

**Írd meg a saját programozás tankönyvedet!**

Ed. BHAX, DEBRECEN,  
2019. május 9, v. 1.0.0

Copyright © 2019 Dr. Bátfai Norbert

Copyright © 2019 Sántha Máté Imre

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

Copyright (C) 2019, Sántha Máté Imre., santha.mate22@gmail.com, mateme@gmail.hu,

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>    |                   |                  |
|---------------|-------------------|-------------------|------------------|
| <i>ACTION</i> | <i>NAME</i>       | <i>DATE</i>       | <i>SIGNATURE</i> |
| WRITTEN BY    | Sántha, Máté Imre | 2019. december 1. |                  |

**REVISION HISTORY**

| NUMBER | DATE       | DESCRIPTION   | NAME             |
|--------|------------|---|------------------|
| 0.0.1  | 2019-02-12 | Az iniciális dokumentum szerkezetének kialakítása.  | nbatfai          |
| 0.0.2  | 2019-02-14 | Inciális feladatlisták összeállítása.   | nbatfai          |
| 0.0.3  | 2019-02-16 | Feladatlisták folytatása. Feltöltés a BHAX csatorna <a href="https://gitlab.com/nbatfai/bhax">https://gitlab.com/nbatfai/bhax</a> repójába. | nbatfai          |
| 0.0.4  | 2019-02-19 | Aktualizálás, javítások.  | nbatfai          |
| 0.0.5  | 2019-03-03 | első fejezet kész   | Sántha Máté Imre |
| 0.0.6  | 2019-03-11 | Második fejezet kész  | Sántha Máté Imre |
| 0.0.7  | 2019-03-18 | Harmadik fejezet kész   | Sántha Máté Imre |
| 0.0.8  | 2019-03-26 | Negyedik fejezet kész .   | Sántha Máté Imre |

**REVISION HISTORY**

| NUMBER | DATE       | DESCRIPTION             | NAME             |
|--------|------------|-------------------------|------------------|
| 0.0.9  | 2019-03-29 | olvasónapló kész        | Sántha Máté Imre |
| 0.1.0  | 2019-04-02 | Ötödik fejezet kész     | Sántha Máté Imre |
| 0.1.1  | 2019-04-10 | Hatodik fejezet kész    | Sántha Máté Imre |
| 0.1.2  | 2019-04-17 | Hetedik fejezet kész    | Sántha Máté Imre |
| 0.1.3  | 2019-04-24 | Nyolcadik fejezet kész  | Sántha Máté Imre |
| 0.1.4  | 2019-04-30 | Kilencedik Fejezet Kész | Sántha Máté Imre |
| 0.1.5  | 2019-05-05 | Hibák javítása.         | nbatfai          |
| 1.0.0  | 2019-05-08 | Első teljes kiadás      | nbatfai          |

# Ajánlás

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

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

DRAFT

# Tartalomjegyzék

|  |          |
|--|----------|
| <b>I. Bevezetés</b>  | <b>1</b> |
| 1. Vízió   | 2        |
| 1.1. Mi a programozás? . . . . .                             | 2        |
| 1.2. Milyen doksikat olvassak el? . . . . .                  | 2        |
| 1.3. Milyen filmeket nézzek meg? . . . . .                   | 2        |
| <b>II. Tematikus feladatok</b>                               | <b>3</b> |
| 2. Helló, Turing!  | 5        |
| 2.1. Végtelen ciklus . . . . .                               | 5        |
| 2.2. Lefagyott, nem fagyott, akkor most mi van? . . . . .    | 6        |
| 2.3. Változók értékének felcserélése . . . . .               | 8        |
| 2.4. Labdapattogás . . . . .                                 | 8        |
| 2.5. Szóhossz és a Linus Torvalds féle BogoMIPS . . . . .    | 10       |
| 2.6. Helló, Google! . . . . .                                | 10       |
| 2.7. 100 éves a Brun téTEL . . . . .                         | 13       |
| 2.8. A Monty Hall probléma . . . . .                         | 14       |
| 3. Helló, Chomsky!   | 17       |
| 3.1. Decimálisból unárisba átváltó Turing gép . . . . .      | 17       |
| 3.2. Az $a^n b^n c^n$ nyelv nem környezetfüggetlen . . . . . | 18       |
| 3.3. Hivatalos nyelv . . . . .                               | 19       |
| 3.4. Saját lexikális elemző . . . . .                        | 20       |
| 3.5. l33t.l . . . . .  | 21       |
| 3.6. A források olvasása . . . . .                           | 23       |
| 3.7. Logikus . . . . .                                       | 24       |
| 3.8. Deklaráció . . . . .                                    | 25       |

