Prog1, C bevezetés

(2 alkalom)

Magasszintű programozási nyelvek 1
BSc előadás

Dr. Bátfai Norbert
egyetemi adjunktus
http://www.inf.unideb.hu/~nbatfai/
Debreceni Egyetem, Informatikai Kar,
Információ Technológia Tanszék
batfai.norbert@inf.unideb.hu
Skype: batfai.norbert

Prog1_1.ppt, v.: 0.0.19, 2012. 02. 14.
http://www.inf.unideb.hu/~nbatfai/#p1

A kurzus szervezése az óra blogján történik: http://progpater.blog.hu/
A Nokia Ovi store-ban is elérhető: http://store.ovi.com/content/100794

Felhasználási engedély

Bátfai Norbert
Debreceni Egyetem, Informatikai Kar, Információ Technológia Tanszék
<nbatfai@inf.unideb.hu, nbatfai gmail com>

Copyright © 2011 2012 Dr. Bátfai Norbert

E közlemény felhatalmazást ad önnek jelen dokumentum sokszorosítására, terjesztésére és/vagy módosítására a Szabad Szoftver Alapítvány által kiadott **GNU Szabad Dokumentációs Licenc 1.2**-es, vagy bármely azt követő verziójának feltételei alapján. Nem változtatható szakaszok: A szerzőről.
Címlap szövegek: Programozó Páternoszter, Bátfai Norbert, Gép melletti fogyasztásra.
Hátlap szövegek: GNU Jávácska, belépés a gépek mesés birodalmába.

Permission is granted to copy, distribute and/or modify this document under the terms of the **GNU Free Documentation License**, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being: A szerzőről, with the Front- Cover Texts being: Programozó Páternoszter, Bátfai Norbert, Gép melletti fogyasztásra,
and with the Back-Cover Texts being: GNU Jávácska, belépés a gépek mesés birodalmába.

http://www.gnu.hu/fdl.html

Célok és tartalom

Előadás (az érdeklődés felkeltése)

a) A kurzus teljesítésének feltételei, szabályai (sillabusz)
b) Általános kép adása a programozásról
c) C nyelvi bevezetés: a C nyelv, típusok, vezérlési szerkezetek, mutatók, deklarációk, kifejezések, függvények, paraméterátadás.

Labor (saját munkastílus kialakítása)

a) szövegszerkesztő használata, forráskód indentálása
b) gcc, g++, cmake használata parancssorból
c) a manuál lapok használata (Linux programmer's Guide lapok)
d) a PP 25-31. oldal példáinak letöltése, kipróbálása, megbeszélése a laborvezetővel.

Laborkártyák

a) Forrás és deklarációs kártyák

Otthoni opcionális feladat

a) GNU/Linux rendszer telepítése, s a japán világbajnok HELIOS csapat szoftvereinek otthoni installálása (rcssserver, rcssmonitor stb.)
http://www.youtube.com/watch?v=BVWkndHk3AE
http://en.sourceforge.jp/projects/rctools/releases/

Kapcsoldó videók, videómagyarázatok és blogok

1) http://progpater.blog.hu/2011/02/01/az_elso_eloadas_1
   http://www.youtube.com/watch?v=r2c-PKh7g8A
   http://www.youtube.com/watch?v=ozuhCwlLbWk
2) http://progpater.blog.hu/2011/02/04/a_szappanoperak_nyelve
   http://www.youtube.com/watch?v=vhMS9hXgDyU
3) http://progpater.blog.hu/2011/02/05/karakterhegyezes
   http://www.youtube.com/watch?v=NOylCvvcsz0
   http://www.youtube.com/watch?v=40--xKEah9k
4) http://progpater.blog.hu/2011/02/09/az_elso_eloadas_fizikailag
   http://www.youtube.com/watch?v=4KbzGCLR86w

2012:

1) http://progpater.blog.hu/2012/02/08/elindult_az_elit_kepzes
2) http://progpater.blog.hu/2012/02/15/retreat_hell

Számonkérés

Az írásbeli és a szóbeli vizsgán bármi (jegyzet, könyv, forráskód, számítógép mobiltelefon stb.) használható! Sőt, a számítógép javallott! Csakis az elektronikus vagy verbális kommunikáció tilos.

Számonkérés

Kérdések a Linux rendszerprogramozás témakörből

Nem lokális ugrások

Mi történik, ha elindítod az alábbi kis progit (csak a lényegi kódcsipet szerepel itt) és három Ctrl+C-t nyomsz?

```
sigjmp_buf jmpbuf;
void
kezdjuk_ujra (int sig)
{
    signal (SIGINT, kezdjuk_ujra);
    printf ("korte\a\n");
    siglongjmp (jmpbuf, 0);
}
int
main (void)
{
    if (signal (SIGINT, kezdjuk_ujra) == SIG_IGN)
        signal (SIGINT, SIG_IGN);
    sigsetjmp (jmpbuf, 1);
    printf("alma");
    for (;;)
        putchar (getchar ());
    return 0;
}
```

http://progpater.blog.hu/2011/06/06/egy_informatikai_targy_vizsgajan

Korábbi teljes tesztek: a Prog1 MEGAPACK-ban: http://www.inf.unideb.hu/~nbatfai/p1/

Minimális gyakorlati cél

A hallgató meg tudjon írni (másolás alapján) és le tudjon fordítani egyszerű kis C programokat, illetve az előadás végi „Helló, Világ!” C++ példát parancssorból cma

iad010.inf.unideb.hu - PuTTY

```
nbatfai@morse:~$ cat > elso.c
#include <stdio.h>
int
main(void)
{
int h = 0;
int n = 0x01;
do
++h;
while(n<<=1);
printf("A szohossz ezen a gepen
return 0;
}
nbatfai@morse:~$ gcc elso.c -o
nbatfai@morse:~$ ./elso
A szohossz ezen a gepen: 32 bit
nbatfai@morse:~$
```

```
def tudatfa(self, szoveg):
    szavak = re.compile("\s").split(unicode(szoveg, 'utf-8'))
    print szavak
    for szo in szavak:
        if self.fa.agak.has_key(szo):
            self.fa = self.fa.agak[szo]
            print szo + u" - mar szerepelt, raleptem"
        else:
            self.fa.agak[szo] = TudatFa()
            self.fa.agak[szo].agak={}
            print szo + " - nem szerepelt, felvettem"
            self.fa = self.gyoker

melyseg = 0
def kiir(self, honnan):
    for szo in honnan.agak.keys():
        m = (self.melyseg+1)*10
        formatum=u"{0:" + str(m) + "d} {1:10s}"
        print formatum.format(self.melyseg, "<"+szo+">")
        self.melyseg +=1
        self.kiir(honnan.agak[szo])
        self.melyseg -=1
```

Python: a program helyességének szükséges feltétele a jó olvashatósága (http://www.inf.unideb.hu/~nbatfai/kk/)

int n=0x

h=1	0000	0000	000					
h=31	1000	0000	000					
h=32	0000	0000	0000	0000	0000	0000	0000	0000

Minimális elméleti cél

1) A hallgató tudjon értelmezni egy állapotátmenet diagramjával megadott Turing gépet (elmondani, hogy *„a gép ebben az állapotban van, ezt olvassa, akkor átmegy ebbe, ezt írja és ide lép”*)
2) Adott egyszerű grammatika esetén ismerje fel, mi a generált nyelv
3) Szintaktikai elemzés fogalmát meg tudja világítani
4) BNF-ben tudjon definiálni egyszerű fogalmakat, például mi egy „egész szám”
5) C nyelv kapcsán: típusok, vezérlési szerkezetek, mutatók, deklarációk, kifejezések, függvények, paraméterátadás

Szabályok (sillabusz)

a) Laboron katalógus, hiányzás >= 3.5 (**laborkártya** miatt tört) esetén nincs aláírás

b) Labor teljesítésének további szükséges feltétele egy saját program bemutatása a laborközösség előtt, a félév közepén. A feladat kötött: adok egy karakterekre (0,1 betűkre) működő algoritmust, azt kell átírni, hogy bináris bemenetre (0, 1 bitekre) működjön (lásd 3 előadás és labor). A félév végén pedig egy saját robofocis fejlesztés bemutatása.

c) Vadászat: 6 győzelem = +1 jeggyel jobb vizsgaeredmény (de ez csak elégtelennél jobb eredményre működik), 18 = +2, 54 = +3 (illetve a szóbeli tételhúzáskor is megfelelő kedvezmények).

d) Laboron bevezetjük a laborkártya rendszert: ez azt szolgálja, hogy a hallgatóság készüljön a laborokra. Egy laborkártya egy vagy néhány egyszerű, előre megadott kérdést tartalmaz, amin a labor elején szóban minden hallgató megválaszol. Aki nem tudja megválaszolni, az a következő laboron biztosan kap egy kártyát. Két „nem tudásonként” egy laborhiányzást könyvelünk el! Újdonság: a hallgatói laborkártya!

Szabályok

e) A szóbeli vizsga tételei az előadások címével egyeznek meg, mivel a tematika még nem végleges, ez módosulhat: http://www.inf.unideb.hu/~nbatfai/#p1 tipikus tétel a „minimális elméleti cél” című fólián megadott

f) Előadáson is van katalógus, aki <= 2 alkalommal hiányzott, annak +5% pont az írásbeli teszten

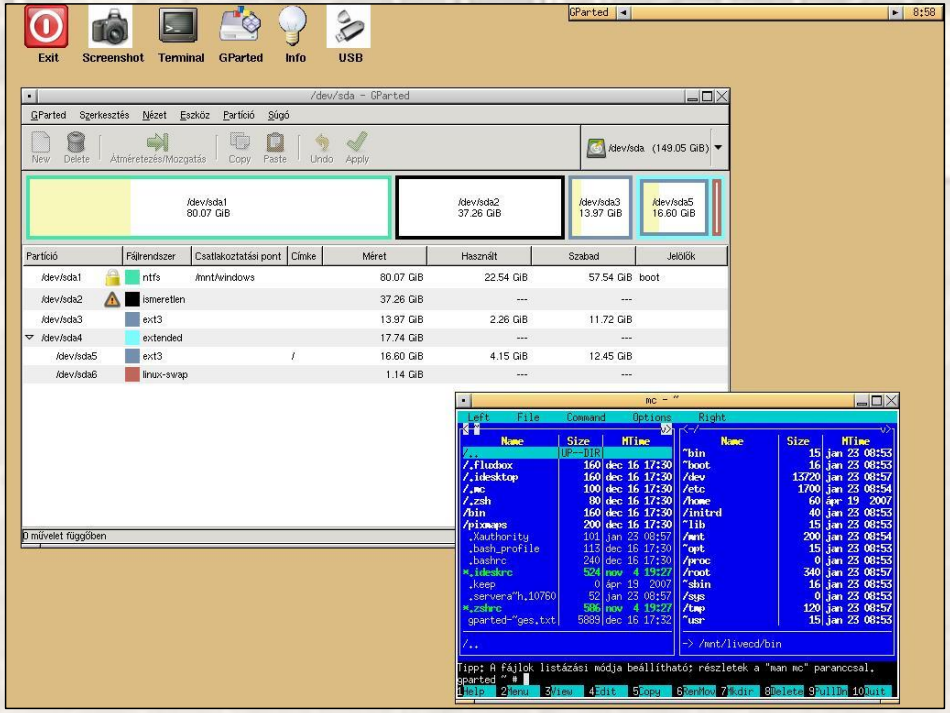
g) Az írásbeli és a szóbeli vizsgán bármi (jegyzet, könyv, forráskód, számítógép, mobiltelefon stb.) használható! (Csak az on-line kommunikáció tiltott!)

h) Jegymegajánlás van a Vadászok Ligája első 3 helyezettjének: http://progpater.blog.hu/2011/05/01/indul_a_vadaszok_ligaja

i) A vadászat saját pontjait a hallgató maga tartja nyilván (mennyit mire mikor) formában, s ha a blogon lévővel nincs összhangban, azt 2 héten belül jelzi nekem a listája elküldésével egyetemben

Ha még nincs Linuxod…

A Gnome Partition Editor http://gparted.sourceforge.net LiveCD-jét (http://gparted-livecd.tuxfamily.org) javaslom kiírni (Windows alatt például a http://infrarecorder.sourceforge.net InfraRecorder programmal tudod ezt megtenni, az *Actions/Burn Image...* menü választásával) Ezzel tudod zsugorítani a Windowst, majd új partíciókat létrehozni stb. Sőt, ha menet közben úgy alakul – Hoppá – hogy már nem tudod bebootolni a fentlévő oprendszereket, abban is segít. Ami fontos, addig, amíg nincs mentésed, bele se kezd semmibe, mert megvan a jó esély, hogy mindent elvesztesz a lemezedről!



Variációk a „papíros programozásra”

„Ha papíron is lehetne programozni, akkor senki nem venne drága elektronikus számítógépeket.”

„Papíron programozni olyan, mint víz alatt lélegezni.”

„Ha papíron lehetne programozni, akkor a MÉH lett volna meghatározó világcég és nem a Microsoft.”

Mottóink

„Csak akkor értesz valamit, ha be tudod programozni. Te magad és nem valaki más! Ha nem tudod beprogramozni, akkor csak úgy gondolod, hogy érted.” - Gregory Chaitin: META MATH! The Quest for Omega

http://www.cs.auckland.ac.nz/CDMTCS/chaitin/omega.html

„Nem tudok kimerítő leírást adni arról, hogy hogyan tudsz megtanulni programozni -- nagyon összetett tudásról van szó. Egyet azonban elárulhatok: a könyvek és tanfolyamok nem érnek túl sokat (sok, valószínűleg a legtöbb hacker autodidakta). Aminek van értelme: (a) kódot olvasni és (b) kódot írni.” - Eric Steven Raymond: How To Become A Hacker

A magyar fordítás: http://esr.fsf.hu/hacker-howto.html

„Talk to other programmers; read other programs. This is more important than any book or training course.” - Peter Norvig: Teach Yourself Programming In Ten Years

http://norvig.com/21-days.html

Web2 diákok Web2 tanárok

Informatika a felsőoktatásban 2011 konferencia Debrecen, 2011. augusztus 24-26.

WEB 2 DIÁKOK WEB 2 TANÁROK

WEB 2.0 TEACHERS AND WEB 2.0 STUDENTS

Bátfai Norbert[1]

Összefoglaló: Ebben a munkában a Debreceni Egyetem Informatikai Kara mérnök informatikus BSc Magas szintű programozási nyelvek című kurzusainak szervezését mutatjuk be. A kurzusok szervezése négy pilléren nyugszik: az előadás prezentációs fóliákon, a labor méréseken, a kurzus on-line segítségként is funkcionáló blogján és a programozási versenyfeladatokon. A kurzus blogja nagyon változatos tartalmakat szolgáltat: a telepítési útmutatóktól, a videóra vett, YouTube-ra kitett előadásokon, a kurzusban készített fotókon át, a C, C++, Java programozási példákig. De a blogon tesszük közzé s kurzus híreit is. A kurzusok tartalmát részletesen is tárgyaljuk. Távolabbi nem titkolt célunk úgy továbbfejleszteni programozás oktatásunkat, hogy az Észak-Kelet Magyarországi régió egyik legerősebb programozási centrumává válhasson. Ezért azt a célt tűzzük magunk elé, hogy egy vezető fejlesztői közösséggé váljunk régiónkban. Hitünk szerint vázolt kurzusaink szervezése majd sikerrel szolgálja ezt a célt.

Kulcsszavak: programozás, oktatás, Programozó Páternoszter, 2010-2011 tanév II. félév

Abstract: In this work we will introduce the organisation of the System Engineering BSc. students' Programming courses at the Faculty of Informatics at the University of Debrecen. The organisation of the

http://nodes.agr.unideb.hu/if2011/dokumentum/IF2011_CD_Kiadvany.pdf

Kötelező olvasmányok

K&R: ***A C programozási nyelv*** KR oldalak

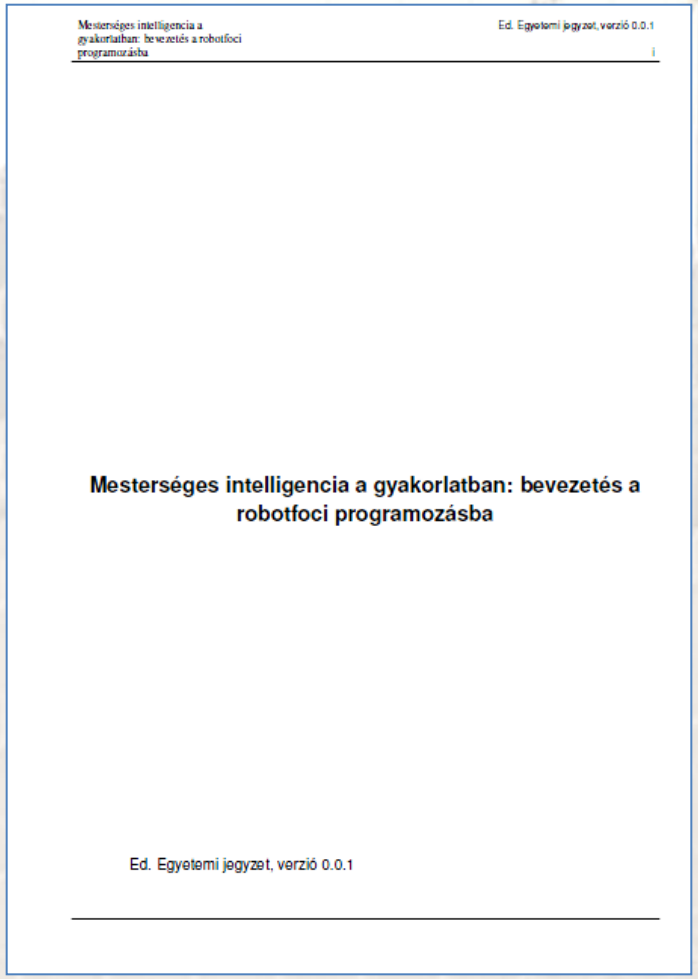
http://webpac.lib.unideb.hu/WebPac/CorvinaWeb?action=onelong&showtype=longlong&recnum=422685&pos=4

BME old.

Benedek Zoltán, Levendovszky Tihamér: ***Szoftverfejlesztés C++ nyelven***, Budapest, 2007, Szak K

http://webpac.lib.unideb.hu/WebPac/CorvinaWeb?action=onelong&showtype=longlong&recnum=469668&pos=2

Könyvek



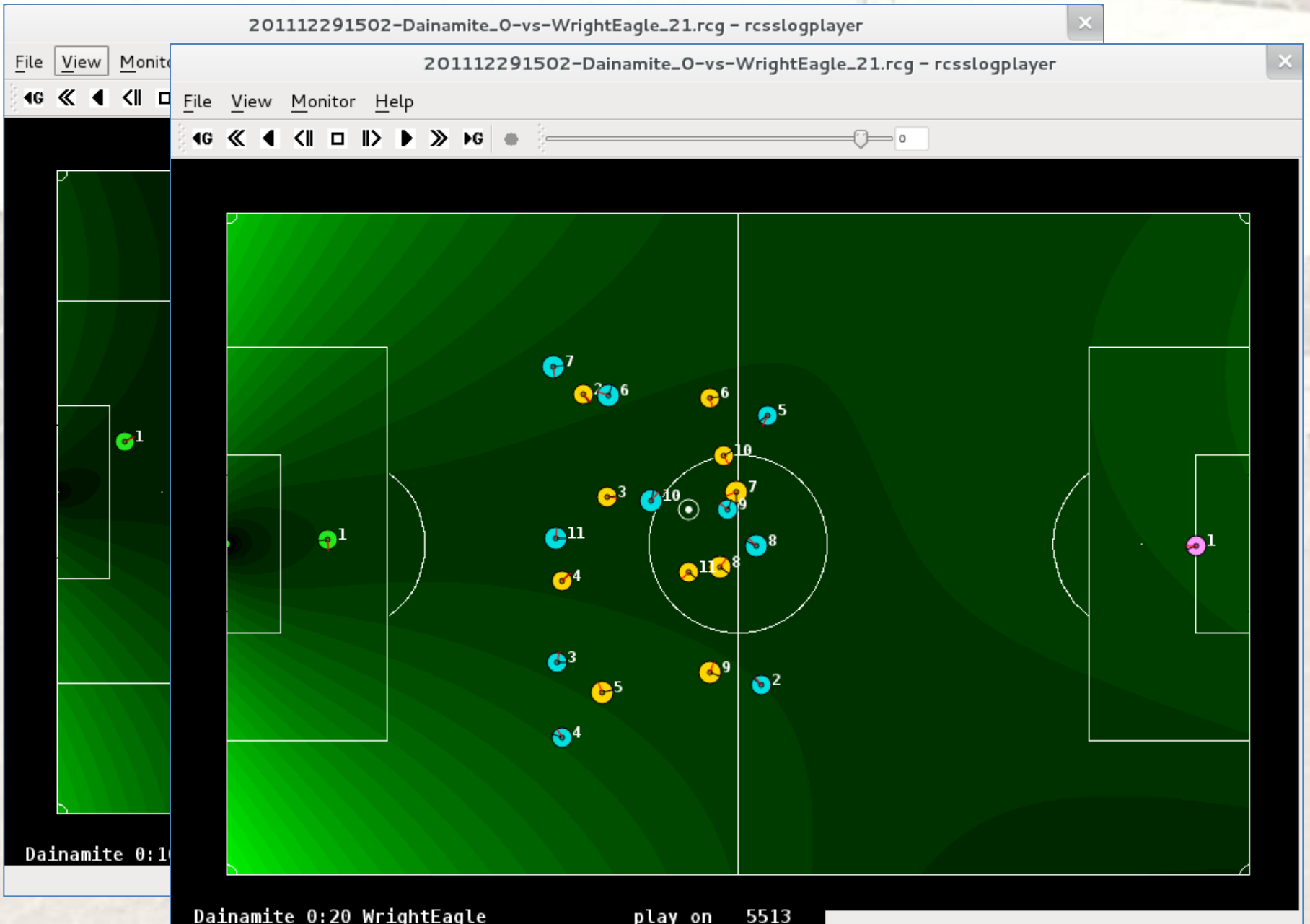
Bátfai Norbert: ***Mesterséges intelligencia a gyakorlatban: bevezetés a robotfoci programozásba***
http://www.inf.unideb.hu/~nbatfai/mircbook.pdf

MIRC oldal

Bátfai Norbert: ***Párhuzamos programozás GNU/Linux környezetben: SysV IPC, P-szálak, OpenMP***

PARP old.

Labor: ki mit programoz éppen?



Előkészületek 1.

