

Univerzális programozás

Így neveld a programozód!

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, Bátfai, Mátyás, Bátfai, Nándor, Bátfai, Margaréta, ÁCs Batizi, Máté	2020. május 21.	

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 https://gitlab.com/nbatfai/bhax repójába.	nbatfai
0.0.4	2019-02-19	A Brun tételes feladat kidolgozása.	nbatfai
0.0.5	2019-02-27	Turing fejezet feltöltése.	Batizi Máté
0.0.6	2019-03-04	Gutenberg fejezet feltöltése.	Batizi Máté
0.0.7	2019-03-11	Chomsky fejezet feltöltése.	Batizi Máté

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
0.0.8	2019-04-01	Caesar fejezet feltöltése.	Batizi Máté
0.0.9	2019-04-10	Mandelbrot fejezet feltöltése.	Batizi Máté
0.0.10	2019-04-15	Welch fejezet feltöltése.	Batizi Máté
0.0.11	2019-04-22	Conway fejezet feltöltése.	Batizi Máté
0.0.12	2019-04-28	Schwarzenegger fejezet feltöltése.	Batizi Máté
0.0.13	2019-05-06	Chaitin fejezet feltöltése.	Batizi Máté

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](#)]

DRAFT

Tartalomjegyzék

I. Bevezetés	1
1. Vízió	2
1.1. Mi a programozás?	2
1.2. Milyen doksikat olvassak el?	2
1.3. Milyen filmeket nézzek meg?	2
II. Tematikus feladatok	4
2. Helló, Turing!	6
2.1. Végtelen ciklus	6
2.2. Lefagyott, nem fagyott, akkor most mi van?	9
2.3. Változók értékének felcserélése	11
2.4. Labdapattogás	12
2.5. Szóhossz és a Linus Torvalds féle BogoMIPS	12
2.6. Helló, Google!	14
2.7. A Monty Hall probléma	16
2.8. 100 éves a Brun téTEL	18
2.9. Vörös Pipacs Pokol/csiga folytonos mozgási parancsokkal	21
3. Helló, Chomsky!	22
3.1. Decimálisból unárisba átváltó Turing gép	22
3.2. Az $a^n b^n c^n$ nyelv nem környezetfüggetlen	22
3.3. Hivatalos nyelv	23
3.4. Saját lexikális elemző	24
3.5. Leetspeak	25

3.6. A források olvasása	27
3.7. Logikus	29
3.8. Deklaráció	29
3.9. Vörös Pipacs Pokol/csiga diszkrét mozgási parancsokkal	33
4. Helló, Caesar!	34
4.1. double ** háromszögmátrix	34
4.2. C EXOR titkosító	36
4.3. Java EXOR titkosító	37
4.4. C EXOR törő	39
4.5. Neurális OR, AND és EXOR kapu	39
4.6. Hiba-visszaterjesztéses perceptron	44
4.7. Vörös Pipacs Pokol/írd ki, mit lát Steve	45
5. Helló, Mandelbrot!	46
5.1. A Mandelbrot halmaz	46
5.2. A Mandelbrot halmaz a std::complex osztályjal	50
5.3. Biomorfok	53
5.4. A Mandelbrot halmaz CUDA megvalósítása	58
5.5. Mandelbrot nagyító és utazó C++ nyelven	62
5.6. Mandelbrot nagyító és utazó Java nyelven	63
5.7. Vörös Pipacs Pokol/csiga diszkrét mozgási parancsokkal	64
6. Helló, Welch!	65
6.1. Első osztályom	65
6.2. LZW	65
6.3. Fabejárás	66
6.4. Tag a gyökér	66
6.5. Mutató a gyökér	66
6.6. Mozgató szemantika	66
7. Helló, Conway!	67
7.1. Hangyasimulációk	67
7.2. Java életjáték	68
7.3. Java életjáték	68
7.4. Qt C++ életjáték	75
7.5. BrainB Benchmark	75
7.6. Vörös Pipacs Pokol/19 RF	97

8. Helló, Schwarzenegger!	98
8.1. Szoftmax Py MNIST	98
8.2. Mély MNIST	99
9. Helló, Chaitin!	102
9.1. Iteratív és rekurzív faktoriális Lisp-ben	102
9.2. Gimp Scheme Script-fu: króm effekt	102
9.3. Gimp Scheme Script-fu: név mandala	102
10. Helló, Gutenberg!	103
10.1. Programozási alapfogalmak	103
10.2. Programozás bevezetés	104
10.3. Programozás	105
III. Második felvonás	106
11. Helló, Arroway!	108
11.1. A BPP algoritmus Java megvalósítása	108
11.2. Java osztályok a Pi-ben	108
IV. Irodalomjegyzék	109
11.3. Általános	110
11.4. C	110
11.5. C++	110
11.6. Lisp	110

Ábrák jegyzéke

2.1. A B_2 konstans közelítése	21
4.1. A double ** háromszögmátrix a memóriában	36
5.1. A Mandelbrot halmaz a komplex síkon	46
5.2.	50

DRAFT

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 alkalmi igény szerint pár robot programozót a felhőből? A programozók dolgozók lesznek vagy papok? Ki tudhatná ma.

Minden esetre 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 gyereknek, 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 gyereknek, 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 mászt 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.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 találod az informatikai szövegek jelölésére használható gazdag „API” elemenkénti bemutatását.

I. rész

Bevezetés

DRAFT

1. fejezet

Vízió

1.1. Mi a programozás?

Ne cifrázzuk: programok írása. Mik akkor a programok? Mit jelent az írásuk?

1.2. Milyen doksikat olvassak el?

- Kezd ezzel: <http://esr.fsf.hu/hacker-howto.html>!
- Olvasgasd aztán 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
- C kapcsán a [**KERNIGHANRITCHIE**] könyv adott részei.
- C++ kapcsán a [**BMECPP**] könyv adott részei.
- Az igazi kockák persze csemegéznek a C nyelvi szabvány [ISO/IEC 9899:2017](#) kódcsipeteiből is.
- Amiből viszont a legeslegjobban lehet tanulni, az a [The GNU C Reference Manual](#), mert gcc specifikus és programozókra van hangolva: szinte csak 1-2 lényegi mondat és apró, lényegi kódcsipete! Aki pdf-ben jobban szereti olvasni: <https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf>
- Az R kódok olvasása kis általános tapasztalat után automatikusan, erőfeszítés nélkül menni fog. A Python nincs ennyire a spektrum magától értetődő végén, ezért ahhoz olvasd el a [**BMECPP**] könyv - 20 oldalas gyorstalpaló részét.

1.3. Milyen filmeket nézzek meg?

- 21 - Las Vegas ostroma, <https://www.imdb.com/title/tt0478087/>, benne a **Monty Hall probléma** bemutatása.
- Kódjátszma, <https://www.imdb.com/title/tt2084970/>, benne a **kódtörő feladat** élménye.

- , , benne a bemutatása.

DRAFT

II. rész

Tematikus feladatok

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

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: <https://youtu.be/m1v0AEZvwQM>

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/infty-f.c, bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/infty-w.c.

Végtelen ciklust általában azért hozunk létre, hogy olyan folyamatot futassunk, aminek nem akarjuk, hogy valamikor is vége legyen, szóval nem kötjük a futását feltételhez. Ezt a megoldást sok helyen tapasztalhatjuk a való életben is, ilyek például az órák, jelzőlámpák, esetleges szerverfolyamatok. Viszont végtelen ciklus bug miatt is létrejöhet és ez nagyban megnehezítheti a dolgunkat, ezért óvatosnak kell lennünk, hogy mikor használunk ilyet.

Egy mag 100 százalékban:

```
int
main ()
{
    for (;;);

    return 0;
}
```

vagy az olvashatóbb, de a programozók és fordítók (szabványok) között kevésbé hordozható

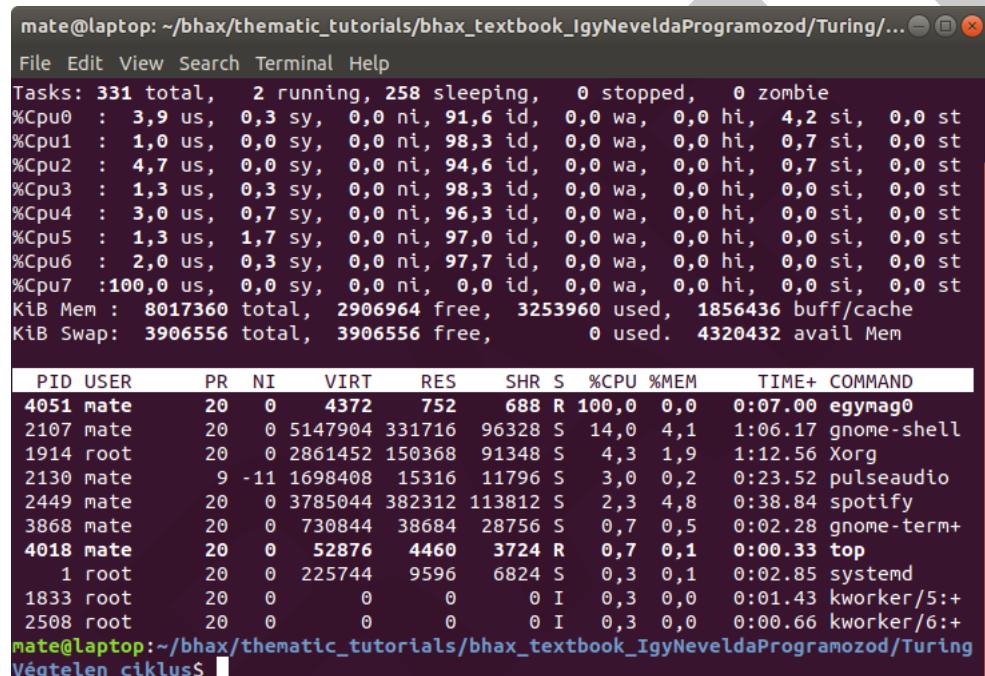
```
int
#include <stdbool.h>
main ()
{
    while(true);

    return 0;
}
```

{}

Egyébként a fordító a `for`-os és `while`-os ciklusból ugyanazt az assembly kódot fordítja:

```
$ gcc -S -o infny-f.S infny-f.c
$ gcc -S -o infny-w.S infny-w.c
$ diff infny-w.S infny-f.S
1c1
< .file "infny-w.c"
---
> .file "infny-f.c"
```



The terminal window displays system statistics and a top command output. The top command shows various processes running on the system, including mate, gnome-shell, Xorg, pulseaudio, spotify, and several kworker processes. The system statistics at the top indicate 331 total tasks, with 2 running, 258 sleeping, 0 stopped, and 0 zombie processes. CPU usage is shown across 8 cores (%Cpu0 to %Cpu7), and memory usage shows 8017360 KiB total memory, 2906964 KiB free, and 3253960 KiB used.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4051	mate	20	0	4372	752	688	R	100,0	0,0	0:07.00	egymag0
2107	mate	20	0	5147904	331716	96328	S	14,0	4,1	1:06.17	gnome-shell
1914	root	20	0	2861452	150368	91348	S	4,3	1,9	1:12.56	Xorg
2130	mate	9	-11	1698408	15316	11796	S	3,0	0,2	0:23.52	pulseaudio
2449	mate	20	0	3785044	382312	113812	S	2,3	4,8	0:38.84	spotify
3868	mate	20	0	730844	38684	28756	S	0,7	0,5	0:02.28	gnome-term+
4018	mate	20	0	52876	4460	3724	R	0,7	0,1	0:00.33	top
1	root	20	0	225744	9596	6824	S	0,3	0,1	0:02.85	systemd
1833	root	20	0	0	0	0	I	0,3	0,0	0:01.43	kworker/5:+
2508	root	20	0	0	0	0	I	0,3	0,0	0:00.66	kworker/6:+

Egy mag 0 százalékban:

```
#include <unistd.h>
int
main ()
{
    for (;;)
        sleep(1);

    return 0;
}
```

```
mate@laptop: ~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/...
File Edit View Search Terminal Help
top - 23:02:25 up 1:34, 1 user, load average: 0,51, 0,64, 0,84
Tasks: 341 total, 2 running, 264 sleeping, 0 stopped, 0 zombie
%Cpu0 : 0,7 us, 0,7 sy, 0,0 ni, 98,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu1 : 0,3 us, 0,3 sy, 0,0 ni, 99,3 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu2 : 1,3 us, 0,7 sy, 0,0 ni, 98,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu3 : 0,0 us, 0,0 sy, 0,0 ni, 100,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu4 : 0,7 us, 0,3 sy, 0,0 ni, 99,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu5 : 1,3 us, 1,0 sy, 0,0 ni, 97,7 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu6 : 0,3 us, 0,7 sy, 0,0 ni, 99,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu7 : 0,0 us, 0,0 sy, 0,0 ni, 100,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem : 8017360 total, 2592252 free, 3296456 used, 2128652 buff/cache
KiB Swap: 3906556 total, 3906556 free, 0 used. 4270044 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2130 mate 9 -11 1698736 15856 12008 S 3,3 0,2 0:38.13 pulseaudio
2449 mate 20 0 3785044 382312 113812 S 3,0 4,8 0:50.69 spotify
4287 mate 20 0 52984 4616 3748 R 0,7 0,1 0:00.09 top
3054 mate 20 0 3168404 407412 163196 S 0,3 5,1 0:28.63 Web Content
3339 mate 20 0 3213080 365500 166884 S 0,3 4,6 1:18.13 Web Content
1 root 20 0 225744 9596 6824 S 0,0 0,1 0:03.32 systemd
2 root 20 0 0 0 0 S 0,0 0,0 0:00.00 kthreadd
3 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 rcu_gp
4 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 rcu_par_gp
6 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 kworker/0:+
9 root 0 -20 0 0 0 I 0,0 0,0 0:00.00 mm_percpu_+
```

Minden mag 100 százalékban:

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

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4509	mate	20	0	4372	856	792	R	100,0	0,0	0:18.10	a.out
2130	mate	9	-11	1698736	16160	12312	R	3,6	0,2	0:43.56	pulseaudio
2449	mate	20	0	3785044	382452	113944	S	3,0	4,8	0:56.54	spotify
1914	root	20	0	2863104	152468	92376	S	0,7	1,9	1:36.85	Xorg
4287	mate	20	0	52984	4616	3748	R	0,7	0,1	0:01.09	top
2107	mate	20	0	5152008	353256	96340	S	0,3	4,4	1:49.11	gnome-shell
1	root	20	0	225744	9596	6824	S	0,0	0,1	0:03.81	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:+

Tanulságok, tapasztalatok, magyarázat... Ha egy magon 0%-os processzorhasználathoz az volt az elgondolás, hogy "elaltatjuk" a programot, akkor nem lesz kimutatható a processzorhasználat. Az egy magnak pedig a 100%-os terhelése egyszerűen úgy működik, hogy a programot egy teljesen "üresen" járatjuk egy végtelen ciklussal.



Werkfilm

- <https://youtu.be/lvmi6tyz-nl>

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 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 épőlő Lefagy2 már nem tartalmaz feltételezett, csak 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.

Ez a feladat egy igen ismert problémát mutat be. A probléma abból áll, hogy el lehet-e dönteni egy programról adott bemenet esetén, hogy végtelen ciklusba kerül-e. Alan Turing bizonyította be, hogy nem lehetséges olyan általános algoritmust írni, amely minden program-bemenet párról megmondja, hogy végtelen ciklusba kerül-e.

2.3. Változók értékének felcserélése

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

Megoldás videó: https://bhaxor.blog.hu/2018/08/28/10_begin_goto_20_avagy_elindulunk

Megoldás forrása:

Viszont ha segédváltozók nélkül szeretnénk megváltoztatni egymás értékeire (ha csak két számunk van) akkor erre itt van egy példa:

Segédváltozó nélkül:

```
#include<stdio.h>

int main(){
    int x=18;
    int y=22;
    printf("a=%d, b=%d\n", a, b);
    y=y-x
    x=x+y
    y=x-y
    printf("a=%d, b=%d\n", a, b);
}
```

Segédváltozó nélkül ezen a példán úgy lehet bemutatni, hogy aaz elején megadott értékek vagyis az $x=18$ és az $y=22$, az első helyen az y egyenlővé tettük " $y=x$ "-el ($22-18=4$), ezután az x -et egyenlővé tettük az " $x+y$ "-al ($18+4=22$) így kaptuk meg az y eredeti értékét és a végén még az y -ból kivontuk a különbséget ($22-4=18$) ez lett az x eredi értéke.

Segédváltozóval:

```
#include<stdio.h>

int main()
{
    int a=5;
    int b=4;
    printf("a=%d, b=%d\n", a, b);
    int tmp = a;
```

```
a=b;  
b=tmp;  
printf("a=%d, b=%d\n", a, b);  
return 0;  
}
```

A változófelcserélés egy igen egyszerű feladat ezt megoldhatjuk segédváltozóval vagy nélküle. Ha segédváltozóval csináljuk akkor be kell vezetni egy új változót, amiben az egyik értékét eltároljuk ennek az eredeti változóját megváltoztatni a másikra, és utána az első változónak a segédváltozóban elmentett értékére megváltoztatni a másikat is.

2.4. 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://youtu.be/VutomH0a8js>

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/Labdapattogtatás/labdapattogtatasifel.c, bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing//Labdapattogtatás/labdapattogtac_c, bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing//Labda_labdaiffel.py.

Ez a feladat jól bemutatja a végtelen ciklus alkalmazhatóságát egy feladatban, mivel a labda addig nem fog leállni amíg nem kap egy külső behatást. Ezt a programot először C nyelven írtam meg utána viszont átírtam Python-ra is, ami először nehéznek bizonyult azonban találtam egy pythonba importálható modult aminek a neve "turtle". Ezzel már jóval egyszerűbb volt egyszerűen megnéztem miként kell hozzáadni egy 2 dimenziós labdát, (csak ezt kell megadni, hogy milyen alakja legyen a "turtle"-nek) ezután lehet állítani háttérszínt hozzá, a turtle színét be lehet állítani, gravitációt hozzáadni.

A programok fordítását a gcc labdapatt_if.c -o labdapatt_if -lncurses , illetve a gcc labdapatt.c -o labdapatt -lncurses paranccsal végezzük.

2.5. Szóhossz és a Linus Torvalds féle BogoMIPS

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

Megoldás videó: https://youtu.be/9KnMqrkj_kU.

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/bogo/szohossz.c

Ami fontos nekünk, hogy bitshift eltolással fogunk tologani addig egy 1-est amíg a memóriacím végre nem ér, amikor ez megtörtnik megvizsgálja, hogy képesek vagyunk mégegyszer eltoni, és amikor kijön, hogy nem addig a számláló ugye számolta ez az eltolás hányszor lehetséges és kilép a program. A végén pedig kiírja mennyi eltolásra volt képes a program.

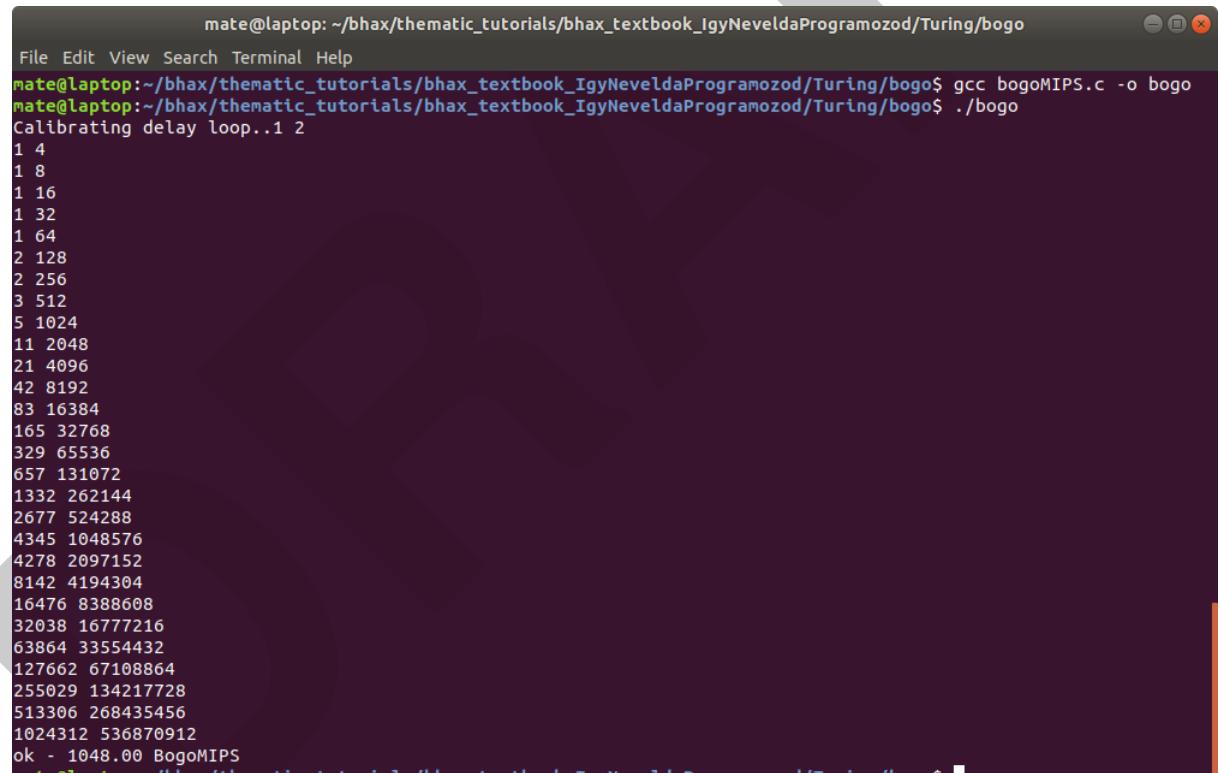
```
#include <iostream>

using namespace std;

int main()
{
    int a = 1;
    int osszeg = 0;

    while (a <= 1)
        osszeg++;
    cout << osszeg+1 << endl;
}
```

A BogoMIPS lényegében ugyanazt a bitshift-es eltolást alkalmazza ebben az esetben egy egyet fog tolni és az hatványozóni fog és ezekkel a számokkal fog műveleteket csinálni, ha ez a lefutási idő túllép ez bizonyos legfutási időt akkor leállítja a programot. Ez valamelyen szinten a gép teljesítményét fogja mérni a program.