|  |           |
|--|-----------|
| <b>4. Helló, Caesar!</b>                                     | <b>27</b> |
| 4.1. int *** háromszögmátrix . . . . .                       | 27        |
| 4.2. C EXOR titkosító . . . . .                              | 29        |
| 4.3. Java EXOR titkosító . . . . .                           | 30        |
| 4.4. C EXOR törő . . . . .                                   | 32        |
| 4.5. Neurális OR, AND és EXOR kapu . . . . .                 | 34        |
| 4.6. Hiba-visszaterjesztéses perceptron . . . . .            | 36        |
| <b>5. Helló, Mandelbrot!</b>                                 | <b>38</b> |
| 5.1. A Mandelbrot halmaz . . . . .                           | 38        |
| 5.2. A Mandelbrot halmaz a std::complex osztállyal . . . . . | 42        |
| 5.3. Biomorfok . . . . .                                     | 45        |
| 5.4. A Mandelbrot halmaz CUDA megvalósítása . . . . .        | 48        |
| 5.5. Mandelbrot nagyító és utazó C++ nyelven . . . . .       | 54        |
| 5.6. Mandelbrot nagyító és utazó Java nyelven . . . . .      | 54        |
| <b>6. Helló, Welch!</b>                                      | <b>59</b> |
| 6.1. Első osztályom . . . . .                                | 59        |
| 6.2. LZW . . . . .   | 60        |
| 6.3. Fabejárás . . . . .                                     | 61        |
| 6.4. Tag a gyökér . . . . .                                  | 62        |
| 6.5. Mutató a gyökér . . . . .                               | 62        |
| 6.6. Mozgató szemantika . . . . .                            | 63        |
| <b>7. Helló, Conway!</b>                                     | <b>64</b> |
| 7.1. Hangyszimulációk . . . . .                              | 64        |
| 7.2. Java életjáték . . . . .                                | 64        |
| 7.3. Qt C++ életjáték . . . . .                              | 65        |
| 7.4. BrainB Benchmark . . . . .                              | 66        |
| <b>8. Helló, Schwarzenegger!</b>                             | <b>68</b> |
| 8.1. Szoftmax Py MNIST . . . . .                             | 68        |
| 8.2. Szoftmax R MNIST (Passz) . . . . .                      | 69        |
| 8.3. Minecraft-MALMÖT(Passz) . . . . .                       | 69        |

