

# Univerzális programozás

---

**Írd meg a saját programozás tankönyvedet!**

Ed. BHAX, DEBRECEN,  
2019. február 19, v. 0.0.4

Copyright © 2019 Dr. Bátfai Norbert

Copyright (C) 2019, Norbert Bátfai Ph.D., batfai.norbert@inf.unideb.hu, nbatfai@gmail.com,

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

<https://www.gnu.org/licenses/fdl.html>

Engedélyt adunk Önnek a jelen dokumentum sokszorosítására, terjesztésére és/vagy módosítására a Free Software Foundation által kiadott GNU FDL 1.3-as, vagy bármely azt követő verziójának feltételei alapján. Nincs Nem Változtatható szakasz, nincs Címlapszöveg, nincs Hátlapszöveg.

<http://gnu.hu/fdl.html>

## COLLABORATORS

	<i>TITLE :</i> Univerzális programozás		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Bátfai, Norbert Ács Koós, Zoltán	2019. március 17.	

## REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
0.0.1	2019-02-12	Az iniciális dokumentum szerkezetének kialakítása.	nbatfai
0.0.2	2019-02-14	Inciális feladatlisták összeállítása.	nbatfai
0.0.3	2019-02-16	Feladatlisták folytatása. Feltöltés a BHAX csatorna <a href="https://gitlab.com/nbatfai/bhax">https://gitlab.com/nbatfai/bhax</a> repójába.	nbatfai
0.0.4	2019-02-19	Aktualizálás, javítások.	nbatfai

# Tartalomjegyzék

<b>I. Bevezetés</b>	<b>1</b>
<b>1. Vízió</b>	<b>2</b>
1.1. Mi a programozás?	2
1.2. Milyen doksikat olvassak el?	2
1.3. Milyen filmeket nézzek meg?	2
<b>II. Tematikus feladatok</b>	<b>3</b>
<b>2. Helló, Turing!</b>	<b>5</b>
2.1. Végtelen ciklus	5
2.2. Lefagyott, nem fagyott, akkor most mi van?	7
2.3. Változók értékének felcserélése	8
2.4. Labdapattogás	9
2.5. Szóhossz és a Linus Torvalds féle BogomIPS	12
2.6. Helló, Google!	13
2.7. 100 éves a Brun tétel	15
2.8. A Monty Hall probléma	16
<b>3. Helló, Chomsky!</b>	<b>18</b>
3.1. Decimálisból unárisba átváltó Turing gép	18
3.2. Az $a^n b^n c^n$ nyelv nem környezetfüggetlen	18
3.3. Hivatkozási nyelv	19
3.4. Saját lexikális elemző	20
3.5. l33t.1	21
3.6. A források olvasása	24
3.7. Logikus	26
3.8. Deklaráció	27

<b>4. Helló, Caesar!</b>	<b>30</b>
4.1. int *** háromszögmátrix	30
4.2. C EXOR titkosító	30
4.3. Java EXOR titkosító	30
4.4. C EXOR törő	30
4.5. Neurális OR, AND és EXOR kapu	31
4.6. Hiba-visszaterjesztéses perceptron	31
<b>5. Helló, Mandelbrot!</b>	<b>32</b>
5.1. A Mandelbrot halmaz	32
5.2. A Mandelbrot halmaz a std::complex osztállyal	32
5.3. Biomorfok	32
5.4. A Mandelbrot halmaz CUDA megvalósítása	32
5.5. Mandelbrot nagyító és utazó C++ nyelven	32
5.6. Mandelbrot nagyító és utazó Java nyelven	33
<b>6. Helló, Welch!</b>	<b>34</b>
6.1. Első osztályom	34
6.2. LZW	34
6.3. Fabejárás	34
6.4. Tag a gyökér	34
6.5. Mutató a gyökér	35
6.6. Mozgató szemantika	35
<b>7. Helló, Conway!</b>	<b>36</b>
7.1. Hangyaszimulációk	36
7.2. Java életjáték	36
7.3. Qt C++ életjáték	36
7.4. BrainB Benchmark	37
<b>8. Helló, Schwarzenegger!</b>	<b>38</b>
8.1. Szoftmax Py MNIST	38
8.2. Szoftmax R MNIST	38
8.3. Mély MNIST	38
8.4. Deep dream	38
8.5. Robotpszichológia	39

<b>9. Helló, Chaitin!</b>	<b>40</b>
9.1. Iteratív és rekurzív faktoriális Lisp-ben . . . . .	40
9.2. Weizenbaum Eliza programja . . . . .	40
9.3. Gimp Scheme Script-fu: króm effekt . . . . .	40
9.4. Gimp Scheme Script-fu: név mandala . . . . .	40
9.5. Lambda . . . . .	41
9.6. Omega . . . . .	41
 <b>III. Második felvonás</b>	 <b>42</b>
<b>10. Helló, Arroway!</b>	<b>44</b>
10.1. A BPP algoritmus Java megvalósítása . . . . .	44
10.2. Java osztályok a Pi-ben . . . . .	44
 <b>IV. Irodalomjegyzék</b>	 <b>45</b>
10.3. Általános . . . . .	46
10.4. C . . . . .	46
10.5. C++ . . . . .	46
10.6. Lisp . . . . .	46

# Ábrák jegyzéke

3.1. Turing gép . . . . .	18
---------------------------	----

DRAFT

## Ajánlás

„To me, you understand something only if you can program it. (You, not someone else!) Otherwise you don't really understand it, you only think you understand it.”

—Gregory Chaitin, *META MATH! The Quest for Omega*, [METAMATH]



# Előszó

Amikor programozónak terveztem állni, ellenezték a környezetemben, mondván, hogy kell szövegszerkesztő meg táblázatkezelő, de az már van... nem lesz programozói munka.

Tévedtek. Hogy egy generáció múlva kell-e még tömegesen hús-vér programozó vagy olcsóbb lesz allokálni igény szerint pár robot programozót a felhőből? A programozók dolgozók lesznek vagy papok? Ki tudhatná ma.

Mindenesetre a programozás a teoretikus kultúra csúcsa. A GNU mozgalomban látom annak garanciáját, hogy ebben a szellemi kalandban a gyerekeim is részt vehessenek majd. Ezért programozunk.

## Hogyan forgasd

A könyv célja egy stabil programozási szemlélet kialakítása az olvasóban. Módszere, hogy hetekre bontva ad egy tematikus feladatcsokrot. Minden feladathoz megadja a megoldás forráskódját és forrásokat feldolgozó videókat. Az olvasó feladata, hogy ezek tanulmányozása után maga adja meg a feladat megoldásának lényegi magyarázatát, avagy írja meg a könyvet.

Miért univerzális? Mert az olvasótól (kvázi az írótól) függ, hogy kinek szól a könyv. Alapértelmezésben gyerekeknek, mert velük készítem az iniciális változatot. Ám tervezem felhasználását az egyetemi programozás oktatásban is. Ahogy szélesedni tudna a felhasználók köre, akkor lehetne kiadása különböző korosztályú gyerekeknek, családoknak, szakköröknek, programozás kurzusoknak, felnőtt és továbbképzési műhelyeknek és sorolhatnánk...

