

# Fordítóprogramok

## 1. előadás

Aszalós László

2015. szeptember 25.

- Jegymegajánló: utolsó hét, előadás időpontja.
- **Feladatgyűjtemény**
- Ajánlott irodalom
  - ▶ Csörnyei Zoltán: **Fordítóprogramok** Typotex, 2006
  - ▶ Fülöp Zoltán: **Formális nyelvek és szintaktikus elemzésük** Poligon, 1999
  - ▶ Gyimóthy Tibor, Havasi Ferenc, Kiss Ákos: **Fordítóprogramok** Typotex 2011

# Mire jó ez nekünk?

- Egy fordító megírása bevezet a nagyméretű programok fejlesztésébe (legnagyobb program az oktatás során)
- Fordítóprogram készítése a számítástudomány diadala, az elmélet segít az eredmény elérésében
- Fordítóprogram írása a programozási nyelvek kutatásának eszköze
- Sok alkalmazásban fel lehet használni az itt megismert módszereket, eszközöket.

# Általános elképzelés

## Input

```
while i<>j do  
  if i>j then i:=i-j else j:=j-i
```

## Output

```
...010101000101001011101010101001010110101...
```

# Fordítóprogramok a valós életben

**struktúra editor** szövegszerkesztő, de ismeri a nyelv kulcsszavait, felajánlja a szerkezeti elemeket (ciklus, stb.), ismeri a zárójelpárokat.

**pretty printer** a forráslistában más színekkel vagy betűtípussal szedi a szerkezeti elemeket, megjegyzéseket, változókat, stb.

**static checker** a program futtatása vagy lefordítása nélkül felfedez szintaktikai vagy szemantikai hibákat (nem definiált változó, típusütközés)

**szövegformázó** a forráskód a formázandó szöveg, amely tartalmazza a formázási parancsokat: alsó-felső index, képletek, ábrák. Pl. TeX, pic-tbl-eqn-troff, Lout, Metapost vagy web-böngésző

**programozható hardver** a forrás egy programkód, az output a neki megfelelő áramkör (FPGA)

**interpreter** a forráskód transzformálása helyett azok azonnali végrehajtása

**lekérdezések interpretere** az SQL a lekérdezéseket az aktuális adatbázis számára érthető formára alakítja.

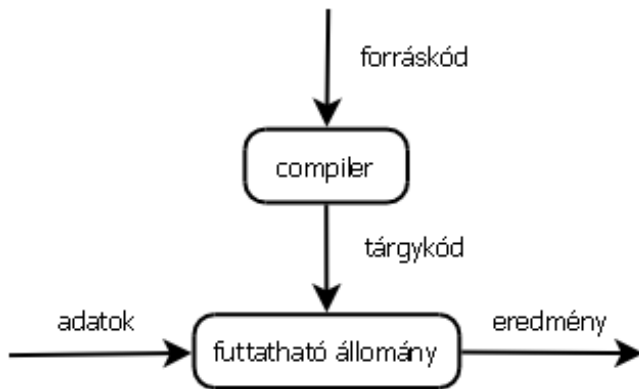
# Lehetőségek

- Döglött vagy redundáns kód megkeresése, eltávolítása
- Tesztek generálása a specifikáció alapján
- Céges kódolási szabályoknak megfelelés ellenőrzése
- A kód átalakítása egységes szerkezetűvé
- Forráskódból dokumentáció készítése
- Magas szintű specifikációnak megfelelés (kommunikáció, protokoll tesztelése, véges automaták)
- Magas szintű specifikációból forráskód generálása
- Kód portolása más nyelvre, környezetre
- folyamatábrák generálása programból

# Eredmények

- 40 000 termből álló logikai kifejezés 80 százalékos egyszerűsítése
- 800 000 soros COBOL kódban 12 százalék, 2,5 millió soros Java kódban 14 százalék, 400 000 soros C kódban 9 százalékos redundancia felfedezése (klónok)
- 1800 C fájlban 1,5 millió sorból kiszűrni a döglött preprocesszor utasításokat