|   |            |
|---|------------|
| <b>9. Helló, Chaitin!</b>   | <b>70</b>  |
| 9.1. Iteratív és rekurzív faktoriális Lisp-ben(Passz) . . . . .       | 70         |
| 9.2. Gimp Scheme Script-fu: króm effekt . . . . .                     | 70         |
| 9.3. Gimp Scheme Script-fu: név mandala . . . . .                     | 74         |
| <b>10. Helló, Gutenberg!</b>  | <b>79</b>  |
| 10.1. Programozási alapfogalmak . . . . .                             | 79         |
| 10.2. Programozás bevezetés . . . . .                                 | 80         |
| 10.3. Programozás . . . . .   | 81         |
| <b>III. Második felvonás</b>  | <b>82</b>  |
| <b>11. Helló, Arroway!</b>  | <b>84</b>  |
| 11.1. OO szemlélet . . . . .  | 84         |
| 11.2. Homokozó . . . . .  | 87         |
| 11.3. „Gagyí” . . . . .   | 87         |
| 11.4. Yoda . . . . .  | 88         |
| 11.5. Kódolás from scratch . . . . .                                  | 89         |
| <b>12. Helló, Liskov!</b>   | <b>91</b>  |
| 12.1. Liskov helyettesítés sértése . . . . .                          | 91         |
| 12.2. Szülő-gyerek . . . . .  | 92         |
| 12.3. Anti OO . . . . .   | 93         |
| 12.4. Hello, Android! . . . . .                                       | 96         |
| <b>13. Helló, Mandelbrot!</b>   | <b>99</b>  |
| 13.1. Reverse engineering UML osztálydiagram . . . . .                | 99         |
| 13.2. Forward engineering UML osztálydiagram . . . . .                | 100        |
| 13.3. BPMN . . . . .  | 101        |
| <b>14. Helló, Chomsky!</b>  | <b>103</b> |
| 14.1. Encoding . . . . .  | 103        |
| 14.2. Paszigráfia Rapszódia OpenGL full screen vizualizáció . . . . . | 105        |
| 14.3. Perceptron osztály . . . . .                                    | 105        |

|   |            |
|---|------------|
| <b>15. Helló, Stroustrup!</b>                               | <b>107</b> |
| 15.1. JDK osztályok . . . . .                               | 107        |
| 15.2. Hibásan implementált RSA törése . . . . .             | 108        |
| 15.3. Változó argumentumszámú ctor . . . . .                | 110        |
| <b>16. Helló, Gödel!</b>                                    | <b>112</b> |
| 16.1. Gengszterek . . . . .                                 | 112        |
| 16.2. Alternatív Tabella rendezése . . . . .                | 113        |
| 16.3. GIMP Scheme hack . . . . .                            | 116        |
| <b>17. Helló, !</b>   | <b>120</b> |
| 17.1. OOCWC Boost ASIO hálózatkezelése . . . . .            | 120        |
| 17.2. SamuCam . . . . .                                     | 120        |
| 17.3. BrainB . . . . .                                      | 123        |
| <b>18. Helló, Schwarzenegger!</b>                           | <b>129</b> |
| 18.1. Port scan . . . . .                                   | 129        |
| 18.2. AOP . . . . .   | 130        |
| 18.3. Junit teszt . . . . .                                 | 130        |
| <b>19. Helló, Calvin!</b>                                   | <b>132</b> |
| 19.1. MNIST . . . . .                                       | 132        |
| 19.2. CIFAR-10 . . . . .                                    | 133        |
| 19.3. Android telefonra a TF objektum detektálója . . . . . | 133        |
| <b>20. Bernes Lee</b>                                       | <b>138</b> |
| 20.1. C++ és Java összehasonlítás . . . . .                 | 138        |
| 20.2. Python . . . . .                                      | 138        |
| <b>IV. Irodalomjegyzék</b>                                  | <b>140</b> |
| 20.3. Általános . . . . .                                   | 141        |
| 20.4. C . . . . .   | 141        |
| 20.5. C++ . . . . .   | 141        |
| 20.6. Lisp . . . . .  | 141        |