## Milyen nyelven nyomjuk?

C (mutatók), C++ (másoló és mozgató szemantika) és Java (lebutított C++) nyelvekből kell egy jó alap, ezt kell kiegészíteni pár R (vektoros szemlélet), Python (gépi tanulás bevezető), Lisp és Prolog (hogy lássuk mást is) példával.

## Hogyan nyomjuk?

Ránts le a <https://gitlab.com/nbatfai/bhax> git repót, vagy méginkább forkolj belőle magadnak egy sajátot a GitLabon, ha már saját könyvön dolgozol!

Ha megvannak a könyv DocBook XML forrásai, akkor az alább látható **make** parancs ellenőrzi, hogy „jól formázottak” és „érvényesek-e” ezek az XML források, majd elkészíti a dblatex programmal a könyved pdf változatát, íme:

```
batfai@entropy:~$ cd glrepos/bhax/thematic_tutorials/bhax_textbook/
batfai@entropy:~/glrepos/bhax/thematic_tutorials/bhax_textbook$ make
rm -f bhax-textbook-fdl.pdf
xmllint --xinclude bhax-textbook-fdl.xml --output output.xml
xmllint --relaxng http://docbook.org/xml/5.0/rng/docbookxi.rng output.xml  ↵
--noout
output.xml validates
rm -f output.xml
dblatex bhax-textbook-fdl.xml -p bhax-textbook.xls
Build the book set list...
Build the listings...
XSLT stylesheets DocBook - LaTeX 2e (0.3.10)
=====
Stripping NS from DocBook 5/NG document.
Processing stripped document.
Image 'dblatex' not found
Build bhax-textbook-fdl.pdf
'bhax-textbook-fdl.pdf' successfully built
```

Ha minden igaz, akkor most éppen ezt a legenerált `bhax-textbook-fdl.pdf` fájlt olvasod.



#### A DocBook XML 5.1 új neked?

Ez esetben forgasd a <https://tdg.docbook.org/tdg/5.1/> könyvet, a végén találsz az informatikai szövegek jelölésére használható gazdag „API” elemenkénti bemutatását.

# **I. rész**

## **Bevezetés**

# 1. fejezet

## Vízió

### Mi a programozás?

### Milyen doksikat olvassak el?

- Olvasgasd a kézikönyv lapjait, kezd a **man man** parancs kiadásával. A C programozásban a 3-as szintű lapokat fogod nézegetni, például az első feladat kapcsán ezt a **man 3 sleep** lapot
- [[KERNIGHANRITCHIE](#)]
- [[BMECPP](#)]
- Az igazi kockák persze csemegéznek a C nyelvi szabvány [ISO/IEC 9899:2017](#) kódcsipeteiből is.

### Milyen filmeket nézzek meg?

- 21 - Las Vegas ostroma, <https://www.imdb.com/title/tt0478087/>, benne a **Monty Hall probléma** bemutatása.

## **II. rész**

### **Tematikus feladatok**

**Bátf41 Haxor Stream**

A feladatokkal kapcsolatos élő adásokat sugároz a <https://www.twitch.tv/nbatfai> csatorna, melynek permanens archívuma a <https://www.youtube.com/c/nbatfai> csatornán található.

---

DRAFT

## 2. fejezet

# Helló, Turing!

### Végtelen ciklus

Írj olyan C végtelen ciklusokat, amelyek 0 illetve 100 százalékban dolgoztatnak egy magot és egy olyat, amely 100 százalékban minden magot!

Megoldás videó:

Megoldás forrása:

A `loop_1core.c` program futtatása 100 százalékban kihasznál egy magot.

```
int main()
{
    while(1) {}
    return 0;
}
```

Ha erről magunk is meg szeretnénk győződni akkor a `top -p `pgrep -u username loop1_core`` parancsot kell beírunk egy külön terminálba, hogy lássuk a várt eredményt.

```
Tasks:   1 total,    1 running,    0 sleeping,    0 stopped,    0 zombie
%Cpu0  : 100,0 us,   0,7 sy,    0,0 ni, 24,6 id,   0,0 wa,   0,0 hi,   0,3 si,   ↵
         0,0 st
%Cpu1  :  8,3 us,    2,4 sy,    0,0 ni, 62,5 id,   0,0 wa,   0,0 hi,   0,0 si,   ↵
         0,0 st
KiB Mem : 3942056 total, 216336 free, 2236064 used, 1489656 buff/cache
KiB Swap:  999420 total,  996336 free,    3084 used. 1123772 avail Mem

PID     USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM    TIME+  ↵
      COMMAND
32098 kewo      20   0   4220    628    560  R   100,0  0,0    4:02.84 ↵
      loop_1core
```

Ha azt szeretnénk elérni, hogy egy magot 0 százalékban terheljen, akkor meg kell hívnunk a `sleep(n)` metódust ami lelassítja az iterálás sebességét.

```
int main()
{
    while(1)
    {
        sleep(1);
    }
    return 0;
}
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	↔
COMMAND											
32243	kewo	20	0	4220	632	564	S	0,0	0,0	0:00.01	↔
loop_0core											

Minden mag 100 százalékos kihasználását szálasítással lehet elérni az OpenMP segítségével.

```
#include <omp.h>
int main()
{
    #pragma omp parallel
    {
        while(1){}
    }
    return 0;
}
```

**gcc loop\_all\_cores.c -o loop\_all\_cores -fopenmp** paranccsal fordítjuk le, majd futtatjuk.

```
Tasks:   1 total,    1 running,    0 sleeping,    0 stopped,    0 zombie
%Cpu0  : 100,0 us,   1,3 sy,    0,0 ni,    0,0 id,    0,0 wa,    0,0 hi,   0,7 si,   ↔
         0,0 st
%Cpu1  : 100,0 us,   2,0 sy,    0,0 ni,    0,0 id,    0,0 wa,    0,0 hi,   0,0 si,   ↔
         0,0 st
KiB Mem : 3942056 total,  233304 free,  2368576 used,  1340176 buff/cache
KiB Swap:  999420 total,  995056 free,    4364 used.  943384 avail Mem

  PID USER      PR  NI   VIRT   RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 32728 kewo      20   0  18968   1532   1408 R  184,3   0,0   0:25.05  ↔
    loop_all_c+
```



## Lefagyott, nem fagyott, akkor most mi van?

Mutasd meg, hogy nem lehet olyan programot írni, amely bármely más programról eldönti, hogy le fog-e fagyni vagy sem!

Megoldás videó:

Megoldás forrása:

Tegyük fel, hogy akkora haxorok vagyunk, hogy meg tudjuk írni a Lefagy függvényt, amely tetszőleges programról el tudja dönteni, hogy van-e benne végtelen ciklus:

