

# 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 Tóth, László Szilárd	2020. október 15.	

**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

# 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]

# 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?	5
2.3. Változók értékének felcserélése	7
2.4. Labdapattogás	7
2.5. Szóhossz és a Linus Torvalds féle BogomIPS	7
2.6. Helló, Google!	7
2.7. 100 éves a Brun tétel	8
2.8. A Monty Hall probléma	8
<b>3. Helló, Chomsky!</b>	<b>9</b>
3.1. Decimálisból unárisba átváltó Turing gép	9
3.2. Az $a^n b^n c^n$ nyelv nem környezetfüggetlen	9
3.3. Hivatkozási nyelv	9
3.4. Saját lexikális elemző	10
3.5. l33t.1	10
3.6. A források olvasása	10
3.7. Logikus	11
3.8. Deklaráció	11

<b>4. Helló, Caesar!</b>	<b>13</b>
4.1. int *** háromszögmátrix . . . . .	13
4.2. C EXOR titkosító . . . . .	13
4.3. Java EXOR titkosító . . . . .	13
4.4. C EXOR törő . . . . .	13
4.5. Neurális OR, AND és EXOR kapu . . . . .	14
4.6. Hiba-visszaterjesztéses perceptron . . . . .	14
<b>5. Helló, Mandelbrot!</b>	<b>15</b>
5.1. A Mandelbrot halmaz . . . . .	15
5.2. A Mandelbrot halmaz a <code>std::complex</code> osztállyal . . . . .	15
5.3. Biomorfok . . . . .	15
5.4. A Mandelbrot halmaz CUDA megvalósítása . . . . .	15
5.5. Mandelbrot nagyító és utazó C++ nyelven . . . . .	15
5.6. Mandelbrot nagyító és utazó Java nyelven . . . . .	16
<b>6. Helló, Welch!</b>	<b>17</b>
6.1. Első osztályom . . . . .	17
6.2. LZW . . . . .	17
6.3. Fabejárás . . . . .	17
6.4. Tag a gyökér . . . . .	17
6.5. Mutató a gyökér . . . . .	18
6.6. Mozgató szemantika . . . . .	18
<b>7. Helló, Conway!</b>	<b>19</b>
7.1. Hangyaszimulációk . . . . .	19
7.2. Java életjáték . . . . .	19
7.3. Qt C++ életjáték . . . . .	19
7.4. BrainB Benchmark . . . . .	20
<b>8. Helló, Schwarzenegger!</b>	<b>21</b>
8.1. Szoftmax Py MNIST . . . . .	21
8.2. Szoftmax R MNIST . . . . .	21
8.3. Mély MNIST . . . . .	21
8.4. Deep dream . . . . .	21
8.5. Robotpszichológia . . . . .	22

<b>9. Helló, Chaitin!</b>	<b>23</b>
9.1. Iteratív és rekurzív faktoriális Lisp-ben . . . . .	23
9.2. Weizenbaum Eliza programja . . . . .	23
9.3. Gimp Scheme Script-fu: króm effekt . . . . .	23
9.4. Gimp Scheme Script-fu: név mandala . . . . .	23
9.5. Lambda . . . . .	24
9.6. Omega . . . . .	24
 <b>III. Második felvonás</b>	 <b>25</b>
<b>10. Helló, Arroway!</b>	<b>27</b>
10.1. OO szemlélet . . . . .	27
10.2. „Gagyí” . . . . .	27
10.3. Yoda . . . . .	28
10.4. Homokózó . . . . .	28
<b>11. Helló, Liskov!</b>	<b>29</b>
11.1. Anti OO . . . . .	29
11.2. Szülő-gyerek . . . . .	30
11.3. Liskov helyettesítés sértése . . . . .	30
11.4. EPAM: Interfész evolúció Java ban . . . . .	31
<b>12. Helló, Mandelbrot!</b>	<b>33</b>
12.1. a . . . . .	33
 <b>IV. Irodalomjegyzék</b>	 <b>35</b>
12.2. Általános . . . . .	36
12.3. C . . . . .	36
12.4. C++ . . . . .	36
12.5. Lisp . . . . .	36

# 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 (hogyan lássuk mást is) példával.

## Hogyan nyomjuk?

Rántsd 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.xsl
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**

DRAFT

# 1. fejezet

## Vízió

### 1.1. Mi a programozás?

### 1.2. 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.

### 1.3. 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!

### 2.1. 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:

Tanulságok, tapasztalatok, magyarázat...

### 2.2. 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 építő 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? Sehogyz, mert ilyen `Lefagy` függvényt, azaz a T100 program nem is létezik.