# A fordítás folyamata



# Alapfogalmak

**forrásprogram** (avagy forrásnyelvű program) a fordító (compiler) inputja

**tárgyprogram** (avagy tárgynyelvű program) a fordító outputja

**fordítási idő** a forrásprogramból tárgyprogram készítéséhez szükséges idő

**futtatási idő** a program végrehajtásához szükséges idő

**$n$  menet** ha a forrásprogramot  $P$ ,

a tárgyprogramot az  $Q$ ,

a fordításifolyamatot  $T$  jelöli, és  $Q = T(P)$ , ahol

$$T = T_1 \circ \dots \circ T_n,$$

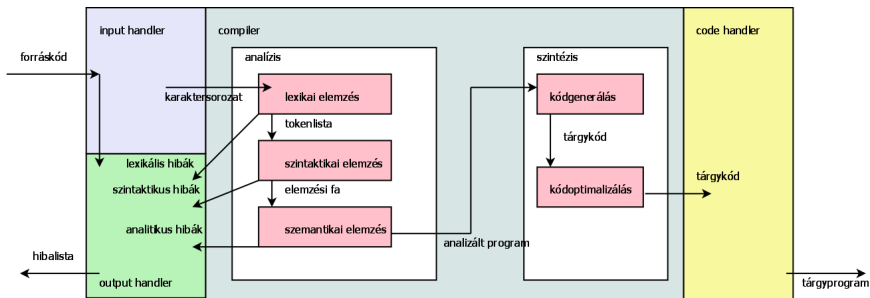
akkor  $n$  menetes fordítóról beszélünk.

$T_i \circ T_{i+1} \circ T_n(P)$ -t a program közbülső programformájának nevezzük.

# Értelmezés folyamata



# Fordítóprogram struktúrája



# Fordítóprogram struktúrája

## Input

forrásprogram

## Output

tárgyprogram és/vagy lista a forrásprogram hibáiról, a tárgyprogram adatairól

# Input-handler

A forrásnyelvű programot a háttértárolóról a memóriába kell juttatni.  
Pufferelt beolvasás (operációs rendszer feladata), megjegyzések,  
soremelések, felesleges szóközők figyelmen kívül hagyása.  
include fájlok, makrók kezelése

## Input

forrásprogram

## Output

karaktársorozat

# Output-handler

Fordítóprogram futásáról ember számára érthető módon beszámolni

## Input

forrásprogram, hibák

## Output

ember számára emészthető hibalista

# Code-handler

Operációs rendszertől függően kell elkészíteni az áthelyezhető, futtatható programot.

## Input

tárgykód

## Output

tárgyprogram

# Fordító/compiler

## Input

karaktersorozat

## Output

tárgykód, hibák

## Analízis

az input feldolgozása, lexikális egységekre bontás

## Szintézis

a forrás alapelemeinek megfelelő tárgykódok összeállítása tárgykóddá

# Lexikális elemző

A forrás felbontása szimbolikus egységekké: változók, kulcsszavak, konstansok, operátorok.

## Input

karaktársorozat

## Output

szimbólumsorozat, lexikális hibák.

## Felhasznált elmélet

reguláris nyelv, reguláris kifejezések, Thompson módszere determinisztikus automata generálására.

## Generátor

(f)lex

# Lexikális analízis ( $x := y + z * 30$ )

$x$  azonosító

$:=$  értékadás jele

$y$  azonosító

$+$  összeadás jele

$z$  azonosító

$*$  szorzás jele

$30$  konstans szám

# Szintaktikus elemző

## Input

szimbólumsorozat

## Output

szintaktikusan elemzett program (elemzési fa/szintaxisfa), szintaktikus hibák

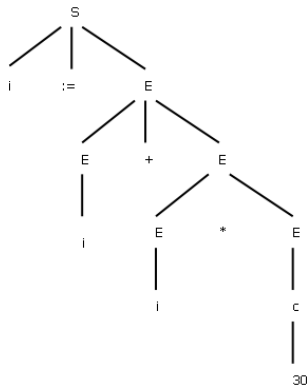
## Módszerek

általános alulról-fel vagy felülről-le elemző, LL, LR, precedencia elemzők

## Generátor

yacc, bison, ANTLR

# Szintaktikus elemzés



# Szemantikus elemző

Annak ellenőrzése, hogy pl. a megfelelő változók, konstansok definiálva vannak-e, megfelelő típusúak-e.

## Input

elemzési fa

## Output

analizált program, analitikus hibák

Ha az előbbi példában  $x$  valós típusú a 30 egész számot a fordító lecserélheti a 30.0 valós számra, illetve egész változót egy `int2real` függvény beillesztésével helyes típusúra hozhatja.