# Ábrák jegyzéke

|  |     |
|--|-----|
| 2.1. PageRank : - Kép Wikipedia:                               | 13  |
| 2.2. Monty Hall Forrás:Wikipedia:                              | 16  |
| 3.1. 1.  | 19  |
| 4.1. Forrás:Batfai Norbert                                     | 29  |
| 5.1. Mandelbrot Forrás :Wikipedia:                             | 42  |
| 5.2. Julia halmaz Forrás :www.t-es-t.h                         | 48  |
| 5.3. CUDA Forrás: NVIDIA Developer                             | 53  |
| 7.1. John Horton Conway-féle életjáték Forrás : Batfai Norbert | 66  |
| 7.2. BrainB Benchmark : Batfai Norbert                         | 67  |
| 8.1. TensorFlow Forrás:SAP Blogs                               | 68  |
| 12.1. Nem sajat kep  | 98  |
| 13.1. Forrás : Sajat kep                                       | 100 |
| 13.2. Forrás : Sajat kep                                       | 100 |
| 13.3. Forrás : Sajat kep                                       | 102 |
| 14.1. mandelbrot   | 104 |
| 17.1. Forrás : Sajat kep                                       | 123 |
| 17.2. BrainB   | 128 |
| 19.1. Forrás : Sajat kep                                       | 135 |
| 19.2. Forrás : Sajat kep                                       | 137 |

# 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 más 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?

### 1.2. Milyen doksikat olvassak el?

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

### 1.3. Milyen filmeket nézzek meg?

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

**II. rész**

**Tematikus feladatok**

**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:

Megoldás forrása: <https://github.com/Samate99>Hello-world/tree/master/Prog1/Turing/Loop>

Ahhoz hogy egy processzormagot 100%-on dolgoztassunk egy végtelen ciklusra van szükségünk . Jelen esetben egy While ciklust láthatunk . A következő esetben hogy minden magot megdolgoztassunk kelleni fog az OpenMP! Itt includeolnunk kell a omp.h könyvtárat illetve a ciklusunkat a #pragma omp parallel függvényben kell elhelyeznünk . Végül pedig hogy egy mag se dolgozzon egy egyszerű sleep függvényt kell alkalmazunk illetve includeolni kell az unistd.h könyvtárat.

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

Tehát itt lathatjuk az első megoldását ! Itt egyetlen egy magot terhelünk 100%ban , nincs is másra szüksé-  
günk csak egy egyszerű while ciklus alkalmazására.

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

```
    return 0;  
  
}  
}
```

A feladat második részében láthatjuk hogy a feladat megfelelő megoldásához includeoldni kellett egy konyvtárat névszerint az om.h konytarat . Ezután nincs is másra szükségünk csak a while programunka a while ciklus elő be kell illesztenünk a #pragma omp parallel függvényt. Ennek segítségével egyszerűen tudjuk dolgoztatni az összes magot a számítógépünkben.

```
include <unistd.h>  
  
int main() {  
    while(1) {  
        sleep(1);  
  
    }  
}
```

Az utolsó részben az volt a feladat hogy egy magot 0 %ban dolgoztassunk. Ez is egy viszonylag egyszerű feladat nincs másra szükségünk mint a sleep funkcióra ennek a programba való beillesztésével elérhetjük hogy egy magunk 0%ban dolgozzon

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

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

Megoldás videó:

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

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

```
}
```

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

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

akár önmagára

```
T100(T100)
false
```

ezt a kimenetet adja.

A T100-as programot felhasználva készítsük most el az alábbi T1000-set, amelyben a Lefagy-ra 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 `Le fagy` függvényt, azaz a T100 program nem is létezik.

Veleményem szerint Nem tudunk olyan programot írni amely képes egy előre megírt programról meghatározni hogy az a program képes lesz e a lefutásra vagy nem lesz e képes azaz le fog fagyni.

## 2.3. Változók értékének felcserélése

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

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

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Turing/valtozocsere>

2 változó felcserélésére többféle módszer alkalmazható . Mintpéldául használhatunk egy segédváltozót vagy különböző muveletekkel is megoldható a feladat. Én személy szerint a második opciót választottam es különböző matematikai műveletek segítésével oldottam meg a feladatot ! Lássuk is hogy műkik a kód. Tehát Van 2 változónk az a és a b ! Az első változó azaz az a helyére felvesszük az a+b összegét azaz az első változó + a második változó összegét . Utána a b helyére felvesszük a-b összegét tehát az a kezdőértéke átkerült a b-be ! Ezután az a valtozóba berakjuk az a-b összegét tehát az (a+b)-((a+b)-b) összegét ezzel pedig a b kezdőértéke átkerült az a-ba.

```
#include <stdio.h>

int main() {

    int a = 6;
    int b = 2;

    printf("a = %d ", a);
    printf("b = %d \n", b);

    a = a+b;
    b = a-b;
    a = a-b;

    printf("a = %d ", a);
    printf("b = %d \n", b);

}
```