Tanulságok, tapasztalatok, magyarázat...

## 2.3. 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:

Tanulságok, tapasztalatok, magyarázat...

## 2.4. Labdapattogás

Először if-ekkel, majd bármiféle logikai utasítás vagy kifejezés használata nélkül írd 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:

Tanulságok, tapasztalatok, magyarázat...

## 2.5. Szóhossz és a Linus Torvalds féle BogomIPS

Írd 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:

Tanulságok, tapasztalatok, magyarázat...

## 2.6. Helló, Google!

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...



## 2.7. 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)

## 2.8. 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)

Tanulságok, tapasztalatok, magyarázat...

## 3. fejezet

# Helló, Chomsky!

### 3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfiájával megadva írd meg ezt a gépet!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 3.2. 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:

Tanulságok, tapasztalatok, magyarázat...

### 3.3. 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:

Tanulságok, tapasztalatok, magyarázat...

### 3.4. 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:

Tanulságok, tapasztalatok, magyarázat...

### 3.5. l33t.l

Lexelj össze egy l33t ciphert!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 3.6. A források olvasása

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

```
if(signal(SIGINT, jelkezeslo)==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 jelkezeslo függvény kezelje. (Miótan 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 megyránzésre, elkapja valamelyiket esetleg a splint vagy a frama?

i.

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

ii.

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

iii.

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

iv. 

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

v. 

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

vi. 

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

vii. 

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

viii. 

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

Megoldás forrása:

Megoldás videó:

Tanulságok, tapasztalatok, magyarázat...

## 3.7. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

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

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

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

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

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)

Tanulságok, tapasztalatok, magyarázat...

## 3.8. 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

Mit vezetnek be a programba a következő nevek?

- `int a;`
- `int *b = &a;`
- `int &r = a;`
- `int c[5];`
- `int (&tr)[5] = c;`
- `int *d[5];`
- `int *h ();`
- `int *(*l) ();`
- `int (*v (int c)) (int a, int b)`
- `int (*(z) (int)) (int, int);`

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 4. fejezet

# Helló, Caesar!

### 4.1. `int ***` háromszögmátrix

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 4.4. C EXOR törő

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

Megoldás videó:

---

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 4.5. 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...

## 4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 5. fejezet

# Helló, Mandelbrot!

### 5.1. A Mandelbrot halmaz

Megoldás videó:

Megoldás forrása:

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

Megoldás videó:

Megoldás forrása:

### 5.3. 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...

### 5.4. A Mandelbrot halmaz CUDA megvalósítása

Megoldás videó:

Megoldás forrása:

### 5.5. 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:

## 5.6. Mandelbrot nagyító és utazó Java nyelven

DRAFT

## 6. fejezet

# Helló, Welch!

### 6.1. 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!

### 6.2. LZW

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

Megoldás videó:

Megoldás forrása:

### 6.3. Fabejárás

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

Megoldás videó:

Megoldás forrása:

### 6.4. 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:

## 6.5. Mutató a gyökér

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

Megoldás videó:

Megoldás forrása:

## 6.6. 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!

### 7.1. 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...

### 7.2. Java életjáték

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 7.3. Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 7.4. BrainB Benchmark