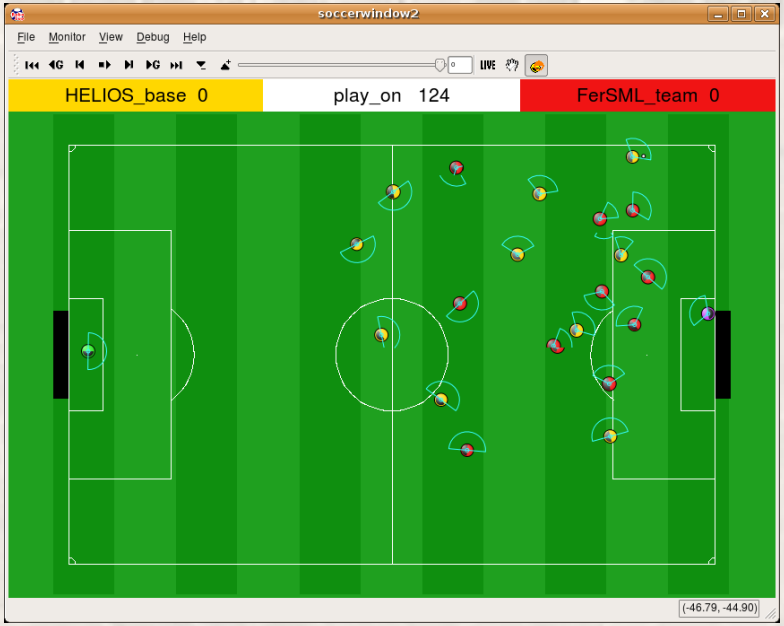
GNU/Linux

A robotfocitika három törvénye posztban bemutatott telepítés reprodukálása:
http://fersml.blog.hu/2010/12/28/a_robotfocitika_harom_torvenye
+soccerwindow:
http://fersml.blog.hu/2011/01/01/fersml_avatar_2_robocup_foci_agens

http://progpater.blog.hu/2011/02/05/a_felkelo_nap_palyaja
kommentjében

Részletes telepítés Linux/Windows alatt

PARP 17



Előkészületek 2.

Kapcsolódó filmek

http://www.imdb.com/title/tt0118884/

A Pi elosztott számításához

http://www.imdb.com/title/tt1210166/

A robotfoci és a ***FerSML*** témák érzelmi megalapozásához

Előkészületek 3.

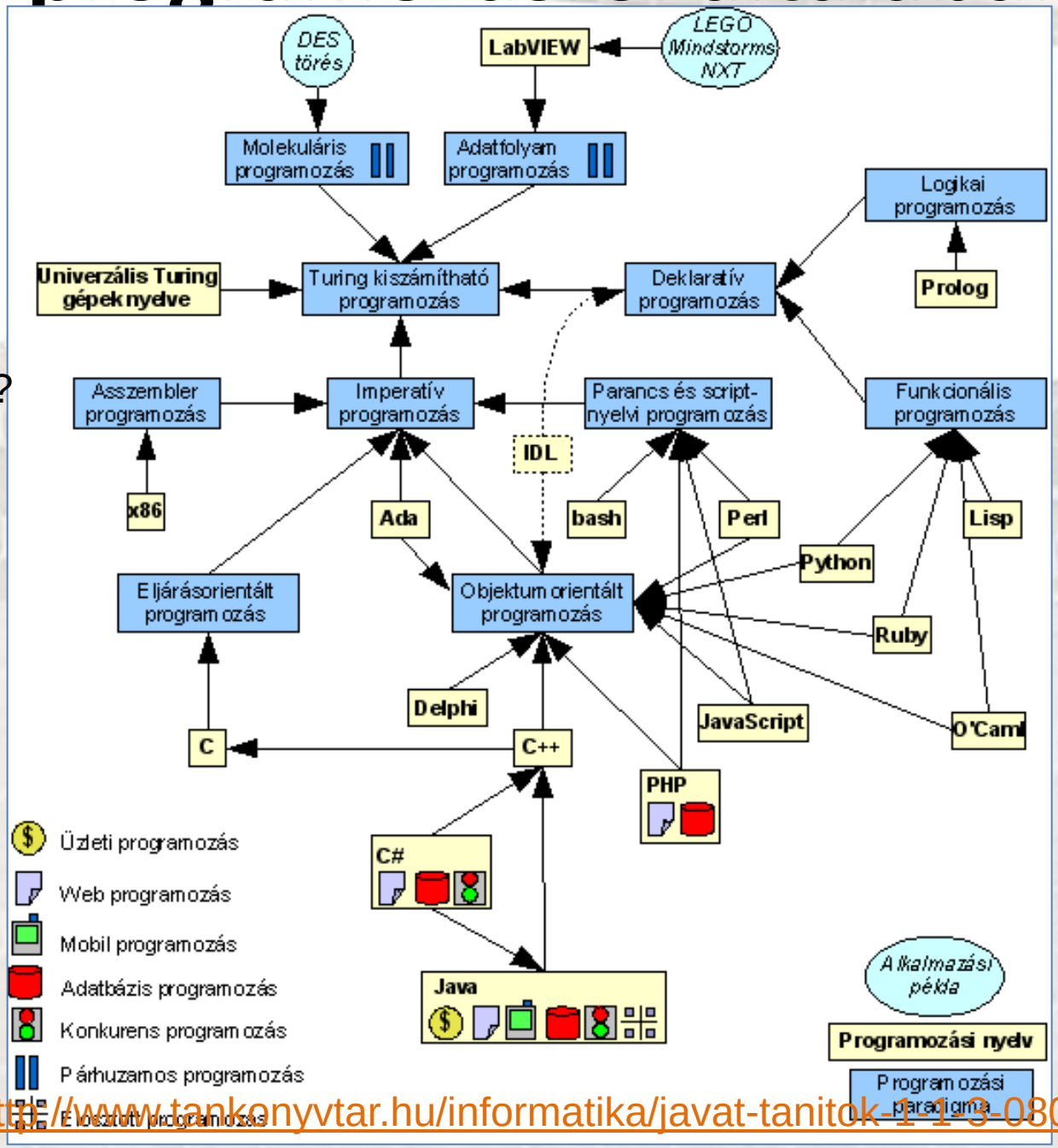
http://www.imdb.com/title/tt1251757/

Hallgatói laborkártya

KR 1~90

A programozásról általában

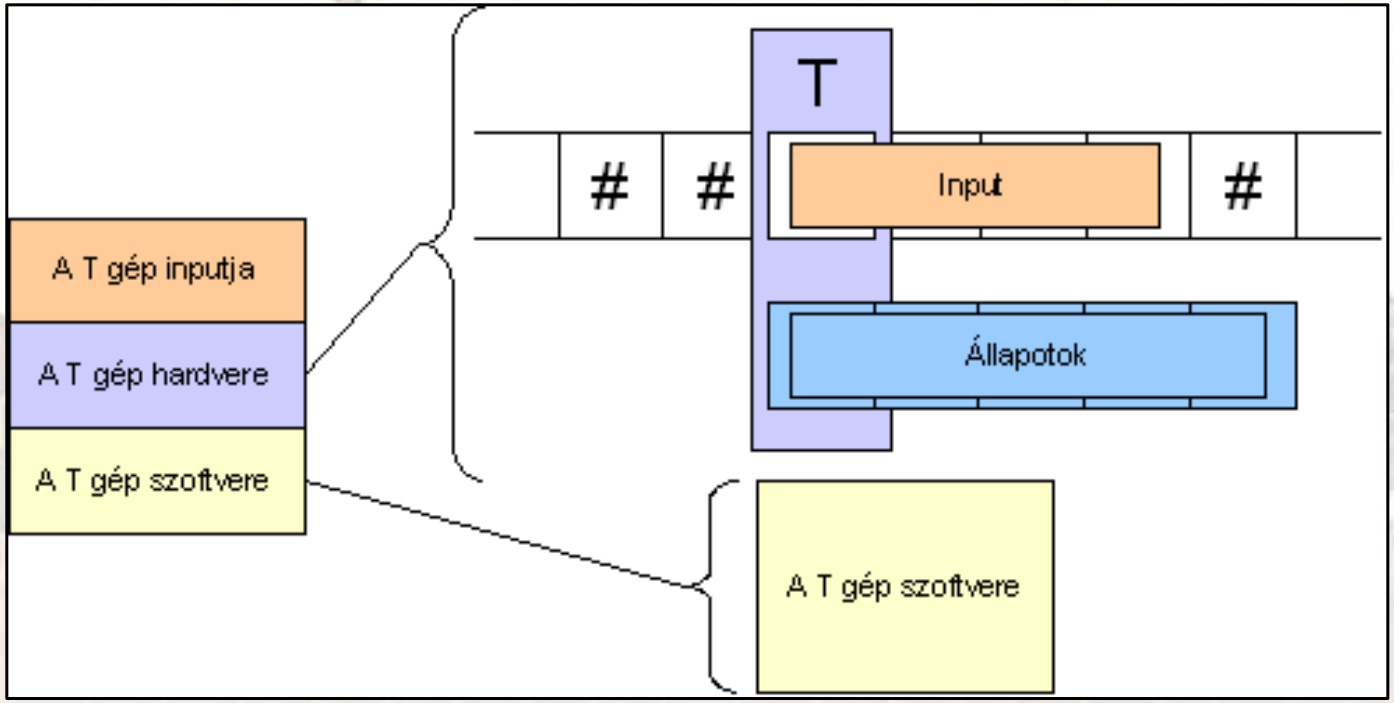
Imperatív
MIT, HOGYAN?



Deklaratív
MIT, HOGYAN?

http://www.tankonyvtar.hu/informatika/javat-tanitok-1-1-3-080904-2

Turing-féle gépek



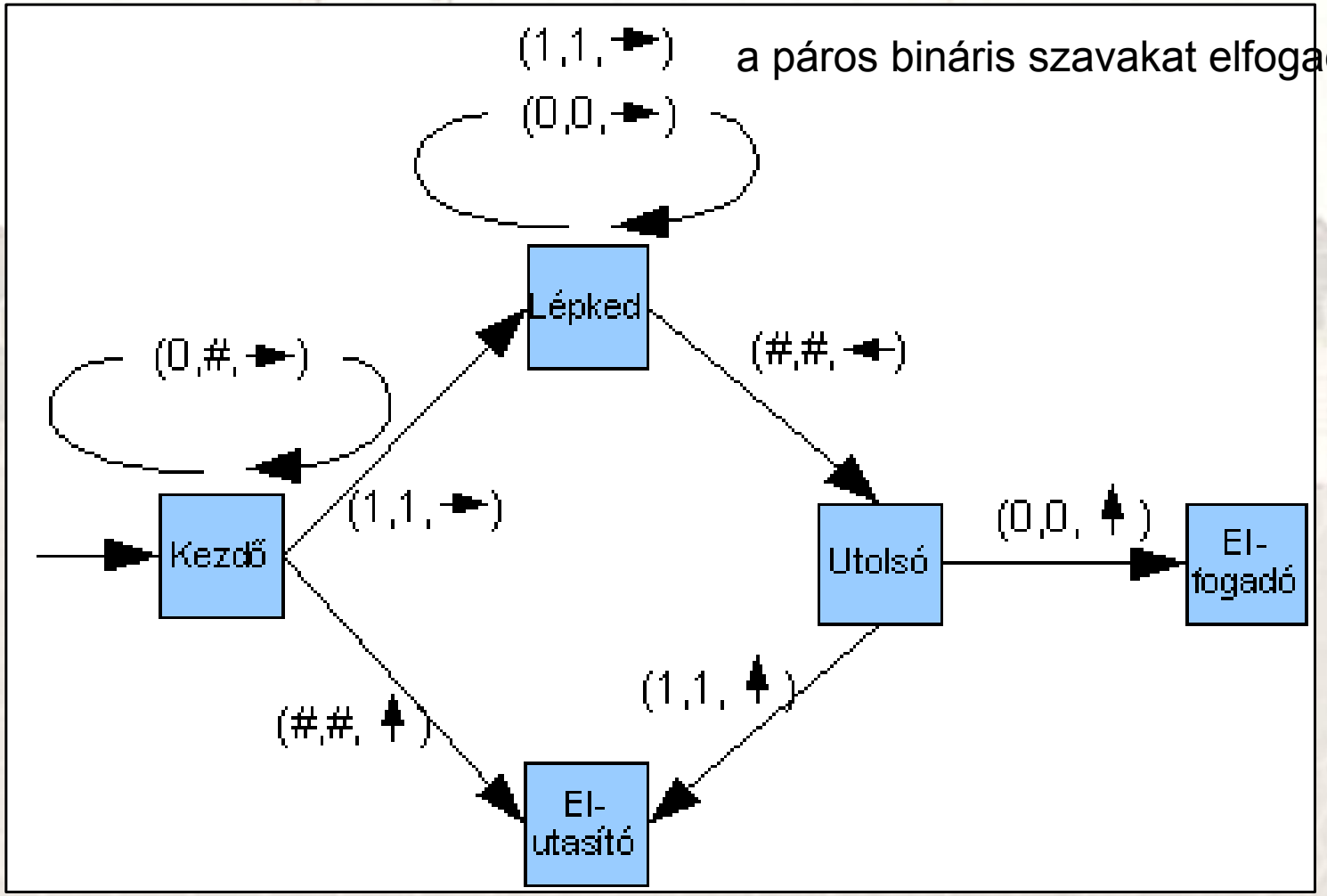
Képek forrása és részletes leírás: Javát tanítok,
http://www.tankonyvtar.hu/main.php?objectID=5314387

Turing színház (9-12 éves korig!)
http://javacska.lib.unideb.hu/seged/szakkor-Turing.pdf

Turing-féle gépek

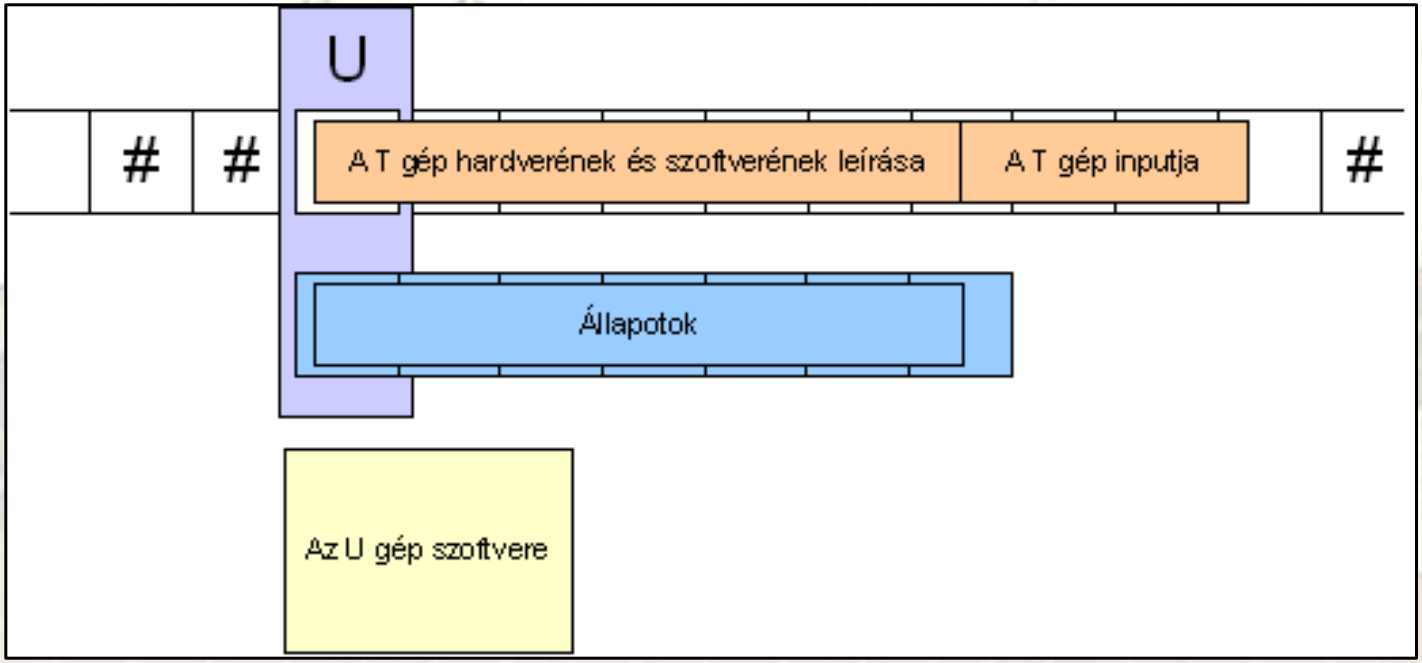
Állapotátmenet diagramal megadott,

a páros bináris szavakat elfogadó gép.



Képek forrása és részletes leírás: Javát tanítok,
http://www.tankonyvtar.hu/main.php?objectID=5314387

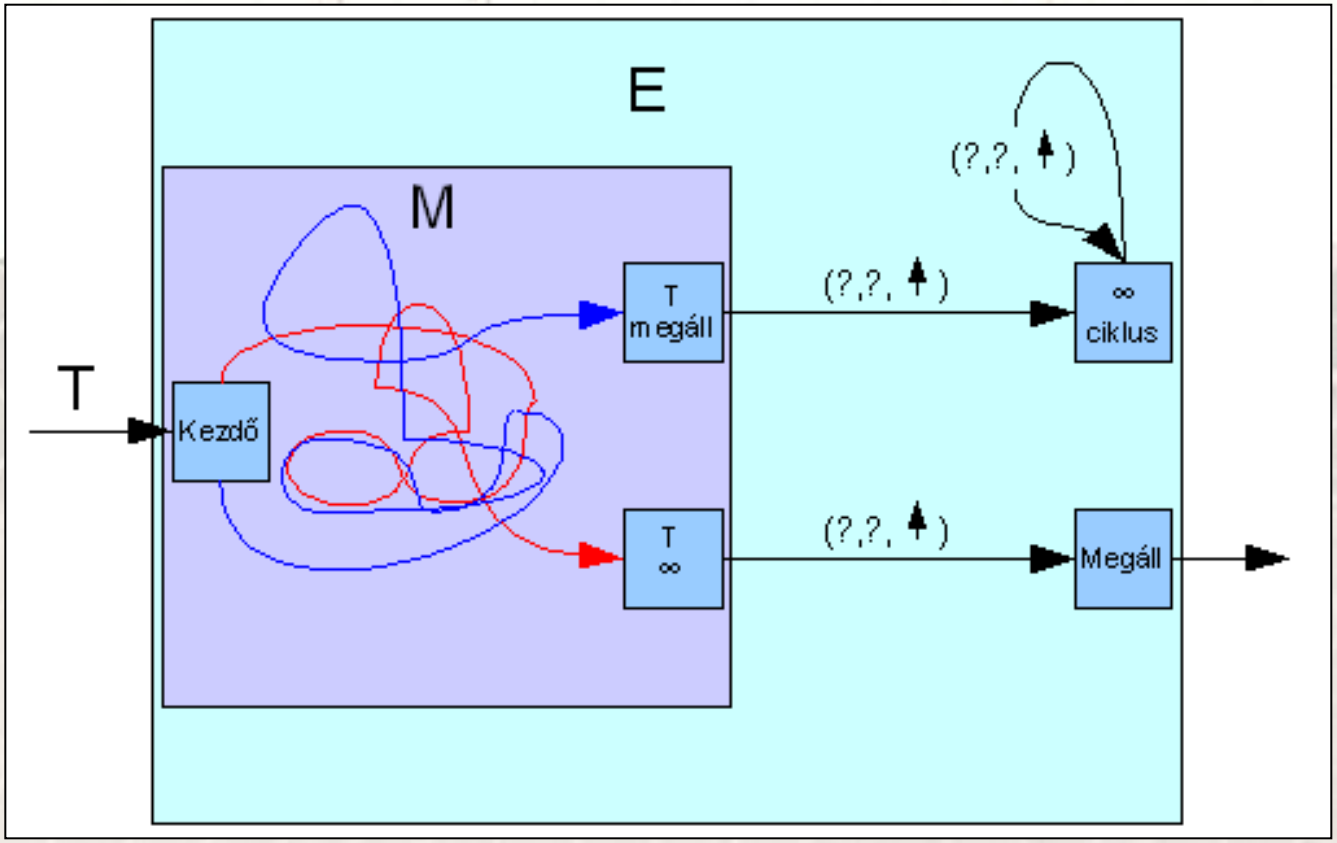
Univerzális Turing gépek



Egy univerzális gép.

Képek forrása: Javát tanítok, http://www.tankonyvtar.hu/main.php?objectID=5314387

Megállási probléma



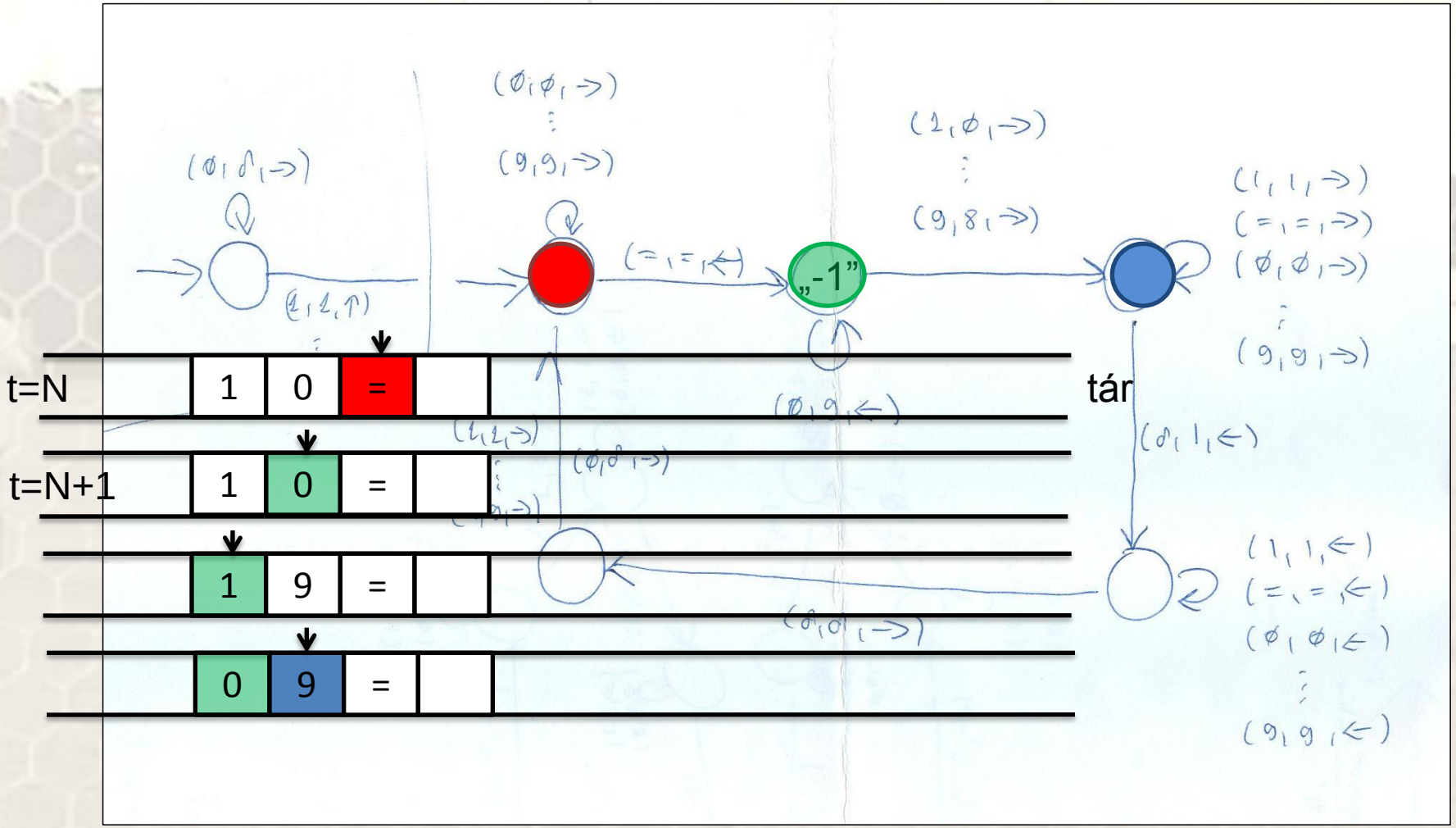
T=E ?