The screenshot shows a terminal window titled "mate@laptop: ~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/bogo". The terminal displays the command "gcc bogoMIPS.c -o bogo" followed by the output of the program. The program prints a sequence of numbers starting from 1, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216, 33554432, 67108864, 134217728, 268435456, 536870912, and finally "ok - 1048.00 BogoMIPS".

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

void
delay (unsigned long long loops)
{
    for (unsigned long long i = 0; i < loops; i++);
}
```

```
int
main (void)
{
    unsigned long long loops_per_sec = 1;
    unsigned long long ticks;

    printf ("Calibrating delay loop..");
    fflush (stdout);

    while ((loops_per_sec <= 1))
    {
        printf(loops_per_sec);
        ticks = clock ();
        delay (loops_per_sec);
        ticks = clock () - ticks;

        if (ticks >= CLOCKS_PER_SEC)
        {
            loops_per_sec = (loops_per_sec / ticks) * CLOCKS_PER_SEC;

            printf ("ok - %llu.%02llu BogoMIPS\n",
                   loops_per_sec / 500000,
                   (loops_per_sec / 5000) % 100);

            return 0;
        }
    }

    printf ("failed\n");
    return -1;
}
```

2.6. Hello, Google!

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

Megoldás forrása: [bhaxi-thematic-tutorials/bhaxi-textbook_IgyNeveldaProgramozod/Turing/PageRank/pagerank.c](https://bhaxi-thematic-tutorials.github.io/bhaxi-textbook/IgyNeveldaProgramozod/Turing/PageRank/pagerank.c)

A PageRank egy olyan algoritmus, amivel weboldalak relatív fontosságát lehet megállapítani. Alapja, hogy egy oldalon minden hivatkozás egy-egy "szavazat" a hivatkozott oldalra. Az alapján meg lehet állapítani egy oldal relatív fontosságát, hogy hány oldalra mutató hivatkozás van az oldalon és, hogy hány oldal hivatkozik erre az oldalra. Az algoritmusban egy jobb minőségű oldal "szavazata" erősebbnek számít, mint egy kis rekatív fontosságúé.

```
#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)
{
    double osszeg = 0.0;
    int i;

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

    return sqrt(osszeg);
}

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 PR[4] = { 0.0, 0.0, 0.0, 0.0 };
    double PRv[4] = { 1.0 / 4.0, 1.0 / 4.0, 1.0 / 4.0, 1.0 / 4.0 };

    int i, j;

    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];
```

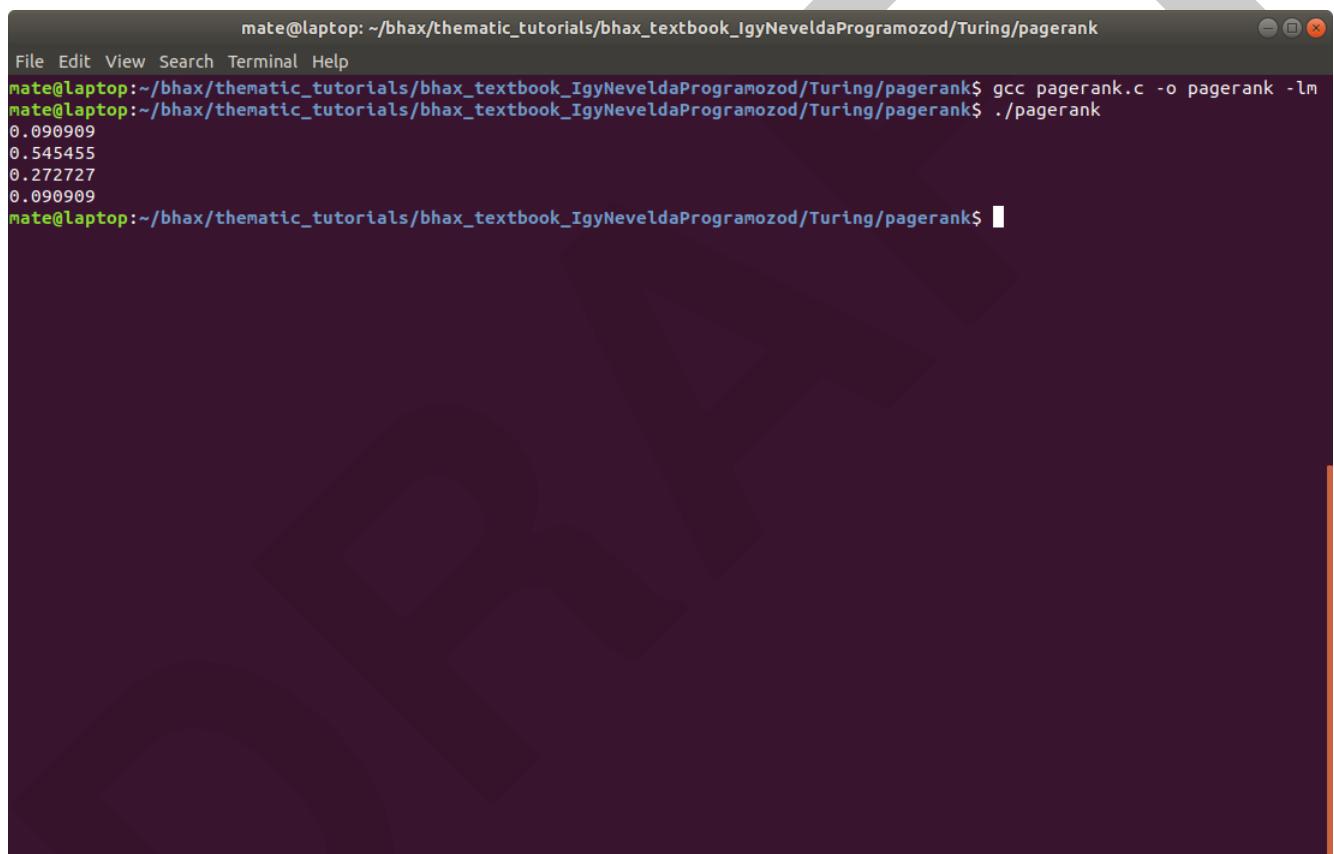
```
    }

    kiir (PR, 4);

    return 0;
}
```

A kódban meg vannak előre adva a pagerank értékek és ez alapján fog dolgozni a program.

A forrásban láthatunk egy végtelen for ciklust, ami folyamatosan fogja iterálni. A program nem fog pontos értéket kiírni viszont közelíteni fog hozzá. Számolás után megnézi a program, hogy az előző értékhez képest makkora az eltérés és ha nagyobb, mint 0.00000001 akkor újrakezdi a számolását, azonban, ha kisebb akkor megáll és kiírja az adott pagerank értékét az oldalnak.



```
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/pagerank
File Edit View Search Terminal Help
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/pagerank$ gcc pagerank.c -o pagerank -lm
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/pagerank$ ./pagerank
0.090909
0.545455
0.272727
0.090909
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/pagerank$ █
```

2.7. A Monty Hall probléma

Írj R szimulációt a Monty Hall problémára!

Megoldás forrása: https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/MontyHall_R

A Monty Hall paradoxon vagy probléma egy régi TV-s műsorban jelent meg először, ahol a nyereményét a játékosnak bezárták egy random kiválasztott ajtó mögé 3-ból, ezután választania kellett egyet. Miután választott egyet a műsorvezető kinyitott egy olyan ajtót, ami mögött nem volt nyeremény és felajánlotta, hogy választhat a két maradt ajtó közül. A lényege az, hogy érdemes-e váltani a jelenlegi ajtóról a másikra

vagy sem. A megoldás az, hogy igen mivel annak az ejtőnök dupla annyi esélye van a nyerésre, mint az elsőre kiválasztottnak.

De hogy ez miért van, azt jól be lehet mutatni egy ábrával:

Monty Hall paradoxon:

A <-- B C (\$)

Tegyük fel, hogy a C ajtó mögött van a nyeremény.

Először a játékos az A ajtót választja. Ekkor a játékvezető csak a B ajtót nyithatja ki és megmutatja, hogy amögött nincsen nyeremény. Most vegyük két részre azt, hogy nyerni fog-e a játékos vagy sem. Ha játékos nem változtatja meg a döntését akkor veszít (ahogyan a táblázatban is látszik), ha megváltoztatja a döntését akkor viszont nyer.

A B <-- C (\$)

Ebben a körben a játékos a B ajtót választja, ekkor a játékvezető kinyitja az A ajtót, ami mögött nyilván nincs semmi. Ekkor a játékos, ha nem változtatja meg a döntését, akkor veszít, ha megváltoztatja nyer pont mint az előző körben.

A B C (\$) <--

Az utolsó körben a játékos a C ajtót választja ami mögött a nyeremény is van. Ha nem változtatja meg a döntését akkor nyer, ha megváltoztatja, akkor veszít.

1.körben

2.körben

3.körben

Megváltoztatja:

Nyer

Nyer

Veszít

Nem változtatja meg:

Veszít

Veszít

Nyer

A táblázatban jól látszik, hogy mikor a játékos megváltoztatta a döntését 2-szer nyert és 1-szer veszített, viszont ha nem akkor 2-szer veszített és csak 1-szer nyert.

2.8. 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

A természetes számok építőelemei a prímszámok. Abban az értelemben, hogy minden természetes szám előállítható prímszámok szorzataként. Például $12=2\cdot2\cdot3$, vagy például $33=3\cdot11$.

Prímszám az a természetes szám, amely csak önmagával és eggyel osztható. Eukleidész görög matematikus már Krisztus előtt tudta, hogy végtelen sok prímszám van, de ma sem tudja senki, hogy végtelen sok ikerprím van-e. Két prím ikerprím, ha különbségük 2.

Két egymást követő páratlan prím között a legkisebb távolság a 2, a legnagyobb távolság viszont bármilyen nagy lehet! Ez utóbbit könnyű bebizonyítani. Legyen n egy tetszőlegesen nagy szám. Akkor szorozzuk össze $n+1$ -ig a számokat, azaz számoljuk ki az $1\cdot2\cdot3\cdots\cdot(n-1)\cdot n\cdot(n+1)$ szorzatot, aminek a neve $(n+1)$ faktoriális, jele $(n+1)!$.

Majd vizsgáljuk meg az a sorozatot:

$(n+1)!+2, (n+1)!+3, \dots, (n+1)!+n, (n+1)!+(n+1)$ ez n db egymást követő azám, ezekre (a jól ismert bizonyítás szerint) rendre igaz, hogy

- $(n+1)!+2=1\cdot2\cdot3\cdots\cdot(n-1)\cdot n\cdot(n+1)+2$, azaz $2\cdot$ valamennyi $+2$, 2 többszöröse, így ami osztható kettővel
- $(n+1)!+3=1\cdot2\cdot3\cdots\cdot(n-1)\cdot n\cdot(n+1)+3$, azaz $3\cdot$ valamennyi $+3$, ami osztható hárommal
- ...
- $(n+1)!+(n-1)=1\cdot2\cdot3\cdots\cdot(n-1)\cdot n\cdot(n+1)+(n-1)$, azaz $(n-1)\cdot$ valamennyi $+(n-1)$, ami osztható $(n-1)$ -el
- $(n+1)!+n=1\cdot2\cdot3\cdots\cdot(n-1)\cdot n\cdot(n+1)+n$, azaz $n\cdot$ valamennyi $+n$, ami osztható n-el
- $(n+1)!+(n+1)=1\cdot2\cdot3\cdots\cdot(n-1)\cdot n\cdot(n+1)+(n-1)$, azaz $(n+1)\cdot$ valamennyi $+(n+1)$, ami osztható $(n+1)$ -el

tehát ebben a sorozatban egy prim nincs, akkor a $(n+1)!+2$ -nél kisebb első prim és a $(n+1)!+(n+1)$ -nél nagyobb első prim között a távolság legalább n.

Az ikerprímszám sejtés azzal foglalkozik, amikor a prímek közötti távolság 2. Azt mondja, hogy az egymástól 2 távolságra lévő prímek végtelen sokan vannak.

A Brun téTEL azt mondja, hogy az ikerprímszámok reciprokaiból képzett sor összege, azaz a $(1/3+1/5)+(1/5+1/7)+(1/11+1/13)+\dots$ véges vagy végtelen sor konvergens, ami azt jelenti, hogy ezek a törtek összeadva egy határt adnak ki pontosan vagy azt át nem lépve növekednek, ami határ számot B_2 Brun konstansnak neveznek. Tehát ez nem dönti el a több ezer éve nyitott kérdést, hogy az ikerprímszámok halmaza végtelen-e? Hiszen ha véges sok van és ezek reciprokait összeadjuk, akkor ugyanúgy nem lépjük át a B_2 Brun konstans értékét, mintha végtelen sok lenne, de ezek már csak olyan csökkenő mértékben járulnának hozzá a végtelen sor összegéhez, hogy így sem lépnék át a Brun konstans értékét.

Ebben a példában egy olyan programot készítettünk, amely közelíteni próbálja a Brun konstans értékét. A repó [bhax/attention_raising/Primek_R/stp.r](https://gitlab.com/nbatfai/bhax/blob/master/attention_raising/Primek_R/stp.r) mevű állománya kiszámolja az ikerprímeket, összegzi a reciprokaikat és vizualizálja a kapott részeredményt.

```
# Copyright (C) 2019 Dr. Norbert Bátfai, nbatfai@gmail.com
#
# 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 <http://www.gnu.org/licenses/>
library(matlab)

stp <- 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=sapply(x, FUN = stp)
plot(x,y,type="b")
```

Soronként értelemezük ezt a programot:

```
primes = primes(13)
```

Kiszámolja a megadott száigmig a prímeket.

```
> primes=primes(13)
> primes
[1]  2  3  5  7 11 13
```

```
diff = primes[2:length(primes)]-primes[1:length(primes)-1]
```

```
> diff = primes[2:length(primes)]-primes[1:length(primes)-1]
> diff
[1] 1 2 2 4 2
```

Az egymást követő prímek különbségét képzi, tehát 3-2, 5-3, 7-5, 11-7, 13-11.

```
idx = which(diff==2)
```

```
> idx = which(diff==2)
> idx
[1] 2 3 5
```

Megnézi a diff-ben, hogy melyiknél lett kettő az eredmény, mert azok az ikerprím párok, ahol ez igaz. Ez a diff-ben lévő 3-2, 5-3, 7-5, 11-7, 13-11 különbségek közül ez a 2., 3. és 5. indexűre teljesül.

```
t1primes = primes[idx]
```

Kivette a primes-ból a párok első tagját.

```
t2primes = primes[idx]+2
```

A párok második tagját az első tagok kettő hozzáadásával képezzük.

```
rt1plust2 = 1/t1primes+1/t2primes
```

Az 1/t1primes a t1primes 3,5,11 értékéből az alábbi reciprokokat képzi:

```
> 1/t1primes
[1] 0.33333333 0.20000000 0.09090909
```

Az 1/t2primes a t2primes 5,7,13 értékéből az alábbi reciprokokat képzi:

```
> 1/t2primes
[1] 0.20000000 0.14285714 0.07692308
```

Az 1/t1primes + 1/t2primes pedig ezeket a törteket rendre összeadja.

```
> 1/t1primes+1/t2primes
[1] 0.53333333 0.34285711 0.1678322
```

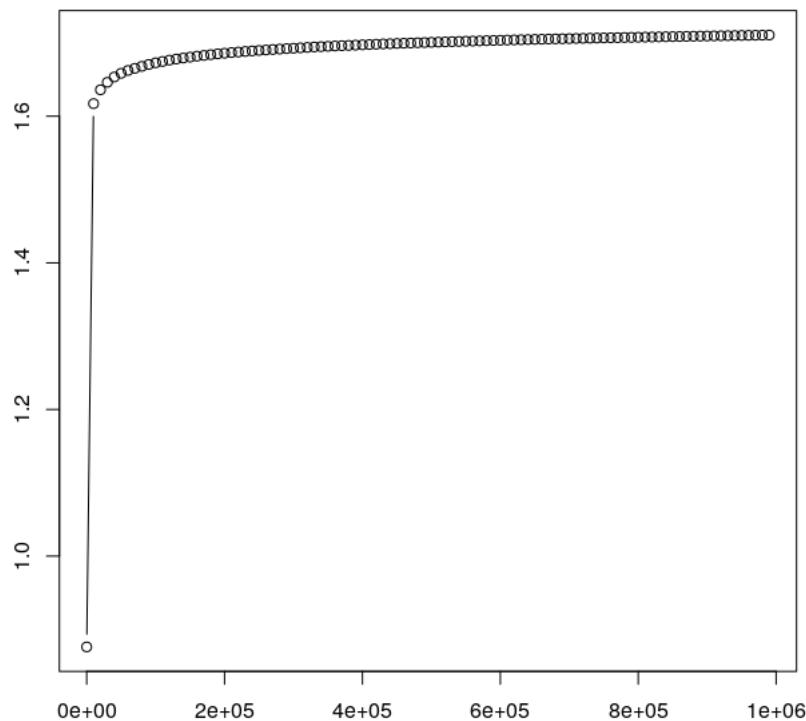
Nincs más dolgunk, mint ezeket a törteket összeadni a sum függvényel.