## 2.4. Labdapattogás

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

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

Megoldás forrása:<https://github.com/Samate99>Hello-world/blob/master/Prog1/Turing/labda>

A feladatunk a labdapattogás feladat lesz . Velemenyem szerint ezzel a feladatban a bevprog során mindenki talalkozott hisz többször is előkerült mind a sima mind az Ifes változata A kódot olvasva lathatjuk hogy több minden is deklarálnunk kell. Meg kell adnunk a labda kezdőhelyzetét ezen felül az ablak méretét illetve a labda mozgásának pattogásának mértékét. Ahogy lathatjuk a kódban egész sok függvényt fogunk alkalmazni ilyen például a refresh ami a nevéből adódóan a program frissítéséért felel ilyen a getmaxyx() amelyel a Dinamikusságot adjuk a programnak vagy eppen ilyen a mvprintw() aminek segítségével magát a labdát rajzolhatjuk ki vagy ilyen az usleep() amelyel a labdánk gyorsaságát állíthatjuk be és továbbá ott van meg az initscr függvény is amelyel a terminal méretétől van kapcsolatban . A Maga program az IF függvények használatával állapítja meg hogy eppen elérte e a labda az adott meretű terminal valamelyik szélét . Ha elérte akkor a labda irányt vált

```
// nem sajat kód
#include <stdio.h>
#include <curses.h>
#include <unistd.h>
int main ( void )

{
    WINDOW *ablak;
    ablak = initscr();

    int x = 0;
    int y = 0;

    int xnov = 1;
    int ynov = 1;

    int mx;
    int my;

    for ( ; ; ) {
        getmaxyx ( ablak, my , mx );
        mvprintw ( y, x, "O" );
        refresh ();
        usleep ( 100000 );

        x = x + xnov;

        y = y + ynov;

        if ( x>=mx-1 ) { // elérte-e a jobb oldalt?
            xnov = xnov * -1;
        }
        if ( x<=0 ) { // elérte-e a bal oldalt?
            xnov = xnov * -1;
        }
        if ( y<=0 ) { //
```

```
        ynov = ynov * -1;
    }
    if ( y>=my-1 ) {
        ynov = ynov * -1;
    }
}
return 0;

}
}
```

## 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:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Turing/shift.c>

Tehát ebben a megoldásban a Bitshift nevezetű operatorral fogunk dolgozni. Ahogy lathatjuk a megoldás során Shiftelésre leszünk szükésgünk . Ez azt jelenti hogy folyamatosan haladunk egy adott irányban. Ezt jelen esetben egy while ciklust használva fogunk megcsinalni de lehetne akar más ciklust is alkalmazni . Miután a elshifteljük 0hoz es sikeres volt a muvelet illetve nem vétettünk hibát gyakorlatilag megkapjuk a szó méretét bitben

```
#include <stdio.h>

int main() {
    int i = 1;
    int a = 0;
    while (i != 0) {
        i <<= 1;
        a++;
    }

    printf("Szóhossz: %d bit\n", a);

}
```