Képek forrása: Javát tanítok, http://www.tankonyvtar.hu/main.php?objectID=5314387

Végtelen ciklus

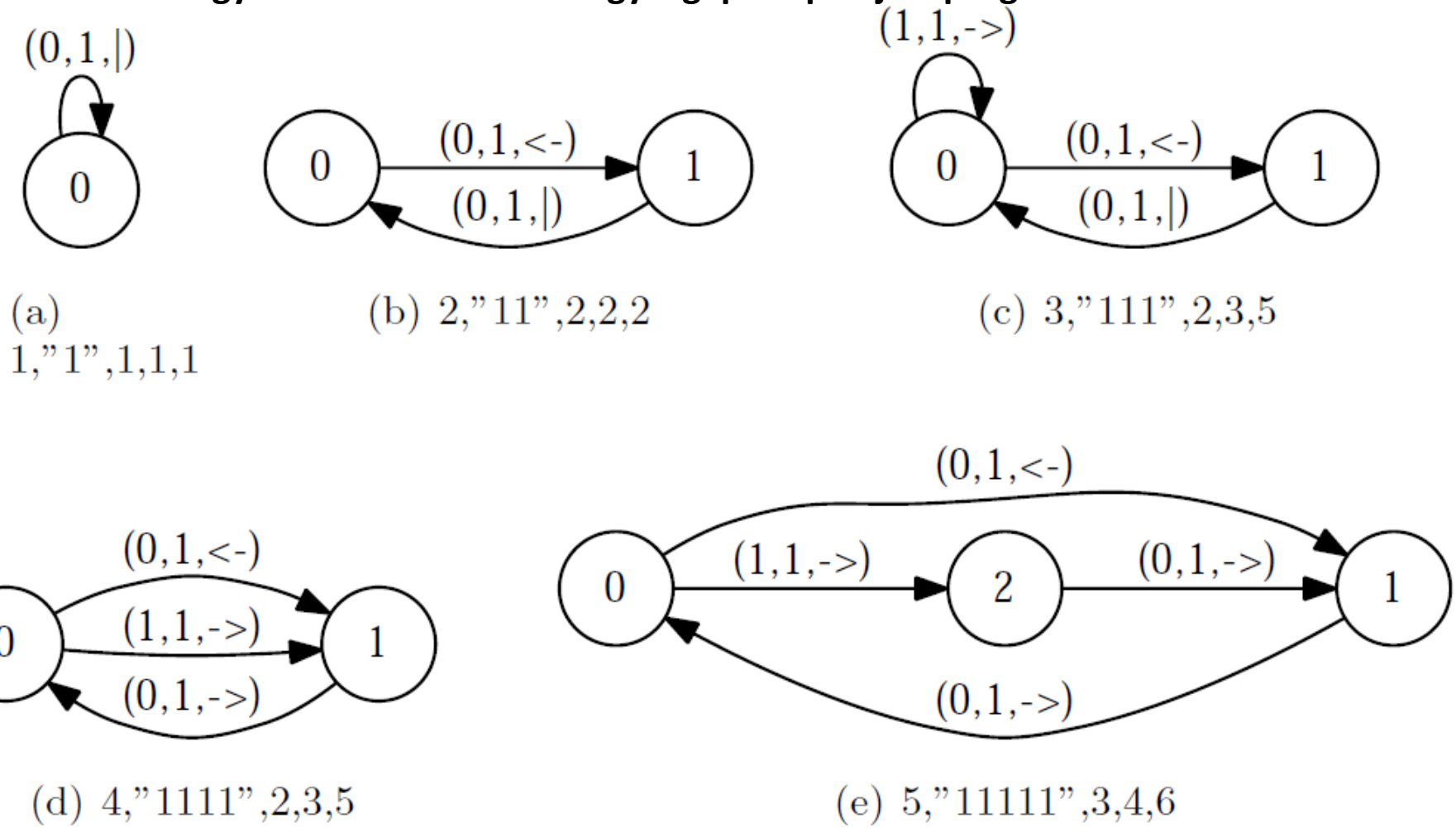
tábla

Decimálisból unárisba átváltó Turing gép



Szorgos Hódok téma

Egyáltalán az emberek vagy a gépek sportja a programozás?



(a) 1,"1",1,1,1

(b) 2,"11",2,2,2

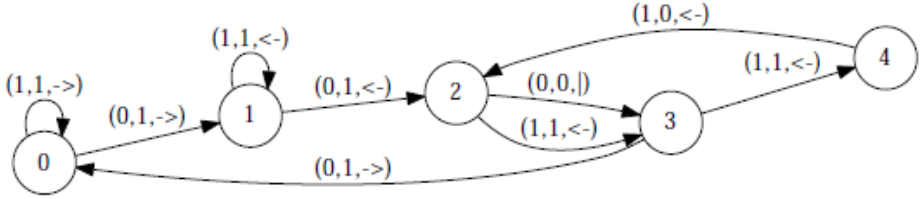
(c) 3,"111",2,3,5

(d) 4,"1111",2,3,5

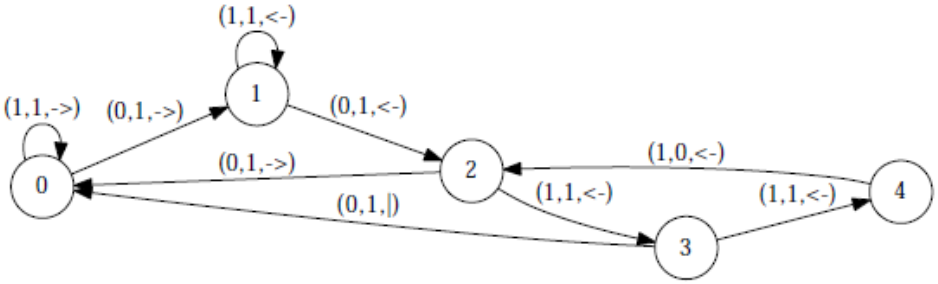
(e) 5,"11111",3,4,6

Kép forrása: http://arxiv.org/abs/0908.1159

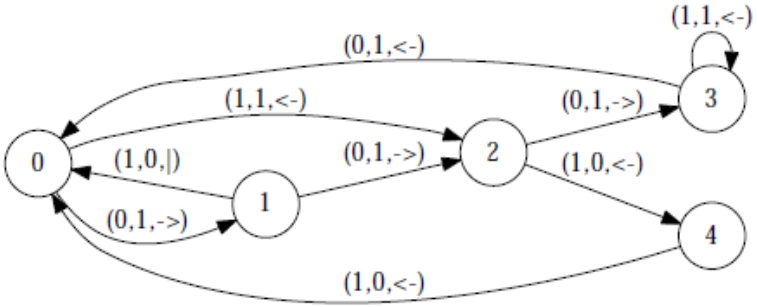
Figure 1: "Placid Platypus machines" are found by our programs, (# of 1's, "$T(\lambda)$", # of states, # of rules, # of steps)



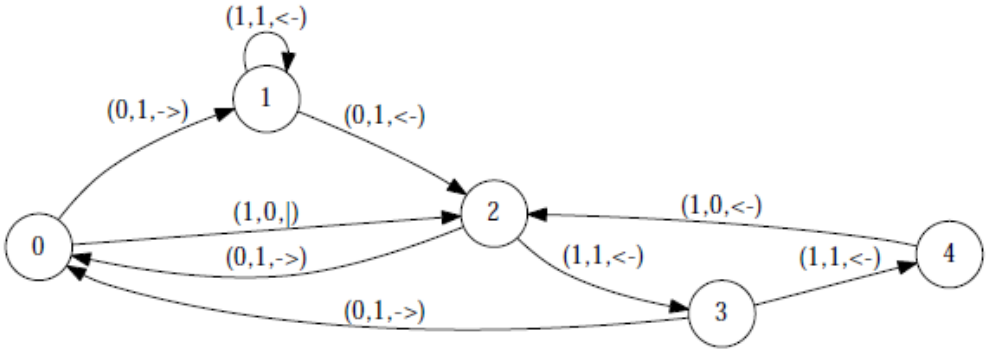
(a) 4097, 11.801.882, (9, 0, 11, 1, 5, 2, 15, 3, 3, 9, 4, 19, 5, 21, 6, 5, 7, 27, 9, 12)



(b) 4097, 11.798.832, (9, 0, 11, 1, 5, 2, 15, 3, 9, 4, 5, 5, 21, 6, 4, 7, 27, 9, 12)



(c) 4097, 70.740.809, (9, 0, 11, 1, 15, 2, 17, 3, 1, 4, 23, 5, 24, 6, 3, 7, 21, 9, 4, 5, 5, 21, 6, 5, 7, 27, 9, 12)
9, 0)



(d) 4097, 17.689.051, (9, 0, 11, 1, 13, 2, 15, 3, 2, 17, 3, 1, 4, 23, 5, 24, 6, 3, 7, 21, 9, 4, 5, 5, 21, 6, 5, 7, 27, 9, 12)

Kép forrása: http://arxiv.org/abs/0908.4013

Chomsky-féle nyelvosztályok

Noam Chomsky, 50-60 évek, MIT, Nyelvészet és matematika

S, X, Y „változók”
a, b, c „konstansok”
S → abc, S → aXbc, Xb → bX, Xc → Ybcc, bY → Yb, aY -> aaX, aY → aa
S-ből indulunk ki

S (S → aXbc)
aXbc (Xb → bX)
abXc (Xc → Ybcc)
abYbcc (bY → Yb)
aYbbcc (aY → aa)
aabbcc

$a^n b^n c^n$

S (S → aXbc)
aXbc (Xb → bX)
abXc (Xc → Ybcc)
abYbcc (bY → Yb)
aYbbcc (aY -> aaX)
aaXbbcc (Xb → bX)
aabXbcc (Xb → bX)
aabbXcc (Xc → Ybcc)
aabbYbccc (bY → Yb)
aabYbbccc (bY → Yb)
aaYbbbccc (aY → aa)
aaabbbccc

S
aXbc
abXc (Xc → Ybcc)
abYbcc

Révész könyv, 12. o. (Bev. a form. nyelvek elméletébe, Akadémiai Kiadó, 1979

Noam Chomsky, 50-60 évek, MIT, Nyelvészet és matematika

A, B, C „változók”
a, b, c „konstansok”
$A \rightarrow aAB$, $A \rightarrow aC$, $CB \rightarrow bCc$, $cB \rightarrow Bc$, $C \rightarrow bc$
S-ből indulunk ki

A ($A \rightarrow aAB$)
aAB ($A \rightarrow aC$)
aaCB ($CB \rightarrow bCc$)
aabCc ($C \rightarrow bc$)
aabbcc

$a^n b^n c^n$

A ($A \rightarrow aAB$)
aAB ($A \rightarrow aAB$)
aaABB ($A \rightarrow aAB$)
aaaABBB ($A \rightarrow aC$)
aaaaCBBB ($CB \rightarrow bCc$)
aaaabCcBB ($cB \rightarrow Bc$)
aaaabCBcB ($cB \rightarrow Bc$)
aaaabCBBc ($CB \rightarrow bCc$)
aaaabbCcBc ($cB \rightarrow Bc$)
aaaabbCBcc ($CB \rightarrow bCc$)
aaaabbbCccc ($C \rightarrow bc$)
aaaabbbbcccc

Révész könyv, 13. o. (Bev. a form. nyelvek elméletébe, Akadémiai Kiadó, 1979)

Noam Chomsky, 50-60 évek, MIT, Nyelvészet és matematika

G=(VN, VT, S, H)

H része (VN U VT)* VN (VN U VT)* X (VN U VT)* (= a bal oldalon legyen legalább egy nemterminális)

Mondatszerkezetű

Környezetfüggő (hossznemcsökkentő)
P1XP2 → P1QP2, P1, P2 eleme (VN U VT)*, X VN beli, Q (VN U VT)+ beli, kivéve S → üres, de akkor S nem lehet jobb oldali egyetlen szabályban sem

Környezetfüggetlen
X → P, X VN beli, P (VN U VT)* beli,

Például programozási nyelvek szintaxisának leírására a BNF.

(Jobb) reguláris
X → pY, X → p, X, Y VN beli, p VT* beli

Egy nyelv akkor ilyen típusú, ha ilyen grammatikával lehet generálni.
Pl. $a^n b^n c^n$ nyelv nem környezetfüggetlen.

Szintaktikai elemzés

1) Szintaxis
2) Lexikai elemzés, lexikális egységek
3) Szintaktikai elemzés, szintaktikai egységek
4) A fordítóprogram feladata

BNF, Backus Normal Form

John Backus, ALGOL 60
Környezetfüggetlen nyelvekhez

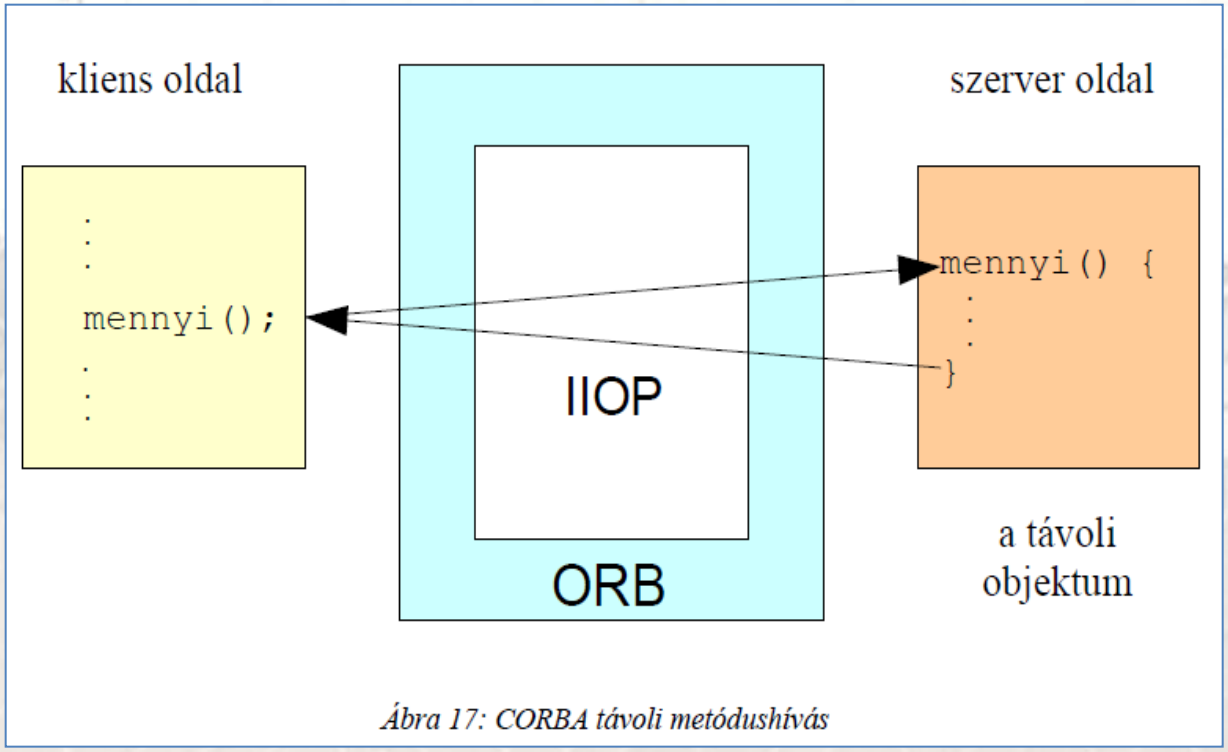
<nem terminális> ::= konkatenációja terminálisoknak, nem terminálisoknak, illetve {iteráció}, [opcionális], alter|natíva

<egész szám> ::= <előjel><szam>
<előjel> ::= [-|+]
<szam> ::= <szamjegy>{<szamjegy>}
<szamjegy> ::= 0|1|2|3|4|5|6|7|8|9

A CORBA világa

Common Object Request Broker Architecture

1) Elosztott
2) Heterogén
3) OO



Ábra 17: CORBA távoli metódushívás

Object Management Group (OMG)
http://www.omg.org
CORBA 3.1 (2008)
http://www.omg.org/spec/CORBA/3.1
C Language Mapping Specification
http://www.omg.org/spec/C/1.0/PDF/
C++ Language Mapping, Version 1.2
http://www.omg.org/spec/CPP/1.2/PDF/

OMG IDL

CORBA 3.1 interfészek, http://www.omg.org/spec/CORBA/3.1/Interfaces/PDF/
39. o.

Részlet az OMG IDL nyelvtanból:

<interface_dcl> ::= <interface_header> “{” <interface_body> “}”
<interface_header> ::= [“abstract” | “local”] “interface” <identifier>
[<interface_inheritance_spec>]

```
module korba
{
    interface Kliens
    {
        void uzenet(in string uzenet);
    };

    interface Szerver
    {
        void beszall(in Kliens kliens);
        void kiszall(in Kliens kliens);
        void uzenet(in string uzenet);
    };
};
```

OMG IDL

CORBA 3.1 interfészek, http://www.omg.org/spec/CORBA/3.1/Interfaces/PDF/
39. o.

<interface_header> ::= [“abstract” | “local”] “interface” <identifier>
[<interface_inheritance_spec>]
<interface_inheritance_spec>::=
“:” <interface_name> { “,” <interface_name> }*

```
interface Hos : Szereplo
{
  attribute long megtalaltErtekek;
  readonly attribute long eletek;
  void megtalaltam(in Kincs kincs);
  boolean megettek();
};
interface Kincs : Szereplo
{
  attribute long ertek;
  readonly attribute boolean megtalalva;
  boolean megtalalt(in Hos hos);
};
interface Szorny : Szereplo
{
```

A legnépszerűbb programozási nyelv

TIOBE index

Position Feb 2012	Position Feb 2011	Delta in Position	Programming Language	Ratings Feb 2012	Delta Feb 2011	Status
1	1	=	Java	17.050%	-1.43%	A
2	2	=	C	16.523%	+1.54%	A
3	6	↑↑↑	C#	8.653%	+1.84%	A
4	3	↓	C++	7.853%	-0.33%	A
5	8	↑↑↑	Objective-C	7.062%	+4.49%	A
6	5	↓	PHP	5.641%	-1.33%	A
7	7	=	(Visual) Basic	4.315%	-0.61%	A
8	4	↓↓↓↓	Python	3.148%	-3.89%	A
9	10	↑	Perl	2.931%	+1.02%	A
10	9	↓	JavaScript	2.465%	-0.09%	A
11	13	↑↑	Delphi/Object Pascal	1.964%	+0.90%	A
12	11	↓	Ruby	1.558%	-0.06%	A
13	14	↑	Lisp	0.905%	-0.05%	A
14	26	↑↑↑↑↑↑↑↑↑↑↑↑	Transact-SQL	0.846%	+0.29%	A
15	17	↑↑	Pascal	0.813%	+0.08%	A
16	22	↑↑↑↑↑↑	Visual Basic .NET	0.796%	+0.21%	A--
17	32	↑↑↑↑↑↑↑↑↑↑	PL/SQL	0.792%	+0.38%	A
18	24	↑↑↑↑↑↑	Logo	0.677%	+0.10%	B
19	16	↓↓↓	Ada	0.632%	-0.17%	B
20	25	↑↑↑↑↑	R	0.623%	+0.06%	B

http://www.tiobe.com/content/paperinfo/tpci/tpci_definition.htm

A C nyelv, történeti háttér

A Unix és a C együtt születtek, ezért a kurzusban ne erőszakoljuk Windows alatt a Unix környezet MinGW - (Minimalist GNU for Windows), Cygwin használatát!