```
sum(rt1plust2)
```

```
> sum(rt1plust2)
[1] 1.044023
```

A következő ábra azt mutatja, hogy a szumma értéke, hogyan nő, egy határértékhez tart, a B₂ Brun konstanshoz. Ezt ezzel a csipettel rajzoltuk ki, ahol először a fenti számítást 13-ig végezzük, majd 10013, majd 20013-ig, egészen 990013-ig, azaz közel 1 millióig. Vegyük észre, hogy az ábra első köre, a 13 értékhez tartozó 1.044023.

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



2.1. ábra. A B_2 konstans közelítése

A Brun-tétel szerint az ikerprímek reciprokösszege egy meghatározható szám felé tart, amelyet a B_2 konstansal jelölnek, amelyértéke $B_2=1,902160583104$. Az ikerprím olyan két egymást követő prímszámot jelent, amelyek különbsége 2.

Werkfilm

- <https://youtu.be/VkMFrgBhN1g>
- <https://youtu.be/aF4YK6mBwf4>

2.9. Vörös Pipacs Pokol/csiga folytonos mozgási parancsokkal

Megoldás videó: <https://youtu.be/VXtpJFjIh1o>

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfjával megadva írd meg ezt a gépet!

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

A fóliabeli Turing gép működési elve a köveztekzőképpen lehet felfogni, a gép egy bizonyos "sávban" tud mozogni jobbra vagy balra és az "alatta" lévő számokat tudja megváltoztatni. Ez úgy működik, hogy az elején a gép az egyenlőségjelen áll és onnan lép majd balra, mindaddig amíg nem talál egy olyan számot ami nem egyenlő 0-val. Amit megtalálja ezt a számot megáll és azt csökkenti egyel(szóval 1-ből 0 lesz, 2-ből 1 lesz, 3-ból 2 lesz...9-ből 8 lesz viszont 0-ból 9 lesz). Miközben a "sávnak" a bal oldalát minden egyel csökkentjük úgy építjük vele együtt a jobb oldali számsort unárisban, miközben a bal oldalt minden egyel csökkentjük egyel úgy írunk egyel több 1-est a jobb oldalhoz.

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:

Szabályok:

$$S \longleftrightarrow aBSc$$

S \longleftrightarrow abc

$$\text{Ba} \longleftrightarrow \text{aB}$$

S --> aBSc --> aBaBScc --> aBaBabccc --> aaBBabccc --> aaBaBbcc --> ←

Section 1

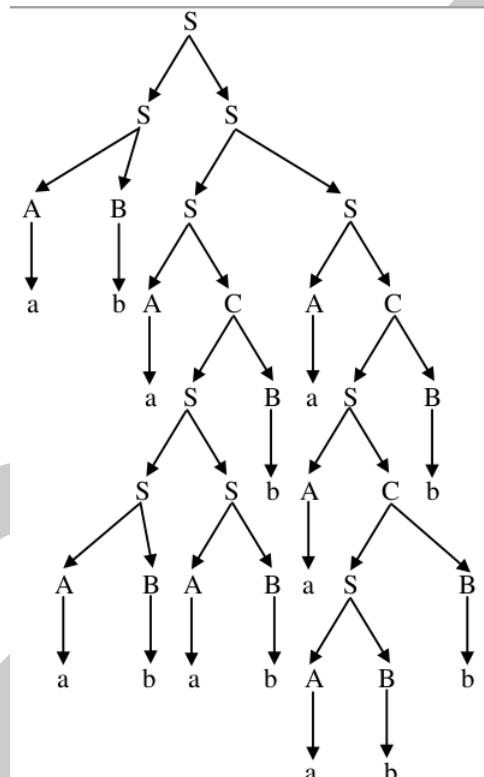
Szabályok

S --> abc

```
Xb --> bX
Xc --> Ybcc
bY --> Yb
aY --> aaX
aY --> aa
```

```
S --> aXbc --> abXc --> abYbcc --> aaXbbcc --> aabXbcc --> aabbXcc --> aabbXcc --> aabbYbcc --> aabYbbccc --> aaYbbbccc --> aaabbccc
```

A generatív nyelvtan elméletét Noam Chomsky alkotta meg, és ő dolgozta ki a Chomsky-hierarchiát. A formális grammaknak 3 típusa van, a környezetfüggetlen nyelvtan, a szabályos nyelvtan és a generatív nyelvtan.



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:

```
#include <stdio.h>
int main()
{
    for (int i = 0; i < 10; i++)
    {
```

```
    printf("%d\n", i);
}
return 0;
}
```

Ha c89-es verzióval fordítjuk akkor egy hibát kapunk:

```
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/ ←
Turing/pagerank$ gcc -std=c89 33.c
33.c: In function 'main':
33.c:4:5: error: 'for' loop initial declarations are only allowed in C99 or ←
C11 mode
    for (int i = 0; i < 10; i++)
      ^~~
33.c:4:5: note: use option -std=c99, -std=gnu99, -std=c11 or -std=gnu11 to ←
compile your code
```

Azonban ha c99-es verzióval fordítjuk akkor lefordul a program:

```
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/ ←
Turing/pagerank$ gcc -std=c99 33.c
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/ ←
Turing/pagerank$ ./a.out
0
1
2
3
4
5
6
7
8
9
```

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 használunk, azaz óriások vállán állunk és ne kispályázzunk!

Megoldás videó: https://youtu.be/9KnMqrkj_kU (15:01-től).

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/realnumber.l

```
% {
#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;
}
```

Ez a lexer egy olyan program ami c programokat ír. A digit-ben definiáljuk, hogy a számok 1-9-ig, a * azt jelenti, hogy akármennyi lehet belőle, az is lehetséges, hogy 0 legyen bármiből. Ezzel ellentétben a végén a + azt jelenti, hogy elgalább 1 számjegynek lennie kell. A pont le van védve a tizedes jegyeknek, ha viszont felismeri a pontot, hogy jelen van akkor utána is kell, hogy számok álljanak. Ha ez megvan akkor növlejük a realnumbers változót. Ezután simán printf függvényel kiiratjuk string-ként az éppen felismert szintaktikai elemet és atof (az atof konverálja a sting-et double-re) függvényel pedig ugyanezt double-ban, szóval két számot fogunk látni.

A main részben először elindítom a lexikális elemzést és ha ennek vége akkor kiiratom az addigra már meghatározott realnumber számot.

3.5. Leetspeak

Lexelj össze egy l33t cipher!

Megoldás videó: https://youtu.be/06C_PqDpD_k

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/l337d1c7.1

```
% {
#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', {"1", "1", "|", "!"}},  
{'j', {"j", "7", "_|", "/"}},  
{'k', {"k", "|<", "1<", "|{"}},  
{'l', {"l", "1", "|", "|_"}},  
{'m', {"m", "44", "(V)", "\\\\"}},  
{'n', {"n", "|\\|", "/\\/", "/V"}},  
{'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 < 100 && r > 0) l337d1c7[i].c = leet[map[l337d1c7[i].c]];  
        }  
    }  
}
```

```
    if(r<91)
        printf("%s", 1337d1c7[i].leet[0]);
    else if(r<95)
        printf("%s", 1337d1c7[i].leet[1]);
    else if(r<98)
        printf("%s", 1337d1c7[i].leet[2]);
    else
        printf("%s", 1337d1c7[i].leet[3]);

    found = 1;
    break;
}

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

}
%%

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

A Leetspeak a szavak keveréke egyfajta számítógépes szabályrendszer, szándékosan helytelenül írt általában a betűk és a számok hasonlóságán alapuló "titkosítás". Ezenkívül a leetspeak használ betűket számokat és ASCII szimbólumokat is. (Például: Bátfai Haxor Stream = B4tf41 H4x0r Str34m)

3.6. 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 megy ránézésre, elkapja valamelyiket esetleg a splint vagy a frama?

i.

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

ii.

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

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

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

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

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

iii.

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

iv.

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

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

Ha a signal nem volt ignorálva, akkor kezelje a jelkezelő! Ha ignorálva volt, akkor továbbra is legyen ignorálva.

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

Ez egy for cílus ami 4-szer fog lefutni. Először növeli az i-t.

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

Az i++ azt jelenti, hogy az i változót minden egyes lefutásnál növeli egyel az érékét. ($i=i+1$) (5-ször fog lefutni)

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

Végigmegy egy 5 elemű tömbön, a 0. elemtől kezdve. 5-ször fog lefutni.

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

Egy for ciklus, ahol az i 0-tól indul és addíg fut a program, amíg i kisebb egy n számnál és d pointer 1-gyel növelt értéke megegyezik az s ponter 1-gyel növelt értékével. minden ciklus előtt az i értéke 1-gyel nő.

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

Kiír 2 integer-t, az első az f függvény return értéke a.

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

Irassuk ki az f függvénynek az a outputját és magát az a-t.

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

2 integer-t, az első az f függvény return értéke a pointer dereferenciált értékével paraméterül, a második maga az a.

Megoldás forrása:

Megoldás videó:

3.7. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

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

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

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

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

Megoldás forrása: https://gitlab.com/nbatfai/bhax/blob/master/attention_raising/MatLog_LaTeX

Megoldás videó: <https://youtu.be/ZexiPy3ZxsA>, https://youtu.be/AJSXOQFF_wk

Az első formula azt mondja, hogy minden x-re létezik egy olyan y, hogy x kisebb, mint y, és y prím. Azaz minden számnál létezik egy nagyobb prímszám.

A második formula azt mondja, hogy minden x-re létezik egy olyan y, hogy x kisebb, mint y, emelett y prím és y rákövetkezőjének rákövezkezője is prím. Tehát léteznek ikerprímek.

A harmadik formula azt monja, hogy létezik olyan y, ami minden x-re igaz, hogy x prím, akkor x kisebb mint y, vagyis minden prímszámnál van nagyobb szám.

A negyedik formula azt mondja, hogy létezik olyan y, hogy minden x-re igaz, hogy ha y kisebb, mint x, akkor x nem prím. Tehát létezik olyan szám, aelytől nem létezik kisebb prímszám.

3.8. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

- egész
- egészre mutató mutató
- egész referenciajára
- 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:

```
#include <stdio.h>

int main()
{
    int a; egész
```

```
int *b = &a; //egészre mutató mutató

int &r = a; //egész referenciaja

int c[5]; //egészek tömbje

int (&tr)[2] = c; //egészek tömbjének referenciaja (nem az első elemé)

int *d[5]; //egészre mutató mutatók tömbje

int *h(); //egészre mutató mutatót visszaadó függvény

int *(*l)(); //egészre mutató mutatót visszaadó függvényre mutató ←
mutató

int (*v(int c))(int a, int b); //egészet visszaadó és két egészet ←
kapó függvényre mutató mutatót visszaadó, egészet kapó függvény

[int (*(*z)(int))(int, int); //függvénymutató egy egészet visszaadó ←
és két egészet kapó függvényre mutató mutatót visszaadó, egészet ←
kapó függvényre

return 0;
}
```

Tna..

Az utolsó két deklarációs példa demonstrálására két olyan kódot írtunk, amelyek összahasonlítása azt mutatja meg, hogy miért érdemes a **typedef** használata: [bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/fptr.c](#), [bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/fptr2.c](#).

```
#include <stdio.h>

int
sum (int a, int b)
{
    return a + b;
}

int
mul (int a, int b)
{
    return a * b;
}

int (*sumormul (int c))(int a, int b)
{
    if (c)
        return mul;
```

```
    else
        return sum;

}

int
main ()
{
    int (*f) (int, int);

    f = sum;

    printf ("%d\n", f (2, 3));

    int (*(*g) (int)) (int, int);

    g = sumormul;

    f = *g (42);

    printf ("%d\n", f (2, 3));

    return 0;
}
```

```
#include <stdio.h>

typedef int (*F) (int, int);
typedef int (*(*G) (int)) (int, int);

int
sum (int a, int b)
{
    return a + b;
}

int
mul (int a, int b)
{
    return a * b;
}

F sumormul (int c)
{
    if (c)
        return mul;
    else
        return sum;
}
```

```
int
main ()
{
    F f = sum;

    printf ("%d\n", f (2, 3));

    G g = sumormul;

    f = *g (42);

    printf ("%d\n", f (2, 3));

    return 0;
}
```

A programunk olvashatóságát nagyban növeli, ha a bonyolultabb típusneveket szinonim nevekkel helyettesítjük. Erre származtatott típusok esetén is a typedef biztosít lehetőséget.

3.9. Vörös Pipacs Pokol/csiga diszkrét mozgási parancsokkal

Megoldás videó: <https://youtu.be/eqfV9-RJ9ek>

4. fejezet

Helló, Caesar!

4.1. double ** háromszögmátrix

Írj egy olyan malloc és free párost használó C programot, amely helyet foglal egy alsó háromszög mátrixnak a szabad tárban!

Megoldás videó: https://youtu.be/W2UTW6_Kb0Q

Megoldás forrása: [bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Caesar/tm.c](https://bhax.com/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Caesar/tm.c)

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

int
main ()
{
    int nr = 5; //megadjuk hogy hány soros legyen a mátrixunk
    double **tm;

    if ((tm = (double **) malloc (nr * sizeof (double *))) == NULL)
    {
        return -1;
    }

    for (int i = 0; i < nr; ++i)
    {
        if ((tm[i] = (double *) malloc ((i + 1) * sizeof (double))) == NULL)
        {
            return -1;
        }
    }

    for (int i = 0; i < nr; ++i)
        for (int j = 0; j < i + 1; ++j)
```

```
tm[i][j] = i * (i + 1) / 2 + j;

for (int i = 0; i < nr; ++i)
{
    for (int j = 0; j < i + 1; ++j)
        printf ("%f, ", tm[i][j]);
    printf ("\n");
}

tm[3][0] = 42.0;
(*(tm + 3))[1] = 43.0; // mi van, ha itt hiányzik a külső ()
*(tm[3] + 2) = 44.0;
*(*(tm + 3) + 3) = 45.0;

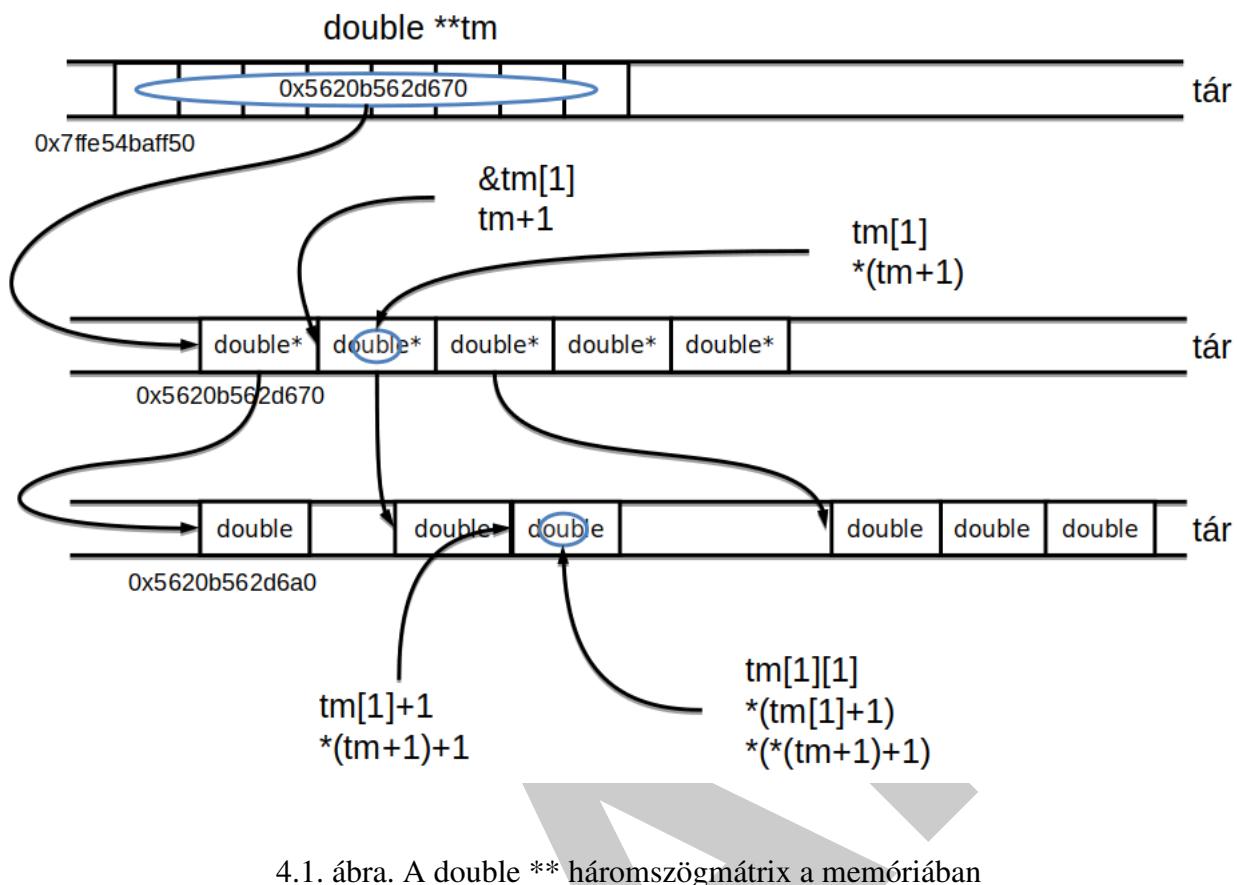
for (int i = 0; i < nr; ++i)
{
    for (int j = 0; j < i + 1; ++j)
        printf ("%f, ", tm[i][j]);
    printf ("\n");
}

for (int i = 0; i < nr; ++i)
    free (tm[i]);

free (tm);

return 0;
}
```





4.1. ábra. A double ** háromszögmátrix a memóriában

4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

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

#define MAX_KULCS 100
#define BUFFER_MERET 256

int main (int argc, char **argv)
{
    char kulcs[MAX_KULCS];
    char buffer[BUFFER_MERET];

    int kulcs_index = 0;
    int olvasott_bajtok = 0;

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

```
while ((olvasott_bajtok = read (0, (void *) buffer, BUFFER_MERET)) ←
)
{
    for (int i = 0; i < olvasott_bajtok; ++i)
    {

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

    }

    write (1, buffer, olvasott_bajtok);

}
}

}
```

Az elején sima include-olás van amiben nincs nagyon semmi különös, azonban az utána lévő sorok jóval érdekesebbek, mert én most először találkozok C-ben define parancsal. Ebben a define parancsban megadjuk a definiálni kívánt, ezesetben MAX_KULCS méretét.

A main függvénynek két paramétere is lesz, az egyik az argc (argument count), ami megszámolja, hány paraméterrel hívjuk meg a fő függvényt. Emellett a **argv függvény pedig egy olyan pointer ami rátámaszt egy pointerekből álló tömbre, amiben rátámaszt char típusú változókra.

Ezután bevezetünk két char típusú változót, amiben elmentjük a MAX_KULCS és a BUFFER_MERET értékeit. Emeltettem megítem, hogy bevezetünk még két változót kulcs_index néven és egyet olvasott_bajtok néven, amiket nyilván az egyiket a kulcs index elmentésére fogjuk alkalmazni a másikat pedig az olvasott bajtok számlálására. Még létrehozunk egy kulcs_meret nevű változót ahol elmentjük az argv[1]-nek a hosszát.

Ezt követően a while cíluson belül gyakorlatilag "össze exorozzuk" a tiszta szövegfájlt és a kulcsot, és a végeredményt kiiratjuk vele egy fájlba.

4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

Megoldás forrása: https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/ch01.html#exor_titkosito

```
public class titkosito {

    public titkosito(String kulcsSzöveg,
                      java.io.InputStream bejövő,
                      java.io.OutputStream kimenő)
```

```
throws java.io.IOException {

    byte [] kulcs = kulcsszöveg.getBytes();
    byte [] buffer = new byte[256];
    int kulcsIndex = 0;
    int olvasottBájtak = 0;

    while((olvasottBájtak =
        bejövőCsatorna.read(buffer)) != -1) {

        for(int i=0; i<olvasottBájtak; ++i) {

            buffer[i] = (byte)(buffer[i] ^ kulcs[kulcsIndex]);
            kulcsIndex = (kulcsIndex+1) % kulcs.length;

        }

        kimenőCsatorna.write(buffer, 0, olvasottBájtak);
    }

}

public static void main(String[] args) {

    try {

        new ExorTitkosító(args[0], System.in, System.out);

    } catch(java.io.IOException e) {

        e.printStackTrace();

    }
}
}
```

Először megadunk egy Class-ot, melyben elsőként a kulcsszöveget, az input és output csatornákat definiáljuk, illetve, hogy esetleg hibát is vissza adhat. Ebben a classban lesz maga a titkosítás végrehajtva.

Először is létrhözünk 2 tömböt: kulcs és buffer. A kulcs tömbbe kerül be maga a kulcs. A buffer tömbbe először a beolvasott szöveg kerül, majd ebben jön létre a titkosított szöveg is. Deklarálunk két változót a kulcsindexet és a beolvasott byteok számának változóját. Ezután jön egy While ciklus amely addig fut amíg a bufferbe van mit olvasni, és az inputot be is olvassa abba, majd ebben egy fpr ciklus, ebben lesz végrehajtva az XOR művelet mellyel "össze-exorozzuk" a a buffer aktuális karakterét a kulccsal. Ezután léptetjük a kulcsindexet, ezt addig csináljuk amíg a forban léptetett i meg nem egyezik az olvasottbájtak-1 el. Végül kiírjuk az outputra a kódolt szöveget.

Jön a main, amiben megpróbáljuk elindítani a titkosító folyamatot, hogyha nem sikerül a catch megpróbálja elkapni a hibát, majd dob egy hiba üzenetet.

4.4. C EXOR törő

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

Az exor törő gyakorlatilag úgy működik, hogy mégegyszer ráengedjük az exor titkosítót az esetben már titkos szövegre és visszakajuk az eredeti szöveget.

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

Ebben a feladatban a cél egy olyan neurális háló létrehozása és tanítása, amely az egyszerű logikai műveletek elvégzésére képes.

A neutrális háló rendelkezik tanulási algoritmussal, vagyis képes megtalálni, hogy mit is akarunk amikor megtanítjuk OR és az AND logikai műveletre. Ebben az R programban megadjuk neki mi a bemenet és, hogy milyen kimenetet várunk erre. Az a1 és az a2 számokat tartalmaz és az OR-ban pedig a kimenetet tartalmazza. Azonban az EXOR műveletnél csak többrétegű neuronokkal lehet tanulni.

