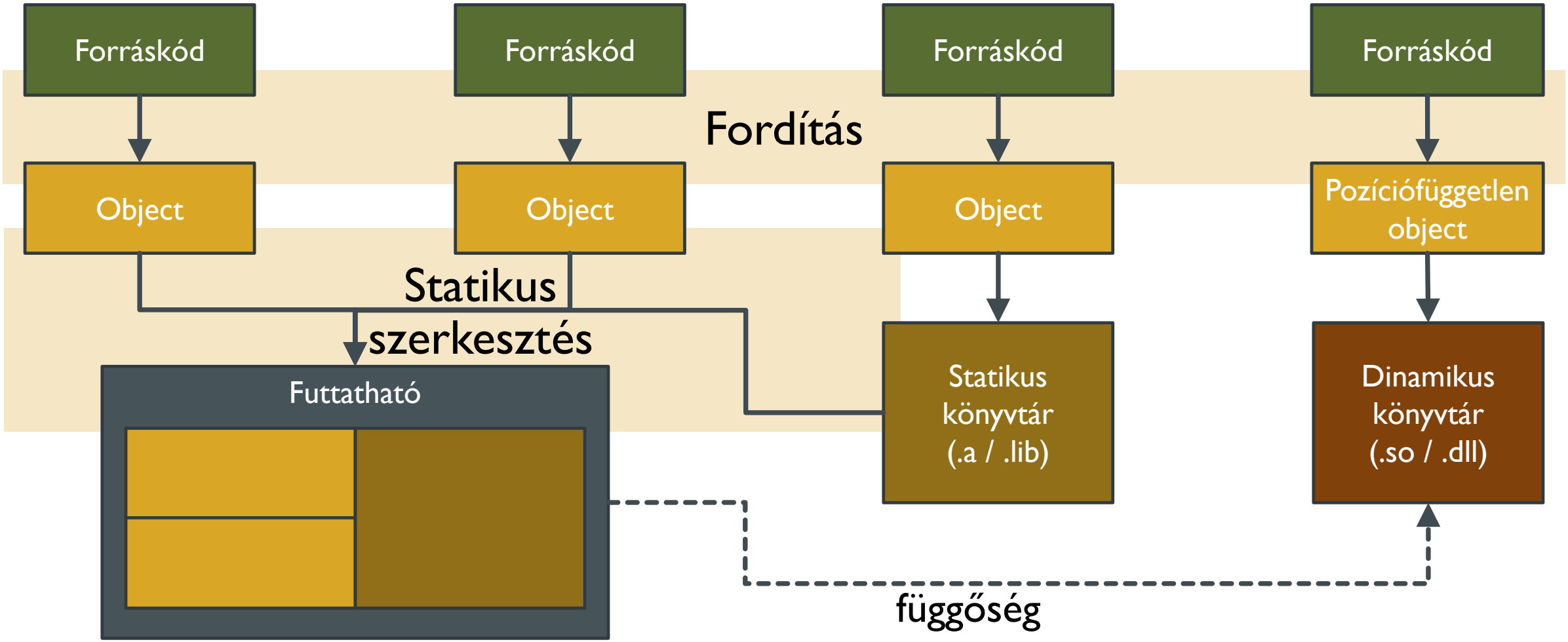


- *Feladat:*
  Kód átalakítása hatékonyságnövelés céljából (pl. sebességnövelés, memóriaigény csökkentés)
- *Bemenet:*
  Belső reprezentáció / tárgykód
- *Kimenet (az utolsó menetben):*
  Belső reprezentáció / tárgykód
- *Eszközök:*
  Statikus elemzés, transzformációs keretrendszerek

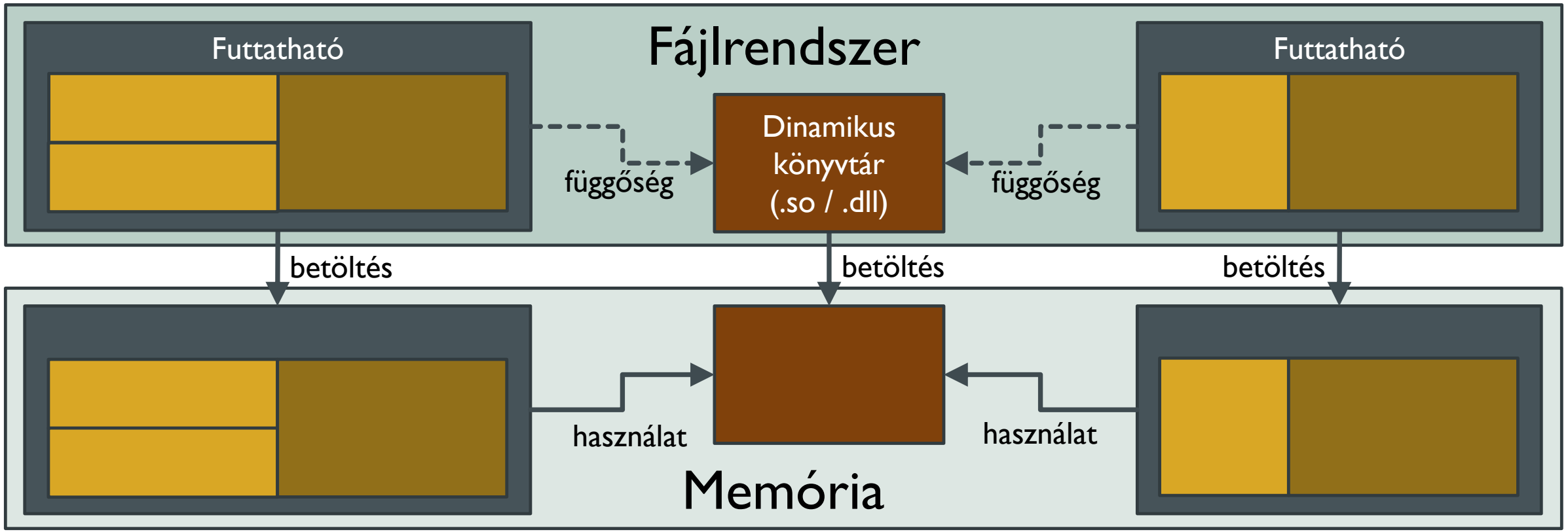
OPTIMALIZÁLÓ ÉS KÓDGENERÁTOR - PÉLDA



FORDÍTÁS ÉS SZERKESZTÉS



VÉGREHAJTÁS



Statikus könyvtárak: Minden futtathatóban külön-külön megtalálható.

Dinamikus (osztott) könyvtárak: Egy példányt használ minden futó processz.