1964, M.I.T, Bell Labs, GE: MULTICS (Ken Thompson-1969 B nyelv, Dennis Ritchie)

1971, UNIX (és C nyelv-Dennis Ritchie)
az AT&T forrásban (C források) is adja oktatási célokra a UNIX-ot

D. Ritchie, K. Thompson:
The UNIX Time-Sharing System (1974)
http://citeseer.ist.psu.edu/viewdoc/summa

1974, Berkeley Egyetem: Berkeley UNIX (1977, Bill Joy, Berkeley Software Distribution, BSD)

1983, GNU (Richard Stallman)

1984, UNIX System V

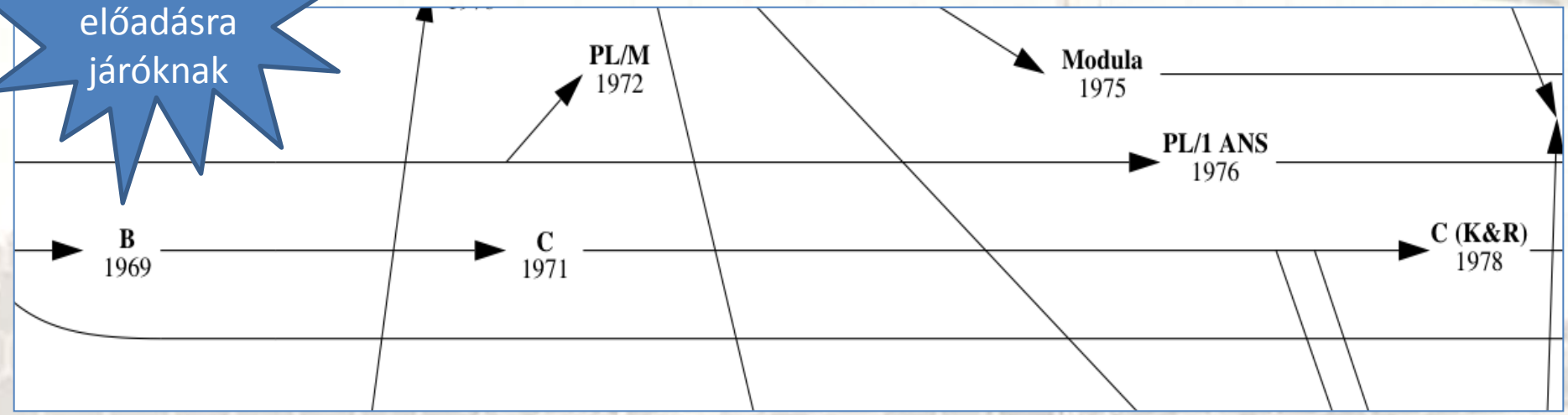
1986, POSIX (Portable OS-IX)

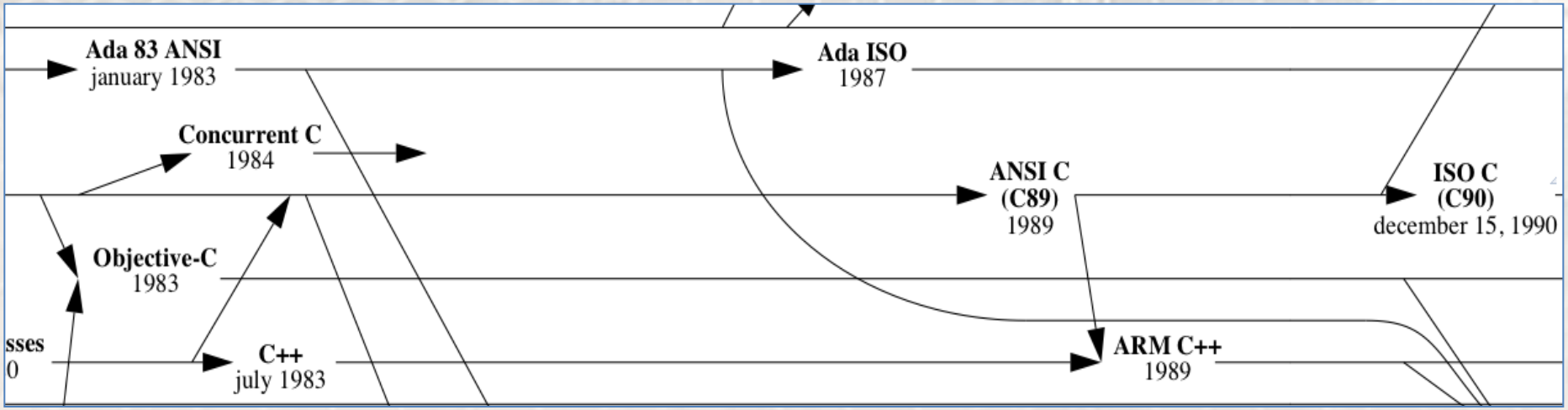
1987, MINIX (Andrew S. Tanenbaum)

1991, Linux (Linus Torvalds hallgatóként)

A C nyelv

+1 trófea előadásra járóknak





Unix Timeline: http://www.levenez.com/unix/ Tökéletes poszterek a szobádba!
Computer Languages Timeline: http://www.levenez.com/lang/

Szabványok és a C fordító

C
1971

C (K&R)
1978

C+
july 1

ANS
(C8
19

IS
(C
decembe

IS
(C
april

C++

IS
dece

C+

C++
2

C++1
july 2

```
(3)                    Linux Programmer's Manual                    PRINTF(3)
```

```
GCC(1)                          GNU                          GCC(1)

NAME
       gcc - GNU project C and C++ compiler

SYNOPSIS
       gcc [-std=standard]...
...
-std=
            Determine the langua
            supported when compi

                   gnu89
                       Default, IS
                      tures).

                   gnu99
                   gnu9x
                       ISO C99 plu
                       in GCC, thi
                       cated.

                   c++98
                       The 1998 IS

                   gnu++98
                       The same as
                       default for
```

```
...
/*
 * Bináris hatványozás mod k,
 * a 16^n mod k értékének kiszámítása.
 *
 * n a kitevő.
 * k a modulus.
 */
long long int
binhatmod (long long int n, long long int k)
{
```

PP 228

Mi történik: gcc … -std=c99, -ansi (-std=c89)?

```
$ gcc pi_bbp.c -o pi_bbp -lm -std=c99
$ ./pi_bbp 0 1000 1
243F6A8885A308D313198A2E03707344A4093822299F3
1D0082EFA98EC4E6C89452821E638D01377BE5466CF34
E90C6CC0AC29B7C97C50DD3F84D5B5B547091...
```

Szabványok

```
for (int i = 0; i < olvasott_bajtok; ++i)
  {

    buffer[i] = buffer[i] ^ kulcs[kulcs_index];
    kulcs_index = (kulcs_index + 1) % kulcs_meret;

  }
```

```
int i = 0;

int kulcs_meret = strlen (argv[1]);
strncpy (kulcs, argv[1], MAX_KULCS);

while (olvasott_bajtok = read (0, (void *) buffer, BUFFER_MERET))
  {

    for (i = 0; i < olvasott_bajtok; ++i)
      {

        buffer[i] = buffer[i] ^ kulcs[kulcs_index];
        kulcs_index = (kulcs_index + 1) % kulcs_meret;

      }
```

Szabványok

A kurzus elején használjuk a K&R körny referencia kézikönyv fejezetét, majd a

C99 szabványt

Rationale for
International Standard—
Programming Languages—
C
http://www.open-std.org/jtc1/sc22/wg14/www/C99RationaleV5.10.pdf

INTERNATIONAL STANDARD ©ISO/IEC ISO/IEC 9899:TC2
Programming languages — C
Munkaverzió:
http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf

A new feature of C99: The **hh** and **ll** length modifiers were added in C99. **ll** supports the new **long long int** type. **hh** adds the ability to treat character types the same as all other

Parancssori használat

1) Fordítás

Fordítsuk a 109. oldal, *Párhuzamos, folyamatok sorával* című pontjának programját a `-Wall` opcióval, illetve anélkül:

```
$ gcc -lnsl -o szerver szerver.c
$ gcc -Wall -lnsl -o szerver szerver.c
szerver.c: In function 'main':
szerver.c:48: warning: unused variable 'kliens'
szerver.c:46: warning: unused variable 'kliensm'
```

Program előállítása

1) Fordítás
2) Relokáció, kapcsolatszerkesztő program

Nézzük meg például a *116*.oldal, *Párhuzamos, POSIX szálakkal* című pont programjának fordítását:

```
$ gcc -lpthread -lnsl -o szerver szerver.c
```

Írassuk ki, hogy ekkor milyen könyvtárakat használ a `szerver`:

```
$ ldd szerver
        libpthread.so.0 => /lib64/libpthread.so.0
(0x0000003841e00000)
        libnsl.so.1 => /lib64/libnsl.so.1 (0x0000003849500000)
        libc.so.6 => /lib64/libc.so.6 (0x0000003841100000)
        /lib64/ld-linux-x86-64.so.2 (0x0000003840f00000)
```

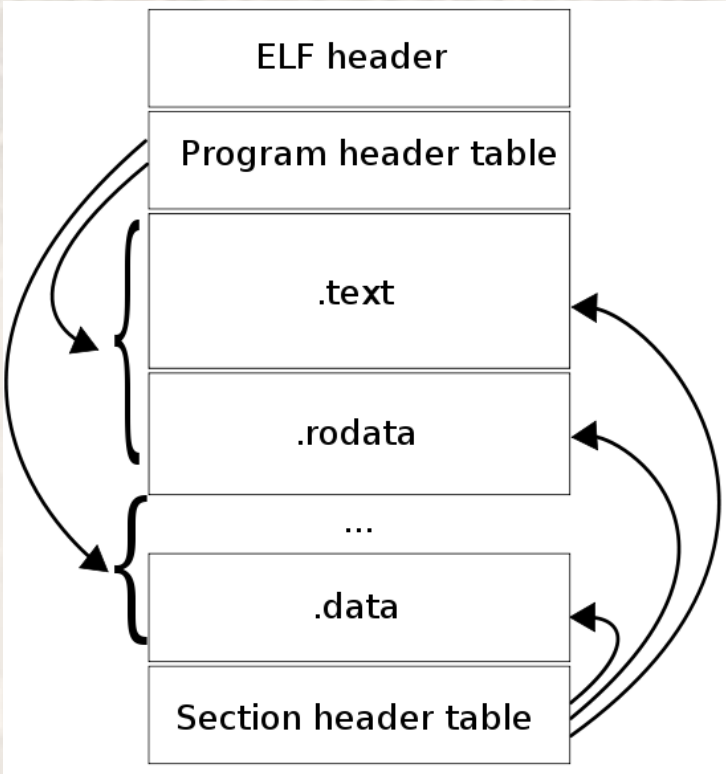
```
nbatfai@morse:~$ ar -t /usr/lib/libc.a |grep "\(strncpy\|printf\)"|more
vfprintf.o
vprintf.o
```

Részletesen lásd majd a 3. előadást!

ELF

(Executable and Linkable Format)

```
nbatfai@morse:~$ gcc szohossz.c -o szohossz
nbatfai@morse:~$ file szohossz
szohossz: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linke
d (uses shared libs), for GNU/Linux 2.6.15, not stripped
```



http://en.wikipedia.org/wiki/File:Elf-layout--en.svg

```
nbatfai@morse:~$ objdump -d -j .data szohossz

szohossz:     file format elf64-x86-64


Disassembly of section .data:

0000000000601020 <alma>:
  601020:       78 56 34 12
```

```
nbatfai@morse:~$ more szohossz.c
#include <stdio.h>
int alma=0x12345678;
int
main(void)
```

Részletesen lásd majd a 3. előadást!

Program a memóriában

```
int main (int argc, char** argv, char** env)
{...
```

PP 272

Környezeti változók

```
int main (int argc, char** argv, char** env)
{...
```

Parancssor argumentumok

```
int main (void)
{
 int a;
 char* p;
```

Lokális változók, paraméterátadás.

Dinamikusan foglalt terület

```
// A számolást végző adott számú gyermekfolyamat létrehozása
proc_pid = (int *) malloc (procsz * sizeof (int));
```

PP 236

```
int g = 16;
int main (void)
{
```

Inicializált adatok (statikus és globális)

```
int g2;
char *p2;
int main (void)
{
 static int si;
```

Inicializálatlan adatok (statikus és globális)

```
Z *= 2;
```

Kód

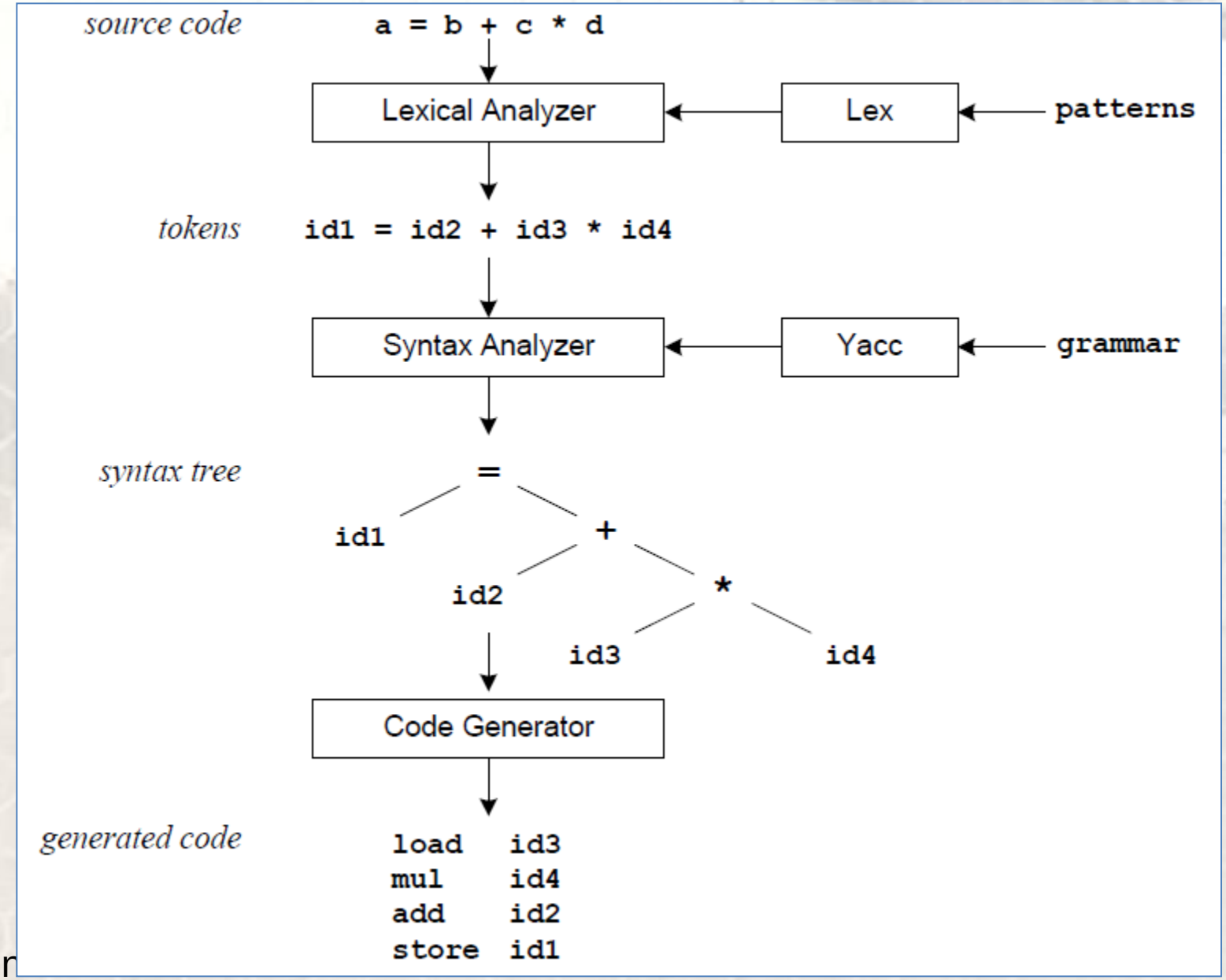
```
int main (void)
{
 const int ci = 16;
 printf(„Hello, Vilag!”);
```

Informatikai analízis

a) Karakterkészlet
b) Lexikális egységek (pl. azonosító, kulcsszó, megjegyzés, literálok) - scanner: LEX, FLEX
c) Szintaktikus egységek - parser: YACC, BISON
d) Utasítások (pl. értékadó, ugró, üres, szelekció, iteráció)
e) Program egységek (pl. blokk, függvény)

A C nyelv esetén lásd a K&R könyv „C referencia-kézikönyv” fejezetét!
Lex, Flex, Yacc, Bison kapcsán lásd: http://bme.ysolt.net/4_felev/Info2/Jegyzet/fony_jegyz

LEX (Lexical Analyser Generator)



Tom Niemann
Portland, Oregon

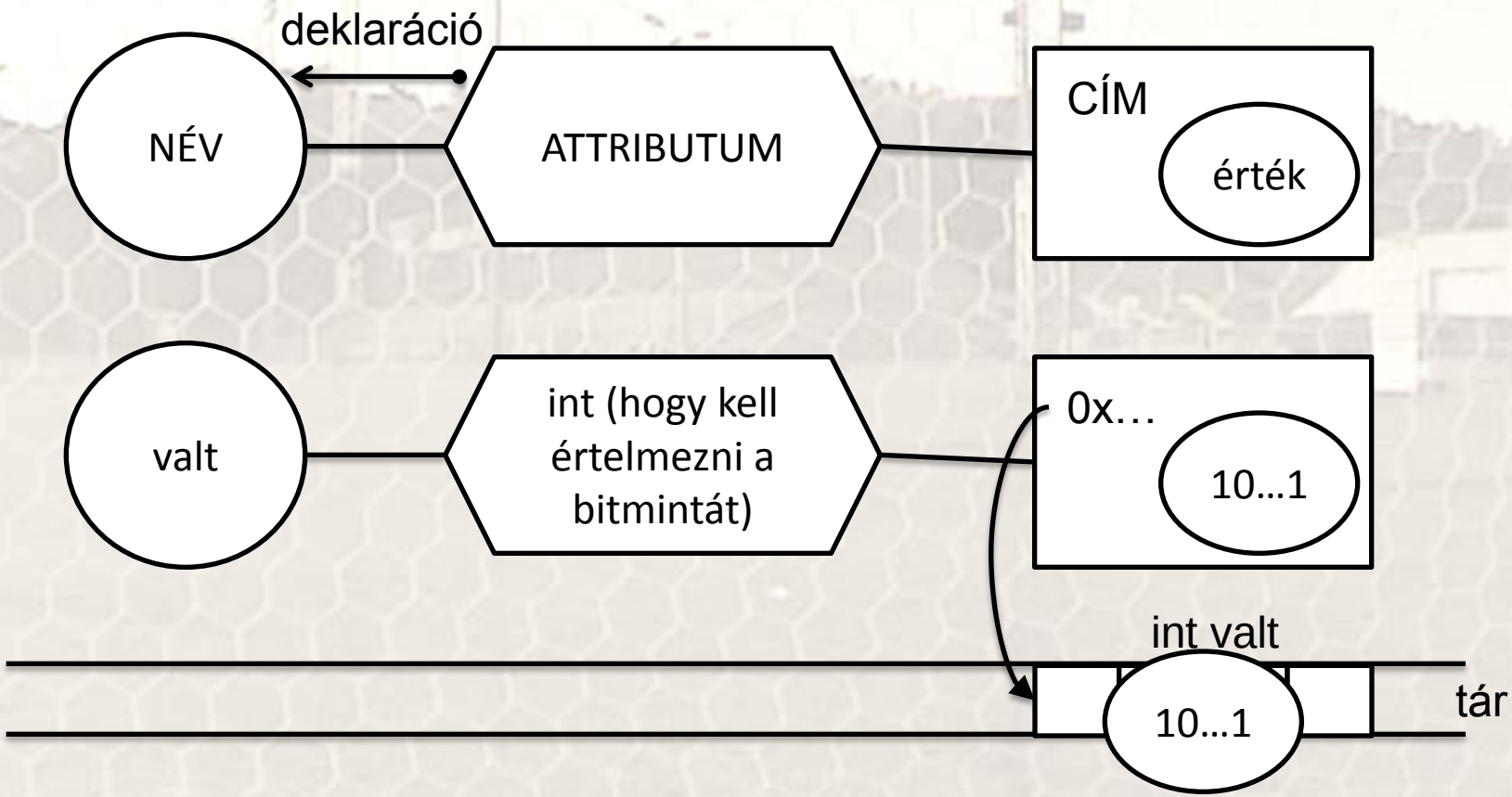
epaperpress.com, http://epaperpress.com/lexandyacc/download/lexyacc.pdf 4.o.

Első saját lexikális elemző

```
%{
  #include <string.h>
  int betuk_szama = 0, szavak_szama = 0, szamok_szama = 0, sorok_szama = 0;
%}
%%
.                       ++betuk_szama;
\n                      ++sorok_szama;
[0-9]+                  {++szavak_szama; ++szamok_szama,
                          printf("szam=[%s]", yytext);
                          betuk_szama += strlen(yytext);}
[a-zA-Z][a-zA-Z0-9]*    {++szavak_szama; betuk_szama += strlen(yytext);}
%%
int
main ()
{
  yylex ();
  printf ("%d betu %d szo %d szam %d sor\n", betuk_szama, szavak_szama,
          szamok_szama, sorok_szama);

  return 0;
}
```

```
nbatfai@hallg:~/c$ lex -o lexikalis.c lexikalis.l
nbatfai@hallg:~/c$ gcc lexikalis.c -o lexikalis -lfl
nbatfai@hallg:~/c$ ./lexikalis
alma 55 5alma
szam=[55]alma 55 5alma
szam=[55]26 betu 6 szo 2 szam 2 sor
nbatfai@hallg:~/c$
```

Informatikai analízis: a változó



Az ábra a szerző Juhász István 1993 évi Programozás 1 előadásán készített előadásjegyzete alapján készült.

C típusok

char, short, int, long,
unsigned, float, double.

```
nbatfai@morse:~$ gcc tipusok.c -o tipusok
nbatfai@morse:~$ ./tipusok
8
16
32
64
32
32
64
64
```

```c
#include <stdio.h>
#include <limits.h>
int main()
{
    printf("%lu\n", sizeof(char) * CHAR_BIT);
    printf("%lu\n", sizeof(short) * CHAR_BIT);
    printf("%lu\n", sizeof(int) * CHAR_BIT);
    printf("%lu\n", sizeof(long) * CHAR_BIT);
    printf("%lu\n", sizeof(unsigned) * CHAR_BIT);
    printf("%lu\n", sizeof(float) * CHAR_BIT);
    printf("%lu\n", sizeof(double) * CHAR_BIT);
    printf("%lu\n", sizeof(char*) * CHAR_BIT);
return 0;
}
```

man limits.h:

```
{CHAR_BIT}
       Number of bits in a type char.
       Value: 8
```

Lásd még a K&R könyv „C referencia-kézikönyv/Típus-specifikátorok” fejezetét!