```
library(neuralnet)

a1      <- c(0,1,0,1)
a2      <- c(0,0,1,1)
OR      <- c(0,1,1,1)

or.data <- data.frame(a1, a2, OR)

nn.or <- neuralnet(OR~a1+a2, or.data, hidden=0, linear.output=FALSE, ←
  stepmax = 1e+07, threshold = 0.000001)

plot(nn.or)

compute(nn.or, or.data[,1:2])

a1      <- c(0,1,0,1)
a2      <- c(0,0,1,1)
OR      <- c(0,1,1,1)
AND    <- c(0,0,0,1)

orand.data <- data.frame(a1, a2, OR, AND)

nn.orand <- neuralnet(OR+AND~a1+a2, orand.data, hidden=0, linear.output= ←
  FALSE, stepmax = 1e+07, threshold = 0.000001)

plot(nn.orand)
```

```
compute(nn.operand, operand.data[,1:2])

a1      <- c(0,1,0,1)
a2      <- c(0,0,1,1)
EXOR    <- c(0,1,1,0)

exor.data <- data.frame(a1, a2, EXOR)

nn.exor <- neuralnet(EXOR~a1+a2, exor.data, hidden=0, linear.output=FALSE, ←
  stepmax = 1e+07, threshold = 0.000001)

plot(nn.exor)

compute(nn.exor, exor.data[,1:2])

a1      <- c(0,1,0,1)
a2      <- c(0,0,1,1)
EXOR    <- c(0,1,1,0)

exor.data <- data.frame(a1, a2, EXOR)

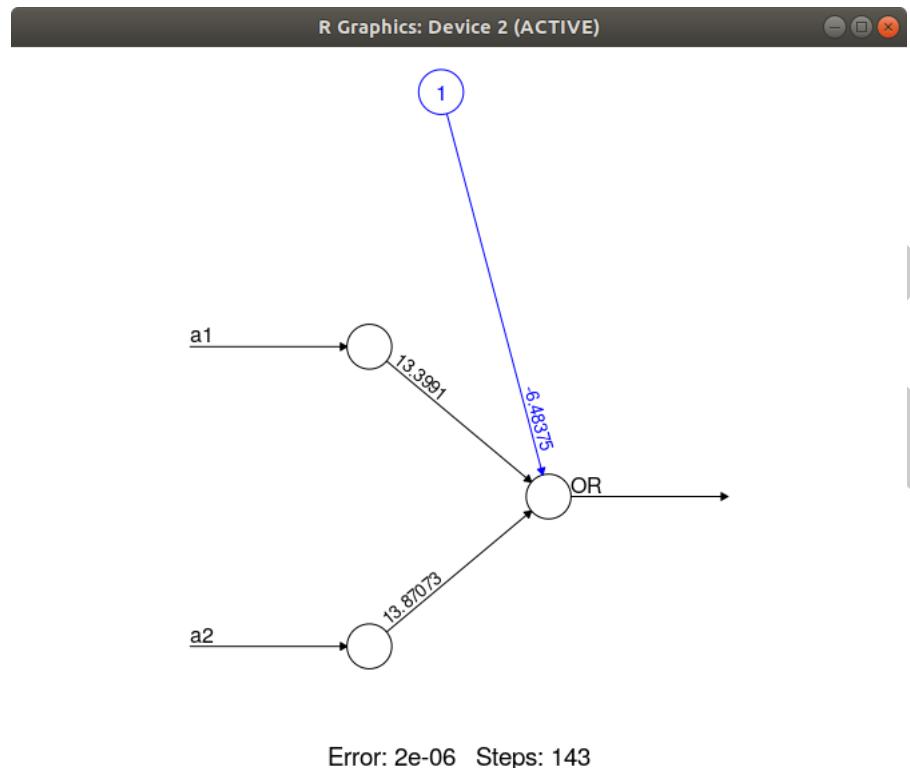
nn.exor <- neuralnet(EXOR~a1+a2, exor.data, hidden=c(6, 4, 6), linear. ←
  output=FALSE, stepmax = 1e+07, threshold = 0.000001)

plot(nn.exor)

compute(nn.exor, exor.data[,1:2])
```

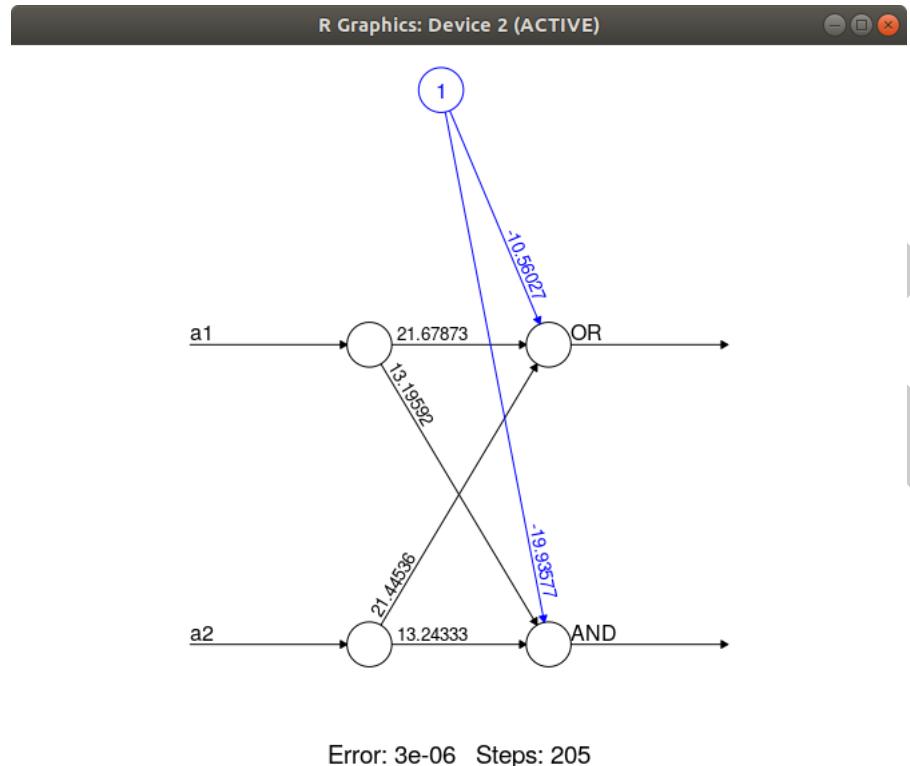
Megtanulta az OR kapcsolatot:

```
$net.result
 [,1]
[1,] 0.001525745
[2,] 0.999008550
[3,] 0.999381121
[4,] 0.999999999
```



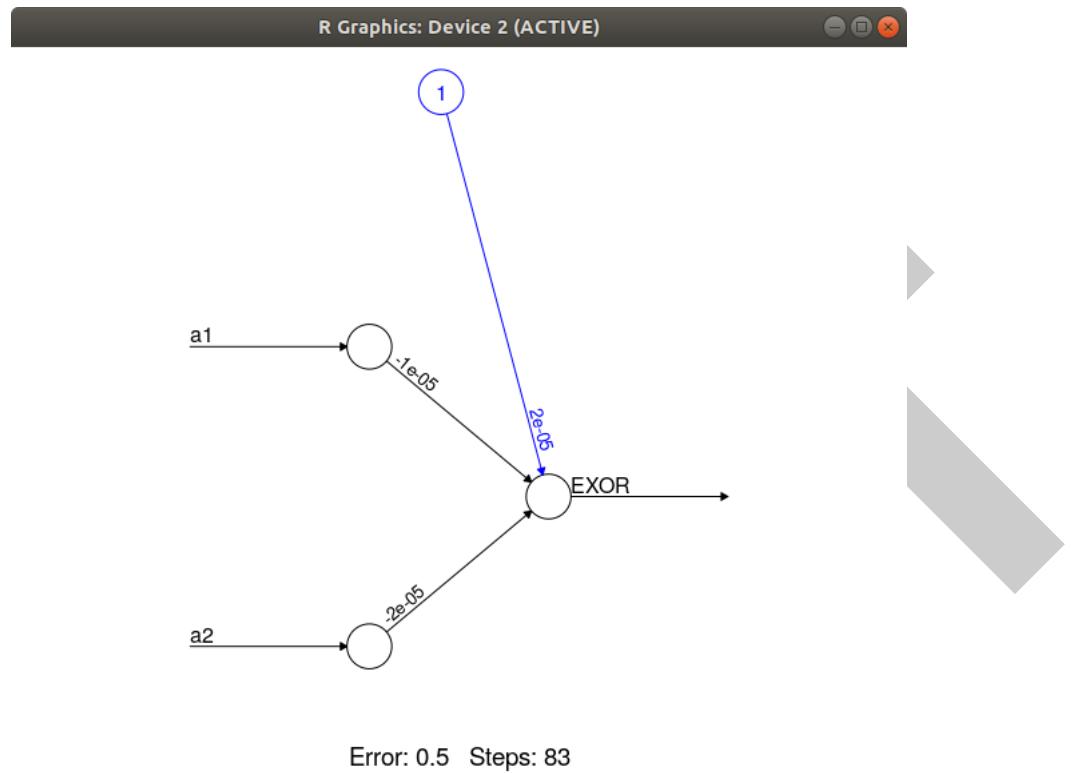
Megtanulta az AND kapcsolatot:

```
$net.result
 [,1]      [,2]
[1,] 0.0000259252 2.197895e-09
[2,] 0.9999851644 1.181429e-03
[3,] 0.9999812648 1.238723e-03
[4,] 1.0000000000 9.985040e-01
```



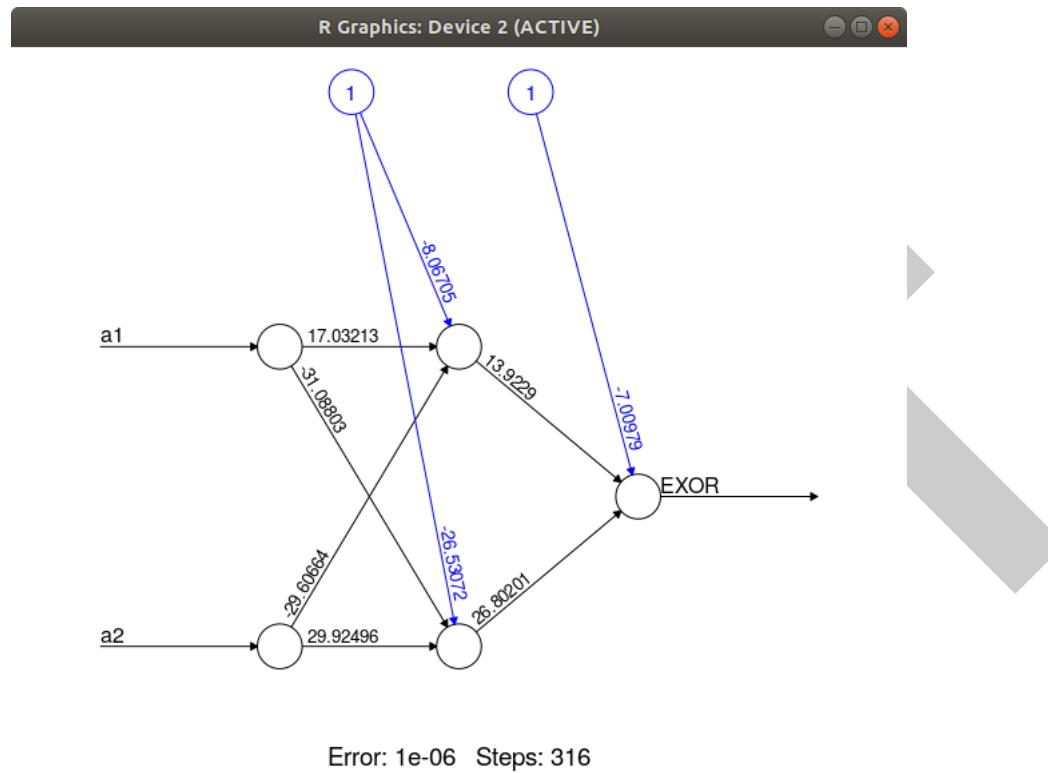
Látható, hogy nem tanulta meg az EXOR kapcsolatot mivel nincs rejtett neuron benne (hidden = 0).

```
$net.result  
      [,1]  
[1,] 0.5000044  
[2,] 0.5000025  
[3,] 0.4999991  
[4,] 0.4999973
```



Azonban, ha rakunk bele 2 rejtett neuront akkor képes rá (hidden = 2):

```
$net.result  
[,1]  
[1,] 0.0009061289  
[2,] 0.9990045591  
[3,] 0.9999999939  
[4,] 0.0009021846
```



4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó: <https://youtu.be/XpBnR31BRJY>

Megoldás forrása: <https://github.com/nbatfai/nahshon/blob/master/ql.hpp#L64>

Itt a mandelbrot által generált kép rgb kódjait rakjuk át a neurális háló bemenetére.

```
#include <iostream>
#include <cmath>
#include "png++/png.hpp"

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        std::cout << "Használat: ./mandelpng fajlnev";
        return -1;
    }

    double a = -2.0, b = .7, c = -1.35, d = 1.35;
```

```
int szelesseg = 600, magassag = 600, iteraciosHatar = 1000;

png::image <png::rgb_pixel> kep(szelesseg, magassag);

double dx = (b-a)/szelesseg;
double dy = (d-c)/magassag;
double reC, imC, rez, imZ, ujrez, ujimZ;

int iteracio = 0;
std::cout << "Szamitas";

for (int j = 0; j < magassag; j++)
{
    for (int k = 0; k < szelesseg; k++)
    {
        reC = a + k *dx;
        imC = d - j*dy;
        rez = imZ = iteracio = 0;

        while (pow(rez, 2) + pow(imZ, 2) < 4 && iteracio < ↪
               iteraciosHatar)
        {
            ujrez = pow(rez, 2) - pow(imZ, 2) + reC;
            ujimZ = 2*rez*imZ + imC;
            rez = ujrez;
            imZ = ujimZ;

            iteracio++;
        }

        kep.set_pixel(k, j, png::rgb_pixel(255 - iteracio % 256, 255 - ↪
                                           iteracio % 256, 255 - iteracio % 256));
    }
    std::cout << "." << std::flush;
}
kep.write(argv[1]);
std::cout << argv[1] << "mentve" << std::endl;
}
```

4.7. Vörös Pipacs Pokol/írd ki, mit lát Steve

Megoldás videó: <https://youtu.be/gH4xrGpdO0c>

Ebben a videóban látható a 3. szakkör megoldása c++-ban a videó végén.

5. fejezet

Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

Írj olyan C programot, amely kiszámolja a Mandelbrot halmazt!

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

Megoldás forrása: bhax/attention_raising/CUDA/mandelpngt.cpp nevű állománya.

5.1. ábra. A Mandelbrot halmaz a komplex síkon

A Mandelbrot halmazt 1980-ban találta meg Benoit Mandelbrot a komplex számsíkon. Komplex számok

azok a számok, amelyek körében válaszolni lehet az olyan egyébként értelmezhetetlen kérdésekre, hogy melyik az a két szám, amelyet összeszorozva -9-et kapunk, mert ez a szám például a 3i komplex szám.

A Mandelbrot halmazt úgy láthatjuk meg, hogy a sík origója középpontú 4 oldalhosszúságú négyzetbe lefektetünk egy, mondjuk 800x800-as rácsot és kiszámoljuk, hogy a rács pontjai mely komplex számoknak felelnek meg. A rács minden pontját megvizsgáljuk a $z_{n+1} = z_n^2 + c$, ($0 <= n$) képlet alapján úgy, hogy a c az éppen vizsgált rácspont. A z_0 az origó. Alkalmazva a képletet a

- $z_0 = 0$
- $z_1 = 0^2 + c = c$
- $z_2 = c^2 + c$
- $z_3 = (c^2 + c)^2 + c$
- $z_4 = ((c^2 + c)^2 + c)^2 + c$
- ... s így tovább.

Azaz kiindulunk az origóból (z_0) és elugrunk a rács első pontjába a $z_1 = c$ -be, aztán a c -től függően a további z -kbe. Ha ez az utazás kivezet a 2 sugarú körből, akkor azt mondjuk, hogy az a vizsgált rácspont nem a Mandelbrot halmaz eleme. Nyilván nem tudunk végletesen sok z -t megvizsgálni, ezért csak véges sok z elemet nézünk meg minden rácsponthoz. Ha eközben nem lép ki a körből, akkor feketére színezzük, hogy az a c rácspont a halmaz része. (Színes meg úgy lesz a kép, hogy változatosan színezzük, például minél későbbi z -nél lép ki a körből, annál sötétebbre).

```
#include <iostream>
#include "png++/png.hpp"
#include <sys/times.h>

#define MERET 600
#define ITER_HAT 32000

void
mandel (int kepadat [MERET] [MERET]) {

    // Mérünk időt (PP 64)
    clock_t delta = clock ();
    // Mérünk időt (PP 66)
    struct tms tmsbuf1, tmsbuf2;
    times (&tmsbuf1);

    // számítás adatai
    float a = -2.0, b = .7, c = -1.35, d = 1.35;
    int szelesseg = MERET, magassag = MERET, iteraciosHatar = ITER_HAT;

    // a számítás
    float dx = (b - a) / szelesseg;
    float dy = (d - c) / magassag;
    float reC, imC, rez, imZ, ujrez, ujimZ;
```

```
// Hány iterációt csináltunk?
int iteracio = 0;
// Végigzongorázzuk a szélesség x magasság rácsot:
for (int j = 0; j < magassag; ++j)
{
    //sor = j;
    for (int k = 0; k < szelesseg; ++k)
    {
        // c = (reC, imC) a rács csomópontjainak
        // megfelelő komplex szám
        reC = a + k * dx;
        imC = d - j * dy;
        // z_0 = 0 = (reZ, imZ)
        reZ = 0;
        imZ = 0;
        iteracio = 0;
        // z_{n+1} = z_n * z_n + c iterációk
        // számítása, amíg |z_n| < 2 vagy még
        // nem értük el a 255 iterációt, ha
        // viszont elértek, akkor úgy vesszük,
        // hogy a kiinduláci c komplex számra
        // az iteráció konvergens, azaz a c a
        // Mandelbrot halmaz eleme
        while (reZ * reZ + imZ * imZ < 4 && iteracio < iteraciosHatar)
        {
            // z_{n+1} = z_n * z_n + c
            ujreZ = reZ * reZ - imZ * imZ + reC;
            ujimZ = 2 * reZ * imZ + imC;
            reZ = ujreZ;
            imZ = ujimZ;

            ++iteracio;
        }

        kepadat[j][k] = iteracio;
    }
}

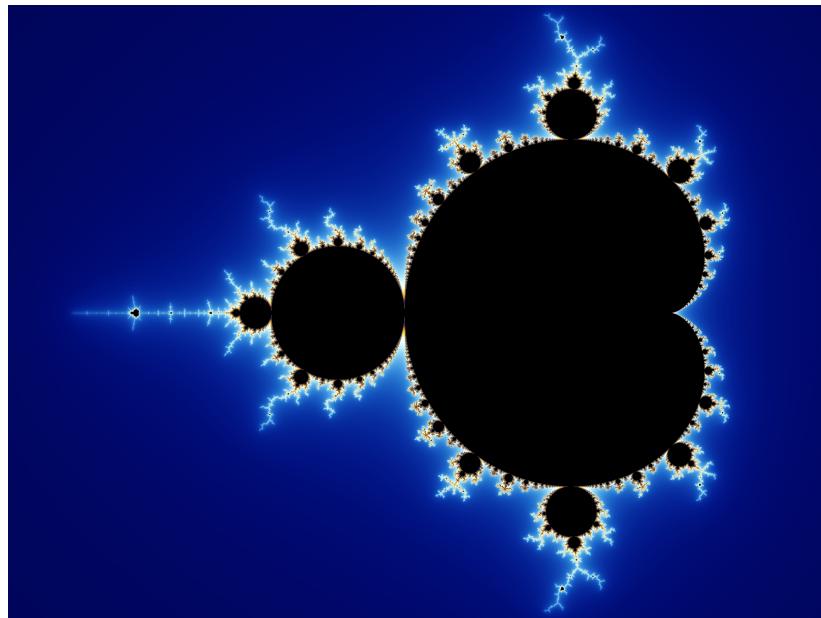
times (&tmsbuf2);
std::cout << tmsbuf2.tms_utime - tmsbuf1.tms_utime
    + tmsbuf2.tms_stime - tmsbuf1.tms_stime << std::endl;

delta = clock () - delta;
std::cout << (float) delta / CLOCKS_PER_SEC << " sec" << std::endl;
}

int
main (int argc, char *argv[])
```

```
{  
  
    if (argc != 2)  
    {  
        std::cout << "Használat: ./mandelpng fajlnev";  
        return -1;  
    }  
  
    int kepadat[MERET][MERET];  
  
    mandel(kepadat);  
  
    png::image<png::rgb_pixel> kep(MERET, MERET);  
  
    for (int j = 0; j < MERET; ++j)  
    {  
        //sor = j;  
        for (int k = 0; k < MERET; ++k)  
        {  
            kep.set_pixel(k, j,  
                          png::rgb_pixel(255 -  
                                         (255 * kepadat[j][k]) / ITER_HAT ↔  
                                         ,  
                                         255 -  
                                         (255 * kepadat[j][k]) / ITER_HAT ↔  
                                         ,  
                                         255 -  
                                         (255 * kepadat[j][k]) / ITER_HAT ↔  
                                         ));  
        }  
    }  
  
    kep.write(argv[1]);  
    std::cout << argv[1] << " mentve" << std::endl;  
}
```

A matematikában a Mandelbrot-halmaz azon a komplex számokból áll, melyekre az alábbi (komplex szám értékű) $x(n)$ rekurzív sorozat: $[x_1 := c \ x_n + 1 := (x_n)^2 + c]$ nem tart a végtelenbe.



5.2. ábra.

5.2. A Mandelbrot halmaz a `std::complex` osztályval

Írj olyan C++ programot, amely kiszámolja a Mandelbrot halmazt!

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

Megoldás forrása:

A **Mandelbrot halmaz** pontban vázolt ismert algoritmust valósítja meg a repó [bhaxor/attention_raising/Mandelbrot/3.1.2.cpp](https://github.com/bhaxor/attention_raising/tree/main/Mandelbrot/3.1.2.cpp) nevű állománya.