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

DRAFT

## 8. fejezet

# Helló, Schwarzenegger!

### 8.1. Szoftmax Py MNIST

aa Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 8.2. Szoftmax R MNIST

R

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 8.3. Mély MNIST

Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

### 8.4. Deep dream

Keras

---

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 8.5. Robotpszichológia

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

DRAFT

## 9. fejezet

# Helló, Chaitin!

### 9.1. Iteratív és rekurzív faktoriális Lisp-ben

Megoldás videó:

Megoldás forrása:

### 9.2. Weizenbaum Eliza programja

Éleszd fel Weizenbaum Eliza programját!

Megoldás videó:

Megoldás forrása:

### 9.3. 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...

### 9.4. 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...

---



## 9.5. Lambda

Hasonlítsd össze a következő programokat!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 9.6. 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!

### 10.1. OO szemlélet

A módosított polártranszformációs normális generátor beprogramozása Java nyelven. Mutassunk rá, hogy a mi természetes saját megoldásunk (az algoritmus egyszerre két normálist állít elő, kell egy példánytag, amely a nem visszaadottat tárolja és egy logikai tag, hogy van-e tárolt vagy futtatni kell az algorit.) és az OpenJDK, Oracle JDK-ban a Sun által adott OO szervezés ua. <https://arato.inf.unideb.hu/batfai.norbert/UDPROG> 22 fólia) Ugyanezt írjuk meg C++ nyelven is! (lásd még UDPROG repó: [source/labor/polargen](#))

Megoldás videó:

Megoldás forrása: <https://github.com/grestemayster/prog2/tree/master/Hell%C3%B3%20Arroway!/-/oo>

Az algoritmus elkészítése "public" "PolárGenerátor" osztályban történik. Deklarálunk egy "nincsTárolt" változót is, amely "boolean" típusú. Deklarálunk egy "tárolt" változót is, amely "double" típusú. Deklarálunk egy "következő" "double" típusú fgv-t is. Amennyiben a "nincsTárolt" igaz, akkor egy do-while cikluson belül a "Math.random+" függvényt használva "u1" illetve "u2" egy véletlen értéket kap, addig ameddig "w" nagyobb mint 1. "v1" illetve "v2" megkapja az előző változók értékeit, ahogy "w" is. Utána kiszámoljuk az "r" értékét is, ahol már a "tárolt" változónk is értéket kap. Ha hamis, akkor a fgv a már eltárolt elemet adja vissza.

### 10.2. „Gagyí”

Az ismert formális2, „while  $x \leq t \ \&\& \ x \geq t \ \&\& \ t \neq x$ ,” tesztkérdéstípusraadj a szokásosnál (miszerint  $x$ ,  $t$  az egyik esetben az objektum által hordozott érték, a másikban meg az objektum referenciája) „mélyebb” választ, írj Java példaprogramot mely egyszer végtelen ciklus, más  $x$ ,  $t$  értékekkel meg nem! A példát építsd a JDK Integer.java forrására3, hogy a 128-nál inkluzív objektum példányokat poolozza!

Megoldás videó:

Megoldás forrása: <https://github.com/grestemayster/prog2/tree/master/Hell%C3%B3%20Arroway!/-/Gagyí>

### 10.3. Yoda

YodaÍrjunk olyan Java programot, ami java.lang.NullPointerException-el leáll, ha nem követjük a Yoda conditions-t! [https://en.wikipedia.org/wiki/Yoda\\_conditions](https://en.wikipedia.org/wiki/Yoda_conditions)

Megoldás videó:

Megoldás forrása: <https://github.com/grestemayster/prog2/tree/master/Hell%C3%B3%20Arroway!/-Yoda>

### 10.4. Homokózó

Írjuk át az első védési programot (LZW binfa) C++ nyelvről Java nyelvre, ugyanúgy működjön! Mutasunk rá, hogy gyakorlatilag a pointereket és referenciákat kell kiírtani és minden máris működik (erre utal a feladat neve, hogy Java-ban minden referencia, nincs választás, hogy mondjuk egy attribútum pointer, referencia vagy tagként tartalmazott legyen). Miután már áttettük Java nyelvre, tegyük be egy Java Servlet-be és a böngészőből GET-es kéréssel (például a böngésző címsorából) kapja meg azt a mintát, amelynek kiszámolja az LZW binfáját!

Megoldás videó:

Megoldás forrása: <https://github.com/grestemayster/prog2/tree/master/Hell%C3%B3%20Arroway!/-LZWBinFa>

# 11. fejezet

## Helló, Liskov!

### 11.1. Anti OO

A BBP algoritmussal a Pi hexadecimális kifejtésének a 0. pozíciótól számított  $10^6$ ,  $10^7$ ,  $10^8$  darab jegyét határozzuk meg C, C++, Java és C# nyelveken és vessük össze a futási időket! <https://www.tankonyvtar.hu/hu/tartalom/tanitok-javat/apas03.html#id561066>

Megoldás videó:

Megoldás forrása: <https://github.com/grestemayster/prog2/tree/master/Hell%C3%B3%20Liskov!/Anti%20OO>

fordítás/futtatás:

C:

```
g++ pi_bbp_bench.c -o pi_bbp_bench -lm
./pi_bbp_bench
```

C++

```
g++ BBP_test.cpp -o bbpTest
./bbpTest
```

Java

```
javac PiBBPBench.java
java PiBBPBench
```

C#

```
mcs PiBBPBench.cs
mono PiBBPBench.exe
```

C:

Kapott eredmények:

C:

$10^6 = 2,111283$

$10^7 = 24,401839$

$10^8 = 282,93285$

C++:

$10^6 = 2,44803$

$10^7 = 27,7696$

$10^8 = 313,902$

Java:

$10^6 = 1,883$

$10^7 = 22,144$

$10^8 = 248,324$

C#:

$10^6 = 1,922333$

$10^7 = 22,595576$

$10^8 = 256,861381$

## 11.2. Szülő-gyerek

Írjunk Szülő-gyerek Java és C++ osztálydefiníciót, amelyben demonstrálni tudjuk, hogy az ősön keresztül csak az ős üzenetei küldhetők! [https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_1.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf) (98. fólia)

Megoldás videó:

Megoldás forrása: <https://github.com/grestemayster/prog2/tree/master/Hell%C3%B3%20Liskov!/Sz%C3%B3%20gyerek>

## 11.3. Liskov helyettesítés sértése

Írjunk olyan OO, leforduló Java és C++ kódcsipetet, amely megsérti a Liskov elvet! Mutassunk rá a megoldásra: jobb OO tervezés.

Megoldás videó:

Megoldás forrása: <https://github.com/grestemayster/prog2/tree/master/Hell%C3%B3%20Liskov!/Liskov%20helyettesites%20sertese>

A Liskov-féle behelyettesítési elv, rövid nevén LSP a következőt mondja ki:

Ha S osztály a T osztály leszármazottja, akkor S szabadon behelyettesíthető a programunkban minden olyan helyre (őraméter, változó...), ahol eredetileg T szülő típust várnánk

fordítás/futtatás:

C++

g++ Liskovsertes.cpp -o Liskovsertes

./Liskovsertes

Java

javac Liskovsertes.java

java Liskovsertes

Mind a kettő esetben azt kapjuk vissza, hogy az Ebr tud láncot javítani, ami nem lehetséges mert az ebr egy kerekes tank.

## 11.4. EPAM: Interfész evolúció Java ban

Írjunk Szülő-gyerek Java és C++ osztálydefiníciót, amelyben demonstrálni tudjuk, hogy az őson keresztül csak az ős üzenetei küldhetők![https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_1.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf)(98. fólia)

Megoldás videó:

A Java 7 2011-ben jelent meg.

Ez volt az első nagyobb Java-frissítés, miután az Oracle megszerezte a Sun Microsystems-t.

A Java 7 a végső Java-verzió, amely hivatalosan támogatná a Windows XP-t.

A Java 7 tartalmazott néhány olyan főbb funkciót, mint például:

Tömörített 64 bites mutatók.

Frissített class-loader architektúra.

Több kivétel kezelése.

JVM támogatás a nyelvek dinamikus támogatásához.

Frissített 1.1 sor és JDBC 4.1.

Karakterlánc objektum egy kapcsoló utasításban.

Automatikus erőforrás-kezelés try utasításban és még sok más.

Java7 vs. Java8

Lambda kifejezések

A lambda kifejezések csak törzset és paraméterlistát tartalmaznak.

A Lambda Expressions használatának előnyei,

Továbbfejlesztett iteratív szintaxis.

Olvashatóság.

Kód újrafelhasználása.

JAR fájl méret csökkentése.

Egyszerűsített változó hatóköre.

A funkcionális programozás ösztönzése.