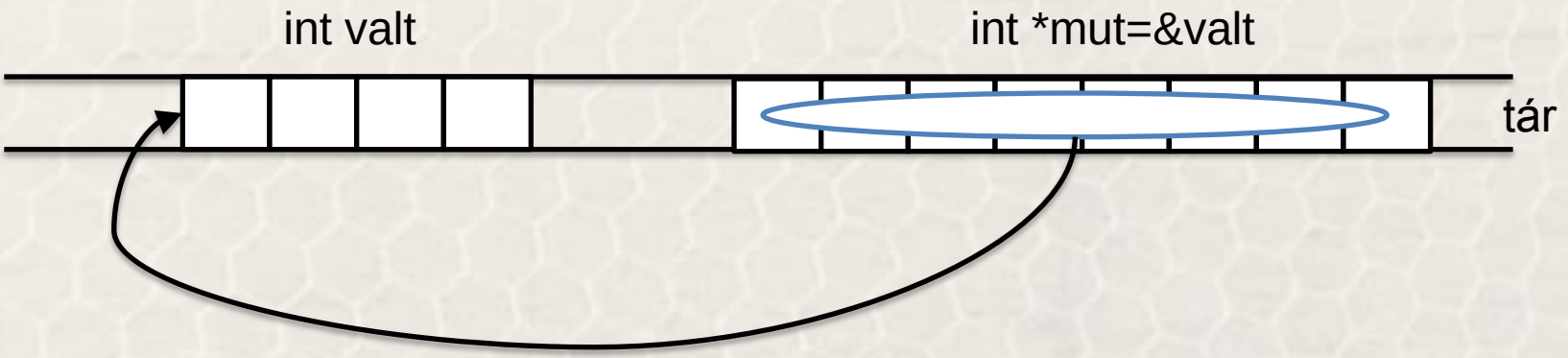
C deklaráció

```
int valt, *mut, tomb[5], *muttomb[5], tomb2d[5][5], *fgv(), (*fgvmut)();
```

```
int valt;           // a valt egy egész
int *mut;           // a mut egy egészre mutató mutató
int tomb[5];        // a tomb egészek tömbje
int *muttomb[5];        // a muttomb egy mutató tömb, 5 darab
                        // egészre mutató mutató van benne
int tomb2d[3][3];       // a tomb2d egy két dimenziós tömb
int *fgv();         // az fgv függvény egészre mutató mutatót ad
                    // vissza
int (*fgvmut)();        // az fgvmut egy egészet visszaadó
                        // függvényre mutató mutató
```

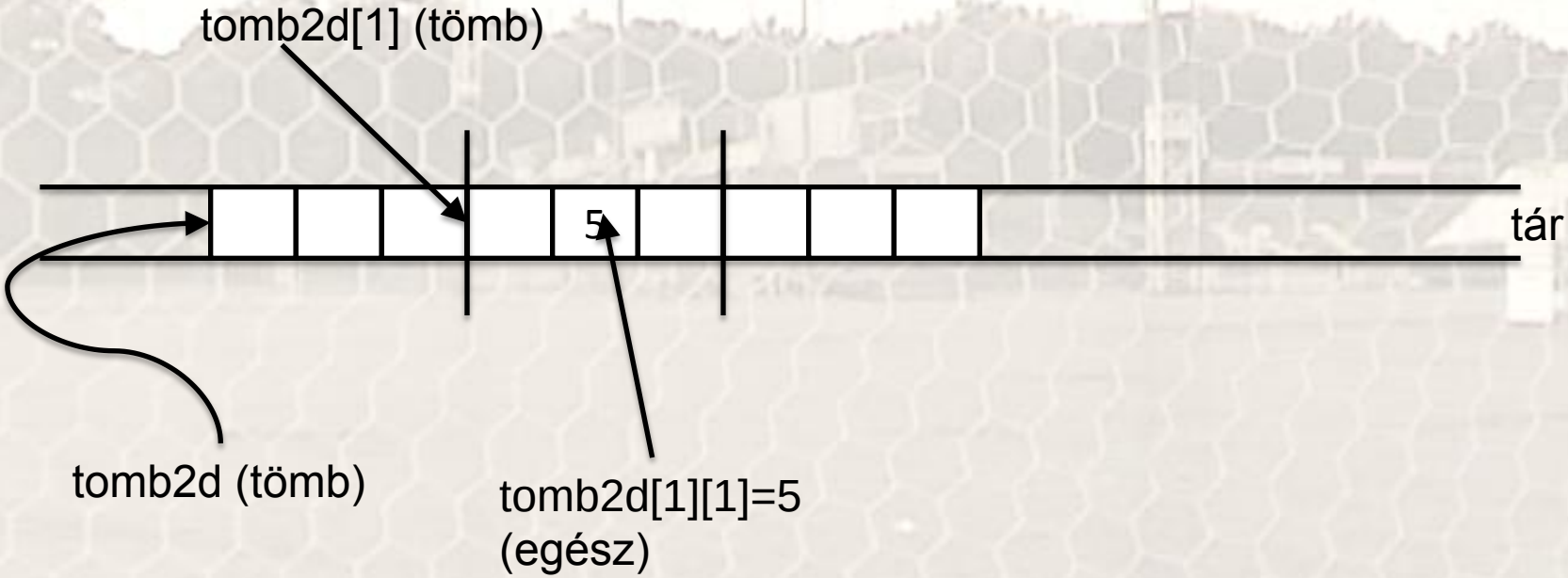
C deklaráció, a mutató

```
int valt;              // a valt egy egész
int *mut=&valt;        // a mut egy egészre mutató mutató
```

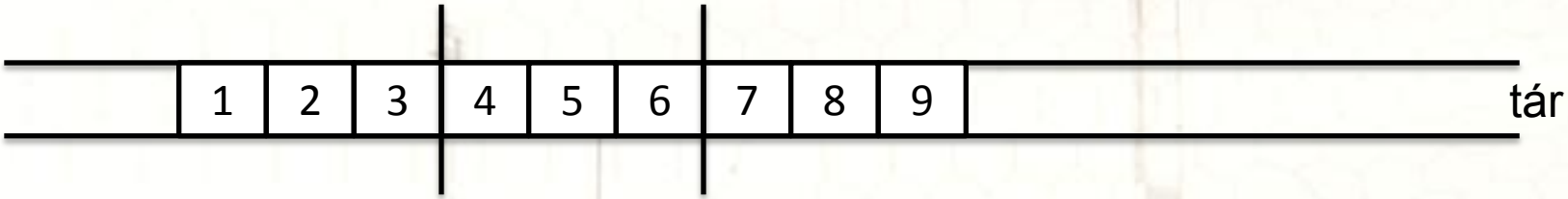


C deklaráció, a tömb

int tomb2d[3][3]; // a tomb2d egy két dimenziós tömb



Tömb inicializálása




```c
#include <stdio.h>

int
main (void)
{
  int i, j, tomb2d[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
  };

  for (i = 0; i < 3; ++i)
    {
      for (j = 0; j < 3; ++j)
        printf ("[%d][%d]=%d ", i, j, tomb2d[i][j]);

      printf ("\n");
    }
  return 0;
}
```

```
nbatfai@hallg:~/c$ gcc t.c -o t
nbatfai@hallg:~/c$ ./t
[0][0]=1 [0][1]=2 [0][2]=3
[1][0]=4 [1][1]=5 [1][2]=6
[2][0]=7 [2][1]=8 [2][2]=9
```

C deklaráció

int (*fgvmut)(); // az fgvmut egy egészet visszaadó
// függvényre mutató mutató

```
PTHREAD_CREATE(P)          POSIX Programmer's Manual          PTHREAD_CREATE(P)



NAME
       pthread_create - thread creation

SYNOPSIS
       #include <pthread.h>

       int pthread_create(pthread_t *restrict thread,
              const pthread_attr_t *restrict attr,
              void *(*start_routine)(void*), void *restrict arg);
```

void *(*start_routine)(void*) ez mit deklarál akkor?

(A „void *” akármire mutathat.)

C deklaráció

void *(*start_routine)(void*) // a start_routine egy void*-ot
// váró, void *-ot visszaadó
// függvényre mutató mutató

```c
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
void *szal(void *id)
{
    printf("Szal: %d, %d\n", *(int *)id, pthread_self());
    *(int *)id *= 2;
    return id;
}
int
main(void)
{
    pthread_t sz1, sz2;
    int s1 = 1, s2 = 2, *r;
    if(pthread_create(&sz1, NULL, szal, (void *)&s1))
        exit(-1);
    if(pthread_create(&sz2, NULL, szal, (void *)&s2))
        exit(-1);
    pthread_join(sz1, (void *) &r);
    printf("%d\n", *r);
    pthread_join(sz2, (void *) &r);
    printf("%d\n", *r);
    return 0;
}
```

void, void *

```c
#include <stdio.h>
int
main(void)
{
    int valt;
    double lebego;

    void *akarmi = &valt;

    *(int *)akarmi = 4;
    printf("%d\n", *(int *)akarmi);

    akarmi = &lebego;

    *(double *)akarmi = 5.0;
    printf("%f\n", *(double *)akarmi);

    return 0;
}
```

```c
int
void1() // C tetsz, C++ void
{
}

int
void2(void)
{
}

…

void1();
void2();

void1(5);
//void2(5);
```

Mi történik, ha itt g++-al fordítasz?

typedef deklaráció

```
typedef int EGESZ, *EGESZP;

EGESZ egesz;                // egesz egy egész
EGESZM egeszmut;            // egeszmut egy egészet
                            // megcímző mutató
```

C deklaráció

```
void (*BADBOY)();   // a BADBOY egy olyan függvényre
                    // mutató mutató, ami nem kap és nem
                    // ad vissza értéket
```

```
typedef void (*BADBOY)();

BADBOY badboy;

...

try_one_crash()
{if (nbytes > 0)
    (*badboy)();
 else if (nbytes == 0)
   while(1);}
```

A forráscsipet: http://packages.debian.org/source/sid/crashme

C deklaráció

```
typedef void (*sighandler_t)(int);
                // a sighandler_t egy olyan függvényre
                // mutató mutató, ami egy egészet kap és nem
                // ad vissza értéket
```

```
SIGNAL(2)                    Linux Programmer's Manual



NAME
       signal - ANSI

SYNOPSIS
       #include <sig

       typedef void

       sighandler_t
```

```c
#include <stdio.h>
#include <signal.h>
void utolso_tennivalo(int sig)
{
    printf("Utolso tennivalo kesz, immar kilephetek\a\n");
    exit(0);
}
int
main(void)
{
    if(signal(SIGINT, utolso_tennivalo) == SIG_IGN)
        signal(SIGINT, SIG_IGN);
    for(;;)
        putchar(getchar());
    return 0;
}
```

C kifejezések

Kifejezéseket rekurzívan építünk fel: a kifejezések egyszerűbb kifejezésekből épülnek fel zárójelek és operátorok segítségével, a rekurziót tipikusan lexikális egységek zárják le.

```
printf("%d\n", sizeof(char) * 8);
```

printf – azonosító
"%d\n"– karakterlánc
8 – állandó
sizeof, char – kulcsszó

Lásd még a K&R könyv „C referencia-kézikönyv/Kifejezések” fejezetét!

C kifejezések

```
printf("%d\n", sizeof(char) * 8);
```

printf – azonosító
"%d\n"– karakterlánc
8 – állandó
sizeof, char – kulcsszó

0 <elsődleges_kifejezés>(<kif_lista>)

1 printf(<kif_lista>)

2 printf(<kif_lista>, <kifejezés>)

3 printf(<kifejezés>, <kifejezés>)

4 printf("%d\n", <kifejezés>)

…
<kifejezés>::=
0 <elsődleges_kifejezés>([<kif_lista>])
…
<elsődleges_kifejezés>
…
* <kifejezés>
<sizeof kifejezés>

…
<elsődleges_kifejezés>::=
…
1 <azonosító>
4 <karakterlánc>
<állandó>

<kif_lista>::=
3 <kifejezés>
2 <kif_lista>, <kifejezés>

Lásd még a K&R könyv „C referencia-kézikönyv/Kifejezések” fejezetét!
„A függvényhívás olyan elsődleges kifejezés, amelyet”

C kifejezések

Kifejezéseket rekurzívan építünk fel: a kifejezések egyszerűbb kifejezésekből épülnek fel zárójelek és operátorok segítségével, a rekurziót tipikusan lexikális egységek zárják le.

```
printf("%d\n", sizeof(char) * 8);
```

printf – azonosító
"%d\n"– karakterlánc
8 – állandó
sizeof, char – kulcsszó

```
…
kifejezés::=
        elsődleges_kifejezés( [kif_lista] )
        …
        elsődleges_kifejezés
        …
        sizeof kifejezés
…
elsődleges_kifejezés::=
        …
        azonosító
        karakterlánc
        állandó
kif_lista::=
        kifejezés
        kif_lista, kifejezés
```

Lásd még a K&R könyv „C referencia-kézikönyv/Kifejezések” fejezetét!

C kifejezések

printf("%d\n", sizeof(char) * 8);

Jelen esetben (függvényhívás) talán nem érezhető triviálisan a kifejezés típusa: a hívás eredménye típusú.

```
#include <stdio.h>

int
main (void)
{

  printf ("%d\n", printf ("%d\n", printf ("%d\n", sizeof (char) * 8)));
  return 0;
}
```

Mit fog kiírni?

C utasítások

A K&R könyv „C referencia-kézikönyv/Utasítások” fejezetében az *összetett_utasítás* a blokk.

<utasítás>::=
<összetett_utasítás>
<kifejezés> ; (alma = 5;)
if(<kifejezés>) <utasítás>
…
for…

…
goto <azonosító> ;
<azonosító> : <utasítás>
; (üres utasítás)

Lásd még a K&R könyv „C referencia-kézikönyv/Utasítások” fejezetét!

C utasítások

a) Üres
   ;
b) Kifejezés utasítás
   ++i;
   printf(”Helló, Világ!”);
   a=0;
c) Blokk
d) Feltételes
   if
   if else
e) Ciklusszervező
   for, while, do while
f) break, continue, return
g) Goto, címkézett utasítás
h) Többszörös elágaztató
   switch

Lásd még a K&R könyv „C referencia-kézikönyv/Utasítások” fejezetét
és természetesen a laborok példaprogramjait!

C utasítások

A)
```
for(i=0; i<3; ++i)
   printf("%d ", i);
```

B)
```
for(i=0; i<3; i++)
   printf("%d ", i);
```

```
if(signal(SIGINT, jelkezelo) == SIG_IGN)
   signal(SIGINT, SIG_IGN);
```

```
if(signal(SIGINT, SIG_IGN) != SIG_IGN)
   signal(SIGINT, jelkezelo);
```

```
SIGNAL(2)                  Linux Programmer's Manual                  SIGNAL(2)

NAME
       signal - ANSI C signal handling

SYNOPSIS
       #include <signal.h>

       typedef void (*sighandler_t)(int);

       sighandler_t signal(int signum, sighandler_t handler);

DESC
RETURN VALUE
       signal()  returns  the previous value of the signal handler, or SIG_ERR
       on error.
```

Mi a különbség a két ciklus között? Mi a különbség a két feltétel között? (beugratós)

C programegységek

Blokk (skatulyázható)
Függvény (nem)

```c
int szamlalo = 0;
void
var (void)
{
  int i, r = 1 + (int) (10000.0 * rand () / (RAND_MAX + 1.0));
  for (i = 0; i < r; ++i);
}

void *
novel_szal (void *id)
{
  int i;
  for (i = 0; i < 100; ++i)
    {
      printf ("Szal: %d, %d\n", *(int *) id, pthread_self ());
      fflush (stdout);
      var ();
      szamlalo = szamlalo + 1;
    }
  return id;
}
```

Lyuk a hatáskörben

```c
#include <stdio.h>

int
main (void)
{
  int blokkban = 0;

  {
    int blokkban = 100;
    printf ("%d\n", blokkban);

    {
      int blokkban = 10000;
      printf ("%d\n", blokkban);
      blokkban = blokkban + 1;
      printf ("%d\n", blokkban);
    }

    printf ("%d\n", blokkban);

  }

  printf ("%d\n", blokkban);

  return 0;
}
```

Mit ír ki ez a program?

```
100
10000
10001
100
0
```

Függvények

A problémamegoldás módszere, illetve újrafelhasználhatóság.

Neve
Opc. formális paraméterei
Törzse
Opc. környezete
Opc. lokális változói

Formális paraméter lista

```
double
tavolsag (double PR[], double PRv[], int n)
{
  double osszeg = 0.0;
  int i;

  for (i = 0; i < n; ++i)
    osszeg += (PRv[i] - PR[i]) * (PRv[i] - PR[i]);

  return sqrt (osszeg);
}
```

(koordináták különbségeinek négyzetösszegéből vont négyzetgyök)

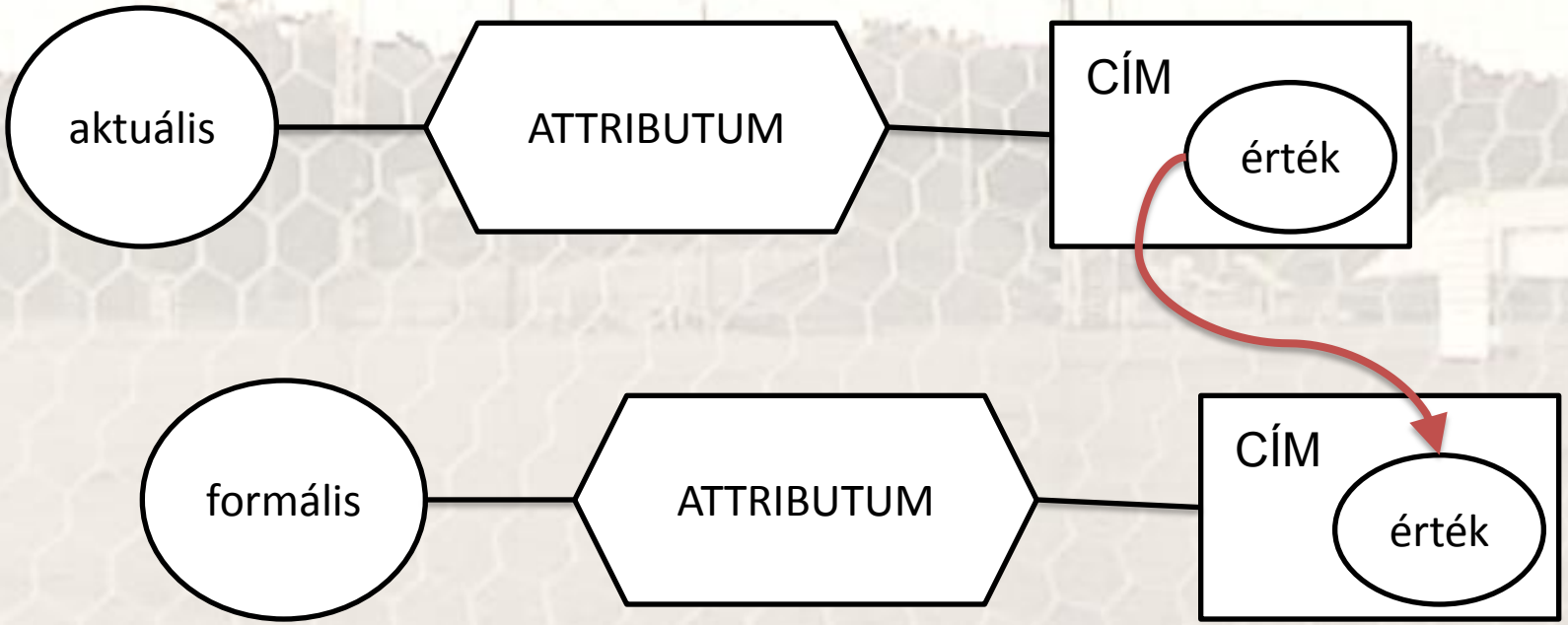
```
for (;;)
  {

    for (i = 0; i < 4; ++i)
      {
        PR[i] = 0.0;
        for (j = 0; j < 4; ++j)
          PR[i] += (L[i][j] * PRv[j]);
      }
    if (tavolsag (PR, PRv, 4) < 0.00000001)
      break;

    for (i = 0; i < 4; ++i)
      PRv[i] = PR[i];

  }
```

Aktuális paraméter lista

http://progpater.blog.hu/2011/02/13/bearazzuk_a_masodik_labort

Informatikai analízis: a paraméterátadás

Érték szerinti, a formális paraméter változtatása hatástalan az aktuálisra nézve.



(Ha viszont mutatót adunk át, akkor a formális paraméteren keresztül történő hozzáférés meg tudja változtatni az aktuális paramétert!)

Az ábra a szerző Juhász István 1993 évi Programozás 1 előadásán készített előadásjegyzete alapján készült.

Szörnyűségek

```
int
f (int x, int y)
{
  return x + y;
}

int
main (void)
{
  int a = 0;
  printf ("%d %d\n", f (a, ++a), f (++a, a));
  a = 0;
  printf ("%d %d\n", f (++a, a), f (a, ++a));
  return 0;
}
```

Formális paraméter lista

Aktuális paraméter lista

4 2
4 2

1

Tgyfh., 1 -el kezdi, azaz hátulról, akkor tgyfh. ezen belül is hátulról, akkor
(,) (1, 0) aktuálisokkal hívja de akkor nem adna 2-t vissza vagy akkor most elölről nézzük:
(,)(1,1)-el? …

A függvényargumentumok kiértékelési sorrendje nincs meghatározva!

Ne írjunk olyan kódot, ami bármilyen értelemben feltételez egy kiértékelési sorrendet! – hogy ne kelljen gyógyszerészünket, vagy orvosunkat kérdezni.

(A paraméterek kiértékelésének sorrendjét a C nyelv nem határozza meg!)

Szörnyűségek

```c
int
f (int x, int y)
{
  return x + y;
}

int
main (void)
{
  int a = 0;
  printf ("%d %d\n", f (a, ++a), f (++a, a));
  a = 0;
  printf ("%d %d\n", f (++a, a), f (a, ++a));
  return 0;
}
```

```
[norbi@sgu ~]$ splint kiert.c
Splint 3.1.2 --- 22 Aug 2009

kiert.c: (in function main)
kiert.c:13:28: Argument 2 modifies a, used by argument 1 (order of evaluation
                  of actual parameters is undefined): f(a, ++a)
  Code has unspecified behavior. Order of evaluation of function parameters or
  subexpressions is not defined, so if a value is used and modified in
  different places not separated by a sequence point constraining evaluation
  order, then the result of the expression is unspecified. (Use -evalorder to
  inhibit warning)
```

Szörnyűségek

```
int
f (int a)
{
  return ++a;
}

int
main (void)
{
  int a = 0;
  printf ("%d %d\n", a, f (a));
  return 0;
}
```

0 1

```
int
f (int *a)
{
  return ++ * a;
}

int
main (void)
{
  int a = 0;
  printf ("%d %d\n", a, f (&a));
  return 0;
}
```

1 1

```
int
f (int *a)
{
  return ++ * a;
}

int
main (void)
{
  int a = 0;
  printf ("%d %d\n", f (&a), a);
  return 0;
}
```

1 0

Ne írjunk olyan kódot, ami bármilyen értelemben feltételez egy kiértékelési sorrendet! – hogy ne kelljen gyógyszerészünket, vagy orvosunkat kérdezni.

```
main (void)
{
  int a = 0;
  printf ("%d %d\n", f (&a), f (&a));
  return 0;
}
```

1

Szörnyűségek

```
int
f  (int  a)
{
  return ++a;
}

int
main (
{
  int  a  =  0;
```

```
[norbi@sgu ~]$ splint sz1.c -exportlocal
Splint 3.1.2 --- 22 Aug 2009

Finished checking --- no warnings
[norbi@sgu ~]$
```

```
int
f  (int  *a)
{
  return ++ * a;
}

int
main  (void)
{
  int  a  =  0;
```

Frama-C

File Project Analyses Help

Source file	Occurrence	Impact
› stdio.h		
⌄ sz2.c		
main		
f		

```
int main(void)
{
 int a ;
 int tmp ;
 int __retres ;
 a = 0;
 tmp = f(& a);
 printf((char const   *)"%d %d\n",tmp,a);
 __retres = 0;
 return (__retres);
}
```

sz2.c

```
 1 #include <stdio.h>
 2
 3 int
 4 f (int *a)
 5 {
 6  return ++ * a;
 7 }
 8
 9 int
10 main (void)
11 {
12  int a = 0;
13  printf ("%d %d\n", f (&a), a);
14  return 0;
15 }
16
```

Szörnyűségek

```c
#include <stdio.h>

int
novel_eggyel (int *a)
{
  return ++ * a;
}

int
csokkent_eggyel (int *a)
{
  return -- * a;
}


int
main (void)
{
  int a = 0;
  printf ("%d\n", novel_eggyel (&a) + csokkent_eggyel (&a));

  a = 0;
  printf ("%d\n", csokkent_eggyel (&a) + novel_eggyel (&a));

  return 0;
}
```

1
-1

Szörnyűségek

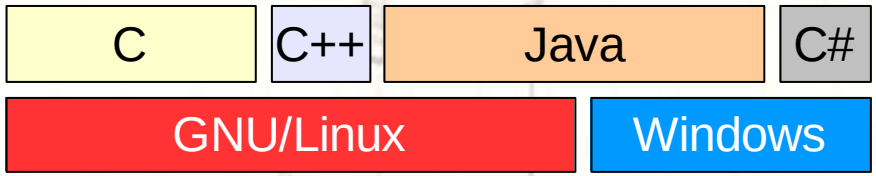
```c
#include <stdio.h>

int
main (void)
{
  int i;
  int tomb[5];

  for (i = 0; i < 5; tomb[i] = i++);

  for (i = 0; i < 5; ++i)
    printf ("%d\n", tomb[i]);

  return 0;
}
```

```
0
1
2
3
4
```

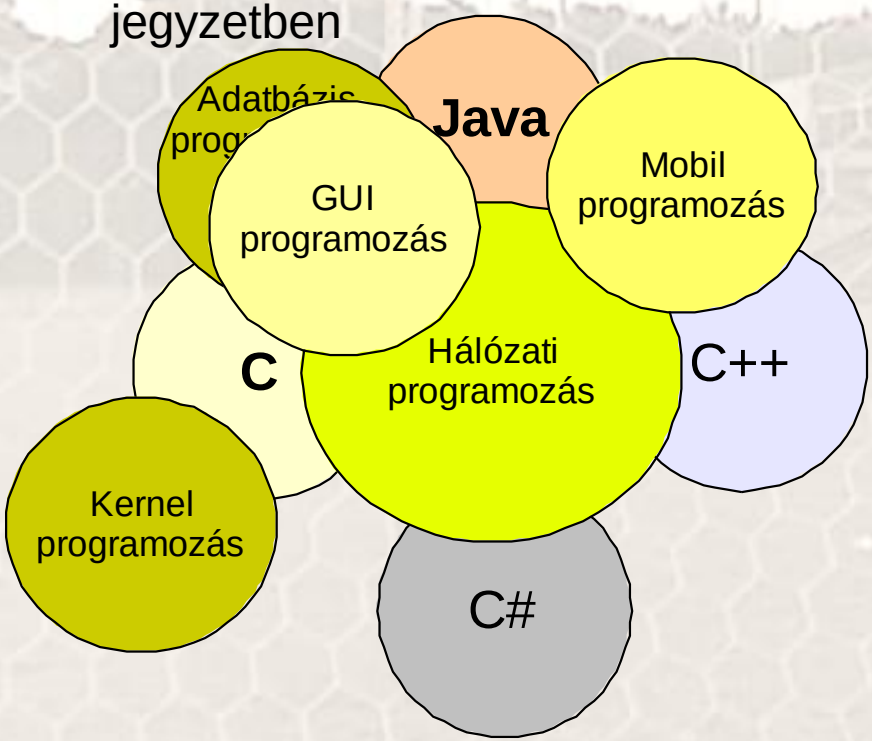
Labor

Az első néhány laboron mindenképpen egyszerű szövegszerkesztőt és parancssort használjunk! A PP javasolta manuál lapokat mindig nyissuk ki, nézzük meg!