```
// Verzio: 3.1.2.cpp
// Forditas:
// g++ 3.1.2.cpp -lpng -O3 -o 3.1.2
// Futtatas:
// ./3.1.2 mandel.png 1920 1080 2040 ←
// -0.01947381057309366392260585598705802112818 ←
// -0.0194738105725413418456426484226540196687 ←
// 0.7985057569338268601555341774655971676111 ←
// 0.798505756934379196110285192844457924366
// ./3.1.2 mandel.png 1920 1080 1020 ←
// 0.4127655418209589255340574709407519549131 ←
// 0.4127655418245818053080142817634623497725 ←
// 0.2135387051768746491386963270997512154281 ←
// 0.2135387051804975289126531379224616102874
// Nyomtatas:
// a2ps 3.1.2.cpp -o 3.1.2.cpp.pdf -l --line-numbers=1 --left-footer="←
// BATF41 HAXOR STR34M" --right-footer="https://bhaxor.blog.hu/" --pro= ←
// color
```

```
// ps2pdf 3.1.2.cpp.pdf 3.1.2.cpp.pdf.pdf
//
//
// 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 <iostream>
#include "png++/png.hpp"
#include <complex>

int
main ( int argc, char *argv[] )
{

    int szelesseg = 1920;
    int magassag = 1080;
    int iteraciosHatar = 255;
    double a = -1.9;
    double b = 0.7;
    double c = -1.3;
    double d = 1.3;

    if ( argc == 9 )
    {
        szelesseg = atoi ( argv[2] );
        magassag = atoi ( argv[3] );
        iteraciosHatar = atoi ( argv[4] );
        a = atof ( argv[5] );
        b = atof ( argv[6] );
        c = atof ( argv[7] );
        d = atof ( argv[8] );
    }
    else
    {
        std::cout << "Használat: ./3.1.2 fajlnev szelesseg magassag n a b c d ←
        " << std::endl;
    }
}
```

```
    return -1;
}

png::image < png::rgb_pixel > kep ( szelesseg, magassag );

double dx = ( b - a ) / szelesseg;
double dy = ( d - c ) / magassag;
double reC, imC, reZ, imZ;
int iteracio = 0;

std::cout << "Szamitas\n";

// j megy a sorokon
for ( int j = 0; j < magassag; ++j )
{
    // k megy az oszlopokon

    for ( int k = 0; k < szelesseg; ++k )
    {

        // c = (reC, imC) a halo racspontjainak
        // megfelelo komplex szam

        reC = a + k * dx;
        imC = d - j * dy;
        std::complex<double> c ( reC, imC );

        std::complex<double> z_n ( 0, 0 );
        iteracio = 0;

        while ( std::abs ( z_n ) < 4 && iteracio < iteraciosHatar )
        {
            z_n = z_n * z_n + c;

            ++iteracio;
        }

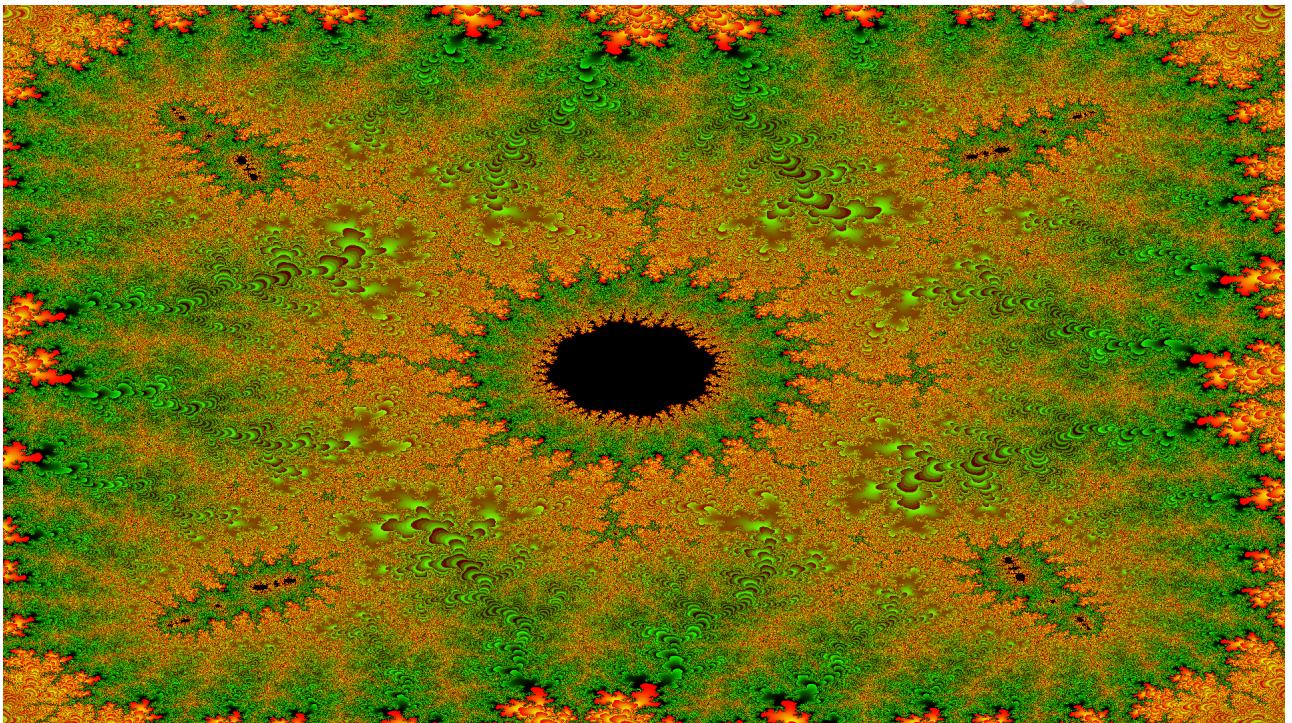
        kep.set_pixel ( k, j,
                        png::rgb_pixel ( iteracio%255, (iteracio*iteracio <=
                           )%255, 0 ) );
    }
}

int szazalek = ( double ) j / ( double ) magassag * 100.0;
std::cout << "\r" << szazalek << "%" << std::flush;
}

kep.write ( argv[1] );
std::cout << "\r" << argv[1] << " mentve." << std::endl;
}
```

Ez a fajta Mandelbrot halmaz-rajzoló program nem sokban különbözik az előző feladatban tárgyalottól, csupán includeoltuk a complex könyvtárat, aminek segítségével könnyen tudunk komplex számokkal dolgozni.

Ezt volt a program kimenetele:



5.3. Biomorfok

Megoldás video: <https://youtu.be/IJMbgrzY76E>

Megoldás forrása: https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/Biomorf

A biomorfokra (a Julia halmazokat rajzoló bug-os programjával) rátaláló Clifford Pickover azt hitte természeti törvényre bukkant: https://www.emis.de/journals/TJNSA/includes/files/articles/Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf (lásd a 2307. oldal aljától).

A különbség a **Mandelbrot halmaz** és a Julia halmazok között az, hogy a komplex iterációban az előbbiben a c változó, utóbbiban pedig állandó. A következő Mandelbrot císpel azt mutatja, hogy a c befutja a vizsgált összes rácspontot.

```
// j megy a soron
for ( int j = 0; j < magassag; ++j )
{
    for ( int k = 0; k < szelesseg; ++k )

        // c = (reC, imC) a halo racspontjainak
        // megfelelo komplex szam
```

```
reC = a + k * dx;
imC = d - j * dy;
std::complex<double> c ( reC, imC );

std::complex<double> z_n ( 0, 0 );
iteracio = 0;

while ( std::abs ( z_n ) < 4 && iteracio < iteraciosHatar )
{
    z_n = z_n * z_n + c;

    ++iteracio;
}
```

Ezzel szemben a Julia halmazos csipetben a cc nem változik, hanem minden vizsgált z rácpontra ugyanaz.

```
// j megy a sorokon
for ( int j = 0; j < magassag; ++j )
{
    // k megy az oszlopokon
    for ( int k = 0; k < szelesség; ++k )
    {
        double rez = a + k * dx;
        double imZ = d - j * dy;
        std::complex<double> z_n ( rez, imZ );

        int iteracio = 0;
        for (int i=0; i < iteraciosHatar; ++i)
        {
            z_n = std::pow(z_n, 3) + cc;
            if (std::real ( z_n ) > R || std::imag ( z_n ) > R)
            {
                iteracio = i;
                break;
            }
        }
    }
}
```

A bimorfos algoritmus pontos megismeréséhez ezt a cikket javasoljuk: https://www.emis.de/journals/TJNSA/includes/files/articles/Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf. Az is jó gyakorlat, ha magából ebből a cikkből from scratch kódoljuk be a sajtunkat, de mi a királyi úton járva a korábbi **Mandelbrot halmazt** kiszámoló forrásunkat módosítjuk. Viszont a program változóinak elnevezését összhangba hozzuk a közlemény jelöléseivel:

```
// Verzio: 3.1.3.cpp
// Forditas:
// g++ 3.1.3.cpp -lpng -O3 -o 3.1.3
// Futtatas:
// ./3.1.3 bmorf.png 800 800 10 -2 2 -2 2 .285 0 10
// Nyomtatas:
```

```
// a2ps 3.1.3.cpp -o 3.1.3.cpp.pdf -l --line-numbers=1 --left-footer=" ↵
// BATF41 HAXOR STR34M" --right-footer="https://bhaxor.blog.hu/" --pro= ↵
// color
//
// BHAX Biomorphs
// Copyright (C) 2019
// Norbert Batfai, 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/>.
//
// Version history
//
// https://youtu.be/IJMbqRzY76E
// See also https://www.emis.de/journals/TJNSA/includes/files/articles/ ↵
// Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf
//

#include <iostream>
#include "png++/png.hpp"
#include <complex>

int
main ( int argc, char *argv[] )
{

    int szelesseg = 1920;
    int magassag = 1080;
    int iteraciosHatar = 255;
    double xmin = -1.9;
    double xmax = 0.7;
    double ymin = -1.3;
    double ymax = 1.3;
    double reC = .285, imC = 0;
    double R = 10.0;

    if ( argc == 12 )
    {
        szelesseg = atoi ( argv[2] );
        magassag = atoi ( argv[3] );
    }
}
```

```
iteraciosHatar = atoi ( argv[4] );
xmin = atof ( argv[5] );
xmax = atof ( argv[6] );
ymin = atof ( argv[7] );
ymax = atof ( argv[8] );
reC = atof ( argv[9] );
imC = atof ( argv[10] );
R = atof ( argv[11] );

}

else
{
    std::cout << "Hasznalat: ./3.1.2 fajlnev szelesseg magassag n a b c ←
        d reC imC R" << std::endl;
    return -1;
}

png::image<png::rgb_pixel> kep ( szelesseg, magassag );

double dx = ( xmax - xmin ) / szelesseg;
double dy = ( ymax - ymin ) / magassag;

std::complex<double> cc ( reC, imC );

std::cout << "Szamitas\n";

// j megy a sorokon
for ( int y = 0; y < magassag; ++y )
{
    // k megy az oszlopokon

    for ( int x = 0; x < szelesseg; ++x )

        double reZ = xmin + x * dx;
        double imZ = ymax - y * dy;
        std::complex<double> z_n ( reZ, imZ );

        int iteracio = 0;
        for (int i=0; i < iteraciosHatar; ++i)
        {

            z_n = std::pow(z_n, 3) + cc;
            //z_n = std::pow(z_n, 2) + std::sin(z_n) + cc;
            if(std::real ( z_n ) > R || std::imag ( z_n ) > R)
            {
                iteracio = i;
                break;
            }
        }
}
```

```
kep.set_pixel ( x, y,
                png::rgb_pixel ( (iteracio*20)%255, (iteracio ←
                    *40)%255, (iteracio*60)%255 ) );
}

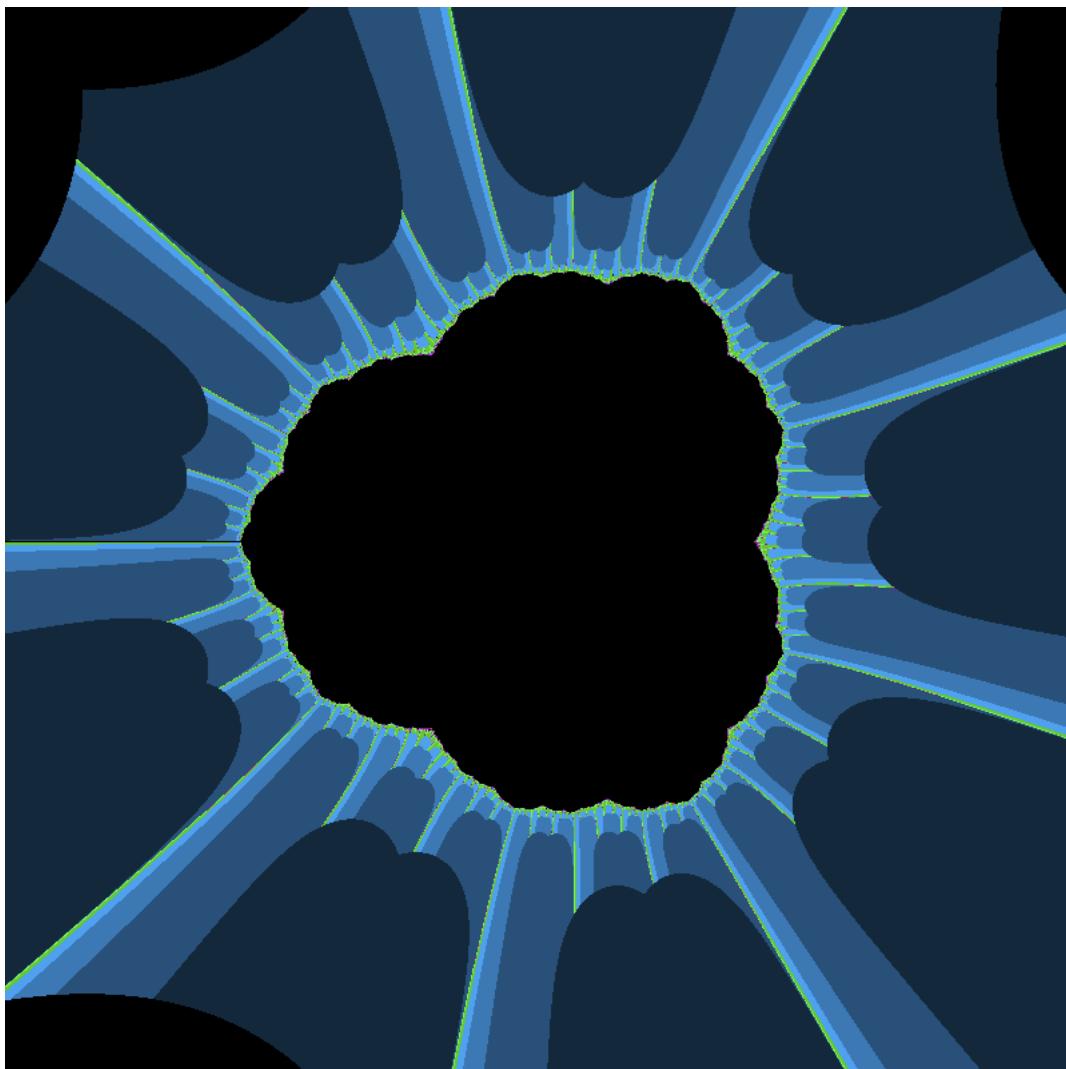
int szazalek = ( double ) y / ( double ) magassag * 100.0;
std::cout << "\r" << szazalek << "%" << std::flush;
}

kep.write ( argv[1] );
std::cout << "\r" << argv[1] << " mentve." << std::endl;
}
```

A biomorfok olyan alakzatok, amelyek ránézésre akár élő organizmusok is lehetnének, viszont nem muszáj természetes eredetűnek lennie az alakzatnak (magyarán, akár lehetnek számítógép által generáltak is).

Ez a program nagyon hasonlít az előzőhöz, ugyanis ennek az alapja a Mandelbrot-halmaz. A legjelentősebb eltérés az előző programhoz képest, hogy itt más a megadott formula a pixelek színeinek számításánál.

A végeredmény pedig:



5.4. A Mandelbrot halmaz CUDA megvalósítása

Megoldás video: <https://youtu.be/gvaqijHlRUs>

Megoldás forrása: [bhax/attention_raising/CUDA/mandelpngc_60x60_100.cu](https://github.com/bhax/attention_raising/CUDA/mandelpngc_60x60_100.cu) nevű állománya.