```
Program T100
{
    boolean Lefagy(Program P)
    {
        if(P-ben van végtelen ciklus)
            return true;
        else
            return false;
    }

    main(Input Q)
    {
        Lefagy(Q)
    }
}
```

A program futtatása, például akár az előző v. c ilyen pszeudókódjára:

```
T100 (t.c.pseudo)
true
```

akár önmagára

```
T100 (T100)
false
```

ezt a kimenetet adja.

A T100-as programot felhasználva készítsük most el az alábbi T1000-set, amelyben a Lefagy-ra épülő Lefagy2 már nem tartalmaz feltételezett, csak konkrét kódot:

```
Program T1000
{
    boolean Lefagy(Program P)
    {
```

```
    if(P-ben van végtelen ciklus)
        return true;
    else
        return false;
}

boolean Lefagy2(Program P)
{
    if(Lefagy(P))
        return true;
    else
        for(;;);
}

main(Input Q)
{
    Lefagy2(Q)
}

}
```

Mit for kiírni erre a T1000 (T1000) futtatásra?

- Ha T1000 lefagyó, akkor nem fog lefagyni, kiírja, hogy true
- Ha T1000 nem fagyó, akkor pedig le fog fagyni...

akkor most hogy fog működni? Sehogy, mert ilyen Lefagy függvényt, azaz a T100 program nem is létezik.

Mondjuk azt, hogy létezik olyan program 'A', ami el tudja dönteni egy másik programról, hogy van-e benne végtelen ciklus. Ha van, akkor megáll a program. Ellenkező esetben pedig végtelen ciklusba kezd. Létrehozunk egy új programot 'B' az előzőt felhasználva és ha A megállt, akkor végtelen ciklusba kezd, ha pedig az A kezdett végtelen ciklusba, akkor megáll. Mi lesz a végeredménye annak, ha önmagára hívjuk meg ezt a programot? Csak akkor áll le a program, ha a második argumentumként kapott saját programunk megáll. Ez viszont ellentmondáshoz vezet. Ezért nem lehetséges ilyen programot írni.

## Változók értékének felcserélése

Írj olyan C programot, amely felcseréli két változó értékét, bármiféle logikai utasítás vagy kifejezés használata nélkül!

Megoldás videó: [https://bhaxor.blog.hu/2018/08/28/10\\_begin\\_goto\\_20\\_avagy\\_elindulunk](https://bhaxor.blog.hu/2018/08/28/10_begin_goto_20_avagy_elindulunk)

Megoldás forrása:

Ezzel a programmal két változó értékét a különbségük kihasználásával segédváltozó nélkül cseréljük meg.

```
#include<stdio.h>
int main()
{
    int firstNumber = 2, secondNumber = 4;

    printf("1. number = %d\n2.number = %d\n", firstNumber, secondNumber);

    firstNumber = ( firstNumber - secondNumber );
    secondNumber = ( firstNumber + secondNumber );
    firstNumber = ( secondNumber - firstNumber );

    printf("1. number = %d\n2. number = %d\n", firstNumber, secondNumber ↵
    );

    return 0;
}
```

## Labdapattogás

Először if-ekkel, majd bármiféle logikai utasítás vagy kifejezés nasználata nélkül írj egy olyan programot, ami egy labdát pattogtat a karakteres konzolon! (Hogy mit értek pattogtatás alatt, alább láthatod a videókon.)

Megoldás videó: <https://bhaxor.blog.hu/2018/08/28/labdapattogas>

Megoldás forrása:

A getmaxyx függvénnyel meghatározzuk a terminál ablakunk méretét. Ezután kiíratjuk a labdát a kezdőpozícióra, majd addig növeljük a pozícióértéket, amíg el nem érjük az ablak széleit, tetejét vagy alját. Ekkor negáljuk azt az értéket amelyet hozzáadunk a hozzárendelt pozícióértékkel. Ez persze mind egy végtelen ciklusban van, tehát a folyamatosan fut a programunk.

```
#include <stdio.h>
#include <curses.h>
#include <unistd.h>

int
main ( void )
{
    WINDOW *ablak;
    ablak = initscr ();

    int x = 0;
    int y = 0;
```

```
int xnov = 1;
int ynov = 1;

int mx;
int my;

for ( ;; ) {

    getmaxyx ( ablak, my , mx );

    mvprintw ( y, x, "O" );

    refresh ();
    usleep ( 100000 );

    x = x + xnov;
    y = y + ynov;

    if ( x>=mx-1 ) { // elerte-e a jobb oldalt?
        xnov = xnov * -1;
    }
    if ( x<=0 ) { // elerte-e a bal oldalt?
        xnov = xnov * -1;
    }
    if ( y<=0 ) { // elerte-e a tetejet?
        ynov = ynov * -1;
    }
    if ( y>=my-1 ) { // elerte-e a aljat?
        ynov = ynov * -1;
    }

}

return 0;
}
```

If ek nélkül is megoldható ez a feladat. Először is létrehozuk a gotoxy függvényt amellyel a kurzort fogjuk pozicionálni. Ezután a main függvényben megadjuk a labda kezdeti helyzetét és a pálya méretét. Meghatározzuk a pálya széleinek helyzetét és már indulhatunk is. Egy végtelen ciklusba kiíratjuk a labda pozícióját és meghívjuk a gotoxy-t ami kiírja nekünk a labdát. Folyamatosan töröljük a terminál ablakát és újra kiíratjuk a labda új helyzetét.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

static void gotoxy(int x, int y) //kurzor pozicionálása
```

```
{
    int i;
    for(i=0; i<y; i++) printf("\n");           //lefelé tolás
    for(i=0; i<x; i++) printf(" ");           //jobbra tolás
    printf("o\n");        //labda ikonja
}

void usleep(int);
int main(void)
{
    int egyx=1;
    int egyy=-1;
    int i;
    int x=10;    //a labda kezdeti pozíciója
    int y=20;
    int ty[23]; //magasság    // a pálya mérete
    int tx[80]; //szélesség

    //pálya széleinek meghatározása

    for(i=0; i<23; i++)
        ty[i]=1;

    ty[1]=-1;
    ty[22]=-1;

    for(i=0; i<79; i++)
        tx[i]=1;

    tx[1]=-1;
    tx[79]=-1;

    for(;;)
    {
        //címsor és pozíció kijelzése
        for(i=0; i<36; i++)
            printf("_");

        printf("x=%2d", x);
        printf("y=%2d", y);

        for(i=0; i<=35; i++)
            printf("_");
    }
}
```

```
(void) gotoxy(x,y);  
//printf("o\n"); Áthelyezve a gotoxy függvénybe  
  
x+=egy x;  
y+=egy y;  
  
egy x*=tx[x];  
egy y*=ty[y];  
  
usleep (200000);  
(void) system("clear");  
}  
  
}
```

## Szóhossz és a Linus Torvalds féle BogoMIPS

Írj egy programot, ami megnézi, hogy hány bites a szó a gépeden, azaz mekkora az int mérete. Használd ugyanazt a while ciklus fejet, amit Linus Torvalds a BogoMIPS rutinjában!