Programozó Páternoszter



Platformok és nyelvek a jegyzetben



Programozási témák a jegyzetben

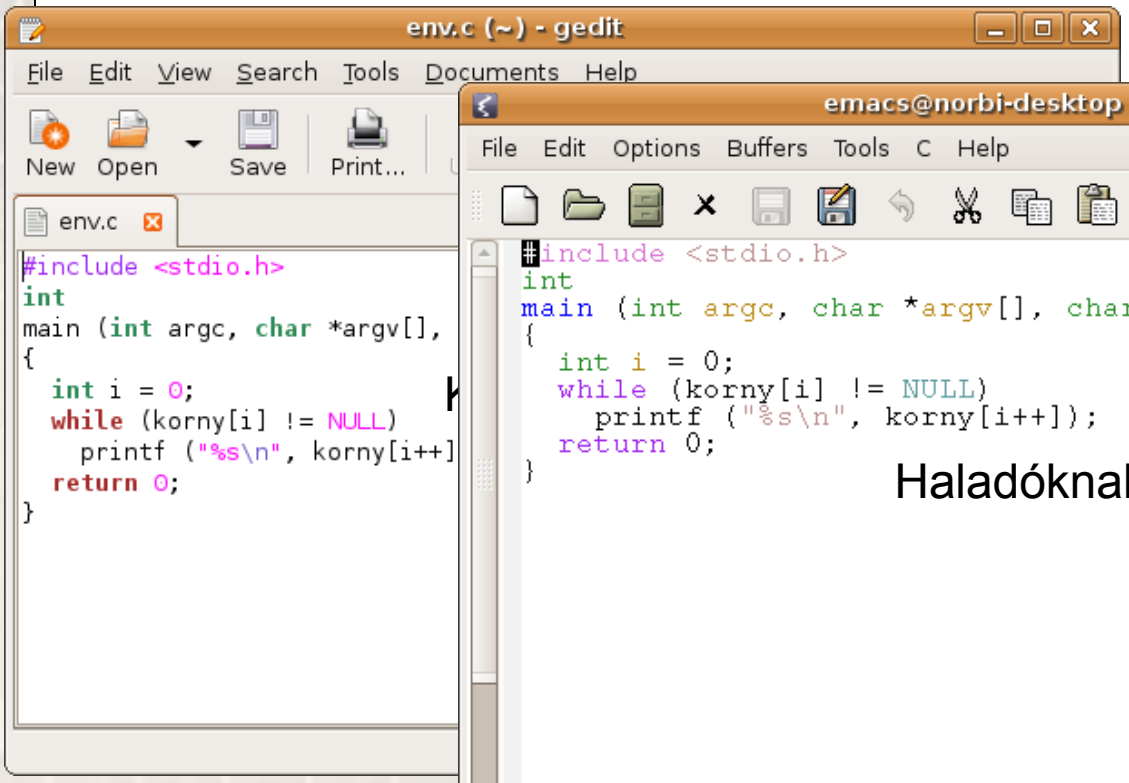
http://www.inf.unideb.hu/~nbatfai/#pp

PP 17

Az oldalszámok a 0.0.247 verzióra vonatkoznak.

Az első laborokon: parancssor

```
$ joe env.c
vagy például
$ vi env.c
Páternoszterből forráskód kivágása és
ide beillesztése...
$ indent env.c              Hogy szép legyen a forrás tördelése
$ gcc env.c -o env  Futtatható fájl készítése a forrásból
$ ./env                              Az elkészített futtatható futtatása
SHELL=/bin/bash
```



Haladóknak

r *korny[])

PP 272

Manuál (kézikönyv) lapok

```
$ man init

INIT(8)                    Linux Rendszergazda Kézikönyve                    INIT(8)

NÉ

ÁT

LE

FUTÁS
       A futási szint egy szoftver konfiguráció, amely csak egy meghatározott
       processz  csoport  létezését  engedi  meg. Az, hogy az init milyen pro-
       cesszeket hozzon létre egy adott futási szinten a /etc/inittab  fájlban
       van  definiálva.  Az  init nyolc futási szinten lehet: 0-6 és S vagy s.
       Futási szintet úgy válthatunk, ha egy  privilegizált  felhasználó  fut-
       tatja a
```

```
$ man w

W(1)                       Linux felhasználói kézikönyv                       W(1)



NÉV
       w - Megmutatja, hogy ki van belépve és mit csinál.

ÁTTEKINTÉS

LEÍ
```

```
$ w

 13:12:14 up  1:00,  4 users,  load average: 0,99, 0,58, 0,25
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
norbi    tty7     :0               12:12    0.00s  1:07m  0.18s /usr/bin/gnome-
norbi    pts/0    :0.0             12:12   18:05m  0.52s  0.52s bash
norbi    pts/1    :0.0             12:42    0.00s  0.48s  0.00s w
norbi    pts/2    :0.0             13:07    3:56m 21.20s  0.14s make
```

A kézikönyv szervezése

1.szint – **Felhasználói parancsok**
man w
2.szint – **Rendszerhívások**
man 2 read
3.szint – **Könyvtári függvények**
man 3 printf
4.szint – **Eszközök**
man 4 stdin
5.szint – **Formátumok, protokollok**
man 5 passwd
6.szint – **Játékok**
man 6 fortune
7.szint – **Egyéb**
man 7 hier
8.szint – **Rendszeradmin**
man 8 reboot

Tipp: hogyan írhatok saját manuál lapot fejlesztett programomhoz?
Nézz meg egy példát és írd át!
http://packages.debian.org/source/sid/crashme

A kézikönyv szervezése

W(1) FreeBSD General Commands Manual W(1)

READ(2) FreeBSD System Calls Manual READ(2)

PRINTF(3) FreeBSD Library Functions Manual PRINTF(3)

FD(4) FreeBSD Kernel Interfaces Manual FD(4)

PASSWD(5) FreeBSD File Formats Manual PASSWD(5)

FORTUNE(6) FreeBSD Games Manual FORTUNE(6)

HIER(7) FreeBSD Miscellaneous Information Manual HIER(7)

REBOOT(8) FreeBSD System Manager's Manual REBOOT(8)

NAME

reboot, halt, fastboot, fasthalt -- stopping and restarting the system

SYNOPSIS

halt [-lnpq] [-k kernel]
reboot [-dlnpq] [-k kernel]
fasthalt [-lnpq] [-k kernel]
fastboot [-dlnpq] [-k kernel]

DESCRIPTION

The halt and reboot utilities flush the file system cache to disk, send all running processes a SIGTERM (and subsequently a SIGKILL) and, respectively, halt or restart the system. The action is logged, including entering a shutdown record into the wtmp(5) file.

The options are as follows:

olvasva = read (kapu, buffer, BUFFER_MERET)

```
$ man 2 read
READ(2)                    FreeBSD System Calls Manual                    READ(2)

NAME
      r

LIBRAR
      S

SYNOPS
      #
      #
      #

      s
      r

 ...

DESCRI
      T
      r
...

STANDA
      T
      (

...
```

```
$ man 2 read

READ(2)                    Linux Programmer's Manual                    READ(2)



NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
       read()  attempts to read up to count bytes from file descriptor fd into
       the buffer starting at buf.

       If count is zero, read() returns zero and has  no  other  results.   If
       count is greater than SSIZE_MAX, the result is unspecified.

RETURN VALUE
       On success, the number of bytes read is returned (zero indicates end of
       file), and the file position is advanced by this number.
...

CONFORMING TO
       SVr4, 4.3BSD, POSIX.1-2001.
...
```

007

```
SHADOW(5)                File Formats and Conversions                SHADOW(5)

NAME
       shadow - encrypted password file

DESCRIPTION
       shadow contains the encrypted password information for user's accounts
       and optional the password aging information. Included is:

       -  login name

       -  encrypted password

...
```

Bebootolunk egy telepítő lemezről vagy éppen (az első előadásban említett) Gparted Live CD-nkről és

```
mkdir /mnt/norbi
mount /mnt/hda4 /mnt/norbi

vi /mnt/norbi/etc/shadow
root:$1$wZ3bMDHK$Xogj2CHjy4.o3MEB2nhp00:13905:0:99999:7:::

umount /mnt/norbi
```

Beboot és a matyi2006 jelszóval root-ok vagyunk!

Karakterhegyezés

Mire utal a const a paraméterátadásnál, például az alábbi *man 3 strcpy* manuál lapon?

STRCPY(3) Linux Programmer's Manual STRCPY(3)

NAME

strcpy, strncpy - copy a string

SYNOPSIS

#include <string.h>

char *strcpy(char *dest, const char *src);

char *strncpy(char *dest, const char *src, size_t n);

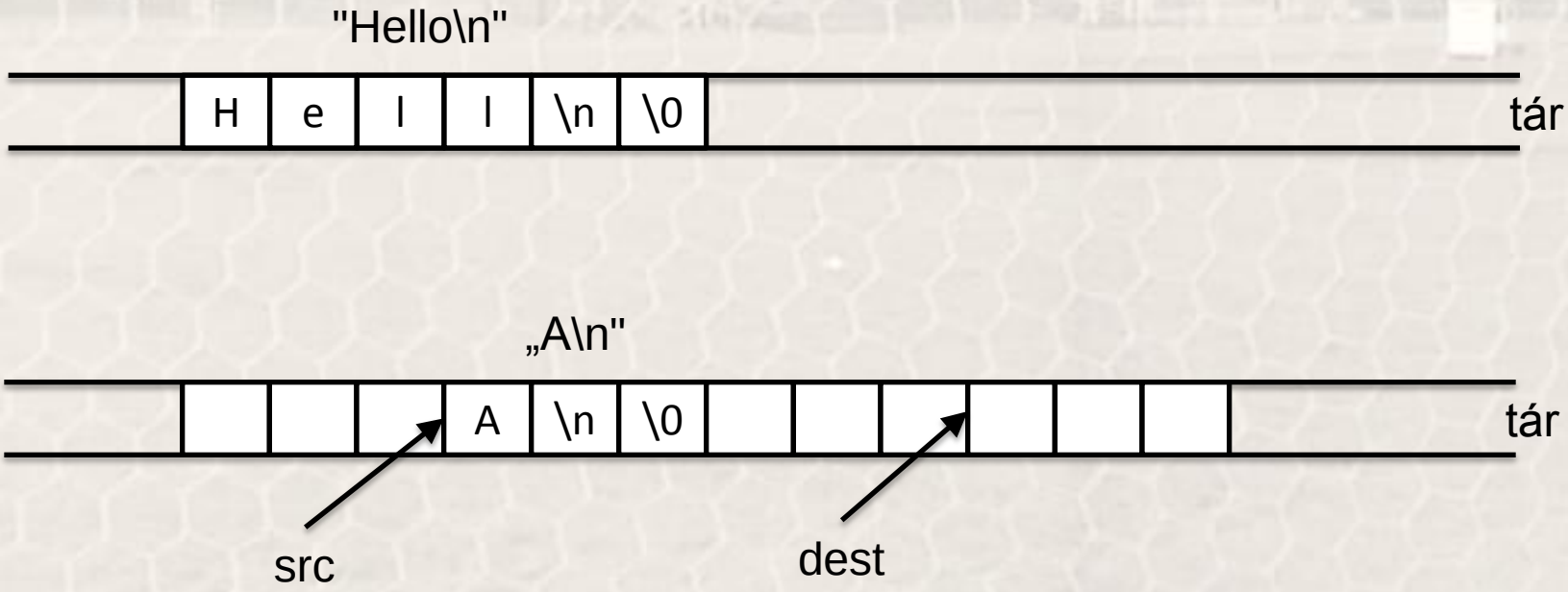
A simple implementation of strncpy() might be:

```
char*
strncpy(char *dest, const char *src, size_t n){
    size_t i;

    for (i = 0 ; i < n && src[i] != '\0' ; i++)
        dest[i] = src[i];
    for ( ; i < n ; i++)
        dest[i] = '\0';

    return dest;
}
```

Karakterhegyezés

A simple implementation of strncpy() might be:

```
char*
strncpy(char *dest, const char *src, size_t n){
    size_t i;

    for (i = 0 ; i < n && src[i] != '\0' ; i++)
        dest[i] = src[i];
    for ( ; i < n ; i++)
        dest[i] = '\0';

    return dest;
}
```



Karakterhegyezés

```c
#include <string.h>
#include <stdio.h>

#define MERET 5

char buffer[MERET];

int
main ()
{
  strncpy (buffer, "aaaaaaaaaaaaaaaaaaaaaaaaaaaa", MERET);
  printf ("%s\n", buffer);
  strncpy (buffer, "bbbbbbbbbbbbbbbbbbbbbbbbbbb", MERET - 1);
  printf ("%s\n", buffer);

  return 0;
}
```

```
aaaaa
bbbba
```

Helló, Világ! - printf

```
#include <stdio.h>

int
main ()
{

  printf ("Hello, Vilag!");

  return 0;
}
```

Könyvtári függvénnyel

```
$ gcc printfhello.c -o printfhello
$ ./printfhello
Hello, Vilag!
```

```
PRINTF(3)                 Linux Programmer's Manual                 PRINTF(3)

NAME
       printf,  fprintf,  sprintf,  snprintf,  vprintf, vfprintf, vsprintf,
       vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
...
```

Könyvtári függvénnyel

Helló, Világ! - write

```
#include <unistd.h>

int
main ()
{

  write (1, "Hello, Vilag!", 14);

  return 0;
}
```

Rendszerhívással

```
$ gcc writehello.c -o writehello
$ ./writehello
Hello, Vilag!
```

```
WRITE(2)                  Linux Programmer's Manual                 WRITE(2)

NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
...
```

Rendszerhívással

(Az előző fólia példában a printf() hívja a write()-ot.)

Helló, Világ! - .S

Rendszerhívással

```
.data

hello:
        .ascii  "Hello, Vilag!"

.text
        .global _start

_start:
        movl $4, %eax
        movl $1, %ebx
        movl $hello, %ecx
        movl $14, %edx

        int $0x80

        movl $1, %eax
        movl $0, %ebx

        int $0x80
```

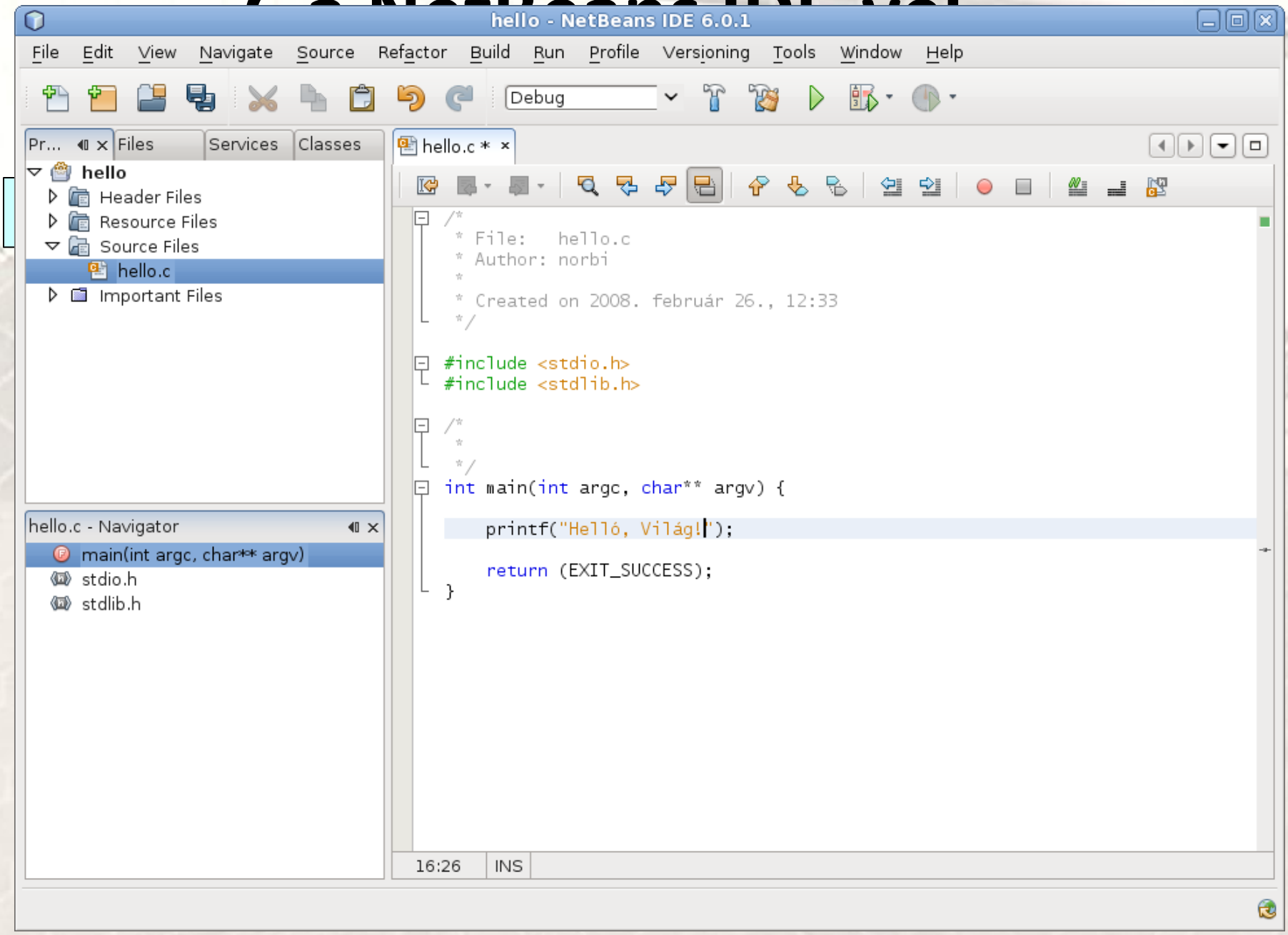
A megfelelő regisztekebe töltjük a rendszerhívás kódját és paramétereit:

```
write (1, "Hello, Vilag!", 14);
 EAX  EBX       ECX         EDX
```

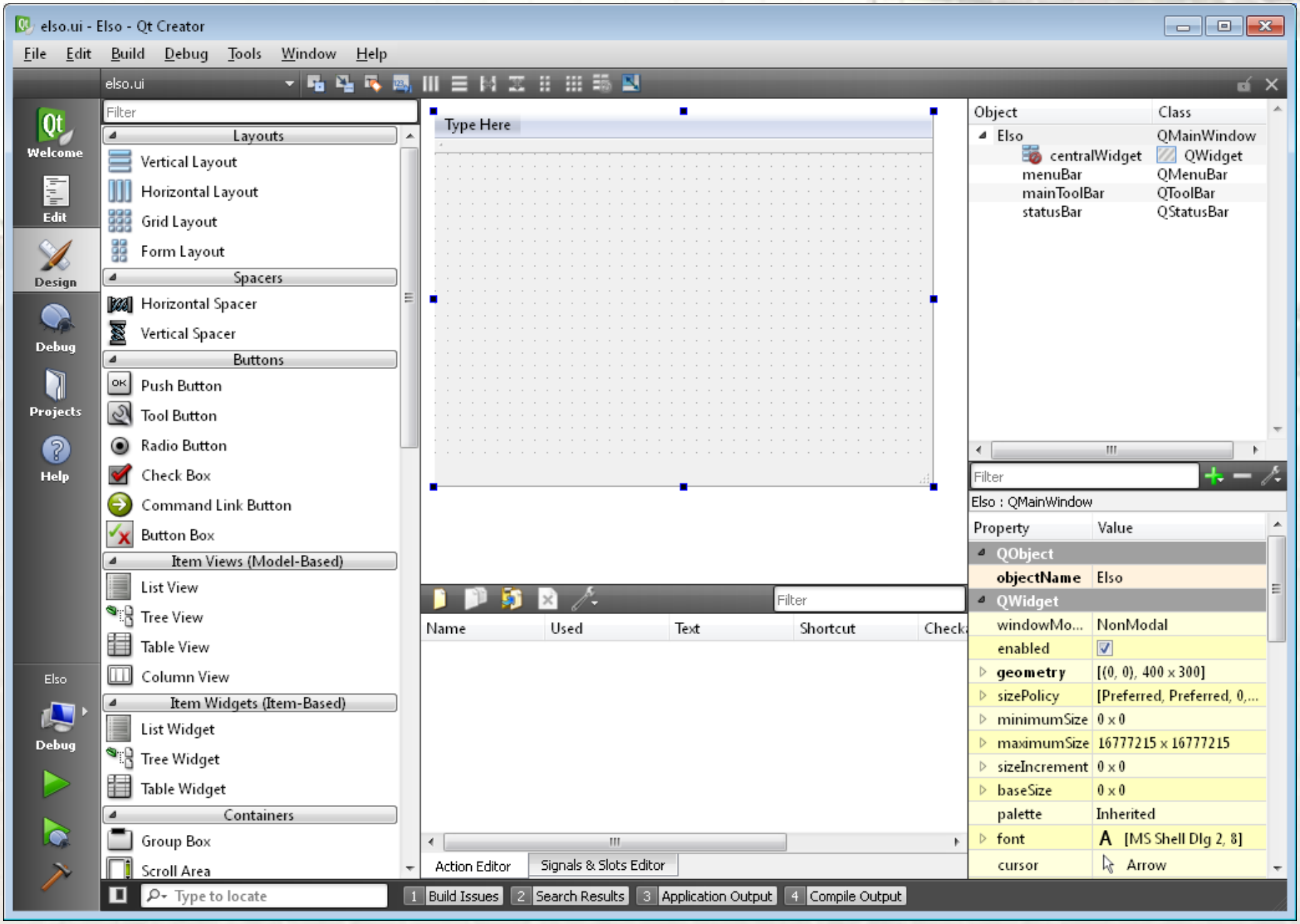
```
exit (0);
 EAX EBX
```

```
$ as asmhello.S -o asmhello.o
$ ld asmhello.o -o asmhello
$ ./asmhello
Hello, Vilag!
```