```
// mandelpngc_60x60_100.cu
// 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/>.
//
// Version history
//
// Mandelbrot png
// Programozó Páternoszter/PARP
// https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0063 ←
// _01_parhuzamos_prog_linux
//
// https://youtu.be/gvaqijHlRUs
//

#include <png++/image.hpp>
#include <png++/rgb_pixel.hpp>

#include <sys/times.h>
#include <iostream>

#define MERET 600
#define ITER_HAT 32000

__device__ int
mandel (int k, int j)
{
    // Végigzongorázza a CUDA a szélesség x magasság rácsot:
    // most eppen a j. sor k. oszlopban vagyunk

    // számítás adatai
    float a = -2.0, b = .7, c = -1.35, d = 1.35;
    int szelesseg = MERET, magassag = MERET, iteraciosHatar = ITER_HAT;

    // a számítás
    float dx = (b - a) / szelesseg;
    float dy = (d - c) / magassag;
    float reC, imC, reZ, imZ, ujreZ, ujimZ;
    // Hány iterációt csináltunk?
    int iteracio = 0;

    // c = (reC, imC) a rács csomópontjainak
    // megfelelő komplex szám
    reC = a + k * dx;
    imC = d - j * dy;
    // z_0 = 0 = (reZ, imZ)
    reZ = 0.0;
    imZ = 0.0;
```

```
iteracio = 0;
// z_{n+1} = z_n * z_n + c iterációk
// számítása, amíg |z_n| < 2 vagy még
// nem értük el a 255 iterációt, ha
// viszont elértek, akkor úgy vesszük,
// hogy a kiinduláci c komplex számra
// az iteráció konvergens, azaz a c a
// Mandelbrot halmaz eleme
while (reZ * reZ + imZ * imZ < 4 && iteracio < iteracionsHatar)
{
    // z_{n+1} = z_n * z_n + c
    ujreZ = reZ * reZ - imZ * imZ + reC;
    ujimZ = 2 * reZ * imZ + imC;
    reZ = ujreZ;
    imZ = ujimZ;

    ++iteracio;

}
return iteracio;
}

/*
__global__ void
mandelkernel (int *kepadat)
{
    int j = blockIdx.x;
    int k = blockIdx.y;

    kepadat[j + k * MERET] = mandel (j, k);

}
*/

__global__ void
mandelkernel (int *kepadat)
{
    int tj = threadIdx.x;
    int tk = threadIdx.y;

    int j = blockIdx.x * 10 + tj;
    int k = blockIdx.y * 10 + tk;

    kepadat[j + k * MERET] = mandel (j, k);

}
```



```
    255 -
    (255 * kepadat[j][k]) / ITER_HAT,
    255 -
    (255 * kepadat[j][k]) / ITER_HAT));
}
}
kep.write (argv[1]);

std::cout << argv[1] << " mentve" << std::endl;

times (&tmsbuf2);
std::cout << tmsbuf2.tms_utime - tmsbuf1.tms_utime
+ tmsbuf2.tms_stime - tmsbuf1.tms_stime << std::endl;

delta = clock () - delta;
std::cout << (float) delta / CLOCKS_PER_SEC << " sec" << std::endl;
}
```

Csak ezzé jellegű kidolgozás, mert nem tudtam lefuttatni a programot.

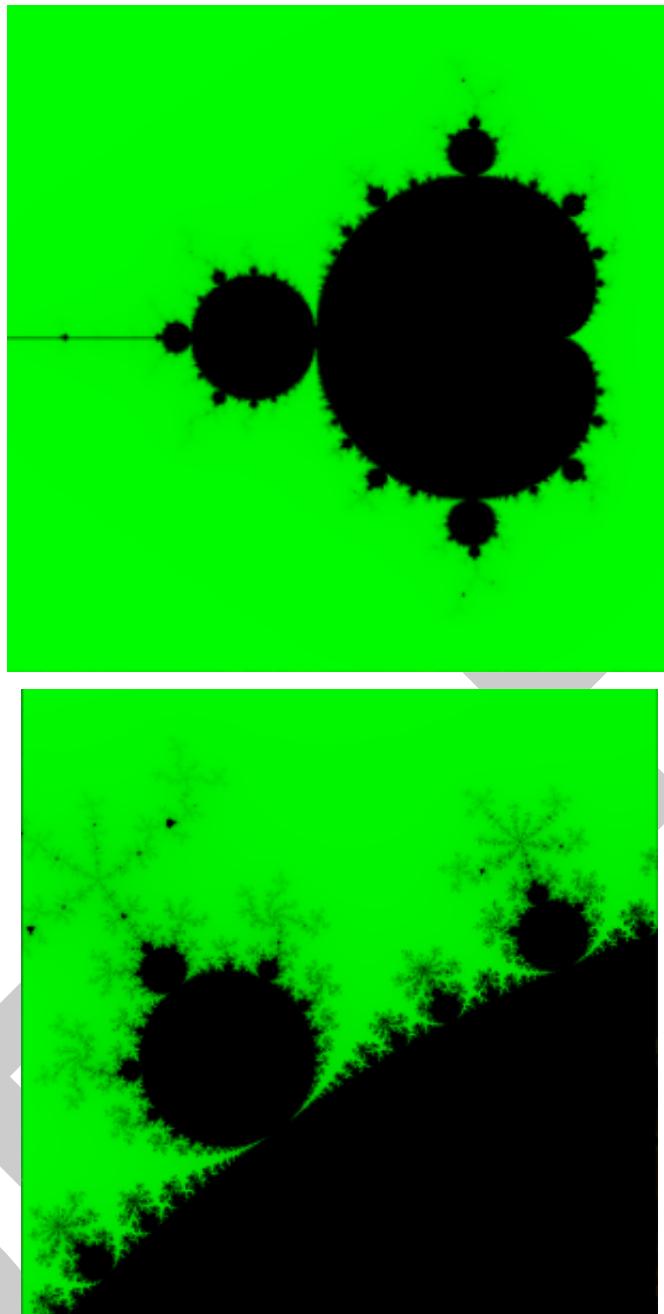
Kezdetben ehhez a programhoz, már a fordításához szükséges egy NVIDIA videókártya, illetve az nvidia-cuda-toolkit. Ezután nvcc - vel illetve -lpng16 al fordítható ,majd ./programnév képnév.png vel futtatható. Ez a program rendkívül szoros köteléket ápol az előző programokkal , a különbség viszont jelentős ugyanis amíg az előző programokban a CPU dolgozik 1 maggal , itt a NVIDIA GPU dolgozik a benne levő CUDA maggal. Az időkülönbség jelentős.

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 videó: Illetve https://bhaxor.blog.hu/2018/09/02/ismerkedes_a_mandelbrot_halmazzal.

A mostani megoldásban nem csak simán kimetnek kapunk egy képet hanem abba képbe bele is tudunk nagyítani a ./frak parancsal. Előtte persze létre kell hozni a képet.



5.6. Mandelbrot nagyító és utazó Java nyelven

Megoldás videó: <https://youtu.be/Ui3B6IJnssY>, 4:27-től. Illetve https://bhaxor.blog.hu/2018/09/02/ismerkedes_a_mandelbrotnagyito_uttazotthon_javaban/

Megoldás forrása: <https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apbs02.html#id570518>

Ez az 5.5 feladat megoldása java-ban. Így a program javac parancsal fog fordulni.

Emelett pár "pluszal" egészül ki, mint példul lehet képet készíteni róla, növelhetjük a halmaz számolásának pontosságát, emellett oda nagyíthat bele az ember ahova csak akar az egér segítségével.

5.7. Vörös Pipacs Pokol/csiga diszkrét mozgási parancsokkal

Megoldás videó:



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 eltéve kiszámolt szám.

Megoldás videó:

Megoldás forrása:

6.2. LZW

Valósítsd meg C++-ban az LZW algoritmus fa-építését!



```
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Welch$ ./z3a184
BT ctor
.....-2 3 0
.....-5 2 0
.....-7 3 0
--8 1 0
.....9 2 0

BT ctor
.....-0 3 0
.....-0 2 0
.....-1 3 0
--1 0
.....0 5 0
.....-0 4 0
.....-0 3 0
.....1 2 0
.....-1 3 0
.....-1 4 0
BT copy ctor
BT ctor
!**!
BT copy assign
BT copy ctor
BT move ctor
BT move assign
BT move assign
BT move assign
BT dtor
BT dtor
!**
BT move ctor
BT move assign
preorder bejaras:
/ 1 1
8 5 2 7 9
lnorder bejaras:
/ 1 1
2 5 4 8 9
postorder bejaras:
1 1 /
2 7 5 9 8
BT dtor
BT dtor
BT dtor
BT dtor
BT dtor
mate@laptop:~/bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Welch$
```

6.3. Fabejárás

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

Az 6.2-es feladatban van.

6.4. Tag a gyökér

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

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Welch/z3a19a_from_scratch.cpp

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 értékkopírást, a mozgató konstruktur legyen a mozgató értékkopíráshoz alapozva!

A megírt programunkban a move assign -on belüli swap végett a mozgató konstruktur, a mozgató értékkopíráshoz alapozva.

Megoldás videó:

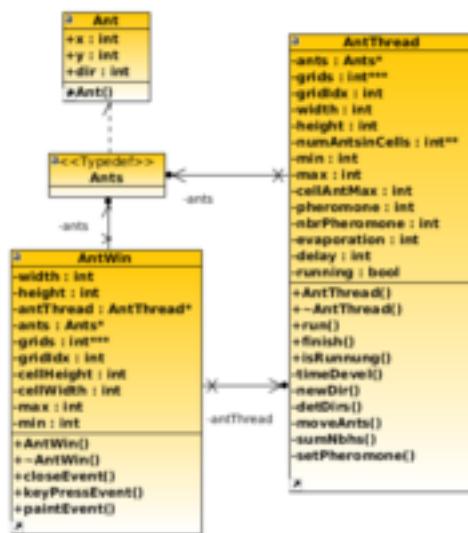
Megoldás forrása:

7. fejezet

Helló, Conway!

7.1. Hangyszimulációk

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



Ez az UML ami lehetővé teszi, hogy grafikusan a való életben is vizualizáljuk az osztályokat, a köztük lévő kapcsolatokat, azok tulajdonságait, elemeit és ezeknek viselkedését. Itt egy részben egy bizonyos osztály látható, annak kapcsolatai a többi osztállyal, illetve az elemei. Ahhoz hogy egy osztályelem láthatóságát definiáljuk speciális karaktereket kell használnunk a `+` a `public` ,`a` `-private`, és a `#` a `protected` ,ezeket az elemek neve elő kell helyeznünk.

Ez a program a hangyák viselkedését igyekezik imitálni. Tudni illik a hangyák nem magányos farkasként élik minden napjaikat, inkább csapatokba tömörülve mennek például élelmet kutatni. A hangyák képesek feromonjaikat a környezetükbe engedni, ezzel jelet hagyva a többi hangyának, hogy merre érdemes menni, kutatni, illetve a többi hangya ezt követi, ugyanis a hangyák nem képesek szóban kommunikálni, ezért valamilyen alternatív megoldást kell alkalmazniuk, hogy megértsék egymást, mint ahogy a többi élőlények is. A programunk is erre az elvre alapszik, látszólag egy ablakban megjelennek kis fekete pixelek melyek mozognak és hagynak maguk után egy színes csíkot, majd ha egy másik pixel betéved ebbe a csíkba,

követi azt. Tehát az elmélet: Hangya kutat, feromont hagy maga után, egy másik hangya követi, és minél több a feromon egy helyen, annál többenn mennek arra.

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

Megoldás forrása:

7.2. Java életjáték

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

Megoldás videó:

Megoldás forrása: https://github.com/ElStl/bhax/tree/master/bhax_textbook_IgyNeveldaProgramozod/Conway/-sejtautamata.java

A myrmecologist.pro fájl az ami összesíti az eddig említett fájlokat ,majd később a qmake el lefordítani és létrehozzuk belőle a futtatható állományt.

7.3. Java életjáték

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

Megoldás forrása: <https://regi.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apb.html#conway>

Az ún. életjáték John Horton Conway ,a Cambridge-i Egyetem matematikusának alkotása mely szabálzai egyszerűek:egy sejt(adott esetben pixel) csak akkor lehet élő ha 2 vagy 3 élő szomszédja van,ha ennél több,vagy kevesebb akkor halott lesz,illetve egy halott sejt cskis akkor kelhet életre ha pont 3 élő szomszédja van,minden más esetben halott marad. Egy sejt környezetét jelen esetben a rácsos felületen az őt körülvevő 8 cella. Játékként való megnevezése csaló lehet ugyanis pontosan 0 személy jatszhat vele, a "játékosnak" minden össze annyi a dolga, hogy elhelzey egy kezdő alakzatot majd figyeli az eseményeket. Láthatjuk még a "SIKLÓT" melynek érdekessége csupánannyiban merül ki,hogy átlósan jobbra mozog az ablak teteje felé.

```
/*
 * Sejtautomata.java
 *
 * DIGIT 2005, Javat tanítok
 * Bátfai Norbert, nbatfai@inf.unideb.hu
 *
 */
/**
 * Sejtautomata osztály.
 *
 * @author Bátfai Norbert, nbatfai@inf.unideb.hu
 * @version 0.0.1
 */
public class Sejtautomata extends java.awt.Frame implements Runnable {
    public static final boolean ÉLŐ = true;
    public static final boolean HALOTT = false;
    protected boolean[][][] rácsok = new boolean[2][][];
}
```

```
protected boolean [][] rács;
protected int rácsIndex = 0;
protected int cellaSzélesség = 20;
protected int cellaMagasság = 20;
protected int szélesség = 20;
protected int magasság = 10;
protected int várakozás = 1000;
private java.awt.Robot robot;
private boolean pillanatfelvétel = false;
private static int pillanatfelvételSzámláló ;
public Sejtautomata(int szélesség, int magasság) {
    this.szélesség = szélesség;
    this.magasság = magasság;
    rácsok[0] = new boolean[magasság][szélesség];
    rácsok[1] = new boolean[magasság][szélesség];
    rácsIndex = 0;
    rács = rácsok[rácsIndex];
    for(int i=0; i<rács.length; ++i)
        for(int j=0; j<rács[0].length; ++j)
            rács[i][j] = HALOTT;
    siklóKilövő(rács, 5, 60);
    addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent e) {
            setVisible(false);
            System.exit(0);
        }
    });
    addKeyListener(new java.awt.event.KeyAdapter() {
        public void keyPressed(java.awt.event.KeyEvent e) {
            if(e.getKeyCode() == java.awt.event.KeyEvent.VK_K) {
                cellaSzélesség /= 2;
                cellaMagasság /= 2;
                setSize(Sejtautomata.this.szélesség*cellaSzélesség,
                        Sejtautomata.this.magasság*cellaMagasság);
                validate();
            } else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_N) {
                cellaSzélesség *= 2;
                cellaMagasság *= 2;
                setSize(Sejtautomata.this.szélesség*cellaSzélesség,
                        Sejtautomata.this.magasság*cellaMagasság);
                validate();
            } else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_S)
                pillanatfelvétel = !pillanatfelvétel;
            else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_G)
                várakozás /= 2;
            else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_L)
                várakozás *= 2;
            repaint();
        }
    });
}
```

```
addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent m) {
        int x = m.getX()/cellaSzélesség;
        int y = m.getY()/cellaMagasság;
        rácsok[rácsIndex][y][x] = !rácsok[rácsIndex][y][x];
        repaint();
    }
});
addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
    public void mouseDragged(java.awt.event.MouseEvent m) {
        int x = m.getX()/cellaSzélesség;
        int y = m.getY()/cellaMagasság;
        rácsok[rácsIndex][y][x] = ÉLŐ;
        repaint();
    }
});
cellaSzélesség = 10;
cellaMagasság = 10;
try {
    robot = new java.awt.Robot(
        java.awt.GraphicsEnvironment.
        getLocalGraphicsEnvironment().
        getDefaultScreenDevice());
} catch(java.awt.AWTException e) {
    e.printStackTrace();
}
setTitle("Sejtautomata");
setResizable(false);
setSize(szélesség*cellaSzélesség,
           magasság*cellaMagasság);
setVisible(true);
new Thread(this).start();
}
/** A sejttér kirajzolása. */
public void paint(java.awt.Graphics g) {
    boolean [][] rács = rácsok[rácsIndex];
    for(int i=0; i<rács.length; ++i) {
        for(int j=0; j<rács[0].length; ++j) {
            if(rács[i][j] == ÉLŐ)
                g.setColor(java.awt.Color.BLACK);
            else
                g.setColor(java.awt.Color.WHITE);
            g.fillRect(j*cellaSzélesség, i*cellaMagasság,
                       cellaSzélesség, cellaMagasság);
            g.setColor(java.awt.Color.LIGHT_GRAY);
            g.drawRect(j*cellaSzélesség, i*cellaMagasság,
                       cellaSzélesség, cellaMagasság);
        }
    }
    if(pillanatfelvétel) {
```

```
pillanatfelvétel = false;
pillanatfelvétel(robot.createScreenCapture
    (new java.awt.Rectangle
        (getLocation().x, getLocation().y,
        szélesség*cellaSzélesség,
        magasság*cellaMagasság)));
}

}
/** 
 * Az kérdezett állapotban lévő nyolcszomszédok száma.
 *
 * @param rács a sejttér rács
 * @param sor a rács vizsgált sora
 * @param oszlop a rács vizsgált oszlopa
 * @param állapot a nyolcszomszédok vizsgált állapota
 * @return int a kérdezett állapotbeli nyolcszomszédok száma.
 */
public int szomszédokSzáma(boolean [][] rács,
    int sor, int oszlop, boolean állapot) {
    int állapotúSzomszéd = 0;
    // A nyolcszomszédok végigzongorázása:
    for(int i=-1; i<2; ++i)
        for(int j=-1; j<2; ++j)
            // A vizsgált sejtet magát kihagyva:
            if(!((i==0) && (j==0))) {
                // A sejttérből szélének szomszédai
                // a szembe oldalakon ("periódikus határfeltétel")
                int o = oszlop + j;
                if(o < 0)
                    o = szélesség-1;
                else if(o >= szélesség)
                    o = 0;

                int s = sor + i;
                if(s < 0)
                    s = magasság-1;
                else if(s >= magasság)
                    s = 0;

                if(rács[s][o] == állapot)
                    ++állapotúSzomszéd;
            }
    return állapotúSzomszéd;
}
/** 
 * A sejttér időbeli fejlődése a John H. Conway féle
 * életjáték sejtautomata szabályai alapján történik.
 * A szabályok részletes ismertetését lásd például a
 * [MATEK JÁTÉK] hivatkozásban (Csákány Béla: Diszkrét
```

```
* matematikai játékok. Polygon, Szeged 1998. 171. oldal.)  
*/  
public void időFejlődés() {  
  
    boolean [][] rácsElőtte = rácsok[rácsIndex];  
    boolean [][] rácsUtána = rácsok[(rácsIndex+1)%2];  
  
    for(int i=0; i<rácsElőtte.length; ++i) { // sorok  
        for(int j=0; j<rácsElőtte[0].length; ++j) { // oszlopok  
  
            int élők = szomszédokSzáma(rácsElőtte, i, j, ÉLŐ);  
  
            if(rácsElőtte[i][j] == ÉLŐ) {  
                /* Élő élő marad, ha kettő vagy három élő  
                szomszedja van, különben halott lesz. */  
                if(élők==2 || élők==3)  
                    rácsUtána[i][j] = ÉLŐ;  
                else  
                    rácsUtána[i][j] = HALOTT;  
            } else {  
                /* Halott halott marad, ha három élő  
                szomszedja van, különben élő lesz. */  
                if(élők==3)  
                    rácsUtána[i][j] = ÉLŐ;  
                else  
                    rácsUtána[i][j] = HALOTT;  
            }  
        }  
    }  
    rácsIndex = (rácsIndex+1)%2;  
}  
/** A sejttér időbeli fejlődése. */  
public void run() {  
  
    while(true) {  
        try {  
            Thread.sleep(várakozás);  
        } catch (InterruptedException e) {}  
  
        időFejlődés();  
        repaint();  
    }  
}  
/**  
 * A sejttérbe "élőlényeket" helyezünk, ez a "sikló".  
 * Adott irányban halad, másolja magát a sejttérben.  
 * Az élőlény ismertetését lásd például a  
 * [MATEK JÁTÉK] hivatkozásban (Csákány Béla: Diszkrét  
 * matematikai játékok. Polygon, Szeged 1998. 172. oldal.)  
 *
```

```
* @param rács      a sejttér ahová ezt az állatkát helyezzük
* @param x         a befoglaló téglalap bal felső sarkának oszlopa
* @param y         a befoglaló téglalap bal felső sarkának sora
*/
public void sikló(boolean [][] rács, int x, int y) {

    rács[y+ 0][x+ 2] = ÉLŐ;
    rács[y+ 1][x+ 1] = ÉLŐ;
    rács[y+ 2][x+ 1] = ÉLŐ;
    rács[y+ 2][x+ 2] = ÉLŐ;
    rács[y+ 2][x+ 3] = ÉLŐ;

}

/**
 * A sejttérbe "élőlényeket" helyezünk, ez a "sikló ágyú".
 * Adott irányban siklókat lő ki.
 * Az élőlény ismertetését lásd például a
 * [MATEK JÁTÉK] hivatkozásban /Csákány Béla: Diszkrét
 * matematikai játékok. Polygon, Szeged 1998. 173. oldal./,
 * de itt az ábra hibás, egy oszloppal told még balra a
 * bal oldali 4 sejtes négyzetet. A helyes ágyú rajzát
 * lásd pl. az [ÉLET CIKK] hivatkozásban /Robert T.
 * Wainwright: Life is Universal./ (Megemlíthetjük, hogy
 * mindenkettő tartalmaz két felesleges sejtet is.)
 *
 * @param rács      a sejttér ahová ezt az állatkát helyezzük
 * @param x         a befoglaló téglalap bal felső sarkának oszlopa
 * @param y         a befoglaló téglalap bal felső sarkának sora
*/
public void siklóKilövő(boolean [][] rács, int x, int y) {

    rács[y+ 6][x+ 0] = ÉLŐ;
    rács[y+ 6][x+ 1] = ÉLŐ;
    rács[y+ 7][x+ 0] = ÉLŐ;
    rács[y+ 7][x+ 1] = ÉLŐ;

    rács[y+ 3][x+ 13] = ÉLŐ;

    rács[y+ 4][x+ 12] = ÉLŐ;
    rács[y+ 4][x+ 14] = ÉLŐ;

    rács[y+ 5][x+ 11] = ÉLŐ;
    rács[y+ 5][x+ 15] = ÉLŐ;
    rács[y+ 5][x+ 16] = ÉLŐ;
    rács[y+ 5][x+ 25] = ÉLŐ;

    rács[y+ 6][x+ 11] = ÉLŐ;
    rács[y+ 6][x+ 15] = ÉLŐ;
    rács[y+ 6][x+ 16] = ÉLŐ;
    rács[y+ 6][x+ 22] = ÉLŐ;
```