Megoldás videó:

Megoldás forrása:

```
#include <stdio.h>  
int main()  
{  
    int word = 1;  
    int length = 0;  
  
    do  
    {  
        length++;  
    }  
    while (word<=1);  
  
    printf("A word %d bites\n", length);  
  
    return 0;  
}
```

Bitshifteléssel fogjuk meghatározni a típus méretét és azt, hogy hány bitet foglal. A while ciklusunkban folyamatosan shiftelünk egyet balra a biteken, közben növeljük a length változó értékét. Ezt addig csinál-

juk amíg csupa 0 bitet nem fog tartalmazni a word változónk. Az int típusú változók 4 byte-on tárolódnak, ezért 32 bites lesz a word.

```
kewo@kewo-X540SA:~/Desktop/programs$ gcc lengthofword.c -o word
kewo@kewo-X540SA:~/Desktop/programs$ ./word
A word 32 bites
```

## Helló, Google!

Írj olyan C programot, amely egy 4 honlapból álló hálózatra kiszámolja a négy lap Page-Rank értékét!

Megoldás videó:

Megoldás forrása:

Kezdekör minden oldalnak van egy szavazati pontja és ha az egyik linkeli a másikat, akkor a linkelt oldal megkapja a linkelő pontját. Kereséskor annál előrébb lesz egy oldal, minél több oldal linkel rá és az is számít, hogy a rá linkelő oldalakra mennyi másik oldal linkel.

```
#include <stdio.h>
#include <math.h>

void
kiir (double tomb[], int db){

    int i;

    for (i=0; i<db; ++i){
        printf("%f\n",tomb[i]);
    }
}

double
tavolsag (double PR[], double PRv[], int n){

    int i;
    double osszeg=0;

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

    return sqrt(osszeg);
}

void
pagerank(double T[4][4]){
    double PR[4] = { 0.0, 0.0, 0.0, 0.0 }; //ebbe megy az eredmény
```

```
double PRv[4] = { 1.0/4.0, 1.0/4.0, 1.0/4.0, 1.0/4.0}; //ezzel szorzok

int i, j;

for(;;){

    // ide jön a mátrix művelet

    for (i=0; i<4; i++){
        PR[i]=0.0;
        for (j=0; j<4; j++){
            PR[i] = PR[i] + T[i][j]*PRv[j];
        }
    }

    if (tavolsag(PR,PRv,4) < 0.0000000001)
        break;

    // ide meg az átpakolás PR-ből PRv-be

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

}

kiir (PR, 4);
}

int main (void){
    double L[4][4] = {
        {0.0, 0.0, 1.0/3.0, 0.0},
        {1.0, 1.0/2.0, 1.0/3.0, 1.0},
        {0.0, 1.0/2.0, 0.0, 0.0},
        {0.0, 0.0, 1.0/3.0, 0.0}
    };

    double L1[4][4] = {
        {0.0, 0.0, 1.0/3.0, 0.0},
        {1.0, 1.0/2.0, 1.0/3.0, 0.0},
        {0.0, 1.0/2.0, 0.0, 0.0},
        {0.0, 0.0, 1.0/3.0, 0.0}
    };

    double L2[4][4] = {
        {0.0, 0.0, 1.0/3.0, 0.0},
        {1.0, 1.0/2.0, 1.0/3.0, 0.0},
        {0.0, 1.0/2.0, 0.0, 0.0},
        {0.0, 0.0, 1.0/3.0, 1.0}
    };
};
```



```
printf("\nAz eredeti mátrix értékeivel történő futás:\n");
pagerank(L);

printf("\nAmikor az egyik oldal semmire sem mutat:\n");
pagerank(L1);

printf("\nAmikor az egyik oldal csak magára mutat:\n");
pagerank(L2);

printf("\n");

return 0;
}
```

Az  $L$  tömbben tároljuk a linkelésből szerzett adatokat, melyik oldalra melyik oldal linkel és mennyit. A végtelen ciklusban nullázzuk  $PR$  összes elemét, majd hozzáadjuk az  $L$  mátrix és  $PR_v$  vektor szorzatainak értékét. Ezután a távolság függvényel végigmegyünk a  $PR$  és  $PR_v$  vektorokon és egy változóban eltároljuk ezek különbségének a négyzetét. Ezután gyököt vonva visszaadjuk az értéket, amely ha kisebb mint  $0.00000001$ , akkor kilépünk a végtelen ciklusból, ellenkező esetben pedig  $PR_v$  tömböt feltöltjük  $PR$  elemeivel és kiirattjuk az értékeket.

## 100 éves a Brun tétel

Írj R szimulációt a Brun tétel demonstrálására!

Megoldás videó: <https://youtu.be/xbYhp9G6VqQ>

Megoldás forrása: [https://gitlab.com/nbatfai/bhax/blob/master/attention\\_raising/Primek\\_R](https://gitlab.com/nbatfai/bhax/blob/master/attention_raising/Primek_R)

A prímszám olyan természetes szám, ami csak önmagával és eggyel osztható és minden természetes szám előállítható prímszámok szorzataként. A prímszámok végtelen felé tartanak.

Az ikerprímek olyan prímszámok, amelyek egymás után következnek és különbségük 2.

Brun tétel: az ikerprímek reciprokából képzett összeg konvergál egy számhoz. Ez a határ a Brun konstans.

A Brun tételt ezzel a programmal tudjuk bemutatni:

```
sumTwinPrimes <- function(x) {
  primes = primes(x)
  diff = primes[2:length(primes)] - primes[1:length(primes)-1]
  idx = which(diff==2)
  t1primes = primes[idx]
  t2primes = primes[idx]+2
  rt1plust2 = 1/t1primes+1/t2primes
  return(sum(rt1plust2))
}
```

```
x=seq(13, 1000000, by=10000)
y=apply(x, FUN = sumTwinPrimes)
plot(x,y,type="b")
```

Létrehozunk egy függvényt aminek megadjuk, hogy meddig számolja az ikerprímszámokat. A `primes(x)` függvény által generáljuk a prímszámokat majd elmentjük őket `primes` változóba. Az egymásután következő prímszámok különbségét `diff` be tároljuk. Ahol ikerprímeket találunk, azaz a `diff=2`, elmentjük `idx` be.

A `t1primes` be kerülnek az ikerprímek első párja, a `t2primes` be pedig a második. Az `rt1plust2` be tároljuk minden párnak a reciprokainak az összegét, majd ezeket összeadjuk és végül visszaadjuk ezt az értéket.

## A Monty Hall probléma

Írj R szimulációt a Monty Hall problémára!