## 2.6. Helló, Google!

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

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Turing/Pagerank>

Szóval a Page-Rank Maga egy algoritmus amely forradalmásította illetve a mai napig meghatározza az interneten való keresést ! A legtöbb keresésre specializálódott oldal ezt használja többet között a google is , nemcsoda hisz a Google 2 alapítója hozta létre ezt az algoritmust még anno 98'ba egyetemi tanulmanyaik során stanford városában Ennek segítségével kapunk számunkra megfelelő releváns oldalakat a kereséseink során !A rendszer ugy működik hogy a PageRank rendszer alapjan listázza ki az oldalakat és ennek alapjan Hozza fel az oldalakat a keressők számára! A rendszer szerint amelyik oldalra többen kiváncsiak azaz hivatkoznak és kattintanak rá tehát nagy a népszerűsége ezáltal nagyon magas a Pagerankja . A Programunk ugy nez ki hogy van egy vegtelen ciklusunk ami gyakorlatilag a végtelenségig fut . Ebben található egy feltétel . A ciklus azonnal megszűnik amint a feltételben említett tavolság kisebb mint 0.00000001. A megoldáshoz mátrixokat és vektorokat kell alkalmaznunk.A program az adatokat letarolja egy matrixba ami mutatja az oldalak közötti helyes sorrendet , és hogy melyik oldal mutat azaz hivatkozik a másik oldalra . !Miutan a programunk lefutott kifogja dobni számunkra a 4 weboldal megfelelő sorrendjét a pagerank keresési algoritmus alapján.

```
// nem sajat kód

#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, 1.0}
    };
}
```

```
{ 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 };

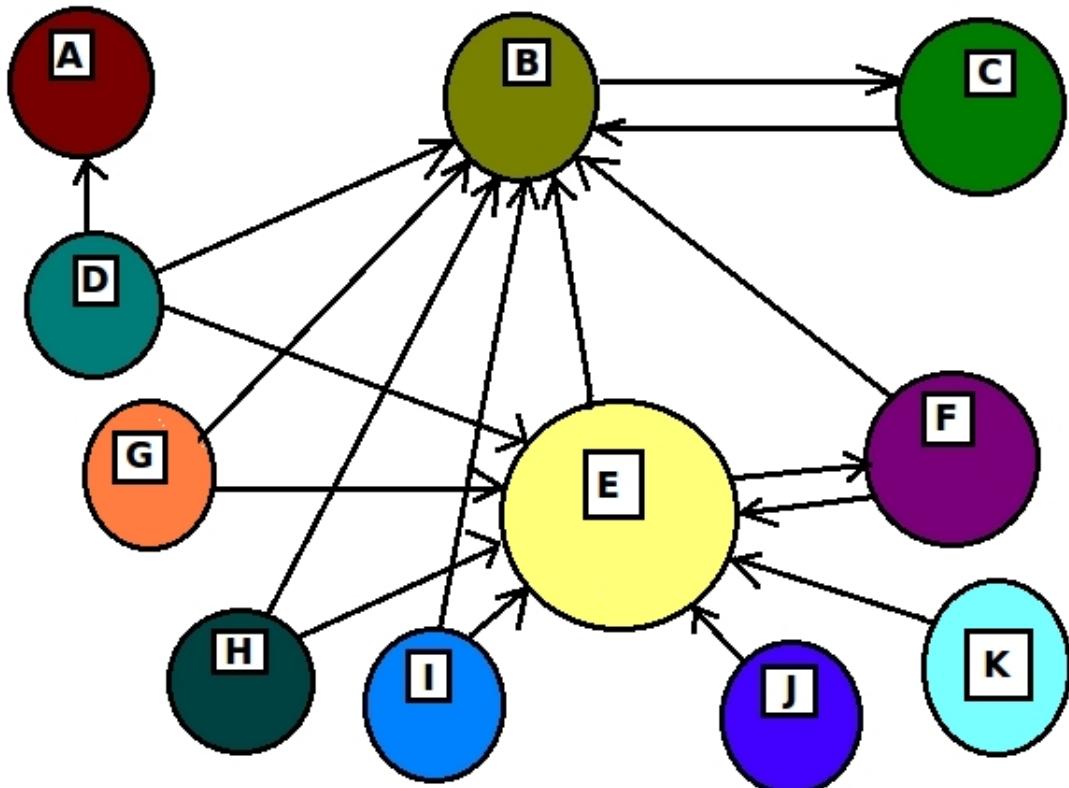
int i, j;
for (;;)
{
    for (i = 0; i < 4; ++i)
    {
        PR[i] = 0.0;
        for (j = 0; j < 4; ++j)
            PR[i] += (L[i][j] * PRv[j]);
    }
    if (tavolsag (PR, PRv, 4) < 0.00000001)

        break;
    for (i = 0; i < 4; ++i)

        PRv[i] = PR[i];
}
kiir (PR, 4);
return 0;

}
```