Null Referencia Sablon, Nevezhetjük Java Opcionális Osztálynak is.



A Java Optional osztály használatának előnyei:

Nincs semmiféle kivétel a futás közben.

Tiszta API-k fejleszthetők.

Nincs szükség semmilyen ellenőrzésre.

Új dátum és idő API

A `java.util.Date` osztályban az évek 1900-tól kezdődnek, a hónapok pedig 0-tól kezdődnek. Ezért fontos tudni, mivel amikor egy évet és egy hónapot szeretnénk kijelölni, annak a következőnek kell lennie: (ÉÉÉÉ-1900) és (HH) -1). Ezt a Java 8 legyőzte a `java.time` csomagban található osztályok használatával.

Az egész számtól vagy a karaktersorozattól eltérően a Dátum osztály változtatható. Ami azt jelenti, hogy a dátum megváltoztatható a tulajdonos osztály tudta nélkül. A Java 8 rendelkezik erre megoldással. A Java 8-ban bevezetett `LocalDate`, `LocalTime`, `LocalDateTime` stb. Változhatatlanok (az eredeti objektum nem változik. Valahányszor változtatás történik, egy új objektum kerül visszaadásra).

Unsigned Integer Arithmetic

Nem világos, hogy a Java 8 mit próbált elérni az előjel nélküli egész számtani előrehaladással. De általában egy előjel nélküli egész szám károsnak tekinthető a Java-ra. Bonyolultabbá teszi a típusátalakító eszközöket, a típusrendszert és a könyvtár API-kat.

Stream API

A Stream API használatának legfőbb előnye, hogy egyértelműen növeli a program sebességét és hatékonyságát. Javában erőműnek számít.

Párhuzamos rendezés

A párhuzamos rendezés több szálát használ a művelethez. Gyorsan és hatékonyan teszi a programot. Ez nem lesz annyira látható egy kis adathalmaz mellett. De ha nagy adatkészletet használunk, a hatékonyság és a teljesítmény növekedése nagyon észrevehető lesz.

Új JavaScript motor

A Java 8-mal bevezetett JavaScript-motor neve Nashorn volt.

## 12. fejezet

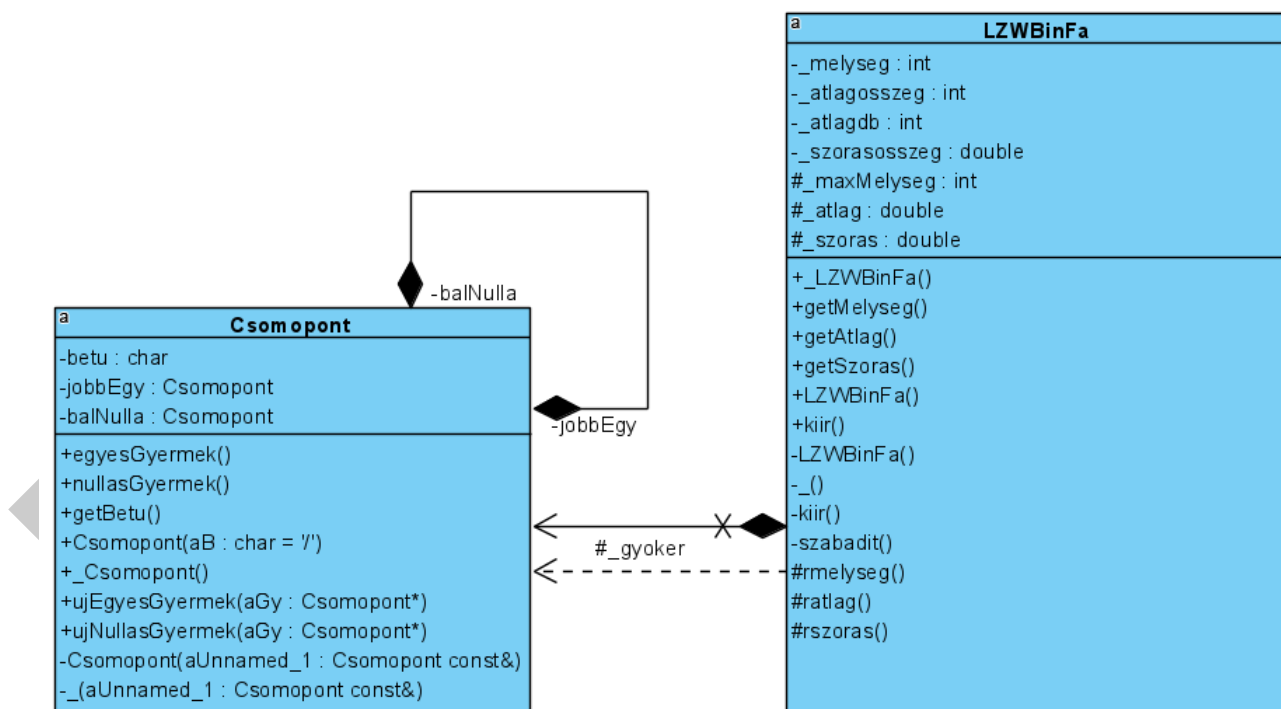
# Helló, Mandelbrot!

### 12.1. a

a

Megoldás videó:

Megoldás forrása:



Az osztálydiagram az osztályok közötti kapcsolatot képviseli a programban. A két osztály két kék színű téglalap. Az téglalap felső része az osztály attribútumait az alsó a metódusait tartalmazza. A láthatóságukat az előtte lévő jelek is megmutatják.

azaz public

azaz private

azaz protected

A feladat külön kéri, hogy térjünk ki a kompozíció és aggregáció kapcsolatára.

Aggregációról akkor beszélhetünk, amikor az egyik objektum részben (vagy akár egészben) is tartalmazza a másikat. Két típusú van:

erős

gyenge

A gyenge aggregációt nevezzük kompozíciónak, azaz a tartamazott objektum illetve a tartalmazó objektum függnék egymástól. Egymás nélkül nem funkcionálnak.

DRAFT

## **IV. rész**

### **Irodalomjegyzék**

DRAFT

## 12.2. Általános

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

## 12.3. C

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

## 12.4. C++

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

## 12.5. 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.