Megoldás videó: [https://bhaxor.blog.hu/2019/01/03/erdos\\_pal\\_mit\\_keresett\\_a\\_nagykonyvben\\_a\\_monty\\_hall-paradoxon\\_kapcsan](https://bhaxor.blog.hu/2019/01/03/erdos_pal_mit_keresett_a_nagykonyvben_a_monty_hall-paradoxon_kapcsan)

Megoldás forrása: [https://gitlab.com/nbatfai/bhax/tree/master/attention\\_raising/MontyHall\\_R](https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/MontyHall_R)

A versenyzőnek ki kell választani a három ajtó közül egyet és a nyereménye az, amit az ajtó rejt. Az egyik ajtó mögött egy értékes nyeremény van, a másik két ajtó mögött pedig semmi. A versenyző választ egy ajtót, majd ezután a műsorvezető kinyit egy olyan ajtót, ami mögött semmi sincs. Ezek után a műsorvezető megkérdezi a játékost, hogy kitart-e az eredeti választása mellett, vagy átvált a másik ajtóra.

Kezdetben a játékos 1 a 3-hoz valószínűséggel választja ki a 3 ajtó közül azt, ami mögött az értékes nyeremény van és 2 a 3-hoz valószínűséggel választ olyan ajtót, ami mögött semmi sincs. Ezután a műsorvezető egy olyan ajtót nyit ki, ami mögött tudja, hogy semmi sincs. A kérdés, hogy ezek után mekkora eséllyel van a játékos által kiválasztott ajtó mögött a nyeremény és mekkora eséllyel van a másik megmaradt ajtó mögött.

A következő programmal fogjuk demonstrálni a megoldást:

```
kiserletek_szama=1000000
kiserlet = sample(1:3, kiserletek_szama, replace=T)
jatekos = sample(1:3, kiserletek_szama, replace=T)
musorvezeto=vector(length = kiserletek_szama)
for (i in 1:kiserletek_szama) {
  if(kiserlet[i]==jatekos[i]){

    mibol=setdiff(c(1,2,3), kiserlet[i])

  }else{

    mibol=setdiff(c(1,2,3), c(kiserlet[i], jatekos[i]))

  }
  musorvezeto[i] = mibol[sample(1:length(mibol),1)]
}
```

```
}
nemvaltoztatesnyer= which(kiserlet==jatekos)
valtoztat=vector(length = kiserletek_szama)
for (i in 1:kiserletek_szama) {
  holvalt = setdiff(c(1,2,3), c(musorvezeto[i], jatekos[i]))
  valtoztat[i] = holvalt[sample(1:length(holvalt),1)]
}
valtoztatesnyer = which(kiserlet==valtoztat)
sprintf("Kiserletek szama: %i", kiserletek_szama)
length(nemvaltoztatesnyer)
length(valtoztatesnyer)
length(nemvaltoztatesnyer)/length(valtoztatesnyer)
length(nemvaltoztatesnyer)+length(valtoztatesnyer)
```

## 3. fejezet

# Helló, Chomsky!

### Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfiájával megadva írd meg ezt a gépet!

Megoldás videó:

Megoldás forrása:

3.1. ábra. Turing gép

Ez a turing gép végigmegy a beolvasandó adatokon, ha egyenlőség jelet talál , akkor kivon egyet az előtte lévő számból. Ez a folyamat addig ismétlődik amíg az előtte lévő szám le nem nullázódik. Mivel itt 0 van, de előtte 1 van, ezért a 0-ból 9 lesz és az 1-ből 0. A kivont egyeseket kiírja a tárolóra, ezért a végén egy egyesekből álló sorozatot kapunk. Az ebben a sorozatban lévő egyesek száma megegyezik a kiindulási szám értékével.

### Az $a^n b^n c^n$ nyelv nem környezetfüggetlen

Mutass be legalább két környezetfüggő generatív grammatikát, amely ezt a nyelvet generálja!

Megoldás videó:

Megoldás forrása:

változók: A, B, C

konstansok: a, b, c

képzési szabályok:

$A \rightarrow aAB$ ,  $A \rightarrow aC$ ,  $CB \rightarrow bCc$ ,  $cB \rightarrow Bc$ ,  $C \rightarrow bc$

A kezdő szimbólum az A.

A (A  $\rightarrow$  aAB)

aAB ( A  $\rightarrow$  aAB)

```

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

```

1-es típusú (környezetfüggő) nyelvtan, ha minden szabálya  $\alpha A \gamma \rightarrow \alpha \beta \gamma$  alakú, ahol  $A \in N, \alpha, \gamma \in (NUT)^*, \beta \in (NUT)^+$ . Ezenkívül megengedhető az  $S \rightarrow \epsilon$ s szabály is, ha  $S$  nem szerepel egyetlen szabály jobb oldalán sem.

változók:  $S, X, Y$

konstansok:  $a, b, c$

képzési szabályok:

$S \rightarrow abc, S \rightarrow aXbc, Xb \rightarrow bX, Xc \rightarrow Ybcc, bY \rightarrow Yb, aY \rightarrow aaX, aY \rightarrow aa$

A kezdő szimbólum az  $S$ .

```

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

```

## Hivatkozási nyelv

A [[KERNIGHANRITCHIE](#)] könyv C referencia-kézikönyv/Utasítások melléklete alapján definiáld BNF-ben a C utasítás fogalmát! Majd mutass be olyan kódcsipeteket, amelyek adott szabvánnyal nem fordulnak (például C89), mással (például C99) igen.

Megoldás videó:

Megoldás forrása:

```

<utasítás> ::=
    <összetett_utasítás> | <feltételes_utasítás> | <iterációs_utasítás> |
    <vezérlés_átadó_utasítás> | <címkézett_utasítás> | <kifejezés_utasítás> |
    <nulla_utasítás>
<összetett_utasítás> ::= <deklaráció_lista> | <utasításlista>
<deklaráció_lista> ::= <deklaráció>
<utasításlista> ::= <utasítás>
<feltételes_utasítás> ::= if | if else | switch
<iterációs_utasítás> ::= while | do while | for
<vezérlés_átadó_utasítás> ::= break | return | goto | continue
<címkézett_utasítás> ::= <azonosító>

```

```
<kifejezés_utasítás> ::= <kifejezés>  
<nulla_utasítás> ::= ;
```

```
#include <stdio.h>  
int main ()  
{  
    // asd  
    printf ("Hello World\n");  
}
```

C99-el lefordul:

```
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$ gcc hello.c -o hello -std= ↵  
c99  
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$
```

C89-el nem fordul le:

```
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$ gcc hello.c -o hello -std= ↵  
c89  
hello.c: In function 'main':  
hello.c:4:7: error: C++ style comments are not allowed in ISO C90  
    // asd  
    ^  
hello.c:4:7: error: (this will be reported only once per input file)  
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$
```

## Saját lexikális elemző

Írj olyan programot, ami számolja a bemenetén megjelenő valós számokat! Nem elfogadható olyan megoldás, amely maga olvassa betűnként a bemenetet, a feladat lényege, hogy lexert használjunk, azaz óriások vállán álljunk és ne kispályázzunk!