2.1. ábra. PageRank : - Kép Wikipedia:

## 2.7. 100 éves a Brun tétele

Írj R szimulációt a Brun tétele demonstrálására!

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

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

Mielőtt belekezdünk a programban érdemes letisztázni hogy miről is kell beszénünk. Fontos megemlítenünk a primeket és ikerprímeket hisz ezekről szól a feladat. Tehát prímeknek azokat a (természetes) számokat nevezzük amelyeknek pontosan 2 osztóval rendelkeznek ilyen peldaul a 3 , az 5 vagy akár a 7 és a 11 is stb...! Ez a 2 osztó az egy Az egy és önmaguk . Kivetéleke közé sorhatjuk a 0 illetve a 1es szamot hisz ezekre igazak a feltételek viszont nem tartoznak a prím számok közé. Az Ikerprímek a prímeknek egy olyan esete amikor már tudunk 2 prímről és ezeknek a prímeknek a különbsége kettő tehát például az 5 és a 7 vagy akár a 11 és a 13 stbb... Nagyon érdekes hogy még a prímről mennyit hallunk az ikerprímek fogalma szinte ismeretlen az átlagembereknek . Gyakorlatilag szinte végtelen ikerprímről beszélhetünk , generalhatunk , rengeteget ismerünk, alkalmazunk ,Tehát a programunk veletlenszeruen generál számunkra prímeket . Miutan ezzel vegzett elkezdi összeveti a megadott feltételek alapjan a prímeket . Megvizsgálja hogy mekkora az első és a második prím különbsége . Ha nem kettő akkor a program dolgozik tehat ugrik a következő 2 primre és vizsgálja azokat még nem talal ikerprímet. Ha talal tehat a vizsgalat alap-

jan a kettő különbsége kettő lesz . Eltarolja az adatokat a megfelelő helyekre A program is ezen az elven mukodik megvizsgálja hogy az első és a második prímnek mekkora a különbsége.Eltárolja az adatokata megfelelő helyekre majd ezelből az adatokból reciprokokat képez , összeadja őket és visszaadja az értéket.Miutan megfelelő mennyiséggü adatot gyűjtött a a program megfelelő működéséhez a már számítógépes matekmatika és vizualizáció óráról is ismert plot függvényel kirajzoltatja nekunk a megfelelő eredményt.

```
// nem sajat kód
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")

}
```

## 2.8. A Monty Hall probléma

Írj R szimulációt a Monty Hall problémára!