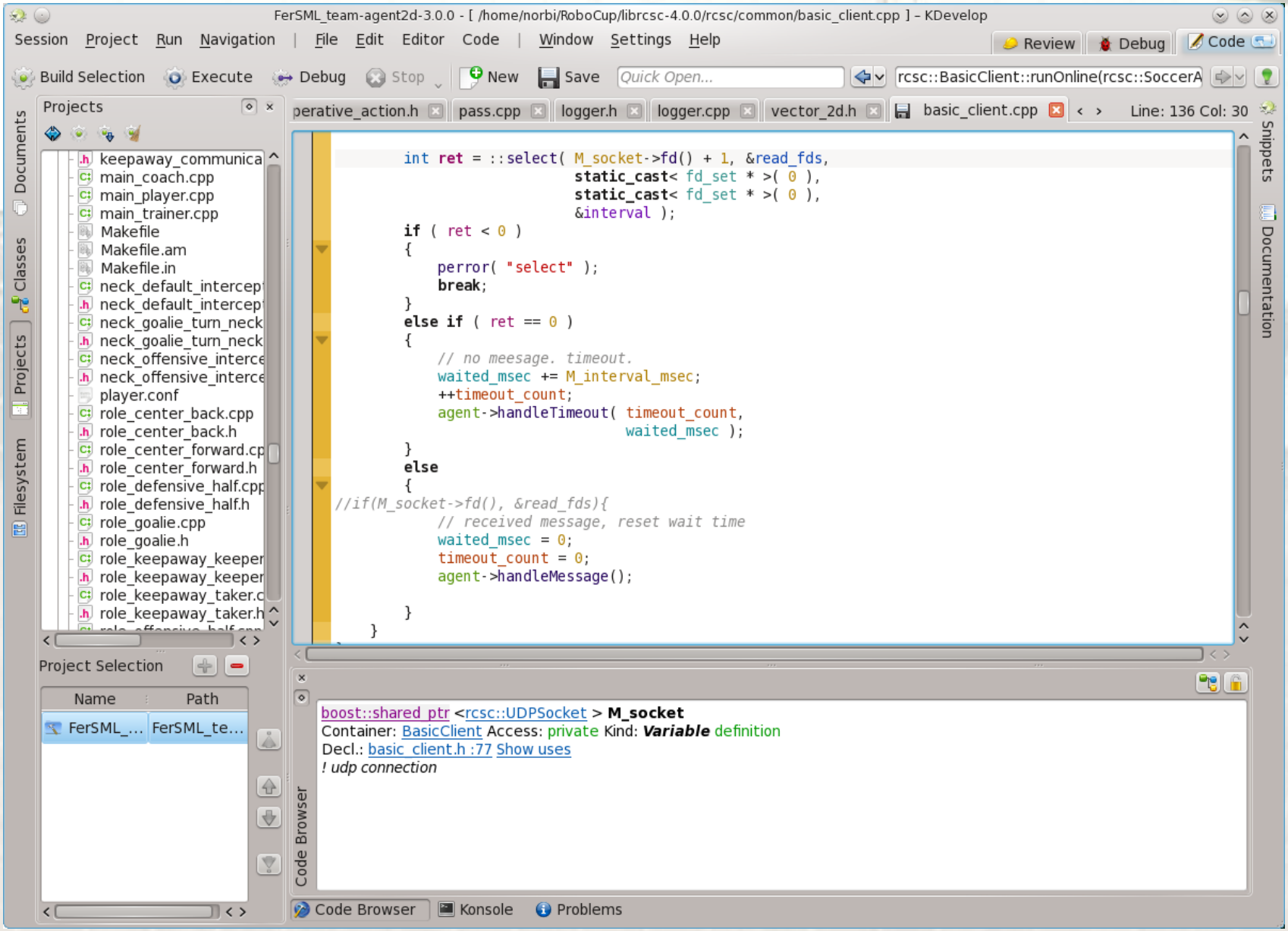
C a NetBeans IDE-vel



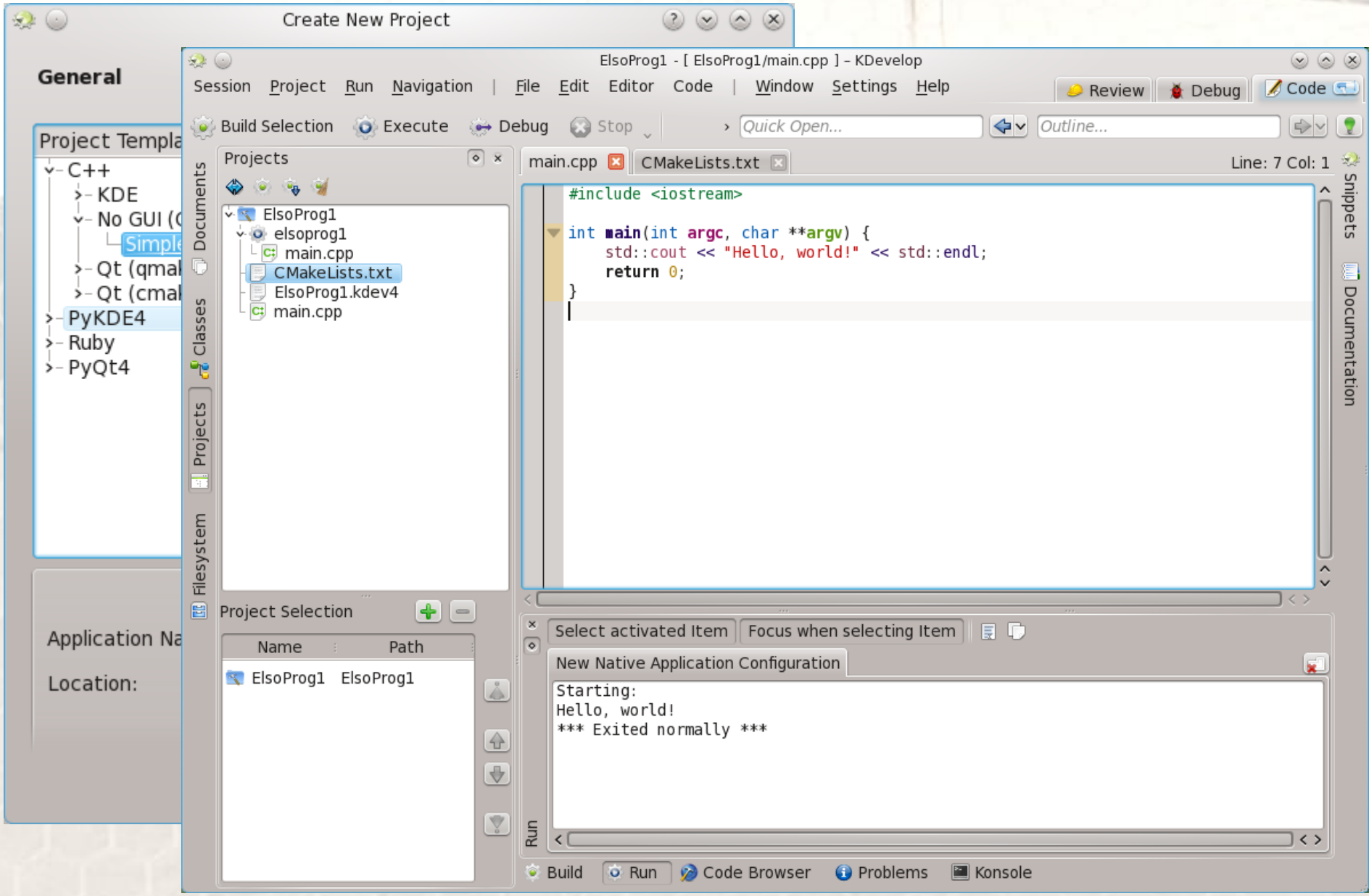
Nokia Qt SDK


elso.ui - Elso - Qt Creator
File Edit Build Debug Tools Window Help
elso.ui
Welcome
Edit
Design
Debug
Projects
Help
Elso
Debug
Filter
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Group Box
Scroll Area
Type Here
Filter
Name
Used
Text
Shortcut
Action Editor
Signals & Slots Editor
Object
Class
Elso
QMainWindow
centralWidget
QWidget
menuBar
QMenuBar
mainToolBar
QToolBar
statusBar
QStatusBar
Filter
Elso : QMainWindow
Property
Value
QObject
objectName
Elso
QWidget
windowMo...
NonModal
enabled
geometry
[(0, 0), 400 x 300]
sizePolicy
[Preferred, Preferred, 0,...
minimumSize
0 x 0
maximumSize
16777215 x 16777215
sizeIncrement
0 x 0
baseSize
0 x 0
palette
Inherited
font
A [MS Shell Dlg 2, 8]
cursor
Arrow
Type to locate
1 Build Issues
2 Search Results
3 Application Output
4 Compile Output