Megoldás videó:

Megoldás forrása:

Ezt a programot lexer segítségével fogjuk megoldani. A lexer egy általunk megadott minta alapján fogja legenerálni a lexikális elemzőnket.

```
%{  
#include <stdio.h>
```

```
int realnumbers = 0;
%}
digit [0-9]
%%
{digit}* (\. {digit}+)? {++realnumbers;
    printf("[realnum=%s %f]", yytext, atof(yytext));}
%%
int
main ()
{
    yylex ();
    printf("The number of real numbers is %d\n", realnumbers);
    return 0;
}
```

Először is létrehozunk egy int változót amiben fogjuk számolni a valós számok előfordulását. Ezután pedig definiáljuk digit néven 0-9 ig a számjegyeket.

Az ezt követő részben adjuk meg a mintát. A digit után a csillag, azt jelenti, hogy digitből bármennyi lehet. A . azt jelenti, hogy bármilyen karakterre rá lehet illeszteni, azonban nekünk le kell védeni \ jellel és így a valós számoknál lévő pontot fogja értelmezni. Ezután megint jönnek a számjegyek, a + pedig azt jelzi, hogy legalább 1 számnak kell lennie a pont után. Ha van találat a mintára, akkor növeljük a realnumbers változót. Ezután kiíratjuk a felismert számot, illetve az atof al átkonvertált double verzióját is. Magában a programban meghívjuk a lexert majd pedig kiíratjuk a valós számok számát. Fordításkor hozzá kell csatolnunk a flex könyvtárat.

```
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$ lex -o valos_szam.c ↵
    valos_szam.l
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$ gcc valos_szam.c -o ↵
    valos_szam -lfl
```

```
kewo@kewo-X540SA:~/Downloads/wat/bhax/thematic_tutorials/bhax_textbook/ ↵
    programs$ ./valos_szam
2324.342 23132 43333 342.232
[realnum=2324.342 2324.342000] [realnum=23132 23132.000000] [realnum=43333 ↵
    43333.000000] [realnum=342.232 342.232000]
The number of real numbers is 4
```

## I33t.l

Lexelj össze egy I33t ciphert!

Megoldás videó:

### Megoldás forrása:

Az előző feladathoz hasonlóan a lexer egy általunk megadott minta alapján jogja létrehozni a lexikális elemzőnket. Ebben a feladatban a begépelt szöveget leet nyelvre fogjuk konvertálni. A leet nyelv az online világ szlengje. Elsősorban üzenetek titkosítására szolgált, majd később a játékosok neveiben fordult leginkább elő.

```
/*
Fordítás:
$ lex -o l337d1c7.c l337d1c7.l
```

```
Futtatás:
$ gcc l337d1c7.c -o l337d1c7 -lfl
(kilépés az input vége, azaz Ctrl+D)
```

```
Copyright (C) 2019
Norbert Bátfai, batfai.norbert@inf.unideb.hu
```

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<https://www.gnu.org/licenses/>>.

```
*/
%{
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>

#define L337SIZE (sizeof l337d1c7 / sizeof (struct cipher))

struct cipher {
    char c;
    char *leet[4];
} l337d1c7 [] = {

    {'a', {"4", "4", "@", "/-\\"}},
    {'b', {"b", "8", "|3", "|"}},
    {'c', {"c", "(", "<", "{"}},
    {'d', {"d", "|)", "|]", "|"}},
    {'e', {"3", "3", "3", "3"}},
    {'f', {"f", "|=", "ph", "#"}},
```



```
{'g', {"g", "6", "[", "[+"}},
{'h', {"h", "4", "|-|", "[-"]}},
{'i', {"i", "1", "1", "|", "!"}},
{'j', {"j", "7", "_|", "_/"}},
{'k', {"k", "<", "1<", "|{"}},
{'l', {"l", "1", "1", "|", "|_"}},
{'m', {"m", "44", "(V)", "\\|"}},
{'n', {"n", "\\|", "/\\", "/V"}},
{'o', {"o", "0", "0", "()", "[]"}},
{'p', {"p", "/o", "|D", "|o"}},
{'q', {"q", "9", "O_", "(,)"}}},
{'r', {"r", "12", "12", "|2"}},
{'s', {"s", "5", "$", "$"}},
{'t', {"t", "7", "7", "'|'"}},
{'u', {"u", "|_|", "(_)", "[_]"}},
{'v', {"v", "\\|", "\\|", "\\|"}},
{'w', {"w", "VV", "\\|\\|", "(/\\|"}},
{'x', {"x", "%", ")(", ")("}},
{'y', {"y", "", "", ""}},
{'z', {"z", "2", "7_", ">_"}}},
```

```
{'0', {"D", "0", "D", "0"}},
{'1', {"I", "I", "L", "L"}},
{'2', {"Z", "Z", "Z", "e"}},
{'3', {"E", "E", "E", "E"}},
{'4', {"h", "h", "A", "A"}},
{'5', {"S", "S", "S", "S"}},
{'6', {"b", "b", "G", "G"}},
{'7', {"T", "T", "j", "j"}},
{'8', {"X", "X", "X", "X"}},
{'9', {"g", "g", "j", "j"}}}
```

```
// https://simple.wikipedia.org/wiki/Leet
};
```

```
%}
```

```
%%
```

```
. {
```

```
int found = 0;
for(int i=0; i<L337SIZE; ++i)
{
    if(l337d1c7[i].c == tolower(*yytext))
    {
        int r = 1+(int) (100.0*rand()/(RAND_MAX+1.0));

        if(r<91)
            printf("%s", l337d1c7[i].leet[0]);
    }
}
```

```
        else if(r<95)
            printf("%s", l337d1c7[i].leet[1]);
        else if(r<98)
            printf("%s", l337d1c7[i].leet[2]);
        else
            printf("%s", l337d1c7[i].leet[3]);

        found = 1;
        break;
    }

}

if(!found)
    printf("%c", *yytext);

}

%%
int
main()
{
    srand(time(NULL)+getpid());
    yylex();
    return 0;
}
```

```
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$ ./l337d1c7
kewo
k3w0
Koos Zoltan
k00s >_0lt4n
```

## A források olvasása

Hogyan olvasod, hogyan értelmezed természetes nyelven az alábbi kódcsipeteket? Például

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

Ha a SIGINT jel kezelése figyelmen kívül volt hagyva, akkor ezen túl is legyen figyelmen kívül hagyva, ha nem volt figyelmen kívül hagyva, akkor a jelkezelő függvény kezelje. (Miután a **man 7 signal** lapon megismertem a SIGINT jelet, a **man 2 signal** lapon pedig a használt rendszerhívást.)



### Bugok

Vigyázz, sok csipet kerülendő, mert bugokat visz a kódba! Melyek ezek és miért? Ha nem megyránézésre, elkapja valamelyiket esetleg a splint vagy a frama?

i.

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

Ha a SIGINT jel kezelése nem volt figyelmen kívül hagyva, akkor a jelkezelő függvény kezelje, máskülönben figyelmen kívül legyen hagyva.

ii.

```
for(i=0; i<5; ++i)
```

A for ciklussal 0-tól 4-ig megyünk, az i-t növelve. Az ++i megnöveli az i értékét majd visszaadja az 1-el megnövelt értéket.

iii.

```
for(i=0; i<5; i++)
```

A for ciklussal 0-tól 4-ig megyünk, az i-t növelve. Az i++ először visszaadja az eredeti értéket majd megnöveli az i értékét.

iv.