```
rács[y+ 6][x+ 23] = ÉLŐ;
rács[y+ 6][x+ 24] = ÉLŐ;
rács[y+ 6][x+ 25] = ÉLŐ;

rács[y+ 7][x+ 11] = ÉLŐ;
rács[y+ 7][x+ 15] = ÉLŐ;
rács[y+ 7][x+ 16] = ÉLŐ;
rács[y+ 7][x+ 21] = ÉLŐ;
rács[y+ 7][x+ 22] = ÉLŐ;
rács[y+ 7][x+ 23] = ÉLŐ;
rács[y+ 7][x+ 24] = ÉLŐ;

rács[y+ 8][x+ 12] = ÉLŐ;
rács[y+ 8][x+ 14] = ÉLŐ;
rács[y+ 8][x+ 21] = ÉLŐ;
rács[y+ 8][x+ 24] = ÉLŐ;
rács[y+ 8][x+ 34] = ÉLŐ;
rács[y+ 8][x+ 35] = ÉLŐ;

rács[y+ 9][x+ 13] = ÉLŐ;
rács[y+ 9][x+ 21] = ÉLŐ;
rács[y+ 9][x+ 22] = ÉLŐ;
rács[y+ 9][x+ 23] = ÉLŐ;
rács[y+ 9][x+ 24] = ÉLŐ;
rács[y+ 9][x+ 34] = ÉLŐ;
rács[y+ 9][x+ 35] = ÉLŐ;

rács[y+ 10][x+ 22] = ÉLŐ;
rács[y+ 10][x+ 23] = ÉLŐ;
rács[y+ 10][x+ 24] = ÉLŐ;
rács[y+ 10][x+ 25] = ÉLŐ;

rács[y+ 11][x+ 25] = ÉLŐ;

}

/** Pillanatfelvételek készítése. */
public void pillanatfelvétel(java.awt.image.BufferedImage felvetel) {
    // A pillanatfelvétel kép fájlneve
    StringBuffer sb = new StringBuffer();
    sb = sb.delete(0, sb.length());
    sb.append("sejtautomata");
    sb.append(++pillanatfelvételszám);
    sb.append(".png");
    // png formátumú képet mentünk
    try {
        javax.imageio.ImageIO.write(felvetel, "png",
            new java.io.File(sb.toString()));
    } catch(java.io.IOException e) {
        e.printStackTrace();
    }
}
```

```
}

// Ne villogjon a felület (mert a "gyári" update()
// lemeszelné a vászon felületét).
public void update(java.awt.Graphics g) {
    paint(g);
}
/***
 * Példányosít egy Conway-féle életjáték szabályos
 * sejttér obektumot.
 */
public static void main(String[] args) {

    new Sejtautomata(100, 75);
}
}
```

7.4. Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó:

Megoldás forrása:

Egy konkrét sejtautomatával talán a legismertebb ilyennel a John Horton Conway-féle életjátékkal foglalkozunk részletesen. Itt egy sejt egy sejttér eleme, a sejt állapota lehet élő vagy halott. A diszkrét időben működő sejttér adott sejtjének állapotát a következő időpillanatban a következő átmeneti szabályok alapján számolhatjuk ki. Élő sejt élő marad, ha kettő vagy három élő szomszédja van, különben halott lesz. Halott sejt halott marad, ha három élő szomszédja van, különben élő lesz. [...]

7.5. BrainB Benchmark

Megoldás videó:

Megoldás forrása:

```
#ifndef BrainBThread_H
#define BrainBThread_H

/***
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBThread.h
 * @author Norbert Bátfai <nbatfai@gmail.com>

```

```
* @version 6.0.1
*
* @section LICENSE
*
* Copyright (C) 2017, 2018 Norbert Bátfai, nbatfai@gmail.com
*
* 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 <http://www.gnu.org/licenses/>.
*
* @section DESCRIPTION
*
*/
#include <QThread>
#include <QSize>
#include <QImage>
#include <QDebug>
#include <sstream>
#include <QPainter>
#include <cstdlib>
#include <ctime>
#include <vector>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>

class Hero;
typedef std::vector<Hero> Heroes;

class Hero
{
public:
    int x;
    int y;
    int color;
    int agility;
    int conds {0};
    std::string name;
```

```
Hero ( int x=0, int y=0, int color=0, int agility=1, std::string name ←
      ="Samu Entropy" ) :
    x ( x ), y ( y ), color ( color ), agility ( agility ), name ( name ←
      )
{ }
~Hero () { }

void move ( int maxx, int maxy, int env ) {

    int newx = x+ ( ( double ) agility*1.0 ) * ( double ) ( std::rand ←
        () / ( RAND_MAX+1.0 ) )-agility/2 ;
    if ( newx-env > 0 && newx+env < maxx ) {
        x = newx;
    }
    int newy = y+ ( ( double ) agility*1.0 ) * ( double ) ( std::rand ←
        () / ( RAND_MAX+1.0 ) )-agility/2 ;
    if ( newy-env > 0 && newy+env < maxy ) {
        y = newy;
    }
}

};

class BrainBThread : public QThread
{
    Q_OBJECT

    //Norbi
    cv::Scalar cBg { 247, 223, 208 };
    cv::Scalar cBorderAndText { 47, 8, 4 };
    cv::Scalar cCenter { 170, 18, 1 };
    cv::Scalar cBoxes { 10, 235, 252 };

    /*
    //Matyi
    cv::Scalar cBg { 86, 26, 228 };
    cv::Scalar cBorderAndText { 14, 177, 232 };
    cv::Scalar cCenter { 232, 14, 103 };
    cv::Scalar cBoxes { 14, 232, 195 };
    */

    Heroes heroes;
    int heroRectSize {40};

    cv::Mat prev {3*heroRectSize, 3*heroRectSize, CV_8UC3, cBg };
    int bps;
    long time {0};
```

```
long endTime {10*60*10};
int delay {100};

bool paused {true};
int nofPaused {0};

std::vector<int> lostBPS;
std::vector<int> foundBPS;

int w;
int h;
int dispShift {40};

public:
    BrainBThread ( int w = 256, int h = 256 );
    ~BrainBThread();

    void run();
    void pause();
    void set_paused ( bool p );
    int getDelay() const {

        return delay;
    }

    void setDelay ( int delay ) {

        if ( delay > 0 ) {
            delay = delay;
        }
    }

    void devel() {

        for ( Hero & hero : heroes ) {

            hero.move ( w, h, ( h < w ) ? h/10:w/10 );
        }
    }

    int nofHeroes () {

        return heroes.size();
    }

    std::vector<int> &lostV () {
```

```
    return lostBPS;
}

std::vector<int> &foundV () {
    return foundBPS;
}

double meanLost () {
    return mean ( lostBPS );
}

double varLost ( double mean ) {
    return var ( lostBPS, mean );
}

double meanFound () {
    return mean ( foundBPS );
}

double varFound ( double mean ) {
    return var ( foundBPS, mean );
}

double mean ( std::vector<int> vect ) {
    double sum = std::accumulate ( vect.begin (), vect.end (), 0.0 );
    return sum / vect.size();
}

double var ( std::vector<int> vect, double mean ) {
    double accum = 0.0;
    std::for_each ( vect.begin (), vect.end (), [&] ( const double d ) ←
    {
        accum += ( d - mean ) * ( d - mean );
    } );
}
```

```
        return sqrt ( accum / ( vect.size()-1 ) );
    }

    int get_bps() const {
        return bps;
    }

    int get_w() const {
        return w;
    }

    bool get_paused() const {
        return paused;
    }

    int get_nofPaused() const {
        return nofPaused;
    }

    void decComp() {
        lostBPS.push_back ( bps );

        if ( heroes.size() > 1 ) {
            heroes.pop_back();
        }

        for ( Hero & hero : heroes ) {
            if ( hero.agility >= 5 ) {
                hero.agility -= 2;
            }
        }
    }

    void incComp() {
        foundBPS.push_back ( bps );
        if ( heroes.size() > 300 ) {
```

```
        return;

    }

/*
Hero other ( w/2 + 200.0*std::rand() / ( RAND_MAX+1.0 )-100,
             h/2 + 200.0*std::rand() / ( RAND_MAX+1.0 )-100,
             255.0*std::rand() / ( RAND_MAX+1.0 ), 11, "New Entropy ←
             " );
*/

double rx = 200.0;
if(heroes[0].x - 200 < 0)
    rx = heroes[0].x;
else if(heroes[0].x + 200 > w)
    rx = w - heroes[0].x;

double ry = 200.0;
if(heroes[0].y - 200 < 0)
    ry = heroes[0].y;
else if(heroes[0].y + 200 > h)
    ry = h - heroes[0].y;

Hero other ( heroes[0].x + rx*std::rand() / ( RAND_MAX+1.0 )-rx/2,
             heroes[0].y + ry*std::rand() / ( RAND_MAX+1.0 )-ry/2,
             255.0*std::rand() / ( RAND_MAX+1.0 ), 11, "New Entropy ←
             " );

heroes.push_back ( other );

for ( Hero & hero : heroes ) {

    ++hero.conds;
    if ( hero.conds == 3 ) {
        hero.conds = 0;
        hero.agility += 2;
    }
}

void draw () {

    cv::Mat src ( h+3*heroRectSize, w+3*heroRectSize, CV_8UC3, cBg );

    for ( Hero & hero : heroes ) {

        cv::Point x ( hero.x-heroRectSize+dispShift, hero.y- ←
```

```
    heroRectSize+dispShift );
cv::Point y ( hero.x+heroRectSize+dispShift, hero.y+ ←
    heroRectSize+dispShift );

cv::rectangle ( src, x, y, cBorderAndText );

cv::putText ( src, hero.name, x, cv::FONT_HERSHEY_SIMPLEX, .35, ←
    cBorderAndText, 1 );

cv::Point xc ( hero.x+dispShift , hero.y+dispShift );

cv::circle ( src, xc, 11, cCenter, CV_FILLED, 8, 0 );

cv::Mat box = src ( cv::Rect ( x, y ) );

cv::Mat cbox ( 2*heroRectSize, 2*heroRectSize, CV_8UC3, cBoxes ←
    );
box = cbox*.3 + box*.7;

}

cv::Mat comp;

cv::Point focusx ( heroes[0].x- ( 3*heroRectSize ) /2+dispShift, ←
    heroes[0].y- ( 3*heroRectSize ) /2+dispShift );
cv::Point focusy ( heroes[0].x+ ( 3*heroRectSize ) /2+dispShift, ←
    heroes[0].y+ ( 3*heroRectSize ) /2+dispShift );
cv::Mat focus = src ( cv::Rect ( focusx, focusy ) );

cv::compare ( prev, focus, comp, cv::CMP_NE );

cv::Mat aRgb;
cv::extractChannel ( comp, aRgb, 0 );

bps = cv::countNonZero ( aRgb ) * 10;

//qDebug() << bps << " bits/sec";

prev = focus;

QImage dest ( src.data, src.cols, src.rows, src.step, QImage::Format_RGB888 );
dest=dest.rgbSwapped();
dest.bits();

emit heroesChanged ( dest, heroes[0].x, heroes[0].y );

}

long getT() const {
```

```
        return time;

    }

void finish () {

    time = endTime;

}

signals:

void heroesChanged ( const QImage &image, const int &x, const int &y );
void endAndStats ( const int &t );

};

#endif // BrainBThread_H
```

```
#ifndef BrainBWin_H
#define BrainBWin_H

/***
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBWin.h
 * @author Norbert Bátfai <nbatfai@gmail.com>
 * @version 6.0.1
 *
 * @section LICENSE
 *
 * Copyright (C) 2017, 2018 Norbert Bátfai, nbatfai@gmail.com
 *
 * 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 <http://www.gnu.org/licenses/>.
```

```
*  
* @section DESCRIPTION  
*  
*/  
  
#include <QKeyEvent>  
#include <QMainWindow>  
#include <QPixmap>  
#include <QPainter>  
#include <QFont>  
#include <QFile>  
#include <QString>  
#include <QCLOSEEvent>  
#include <QDate>  
#include <QDir>  
#include <QDateTime>  
#include "BrainBThread.h"  
  
enum playerstate {  
    lost,  
    found  
};  
  
class BrainBWin : public QMainWindow  
{  
    Q_OBJECT  
  
    BrainBThread *brainBThread;  
    QPixmap pixmap;  
    Heroes *heroes;  
  
    int mouse_x;  
    int mouse_y;  
    int yshift {50};  
    int nofLost {0};  
    int nofFound {0};  
  
    int xs, ys;  
  
    bool firstLost {false};  
    bool start {false};  
    playerstate state = lost;  
    std::vector<int> lost2found;  
    std::vector<int> found2lost;  
  
    QString statDir;  
  
public:  
    static const QString appName;  
    static const QString appVersion;
```

```
BrainBWin ( int w = 256, int h = 256, QWidget *parent = 0 );

void closeEvent ( QCloseEvent *e ) {

    if ( save ( brainBThread->getT() ) ) {
        brainBThread->finish();
        e->accept();
    } else {
        e->ignore();
    }

}

virtual ~BrainBWin();
void paintEvent ( QPaintEvent * );
void keyPressEvent ( QKeyEvent *event );
void mouseMoveEvent ( QMouseEvent *event );
void mousePressEvent ( QMouseEvent *event );
void mouseReleaseEvent ( QMouseEvent *event );

double mean ( std::vector<int> vect ) {

    if ( vect.size() > 0 ) {
        double sum = std::accumulate ( vect.begin (), vect.end (), 0.0 );
        return sum / vect.size();
    } else {
        return 0.0;
    }
}

double var ( std::vector<int> vect, double mean ) {

    if ( vect.size() > 1 ) {

        double accum = 0.0;

        std::for_each ( vect.begin (), vect.end (), [&] ( const double & d ) {
            accum += ( d - mean ) * ( d - mean );
        } );

        return sqrt ( accum / ( vect.size()-1 ) );
    } else {
        return 0.0;
    }
}

void millis2minsec ( int millis, int &min, int &sec ) {
```

```
sec = ( millis * 100 ) / 1000;
min = sec / 60;
sec = sec - min * 60;

}

bool save ( int t ) {

    bool ret = false;

    if ( !QDir ( statDir ).exists() )
        if ( !QDir().mkdir ( statDir ) ) {
            return false;
        }

    QString name = statDir + "/Test-" + QString::number ( t );
    QFile file ( name + "-screenimage.png" );
    if ( file.open ( QIODevice::WriteOnly ) ) {
        ret = pixmap.save ( &file, "PNG" );
    }

    QFile tfile ( name + "-stats.txt" );
    ret = tfile.open ( QIODevice::WriteOnly | QIODevice::Text );
    if ( ret ) {
        QTextStream textStremam ( &tfile );

        textStremam << appName + " " + appVersion << "\n";
        textStremam << "time" : " " << brainBThread->getT() << "\n";
        textStremam << "bps" : " " << brainBThread->get_bps() << "\n";
        textStremam << "noc" : " " << brainBThread->nofHeroes() << "\n";
        textStremam << "nop" : " " << brainBThread->get_nofPaused() << "\n";

        textStremam << "lost" : " " << "\n";
        std::vector<int> l = brainBThread->lostV();
        for ( int n : l ) {
            textStremam << n << ' ';
        }
        textStremam << "\n";
        int m = mean ( l );
        textStremam << "mean" : " " << m << "\n";
        textStremam << "var" : " " << var ( l, m ) << "\n";

        textStremam << "found" : " ";
        std::vector<int> f = brainBThread->foundV();
        for ( int n : f ) {
            textStremam << n << ' ';
        }
    }
}
```

```
        }

    textStremam << "\n";
    m = mean ( f );
    textStremam << "mean      : " << m << "\n";
    textStremam << "var       : " << var ( f, m ) << "\n";

    textStremam << "lost2found: " ;
    for ( int n : lost2found ) {
        textStremam << n << ' ';
    }
    textStremam << "\n";
    int m1 = m = mean ( lost2found );
    textStremam << "mean      : " << m << "\n";
    textStremam << "var       : " << var ( lost2found, m ) << "\n" <-
        ;

    textStremam << "found2lost: " ;
    for ( int n : found2lost ) {
        textStremam << n << ' ';
    }
    textStremam << "\n";
    int m2 = m = mean ( found2lost );
    textStremam << "mean      : " << m << "\n";
    textStremam << "var       : " << var ( found2lost, m ) << "\n" <-
        ;

    if ( m1 < m2 ) {
        textStremam << "mean(lost2found) < mean(found2lost)" << "\n" <-
            ;
    }

    int min, sec;
    millis2minsec ( t, min, sec );
    textStremam << "time      : " << min << ":" << sec << "\n";

    double res = ( ( ( double ) m1+ ( double ) m2 ) /2.0 ) /8.0 ) <-
        /1024.0;
    textStremam << "U R about " << res << " Kilobytes\n";

    tfile.close();
}
return ret;
}

public slots :

void updateHeroes ( const QImage &image, const int &x, const int &y );
//void stats ( const int &t );
void endAndStats ( const int &t );
};
```

```
#endif // BrainBWin

/*
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBThread.cpp
 * @author Norbert Bátfai <nbatfai@gmail.com>
 * @version 6.0.1
 *
 * @section LICENSE
 *
 * Copyright (C) 2017, 2018 Norbert Bátfai, nbatfai@gmail.com
 *
 * 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 <http://www.gnu.org/licenses/>.
 *
 * @section DESCRIPTION
 *
 */

#include "BrainBThread.h"

BrainBThread::BrainBThread ( int w, int h )
{
    dispShift = heroRectSize+heroRectSize/2;

    this->w = w - 3 * heroRectSize;
    this->h = h - 3 * heroRectSize;

    std::srand ( std::time ( 0 ) );

    Hero me ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ) - ←
              100,
              this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ) - ←
              100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), 9 );
```

```
Hero other1 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
    ) - 100,
              this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
                  ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ←
                  5, "Norbi Entropy" );
Hero other2 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
    ) - 100,
              this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
                  ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ←
                  3, "Greta Entropy" );
Hero other4 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
    ) - 100,
              this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
                  ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ←
                  5, "Nandi Entropy" );
Hero other5 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
    ) - 100,
              this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ←
                  ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ←
                  7, "Matyi Entropy" );

heroes.push_back ( me );
heroes.push_back ( other1 );
heroes.push_back ( other2 );
heroes.push_back ( other4 );
heroes.push_back ( other5 );

}

BrainBThread::~BrainBThread()
{
}

void BrainBThread::run()
{
    while ( time < endTime ) {

        QThread::msleep ( delay );

        if ( !paused ) {

            ++time;

            devel();

        }

        draw();
    }
}
```

```
        }

        emit endAndStats ( endTime );

    }

void BrainBThread::pause()
{

    paused = !paused;
    if ( paused ) {
        ++nofPaused;
    }

}

void BrainBThread::set_paused ( bool p )
{

    if ( !paused && p ) {
        ++nofPaused;
    }

    paused = p;

}
```

```
/***
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBWin.cpp
 * @author Norbert Bátfai <nbatfai@gmail.com>
 * @version 6.0.1
 *
 * @section LICENSE
 *
 * Copyright (C) 2017, 2018 Norbert Bátfai, nbatfai@gmail.com
 *
 * 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 <http://www.gnu.org/licenses/>.
*
* @section DESCRIPTION
*
*/
```

```
#include "BrainBWin.h"

const QString BrainBWin::appName = "NEMESPOR BrainB Test";
const QString BrainBWin::appVersion = "6.0.3";