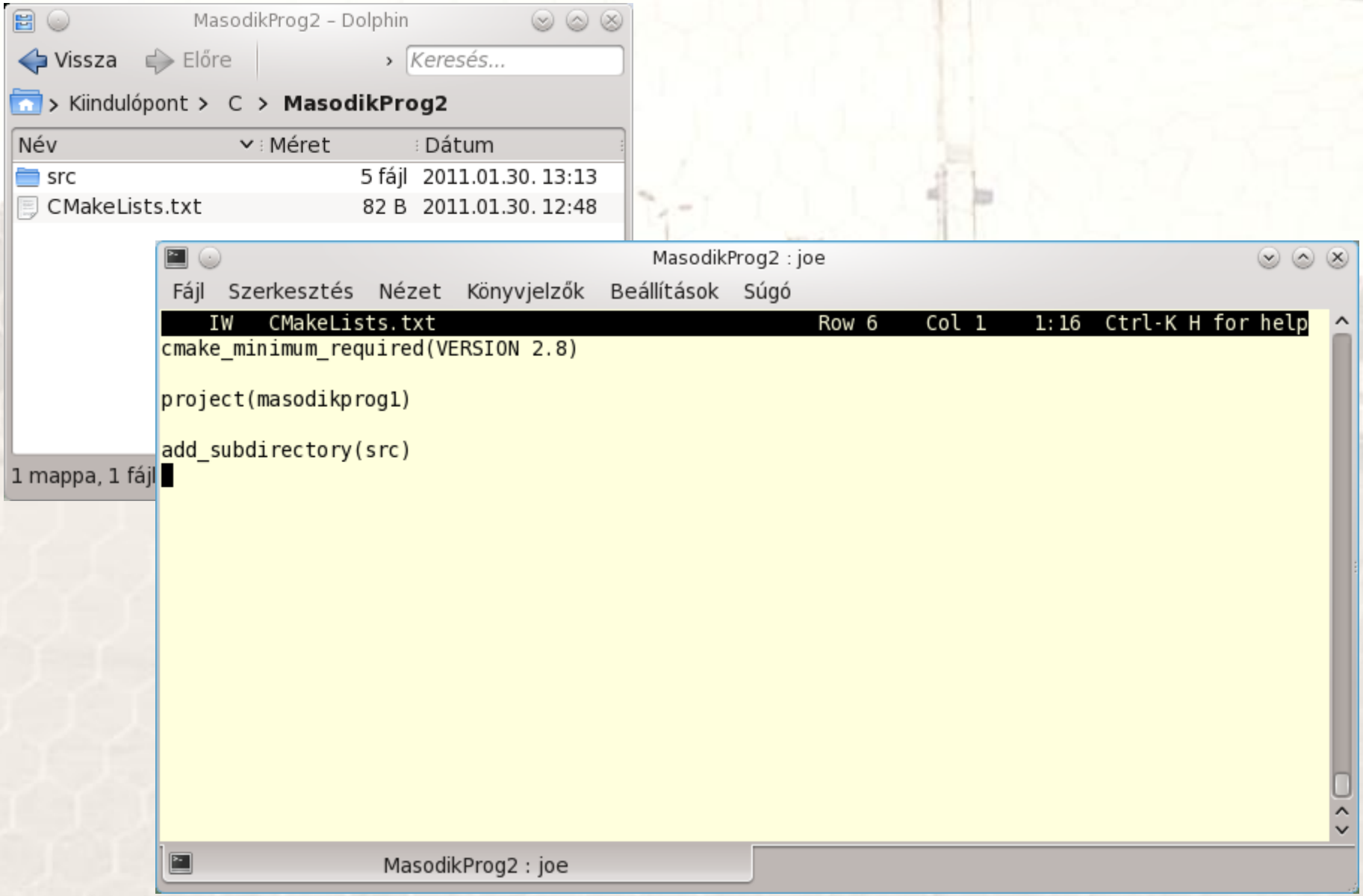
Fejlesztőeszköz: KDevelop



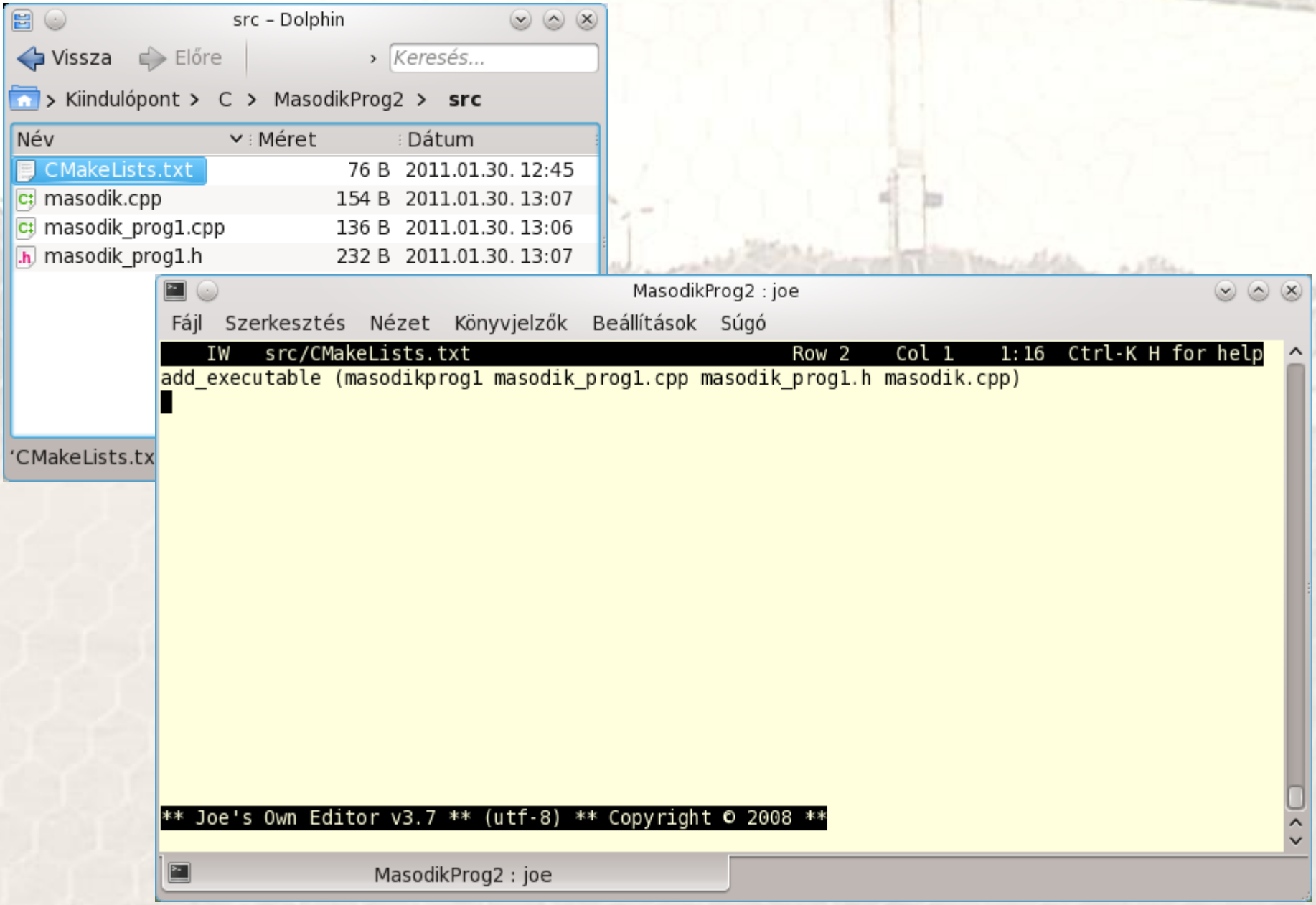
Fejlesztőeszköz: KDevelop



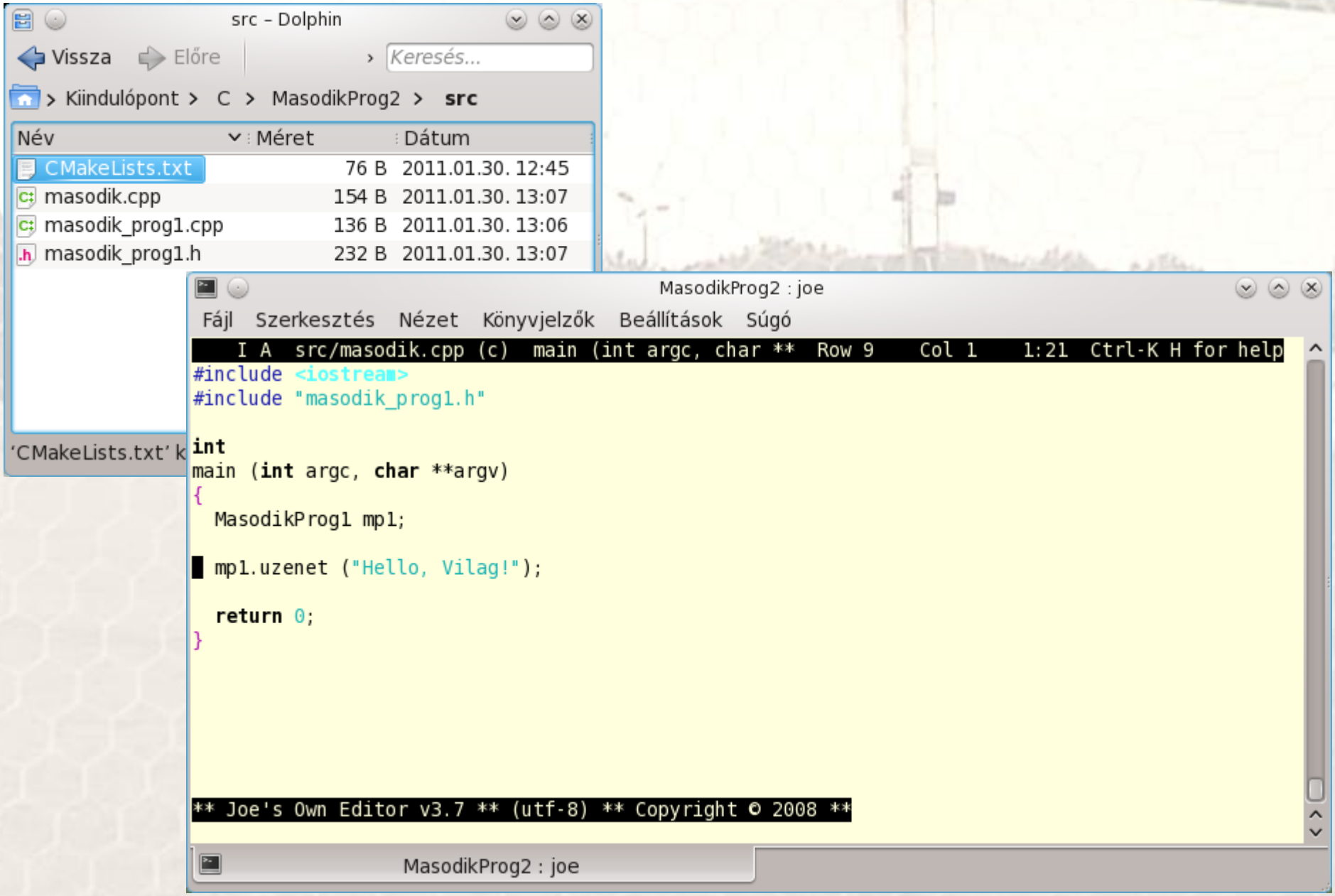
cmake parancssorból



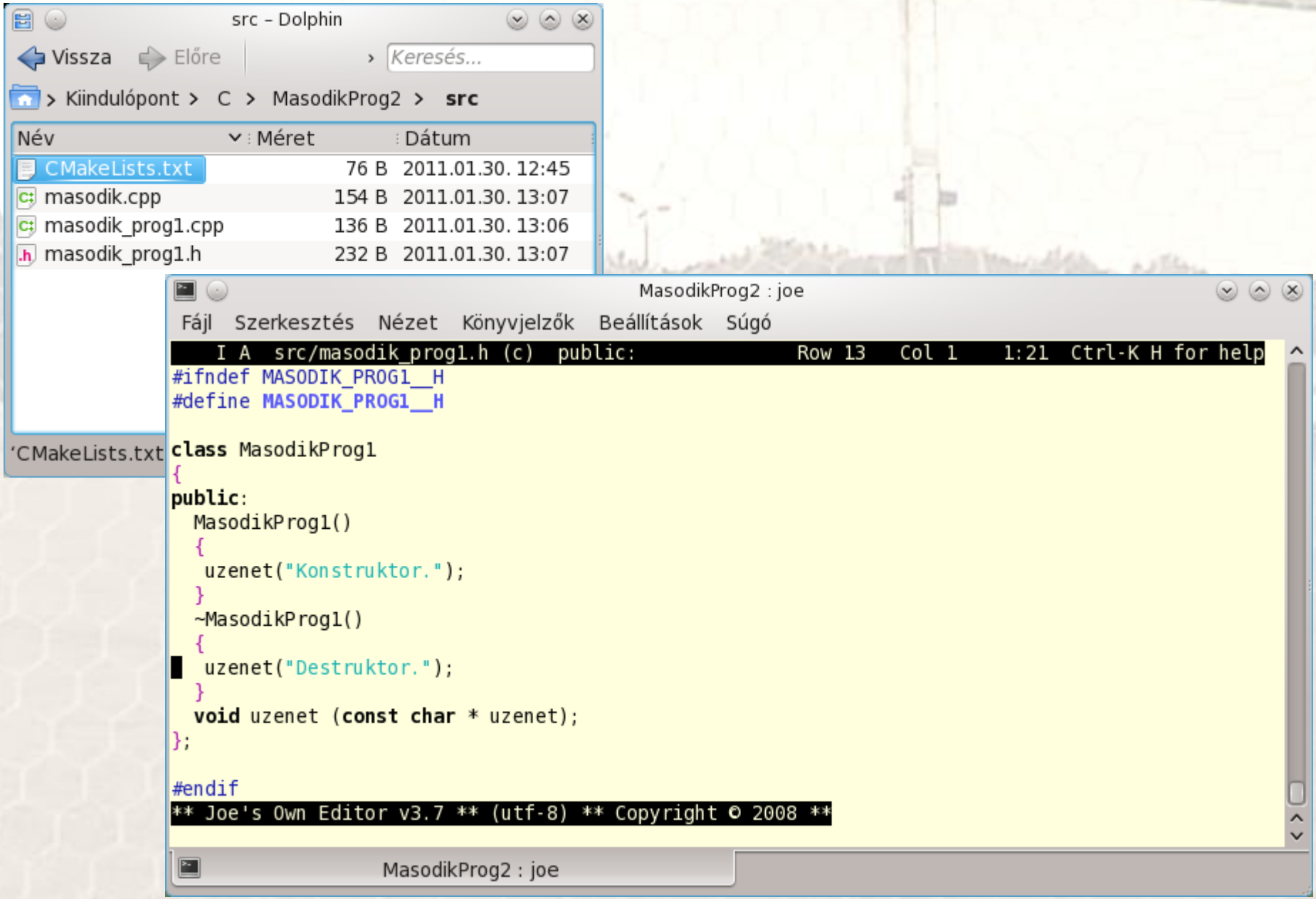
cmake parancssorból



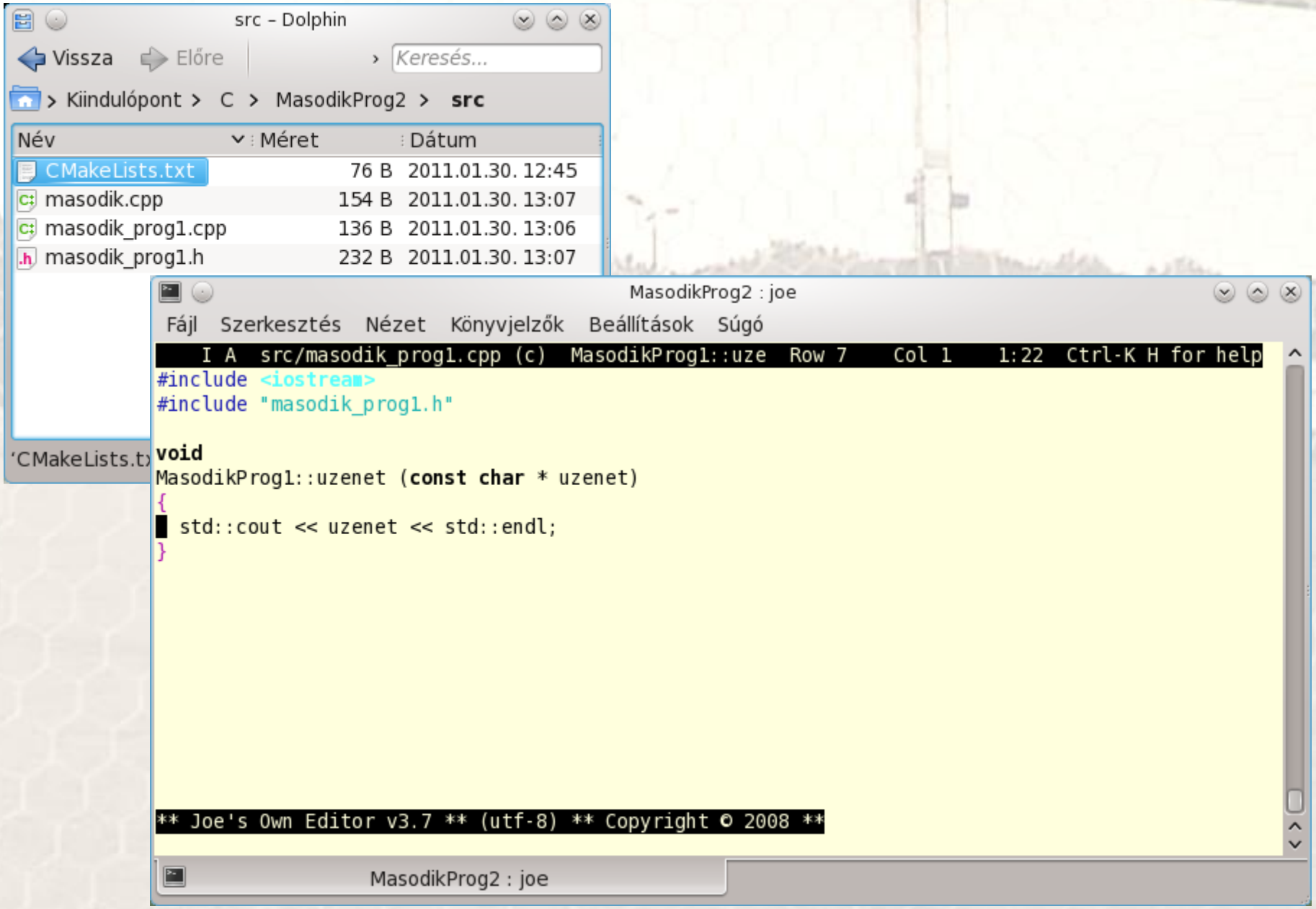
Helló, Világ!


src – Dolphin
Vissza
Előre
Keresés...
Kiindulópont > C > MasodikProg2 > src
Név
Méret
Dátum
CMakeLists.txt 76 B 2011.01.30. 12:45
masodik.cpp 154 B 2011.01.30. 13:07
masodik_prog1.cpp 136 B 2011.01.30. 13:06
masodik_prog1.h 232 B 2011.01.30. 13:07
'CMakeLists.txt' k
MasodikProg2 : joe
Fájl Szerkesztés Nézet Könyvjelzők Beállítások Súgó
I A src/masodik.cpp (c) main (int argc, char ** Row 9 Col 1 1:21 Ctrl-K H for help
#include <iostream>
#include "masodik_prog1.h"
int
main (int argc, char **argv)
{
MasodikProg1 mp1;
mp1.uzenet ("Hello, Vilag!");
return 0;
}
** Joe's Own Editor v3.7 ** (utf-8) ** Copyright © 2008 **
MasodikProg2 : joe

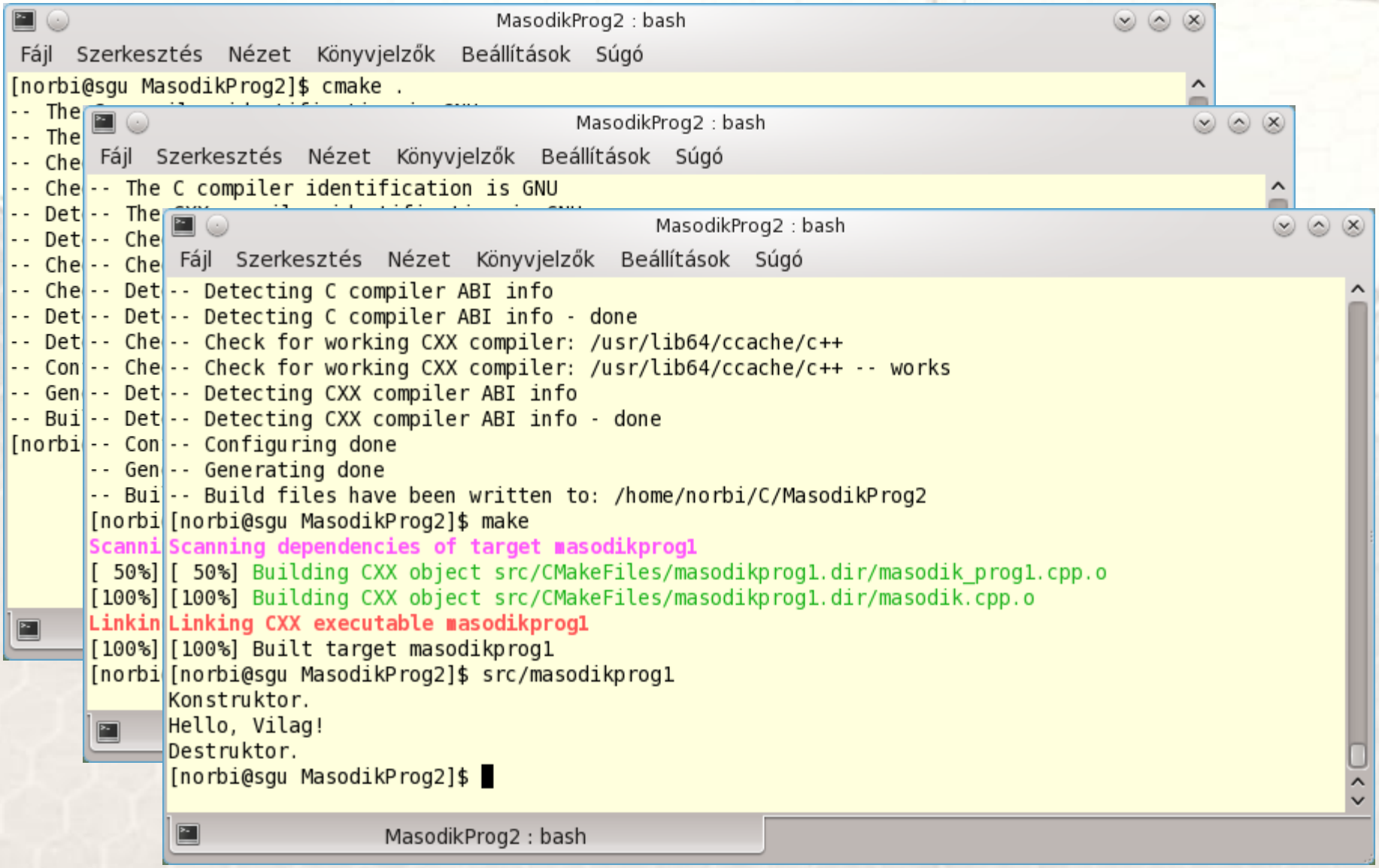
Helló, Világ!


src – Dolphin
Vissza
Előre
Keresés...
Kiindulópont > C > MasodikProg2 > src
Név
Méret
Dátum
CMakeLists.txt 76 B 2011.01.30. 12:45
masodik.cpp 154 B 2011.01.30. 13:07
masodik_prog1.cpp 136 B 2011.01.30. 13:06
masodik_prog1.h 232 B 2011.01.30. 13:07
'CMakeLists.txt
MasodikProg2 : joe
Fájl Szerkesztés Nézet Könyvjelzők Beállítások Súgó
I A src/masodik_prog1.h (c) public: Row 13 Col 1 1:21 Ctrl-K H for help
#ifndef MASODIK_PROG1__H
#define MASODIK_PROG1__H
class MasodikProg1
{
public:
MasodikProg1()
{
uzenet("Konstruktor.");
}
~MasodikProg1()
{
uzenet("Destruktor.");
}
void uzenet (const char * uzenet);
};
#endif
** Joe's Own Editor v3.7 ** (utf-8) ** Copyright © 2008 **
MasodikProg2 : joe

Helló, Világ!


src – Dolphin
Vissza
Előre
Keresés...
Kiindulópont > C > MasodikProg2 > src
Név
Méret
Dátum
CMakeLists.txt 76 B 2011.01.30. 12:45
masodik.cpp 154 B 2011.01.30. 13:07
masodik_prog1.cpp 136 B 2011.01.30. 13:06
masodik_prog1.h 232 B 2011.01.30. 13:07
MasodikProg2 : joe
Fájl Szerkesztés Nézet Könyvjelzők Beállítások Súgó
I A src/masodik_prog1.cpp (c) MasodikProg1::uze Row 7 Col 1 1:22 Ctrl-K H for help
#include <iostream>
#include "masodik_prog1.h"
void
MasodikProg1::uzenet (const char * uzenet)
{
std::cout << uzenet << std::endl;
}
** Joe's Own Editor v3.7 ** (utf-8) ** Copyright © 2008 **
MasodikProg2 : joe

Fordítás, futtatás



Laborkártyák

```c
#include <stdio.h>
int
main (void)
{
  int h = 0;
  int n = 0x01;
  do
    {
      ++h;
      printf ("[%d %u]\n", h, n);
    }
  while (n <<= 1);
  printf ("A szohossz ezen a gepen: %d bites\n", h);
  return 0;
}
```

Mit ír ki kockás zárójelek között?
Mit írna ki printf ("[%d %d]\n", h, n); esetén?

```c
int
main (void)
{
  for (;;)
    ;
}
```

Mit csinál ez a program?

Labor elején 16 hallgatónak 1 kérdés, gondolkodási idő nincs, azonnali válaszok! (A példákat nem kell begépelni, ugyanezek vagy hasonlók a PP-ben, elég bemásolni.)

Laborkártyák

```c
#include <stdio.h>
int
main (void)
{
  int c;
  while ((c = getchar ()) != EOF)
    putchar (c);
  return 0;
}
```

Mit csinál ez a program?

```c
#include <stdio.h>
#include <stdlib.h>
int
main (void)
{
  int i, r;
  for (i = 0; i < 10; ++i)
    {
      r = 1 + (int) (10.0 * rand () / (RAND_MAX + 1.0));
      printf ("%d ", r);
    }
  printf ("\n");
  return 0;
}
```

Mit csinál ez a program?

Mit csinál ez a program, ha többször futtatod egymás után?

Laborkártyák

```c
#include <stdio.h>
#include <stdlib.h>
int
main (void)
{
  int i = 0;
  double r, s = 0.0;
  while (scanf ("%lf", &r) > 0)
    {
      ++i;
      s += r;
    }
  printf ("%lf\n", s / i);
  return 0;
}
```

Mit csinál ez a program? (input vége Ctrl+D)

Laborkártyák

```c
// rszamok.c

#include <stdio.h>
#include <stdlib.h>
int
main (void)
{
  int i, r;
  for (i = 0; i < 1000; ++i)
    {
      r = 1 + (int) (10.0 * rand () / (RAND_MAX + 1.0));
      printf ("%d ", r);
    }
  printf ("\n");
  return 0;
}
```

```c
// atlag.c

#include <stdio.h>
#include <stdlib.h>
int
main (void)
{
  int i = 0;
  int r, s = 0;
  while (scanf ("%d", &r) > 0)
    {
      ++i;
      s += r;
    }
  printf ("%d\n", s / i);
  return 0;
}
```

```
nbatfai@hallg:~/c$ gcc  atlag.c -o atlag
nbatfai@hallg:~/c$ gcc rszamok.c -o rszamok
nbatfai@hallg:~/c$ ./rszamok|./atlag
```

Mit ír ki és mit jelent az utolsó parancs?

Laborkártyák

A)
```
for(i=0; i<3; ++i)
   printf("%d ", i);
```

B)
```
for(i=0; i<3; i++)
   printf("%d ", i);
```

Mi a különbség a két ciklus között?

```
if(signal(SIGINT, jelkezelo) == SIG_IGN)
   signal(SIGINT, SIG_IGN);
```

```
if(signal(SIGINT, SIG_IGN) != SIG_IGN)
   signal(SIGINT, jelkezelo);
```

Mi a különbség a két feltétel között?

Laborkártyák

Mi a különbség: **man passwd** és **man 5 passwd** között?

```
PASSWD(1)                                                    PASSWD(1)

NÉV
       passwd - Felhasználói jelszó megváltoztatása

ÁTTEK
```

```
PASSWD(5)                  Linux Programozói Kézikönyv                  PASSWD(5)

NÉV
       passwd - Jelszófájl

LEÍRÁS
LEÍRÁ  A  Passwd egy, a felhasználók listáját és a belépésükhöz szükséges jel-
       szavakat tartalmazó ASCII file. Mindenkinek tudnia  kell  olvasni  (sok
       segédprogram,  pl.  a  ezt  használja hogy a felhasználói azonosítókhoz
       (UID-khez) neveket rendeljen), de írási jogot csak  a  rendszergazdának
       szabad kapnia.

       A  régi szép időkben nem is volt ezzel semmi baj: mindenki olvashatta a
       felhasználók kódolt jelszavait, de mivel az akkori hardware-körülmények
       nem  tették  lehetővé  a  jól  megválasztott  jelszavak  törését,  és a
    Je felasználók alapvetően jóindulatűak voltak, ez nem  okozott  problémát.
       Manapság  mindenki  igyekszik shadow-zott jelszavakat használni, amikor
       is a /etc/passwd file-ban a jelszavak helyett csak egy * karakter  áll,
       és  az  igazi jelszavak a /etc/shadow file-ban találhatóak, amit csak a
       rendszergazda olvashat.

       Amikor új felhasználót készítesz, célszerű a jelszó helyét üresen  hag-
       yni,  és  a  passwd(1)  programmal  kitölteni.  Egy * karakter a jelszó
```

Laborkártyák

```
int a;
int *b;
int c[5];
int *d[5];
int e[3][3];
int *f ();
int (*g)();
```

```
char a;
char *b;
char c[5];
char *d[5];
char e[3][3];
char *f ();
char (*g)();
```

```
int a;
int *b;
int c[5];
int *d[5];
int e[3][3];
int *f ();
void (*g)(int);
```

```
int a;
int *b;
int c[5];
int *d[5];
int e[3][3];
int *f ();
void (*g)();
```

```
int a;
int *b;
int c[5];
int *d[5];
int e[3][3];
int *f ();
void* (*g)(void *);
```

Mit takar az a, b, c, d, e, f, g?

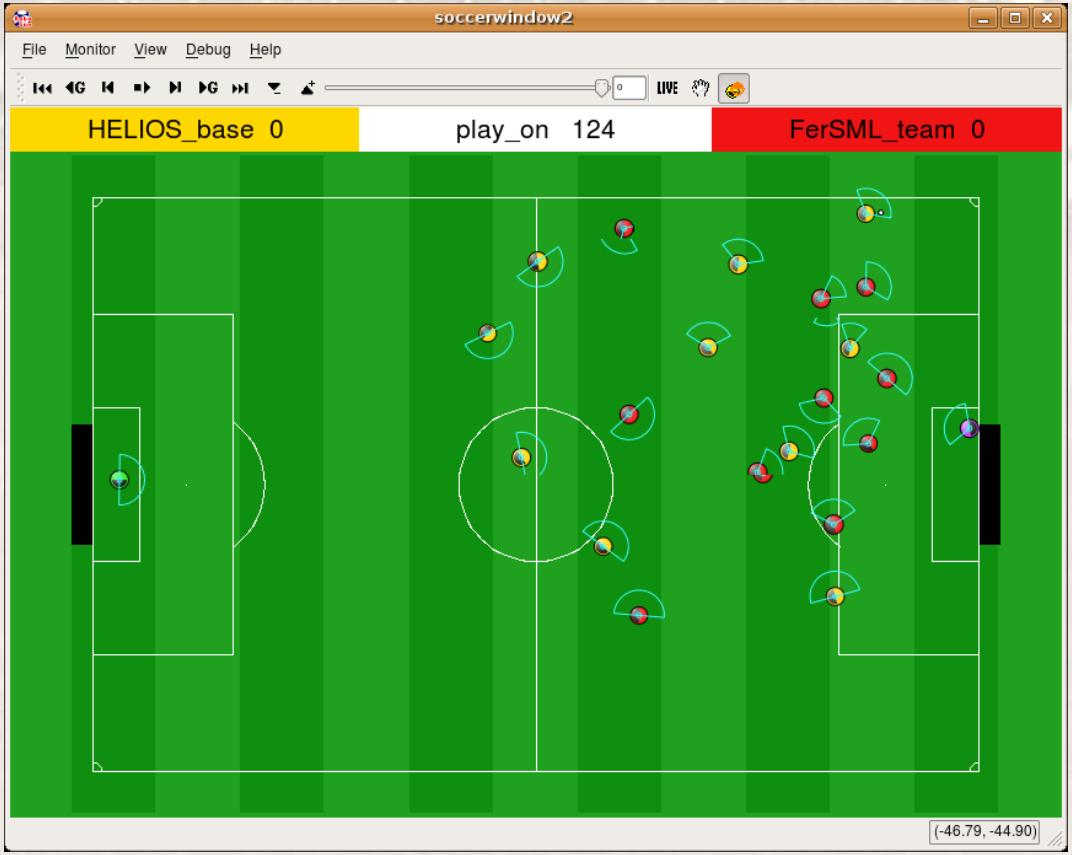
Labor-kártyák

Mit ír ki ez a program?

```c
#include <stdio.h>

#define MERET 5
char buffer[MERET];

char *
string_masolo_man_pl_alapjan (char *dest, const char *src, int n)
{
  int i;

  for (i = 0; i < n && src[i] != '\0'; i++)
    dest[i] = src[i];
  for (; i < n; i++)
    dest[i] = '\0';

  return dest;
}

int
main ()
{
  string_masolo_man_pl_alapjan (buffer, "aaaaaaaaaaaaaaaaaaaaaaaaaaaa",
                                MERET);
  printf ("%s\n", buffer);
  string_masolo_man_pl_alapjan (buffer, "bbbbbbbbbbbbbbbbbbbbbbbbbbb",
                                MERET - 1);
  printf ("%s\n", buffer);

  return 0;
}
```

Otthoni opcionális feladat

A robotfocitika három törvénye posztban bemutatott telepítés reprodukálása:
http://fersml.blog.hu/2010/12/28/a_robotfocitika_harom_torvenye
+soccerwindow:
http://fersml.blog.hu/2011/01/01/fersml_avatar_2_robocup_foci_agens



Kötelező olvasmány

K&R könyvből olvassuk el (többször!) az első három fejezetet:

a) Bevezetés
b) Típusok, operátorok és kifejezések
c) Vezérlési szerkezetek

(kb. 70 o.)