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

A for ciklussal 0-tól 4-ig megyünk, közben tomb i edik elemét megváltoztatjuk arra az i-re ami megnövelés előtt volt.

```
kewo@kewo-X540SA:~/Downloads/wat/bhax/programs$ splint wat4.c
Splint 3.1.2 --- 03 May 2009
```

```
wat4.c: (in function main)
```

```
wat4.c:12:28: Expression has undefined behavior (left operand uses i, ←
    modified
```

```
        by right operand): tomb[i] = i++
```

```
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)
```

```
Finished checking --- 1 code warning
```

v.

```
for(i=0; i<n && (*d++ = *s++); ++i)
```

A for ciklussal 0-tól megyünk, addig amíg i kisebb mint n és a d mutató által mutatott tömb eleme egyenlő az s mutató által mutatott tömb elemével.

```
wat4.c:13:19: Right operand of && is non-boolean ( ←
    int): i < n && (*d++ = *s++)
```

```
The operand of a boolean operator is not a boolean.
```

vi.

```
printf("%d %d", f(a, ++a), f(++a, a));
```

Kiíratunk két egész számot. Mindkettőt az  $f$  függvény adja vissza. Először az  $f$  függvénynek átadjuk az a változót és az a változó 1-el megnövelt értékét. Másodszor pedig az  $f$ -nek, az a változó 1-el megnövelt értékét és  $a$ -t adjuk.

```
wat4.c:13:19: Argument 2 modifies a, used by argument 3 (order of ↵
  evaluation of
    actual parameters is undefined): printf("%d %d", f(a, ++a), 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)
```

vii.

```
printf("%d %d", f(a), a);
```

Két egész számot íratunk ki, az egyik az  $f$  függvény a változó felhasználása által visszaadott szám, a másik pedig az a változó.

viii.

```
printf("%d %d", f(&a), a);
```

Két egész számot íratunk ki, az egyik az  $f$  függvény a változó memóriacímének felhasználásával visszaadott szám, a másik pedig az a változó.

Megoldás forrása:

Megoldás videó:

## Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

```
 $\$(\forall x \exists y ((x < y) \wedge (y \text{ prím})))\$$ 
```

```
 $\$(\forall x \exists y ((x < y) \wedge (y \text{ prím})) \wedge (\exists y \text{ prím})) \leftrightarrow$   
 $)\$$ 
```

```
 $\$(\exists y \forall x (x \text{ prím}) \supset (x < y)) \$$ 
```

```
 $\$(\exists y \forall x (y < x) \supset \neg (x \text{ prím}))\$$ 
```

Megoldás forrása: [https://gitlab.com/nbatfai/bhax/blob/master/attention\\_raising/MatLog\\_LaTeX](https://gitlab.com/nbatfai/bhax/blob/master/attention_raising/MatLog_LaTeX)

Megoldás videó: <https://youtu.be/ZexiPy3ZxsA>, [https://youtu.be/AJSXOQFF\\_wk](https://youtu.be/AJSXOQFF_wk)

A feladatban szereplő kifejezések jelentése:

```
-forall: univerzális kvantor  
-exist: egzisztenciális kvantor  
-wedge: implikáció  
-supset: konjukció  
-neg: negáció  
-S: rákövetkező
```

Ezeket a kifejezéseket felhasználva az alábbi mondatokat állíthatjuk elő:

1. Minden  $x$ -hez létezik olyan  $y$ , amelynél ha  $x$  kisebb, akkor  $y$  prím.
2. Minden  $x$ -hez létezik olyan  $y$ , amelynél ha  $x$  kisebb, akkor  $y$  prím, és  $\leftrightarrow$  ha  $y$  prím, akkor annak második rákövetkezője is prím.
3. Létezik olyan  $y$ , amelyhez minden  $x$  esetén az  $x$  prím, és  $x$  kisebb, mint  $\leftrightarrow y$ .
4. Létezik olyan  $y$ , amelyhez minden  $x$  esetén az  $x$  nagyobb, és  $x$  nem prím.

## Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

- egész
- egészre mutató mutató
- egész referenciája
- egészek tömbje
- egészek tömbjének referenciája (nem az első elemé)
- egészre mutató mutatók tömbje
- egészre mutató mutatót visszaadó függvény
- egészre mutató mutatót visszaadó függvényre mutató mutató
- egészet visszaadó és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó függvény
- függvénymutató egy egészet visszaadó és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó függvényre

```
#include <iostream>

int i = 0;
int a = 5;
int *b = &a;
int &r = a;
int c[5] = {0,1,2,3,4};
int (&tr)[5] = c;
int *d[5];
int *h ();
int *(*l) ();
int (*v (int c)) (int a, int b);
int (*(z) (int)) (int, int);

int main()
{

    return 0;
}
```

Mit vezetnek be a programba a következő nevek?

- `int a;`  
Egy `int` típusú a nevű változó.
- `int *b = &a;`  
`b` egésze mutató mutató `a`-ra mutat.
- `int &r = a;`  
`r` egésze mutató mutató, ami a címet tartalmazza.
- `int c[5];`  
Öt elemű egészekből álló tömb.
- `int (&tr)[5] = c;`  
Öt elemű egészekből álló tömbre mutató mutató, ami a `c` tömbre mutat.
- `int *d[5];`  
5 elemű egészekre mutató mutatókból álló tömb.
- `int *h ();`

Egy egészszel visszatérő paraméter nélküli függvényre mutató mutató.

- ```
int *(*l) ();
```

Egy egészre mutató mutatóval visszatérő, paraméter nélküli függvényre mutató mutató.

- ```
int (*v (int c)) (int a, int b)
```

Egy egészszel visszatérő 2 egészet váró függvényre mutató mutatóval visszatérő 2 egészet váró függvény.

- ```
int ((*z) (int)) (int, int);
```

Függvénymutató egy egészet visszaadó és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó függvényre mutató mutató.

Megoldás videó:

Megoldás forrása:

## 4. fejezet

# Helló, Caesar!

### int \*\*\* háromszögmátrix

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### C EXOR törő

Írj egy olyan C programot, amely megtöri az első feladatban előállított titkos szövegeket!

Megoldás videó:



Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## Neurális OR, AND és EXOR kapu

R

Megoldás videó: <https://youtu.be/Koyw6IH5ScQ>

Megoldás forrása: [https://gitlab.com/nbatfai/bhax/tree/master/attention\\_raising/NN\\_R](https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/NN_R)

Tanulságok, tapasztalatok, magyarázat...

## Hiba-visszaterjesztéses perceptron

C++

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 5. fejezet

# Helló, Mandelbrot!

### A Mandelbrot halmaz

Megoldás videó:

Megoldás forrása:

### A Mandelbrot halmaz a `std::complex` osztállyal

Megoldás videó:

Megoldás forrása:

### Biomorfok

Megoldás videó: <https://youtu.be/IJMbgRzY76E>

Megoldás forrása: [https://gitlab.com/nbatfai/bhax/tree/master/attention\\_raising/Biomorf](https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/Biomorf)

Tanulságok, tapasztalatok, magyarázat...