# Kódgenerátor

## Input

analizált program

## Output

tárgykód (általános, vagy processzorfüggő)

# Általános tárgykódú kódgenerátor

- tripod: háromcímű kód, egy utasításnak maximum három argumentuma lehet

```
temp1:=int2real(30)
temp2:=id3*temp1
temp3:=id2+temp2
id1:=temp3
```

- fordított lengyel jelölés

$x \ y \ z \ 30.0 \ * \ + \ :=$

Egy egyszerűbb fordító a processzor utasításainak kb. 20%-át használja.

# Kódoptimalizáló

## Hatékony program készítése

- azonos kódrészletek alprogramba helyezése,
- ismétlődő számítások kiemelése a ciklusokból
- ismétlődő kódrészletek alprogramba szervezése
- ciklusok szétbontása

## Input

tárgykód

## Output

tárgykód

# Kódoptimizálás

## Input

```
temp1:=int2real(30)
temp2:=id3*temp1
temp3:=id2+temp2
id1:=temp3
```

## Output

```
temp1:=id3*30.0
id1:=id2+temp1
```

# Kódgenerátor

## Input

```
temp1:=id3*30.0  
id1:=id2+temp1
```

## Output

```
LOAD   id3  
MUL    #30.0  
STORE  temp1  
LOAD   id2  
ADD    temp1  
STORE  id1
```

# További alapfogalmak

## Front-end

azon fázisok összessége, amely a forrásnyelvtől függ és független a tárgynyelvtől (lexikai, szintaktikai és szemantikai elemző, szimbólumtábla létrehozása, közbülső kód generálása, hibakezelés, esetleges kódoptimalizálás)

## Back-end

azon fázisok összessége, amely a tárgynyelvtől függ és független a forrásnyelvtől. (bizonyos kódoptimalizálások, kódgenerálás, a hibák és a szimbólumtábla kezelése)

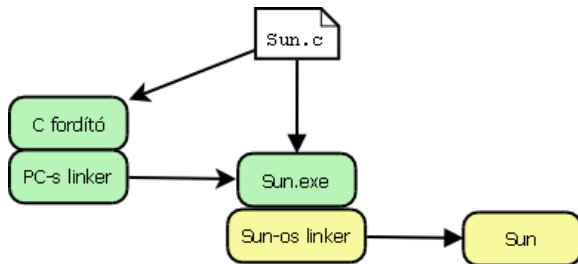
# Többszörös front- és back-end

Szokásos módszer, hogy több különböző nyelvi fordítónál is ugyanazt a közbülső kódot használjuk, így eme fordítók back end-je megegyezhet. Egy front end-hez tartozhat több back end, ezzel keresztfordítás készíthető.

Feladat Sun-os C fordító készítése PC-s C fordító felhasználásával

- SUN C fordító írása C-ben
- SUN C fordító készítése PC-s C fordítóval (tárgykód), majd PC-s szerkesztővel (PC-n futtatható kód)
- A PC-n futtatható SUN C fordítóval lefordítani a SUN C fordító forrását (SUN tárgykódú program)
- SUN-os szerkesztővel a tárgykódú program SUN-on futtatható alakjának elkészítése

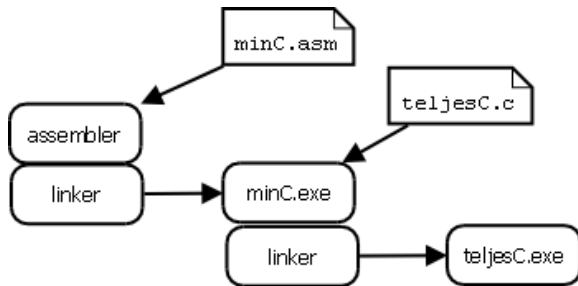
# Keresztfordítás



# Csizmahúzás - bootstrapping

- minimális C fordító írása assembly nyelven
- minimális C fordító készítése assemblerrel (tárgykód) majd szerkesztővel (futtatható kód)
- teljes C fordító írása minimális C nyelven
- teljes C fordító készítése minimális C fordítóval (tárgykód) majd szerkesztővel

# Csizmahúzás - bootstrapping



# Virtuális processzorra fordítás

- p-code machine (virtuális gép – elképzelt processzor gépi kódjának futtatására)
  - ▶ Pascal p-code 1973
  - ▶ Pascal-S compiler 1975
  - ▶ Python
  - ▶ Visual Basic
  - ▶ Java (JVM)
- Byte-code
  - ▶ Adobe Flash
  - ▶ CLISP, Emacs
  - ▶ Icon
  - ▶ Lua
  - ▶ m-code (MATLAB)
- JIT compiler