BrainBWin::BrainBWin ( int w, int h, QWidget *parent ) : QMainWindow ( ←
    parent )
{
    //    setWindowTitle(appName + " " + appVersion);
    //    setFixedSize(QSize(w, h));

    statDir = appName + " " + appVersion + " - " + QDate::currentDate() ←
        .toString() + QString::number ( QDateTime::←
        currentMSecsSinceEpoch() ) ;

    brainBThread = new BrainBThread ( w, h - yshift );
    brainBThread->start();

    connect ( brainBThread, SIGNAL ( heroesChanged ( QImage, int, int ) ←
        ),
              this, SLOT ( updateHeroes ( QImage, int, int ) ) );

    connect ( brainBThread, SIGNAL ( endAndStats ( int ) ),
              this, SLOT ( endAndStats ( int ) ) );
}

void BrainBWin::endAndStats ( const int &t )
{
    qDebug() << "\n\n\n";
    qDebug() << "Thank you for using " + appName;
    qDebug() << "The result can be found in the directory " + statDir;
    qDebug() << "\n\n\n";

    save ( t );
    close();
}
```

```
void BrainBWin::updateHeroes ( const QImage &image, const int &x, const int &y )
{
    if ( start && !brainBThread->get_paused() ) {

        int dist = ( this->mouse_x - x ) * ( this->mouse_x - x ) + ( this->mouse_y - y ) * ( this->mouse_y - y );

        if ( dist > 121 ) {
            ++nofLost;
            nofFound = 0;
            if ( nofLost > 12 ) {

                if ( state == found && firstLost ) {
                    found2lost.push_back ( brainBThread->get_bps() );
                }

                firstLost = true;

                state = lost;
                nofLost = 0;
                //qDebug() << "LOST";
                //double mean = brainBThread->meanLost();
                //qDebug() << mean;

                brainBThread->decComp();
            }
        } else {
            ++nofFound;
            nofLost = 0;
            if ( nofFound > 12 ) {

                if ( state == lost && firstLost ) {
                    lost2found.push_back ( brainBThread->get_bps() );
                }

                state = found;
                nofFound = 0;
                //qDebug() << "FOUND";
                //double mean = brainBThread->meanFound();
                //qDebug() << mean;

                brainBThread->incComp();
            }
        }
    }
}
```

```
    }

    pixmap = QPixmap::fromImage ( image );
    update();
}

void BrainBWin::paintEvent ( QPaintEvent * )
{
    if ( pixmap.isNull() ) {
        return;
    }

    QPainter qpainter ( this );

    xs = ( qpainter.device()->width() - pixmap.width() ) /2;
    ys = ( qpainter.device()->height() - pixmap.height() +yshift ) /2;

    qpainter.drawPixmap ( xs, ys, pixmap );

    qpainter.drawText ( 10, 20, "Press and hold the mouse button on the ←
                        center of Samu Entropy" );

    int time = brainBThread->getT();
    int min, sec;
    millis2minsec ( time, min, sec );
    QString timestr = QString::number ( min ) + ":" + QString::number ( ←
        sec ) + "/10:0";
    qpainter.drawText ( 10, 40, timestr );

    int bps = brainBThread->get_bps();
    QString bpsstr = QString::number ( bps ) + " bps";
    qpainter.drawText ( 110, 40, bpsstr );

    if ( brainBThread->get_paused() ) {
        QString pausedstr = "PAUSED (" + QString::number ( ←
            brainBThread->get_nofPaused() ) + ")";
        qpainter.drawText ( 210, 40, pausedstr );
    }

    qpainter.end();
}

void BrainBWin::mousePressEvent ( QMouseEvent *event )
{
    brainBThread->set_paused ( false );
}

void BrainBWin::mouseReleaseEvent ( QMouseEvent *event )
```

```
{  
  
    //brainBThread->set_paused(true);  
  
}  
  
void BrainBWin::mouseMoveEvent ( QMouseEvent *event )  
{  
  
    start = true;  
  
    mouse_x = event->pos().x() -xs - 60;  
    //mouse_y = event->pos().y() - yshift - 60;  
    mouse_y = event->pos().y() - ys - 60;  
}  
  
void BrainBWin::keyPressEvent ( QKeyEvent *event )  
{  
  
    if ( event->key() == Qt::Key_S ) {  
        save ( brainBThread->getT() );  
    } else if ( event->key() == Qt::Key_P ) {  
        brainBThread->pause();  
    } else if ( event->key() == Qt::Key_Q || event->key() == Qt::Key_Escape ) {  
        close();  
    }  
}  
  
BrainBWin::~BrainBWin()  
{  
  
}
```

```
/**  
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games  
 *  
 * @file main.cpp  
 * @author Norbert Bátfai <nbatfai@gmail.com>  
 * @version 6.0.1  
 *  
 * @section LICENSE  
 *  
 * Copyright (C) 2017, 2018 Norbert Bátfai, nbatfai@gmail.com  
 *  
 * 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 <http://www.gnu.org/licenses/>.
*
* @section DESCRIPTION
*
*/
```

```
#include <QApplication>
#include <QTextStream>
#include <QtWidgets>
#include "BrainBWin.h"

int main ( int argc, char **argv )
{
    QApplication app ( argc, argv );

    QTextStream qout ( stdout );
    qout.setCodec ( "UTF-8" );

    qout << "\n" << BrainBWin::appName << QString::fromUtf8 ( " ↵
Copyright (C) 2017, 2018 Norbert Bátfai" ) << endl;

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

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

    qout << QString::fromUtf8 ( "Ez a program szabad szoftver; ↵
        terjeszthető illetve módosítható a Free Software" ) << endl;
    qout << QString::fromUtf8 ( "Foundation által kiadott GNU General ↵
        Public License dokumentumában leírtak;" ) << endl;
    qout << QString::fromUtf8 ( "akár a licenc 3-as, akár (tetszőleges) ↵
```

```
későbbi változata szerint.\n" ) << endl;

qout << QString::fromUtf8 ( "Ez a program abban a reményben kerül ←
    közreadásra, hogy hasznos lesz, de minden" ) << endl;
qout << QString::fromUtf8 ( "egyéb GARANCIA NÉLKÜL, az ←
    ELADHATÓSÁGRA vagy VALAMELY CÉLRA VALÓ" ) << endl;
qout << QString::fromUtf8 ( "ALKALMAZHATÓSÁGRA való származtatott ←
    garanciát is beleértve. További" ) << endl;
qout << QString::fromUtf8 ( "részleteket a GNU General Public ←
    License tartalmaz.\n" ) << endl;

qout << "http://gnu.hu/gplv3.html" << endl;

QRect rect = QApplication::desktop()->availableGeometry();
BrainBWin brainBWin ( rect.width(), rect.height() );
brainBWin.setWindowState ( brainBWin.windowState() ^ Qt::←
    WindowFullScreen );
brainBWin.show();
return app.exec();
}
```

```
#
# @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
#
# @file BrainB.pro
# @author Norbert Bátfai <nbatfai@gmail.com>
# @version 0.0.1
#
# @section LICENSE
#
# Copyright (C) 2017 Norbert Bátfai, nbatfai@gmail.com
#
# 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 <http://www.gnu.org/licenses/>.
#
# @section DESCRIPTION
#
#
```

```
QT += widgets core
CONFIG += c++11 c++14 c++17
QMAKE_CXXFLAGS += -fopenmp
LIBS += -fopenmp
LIBS += `pkg-config --libs opencv`

TEMPLATE = app
TARGET = BrainB
INCLUDEPATH += .

HEADERS += BrainBThread.h BrainBWin.h
SOURCES += BrainBThread.cpp BrainBWin.cpp main.cpp
```

Tanulságok, tapasztalatok, magyarázat...

7.6. Vörös Pipacs Pokol/19 RF

Megoldás videó: <https://youtu.be/VP0kfvRYD1Y>

Megoldás forrása: <https://github.com/nbatfai/RedFlowerHell>

Tanulságok, tapasztalatok, magyarázat... ezt kell az olvasónak kidolgozna, mint labor- vagy otthoni mérési feladatot! Ha mi már megtettük, akkor használd azt, dolgozd fel, javítsd, adj hozzá értéket!

8. fejezet

Helló, Schwarzenegger!

8.1. Szoftmax Py MNIST

Python

Megoldás videó: <https://youtu.be/j7f9SkJR3oc>

Megoldás forrása: <https://github.com/tensorflow/tensorflow/releases/tag/v0.9.0> (/tensorflow-0.9.0/tensorflow/exa...
https://progpater.blog.hu/2016/11/13/hello_samu_a_tensorflow-bol

A program elején importáljuk a tensorflow libralyt amire ezetntől tf ként fogunk hivatkozni. Ezutan lekreáljuk az mnist objectumot, majd betöljtük és igazítjuk a Train és teszt adatbázisokat. Jöhet a mondel készítése,ez a gépi tanulásnak/deep learningnek a legfontosabb része. Ebben segít a tensorflowba beipitett keras modellek. Ezután ami feltűnik az az a folyamat melynem során ún. prediction-t adunk meg, ez gyakorlatilag a neurális háló kimenetét adja megne persze ezt fotmáznunk kell, áttranszformáljuk osztályokká. Itt kerül konkrétan hasznákatba a softmax. Láthatjuk a loss függvényt ami konkrétan a program tanításában játszik szerepet. Ezután a modellünket összeállítjuk a compile al , majd a fit 1 kezdődhet is a tanítás. Ide átadjuk a train adatbázist, és hogy hányszor fussen a program . A model-evaluate el adunk neki egy kiértékelést

```
import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
])

predictions = model(x_train[:1]).numpy()
predictions
```

```
tf.nn.softmax(predictions).numpy()

loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

loss_fn(y_train[:1], predictions).numpy()

model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test, verbose=2)

probability_model = tf.keras.Sequential([
    model,
    tf.keras.layers.Softmax()
])

probability_model(x_test[:5])
```

8.2. Mély MNIST

Python

Megoldás videó:

Megoldás forrása:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten, ↵
    MaxPooling2D

from PIL import Image
import numpy as np
import sys

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
```

```
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.2))
model.add(Dense(10, activation=tf.nn.softmax))

tb_log_dir = "./cnn_tb"
file_writer = tf.summary.create_file_writer(tb_log_dir)
file_writer.set_as_default()
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=tb_log_dir, ←
    profile_batch=0)

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x=x_train, y=y_train, epochs=10, callbacks=[tensorboard_callback])

model.evaluate(x_test, y_test)

input_image = np.array(Image.open(sys.argv[1]).getdata(0).resize((28, 28), ←
    0))

pred = model.predict(input_image.reshape(1, 28, 28, 1))

print (pred)

print("The number is = ", pred.argmax())
</para>
    <para>
        Tanulságok, tapasztalatok, magyarázat...
    </para>
</section>
<!--
<section>
    <title>Deep dream</title>
    <para>
        Keras
    </para>
    <para>
        Megoldás videó:
    </para>
```

```
</para>
<para>
    Megoldás forrása:
</para>
<para>
    Tanulságok, tapasztalatok, magyarázat...
</para>
</section>
-->
<section>
    <title>Minecraft-MALMÖ</title>
    <para>
    </para>
    <para>
        Megoldás videó: <link xlink:href="https://youtu.be/bAPSu3Rndi8" ↵
                    >https://youtu.be/bAPSu3Rndi8</link>
    </para>
    <para>
        Megoldás forrása:
    </para>
    <para>
        Tanulságok, tapasztalatok, magyarázat...
    </para>
```

DRY

9. fejezet

Helló, Chaitin!

9.1. Iteratív és rekurzív faktoriális Lisp-ben

Megoldás videó: <https://youtu.be/z6NJE2a1zIA>

Megoldás forrása:

A programban egy különös programozási nyelvel ismerkedtem meg, ahol valamemnnyire "felrúgja" a normális programozási nyelvnek a törvényeit és újakat vezet be. Az elején különösnek tűnik mikor elkezdi az ember, de gyorsan bele lehet tanulni és goldolkodásmódjának alapjai gyorsan elsajátítható.

9.2. 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

Megoldás forrása: https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/GIMP_Lisp/Chrome

Tanulságok, tapasztalatok, magyarázat...

9.3. 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

Megoldás forrása: https://gitlab.com/nbatfai/bhax/tree/master/attention_raising/GIMP_Lisp/Mandala

Tanulságok, tapasztalatok, magyarázat...

10. fejezet

Helló, Gutenberg!

10.1. Programozási alapfogalmak

[?]

Bevezető

A számítógépek programozására kialakult nyelveknek három szintjét különböztetjük meg: gépi nyelv, assembly szintű nyelv, magas szintű nyelv. A magas szintű nyelven megírt programot forrásprogramnak, vagy forrásszövegnek nevezzük. Szintaktikai szabályok: hogy írjuk meg a programot úgy, hogy az lefusson és azt csinálja amit kell vagyis a forrásszöveg összeállítására vonatkozó nyelvtani szabályok szemantikai szabályok: A tartalmi vagy jelentésbeli szabályok. Egy magas szintű programozási nyelvet szintaktikai és szemantikai szabályainak együttese határoz meg. minden processzor rendelkezik egy gépi nyelvvel, csak azon tud kommunikálni, csak az azon nyelven megírt programokat képes futtatni. Ez azt eredményezi hogy az emberi nyelven megírt kódolatot kell generálni, például fordító programmal ami a magas szintű nyelven megírt forráskódóból gépi kódot generál a következő képpen: lexikális elemzés (lexikális egységekre darabolja a kódot) szintaktikai (ellenírja, hogy teljesülnek-e az adott nyelv szintaktikai szabályai) majd szemantikai elemzés majd kódgenerálás. A futtatható programot a szerkesztő vagy kapcsolatszerkesztő állít elő, ezután a futtatható programot a betöltő helyezi el a tárban és adja át neki a vezérlést. minden programnyelvnek megvan a saját szabványa, amit hivatkozási nyelvnek hívunk. Ebben pontosan definiálva vannak a szintaktikai és a szemantikai szabályok. A hivatkozási nyelvek mellett léteznek az implementációk (adott platformon realizált fordítók vagy interpreterek). Napjainkban a programok írásához grafikus integrált fejlesztői környezetek állnak rendelkezésünkre. Ezek tartalmaznak szövegszerkesztőt, fordítót, kapcsolatszerkesztőt, betöltőt, futtató rendszert és belövit. Programok osztályozása: Imperativ nyelvek (Szorosan kötődnek a Neumann-architektúrához), deklaratív nyelvek, máselvő (egyéb) nyelvek. Az Imperativ nyelvre jellemző, hogy algoritmikusak azaz a feladat utasításokból áll, változókat használ, folyamatos lefutású, véges. A deklaratív nyelvek nem algoritmikusak, nincs lehetőség memóriuműveletekre, emelett a programozó csak a problémát adja meg, az implementáció keresi a megoldást. Fontos megjegyzendő dolog, hogy valaki csak papiron/elméletben nem fog megtanulni programozni, rengeteg gyakorlás szükséges. A forrásszöveg legkisebb alkotórészei a karakterek. Alapvető a karakterkészlet, ezekből állíthatók össze a bonyolult nyelvi elemek. Az eljárásorientált nyelvek esetén ezek a következik: lexikális egységek, szintaktikai egységek, utasítások, programegységek, fordítási egységek, program. minden nyelv definiálja a saját karakterkészletét, például: betűk, számok, egyéb karakterek. minden programnyelvben betű az angol ABC 26 nagybetűje. A nyelvek továbbá betű kategóriájú karakterek tekintik gyakran az _, \$, #, @ karaktereket.

is. Ez viszont sokszor implementációfüggő. Egyéb karakterek közé tartoznak a +, -, *, / műveletjelek, vagy a [,], .. ;, { , }, ', ", ; elhatárolójelek, vagy például c ben a többkarakteres szimbólumok :++, --/*, */.

Benedek Zoltán, Levendovszki Tihámér: Szoftverfejlesztés C++ nyelven

Kezdetekben megismerhetjük a c++ nyelv rövid történetét, és kifejezésre kerül az is hogy a C++ a C re épül. A C++ viszont támogatja az objektumorientált és generikus programozást. ezután nem sokkal összehasonlításra kerül a C nyelvel, például a C-ben nem szükséges paramétert megadnunk függvény létrehozásakor, ilyenkor Ez Int tipusu lesz. C++ ban ha paramétereket szeretnénk a main függvénybe, azt argc és argv[] vel tehetjük meg, illetve itt nem kötelező a return 0; parancs használata, nem mint C ben. A bool típus bevezetése is meg van említve, ez logikai (igaz-hamis) értékre utal, viszont C ben még nem volt. változókat használat előtt deklarálni kell, változtatásokat utasításban végezhetünk.

10.2. Programozás bevezetés

[KERNIGHANRITCHIE]

Megoldás videó: <https://youtu.be/zmfT9miB-jY>

Ahhoz hogy elsajátitsunk egy új programozási nyelvet csak egy út létezik.... programokat kell írnunk az adott nyelven, például a már sokak által jól ismert Hello World programot. Képesnek kell lennünk egy program létrehozására, annak sikeres lefordítására, betöltésére, futtatására, és ki kell találnunk, hogy a kiírt szöveg hol jelenik meg, ha ezek megvannak az alap lépésekkel készzen is vagyunk. A program futtatásának módja az általunk használt rendszertől függ. Például PL UNIX alatt a cc paracsal, természetesen az adott állománynak melyben a program szerepel .c re kell végződni. Egy C program MINDIG c függvényekből és változókból áll, azaz azokat amelyeket megenged az adott nyelv pl jelen esetben a printf() függvény. A #include stdio.h sor éppen azt mondja meg a fordítóprogramnak, hogy a fordítás során a programba foglalja bele a standard bemeneti/kimeneti könyvtárra vonatkozó információkat. A függvények közti adatcsere egyik módszere, hogy a hívó függvény adatokból álló listát, az ún. argumentumokat adja át a hívott függvénynek. A függvény neve utáni () zárójelek ezt az argumentumlistát határolják. Igy hát a printf("Halló mindenki!\n"); utasítás a "" jelek közötti szöveget a standard outputra irányítja. Az első programjainkat mindenkoréppen from scratch, azaz a nulláról írjuk, így garantáltan elsajátítjuk a nyelv alapjait. A fejezet további részében az író bemutat egy kicsivel komplexebb C programot ami a Celsius és Fahrenheit hőmérsékleteket táblázatos alakban iratja ki. Továbbá megismerkedhetünk az int, float tipusokkal, emelett a while ciklus is részletes bemutatásra kerül. Ezután a for ciklus kerül részletesebb bemutatásra majd összehasonlításra a while ciklussal. Később megismerkedhetünk a szimbolikus állandókkal is mint például a #define. Bemutatja a char adattípust ami karakterekre utal, megismerjük a getchar() függvényt. Későbbieknben az iro bemutatja a tömbököt A C nyelvben a tömbök indexe mindenkorán nullától indul, tehát ha n elemű a tömb, akkor n-1 lesz a legnagyobb index. Láthatjuk az IF utasítás ELSE ágát is ami akkor lép érvénybe ha az if ágban vett utasítás nem teljesül, ha nincs else águnk a program szimplán tovább lép. Bármikor létrahozhatunk egy nekünk megfelelő függvényt amit akkor használhatunk amikor szükség van rá, ezzel a programunk átláthatóbb lesz, egy számítást elég egyszer elvégezni. Itt szemléltetik pontosan egy függvény argumentumainak/paramétereinek szerepét és fontosságát, illetve egy függvény return utasítás által visszaadott visszatérési értékét. Ha egy függvényen belül dekláralunk valamit az csak abban a függvényben lesz érvényes, azaz azon kívül nem használható. Deklarálhatunk azonban globális változókat, melyeket a program bármely részén lehet használni, ezek deklárációkor 0 értéket kapnak, ha csak a programozó nem ad más értéket.

10.3. Programozás

[BMECPP]

Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba

A python programozási nyelv alkalmas összetett alkalmazások elkészítésére ,ugyanis sokkal kompaktabb és egyszerűbb mint a C, C++ vagy JAVA Nincs szükség fordításra,linkelésre ezért is gyorsít a tipikus programirás/fordítás/tesztelés/újrafordítás cikluson. Nem kell külön változódeklarálás sem. A kódcsoporthoz kicsit furcs lehet ugyanis itt nem találhatóak meg a szokásos {} vagy begin end csoportosítók, és még ; sem kell az utasítások végére ,személy szerint kicsit furcsának tartottam eleim, hogy azzal mondjam meg hogy egy while vagy for cílusok mely utasítások tartoznak,hogy TAB ot rakok elé,ugyanakkor így 100% hogy átláthatóbb a kód. A nyelvben a változóknak nincsenek típusai, akár több típusú objektumra is hivatkozhatnak. Igy azt sem kell definiálnunk hogy egy változó lebegőpontos értéket tárol e hanem egyszerűen példával élve a = 3.0. Itt is definíálhatók globális illetve lokális változók, alapból lokális változónak vannak deklarálva, viszont például: global a globális változónak lesz definiálva. Érdekesebb dolgok közé tartozik az elif is, ami a következő módon működik : egy if ()else if()else kiompaktabban van leírva IF: ELIF: else, így rovidithetünk a kódon, és átláthatóbbá tehetjük. A függvényeket sem kell túlkomplikálni, elég a def függvény():..

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

11. fejezet

Helló, Arroway!

11.1. A BPP algoritmus Java megvalósítása

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

11.2. 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

11.3. Általános

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

11.4. C

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

11.5. C++

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

11.6. Lisp

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

Köszönet illeti a NEMESPOR, <https://groups.google.com/forum/#!forum/nemespor>, az UDPORG 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 érdeklődésekért és hasznos észrevételeikért.

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