### A Mandelbrot halmaz CUDA megvalósítása

Megoldás videó:

Megoldás forrása:

### Mandelbrot nagyító és utazó C++ nyelven

Építs GUI-t a Mandelbrot algoritmusra, lehessen egérrel nagyítani egy területet, illetve egy pontot egérrel kiválasztva vizualizálja onnan a komplex iteráció bejárta  $z_n$  komplex számokat!

Megoldás forrása:

Megoldás videó:

Megoldás forrása:

## Mandelbrot nagyító és utazó Java nyelven

DRAFT

## 6. fejezet

# Helló, Welch!

### Első osztályom

Valósítsd meg C++-ban és Java-ban az módosított polártranszformációs algoritmust! A matek háttér teljesen irreleváns, csak annyiban érdekes, hogy az algoritmus egy számítása során két normálist számol ki, az egyiket elspájzolod és egy további logikai taggal az osztályban jelzed, hogy van vagy nincs eltérve kiszámolt szám.

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat... térj ki arra is, hogy a JDK forrásaiban a Sun programozói pont úgy csinálták meg ahogyan te is, azaz az OO nemhogy nem nehéz, hanem éppen természetes neked!

### LZW

Valósítsd meg C-ben az LZW algoritmus fa-építését!

Megoldás videó:

Megoldás forrása:

### Fabejárás

Járd be az előző (inorder bejárású) fát pre- és posztorder is!

Megoldás videó:

Megoldás forrása:

### Tag a gyökér

Az LZW algoritmust ültesd át egy C++ osztályba, legyen egy Tree és egy beágyazott Node osztálya. A gyökér csomópont legyen kompozícióban a fával!

Megoldás videó:

Megoldás forrása:

## Mutató a gyökér

Írd át az előző forrást, hogy a gyökér csomópont ne kompozícióban, csak aggregációban legyen a fával!

Megoldás videó:

Megoldás forrása:

## Mozgató szemantika

Írj az előző programhoz mozgató konstruktort és értékadást, a mozgató konstruktor legyen a mozgató értékadásra alapozva!

Megoldás videó:

Megoldás forrása:

## 7. fejezet

# Helló, Conway!

### Hangyaszimulációk

Írj Qt C++-ban egy hangyaszimulációs programot, a forrásaidról utólag reverse engineering jelleggel készíts UML osztálydiagramot is!

Megoldás videó: <https://bhaxor.blog.hu/2018/10/10/myrmecologist>

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### Java életjáték

Írd meg Java-ban a John Horton Conway-féle életjátékot, valósítsa meg a sikló-kilövőt!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## BrainB Benchmark

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

DRAFT

## 8. fejezet

# Helló, Schwarzenegger!

### Szoftmax Py MNIST

aa Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### Szoftmax R MNIST

R

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### Mély MNIST

Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### Deep dream

Keras



Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## Robotpszichológia

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

DRAFT

## 9. fejezet

# Helló, Chaitin!

### Iteratív és rekurzív faktoriális Lisp-ben

Megoldás videó:

Megoldás forrása:

### Weizenbaum Eliza programja

Éleszd fel Weizenbaum Eliza programját!

Megoldás videó:

Megoldás forrása:

### Gimp Scheme Script-fu: króm effekt

Írj olyan script-fu kiterjesztést a GIMP programhoz, amely megvalósítja a króm effektet egy bemenő szövegre!

Megoldás videó: [https://youtu.be/OKdAkI\\_c7Sc](https://youtu.be/OKdAkI_c7Sc)

Megoldás forrása: [https://gitlab.com/nbatfai/bhax/tree/master/attention\\_raising/GIMP\\_Lisp/Chrome](https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/GIMP_Lisp/Chrome)

Tanulságok, tapasztalatok, magyarázat...

### Gimp Scheme Script-fu: név mandala

Írj olyan script-fu kiterjesztést a GIMP programhoz, amely név-mandalát készít a bemenő szövegből!

Megoldás videó: [https://bhaxor.blog.hu/2019/01/10/a\\_gimp\\_lisp\\_hackelese\\_a\\_scheme\\_programozasi\\_nyelv](https://bhaxor.blog.hu/2019/01/10/a_gimp_lisp_hackelese_a_scheme_programozasi_nyelv)

Megoldás forrása: [https://gitlab.com/nbatfai/bhax/tree/master/attention\\_raising/GIMP\\_Lisp/Mandala](https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/GIMP_Lisp/Mandala)

Tanulságok, tapasztalatok, magyarázat...

## Lambda

Hasonlítsd össze a következő programokat!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## Omega

Megoldás videó:

Megoldás forrása:

DRAFT

## **III. rész**

### **Második felvonás**

DRAFT

**Bátf41 Haxor Stream**

A feladatokkal kapcsolatos élő adásokat sugároz a <https://www.twitch.tv/nbatfai> csatorna, melynek permanens archívuma a <https://www.youtube.com/c/nbatfai> csatornán található.

---

DRAFT

---

## 10. fejezet

# Helló, Arroway!

### A BPP algoritmus Java megvalósítása

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### Java osztályok a Pi-ben

Az előző feladat kódját fejleszd tovább: vizsgáld, hogy Vannak-e Java osztályok a Pi hexadecimális kifejtésében!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## **IV. rész**

### **Irodalomjegyzék**

DRAFT

## Általános

[MARX] Marx, György, *Gyorsuló idő*, Typotex , 2005.

## C

[KERNIGHANRITCHIE] Kernighan, Brian W. És Ritchie, Dennis M., *A C programozási nyelv*, Bp., Műszaki, 1993.

## C++

[BMECPP] Benedek, Zoltán És Levendovszky, Tihamér, *Szoftverfejlesztés C++ nyelven*, Bp., Szak Kiadó, 2013.

## Lisp

[METAMATH] Chaitin, Gregory, *META MATH! The Quest for Omega*, [http://arxiv.org/PS\\_cache/math/pdf/0404/0404335v7.pdf](http://arxiv.org/PS_cache/math/pdf/0404/0404335v7.pdf) , 2004.

Köszönet illeti a NEMESPOR, <https://groups.google.com/forum/#!forum/nemespor>, az UDPROG tanulószoba, <https://www.facebook.com/groups/udprog>, a DEAC-Hackers előszoba, <https://www.facebook.com/groups/DEACHackers> (illetve egyéb alkalmi szerveződésű szakmai csoportok) tagjait inspiráló érdeklődésükért és hasznos észrevételeikért.

Ezen túl kiemelt köszönet illeti az említett UDPROG közösséget, mely a Debreceni Egyetem reguláris programozás oktatása tartalmi szervezését támogatja. Sok példa eleve ebben a közösségben született, vagy itt került említésre és adott esetekben szerepet kapott, mint oktatási példa.