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

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

Ez egy nagyon érdekes televíziós elkezelés . Több televíziós műsorban is megfigyelhető volt az évek alatt. Az elkepzeles szerint 3 különböző ajtó található es a jatek folyaman ezek közül kell választanunk a megfelelőt , az egyik ajtó mögött egy nyeremény vagy egyéb érték található a másik kettő ajtó mögötti tér viszont üres ! A játékos célja hogy eltalalja a megfelelő ajtót , azaz ,megnyerje a jateket , értékes nyereményekkel legyen gazdagabb a jatek után. . Tehat a jateket magát egy musorvezető irányítja az ő feladata nagyon fontos lesz . Az ő jelzésére kell a jatekosnak kivalasztani egy ajtót ..Miutan ez megtörtént A musorvezető kinyitja az egyik üres ajtót azaz most mar csak kettő ből kell eltalálni a megfelelő ajtót , tehát a legtöbb ember szerint most mar 50%-50% esélyünk van a nyerni jatekosként . Gyakorlatilag viszont ha a megvaltoztatjuk az eredetileg alkotott döntésünk és a fentmarad két ajtóból azt az ajtót választjuk amelyre eddig nem szavaztunk 2/3 esélyünk van a nyeremény megnyerésére .A program elég sok random esetet generál amelyekben kulonbozo variaciok és konbinaciok jatszódnak le . Meg kell hataroznunk az esetek számát hogy ne a végtelenségig generalja az eseteket .Ezután egy ciklussal vegig megyünk es megnezzük a jatekok alltal adot eredményket .Hogy éppen az adott jatekban ki kapott montot az ekkora kapott pontokat variaciokat konbinaciokat , eredményeket kiertekeljük Majd az osszegzett eredményket kiiratjuk.

```
// nem sajat kód
kiserletek_szama=10000000
kiserlet = sample(1:3, kiserletek_szama, replace=T)
jatekos = sample(1:3, kiserletek_szama, replace=T)
musorvezeto=vector(length = kiserletek_szama)

for (i in 1:kiserletek_szama) {

  if(kiserlet[i]==jatekos[i]) {

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

  }else{

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

  }

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

}

nemvaltoztatesnyer= which(kiserlet==jatekos)
valtoztat=vector(length = kiserletek_szama)

for (i in 1:kiserletek_szama) {

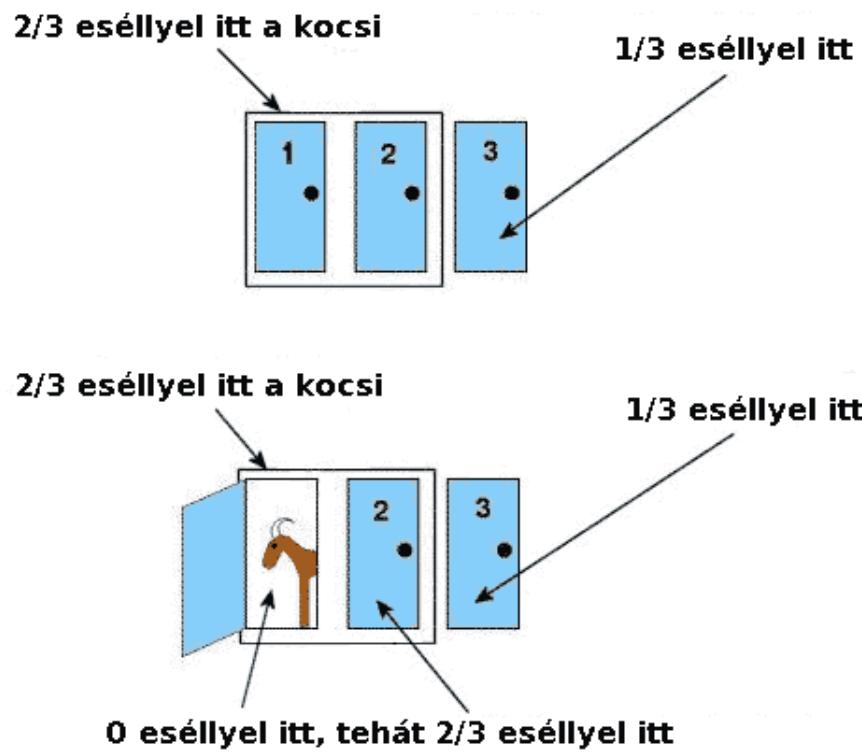
  holvalt = setdiff(c(1,2,3), c(musorvezeto[i], jatekos[i]))
  valtoztat[i] = holvalt[sample(1:length(holvalt),1)]

}

valtoztatesnyer = which(kiserlet==valtoztat)

sprintf("Kiserletek szama: %i", kiserletek_szama)
length(nemvaltoztatesnyer)
length(valtoztatesnyer)
length(nemvaltoztatesnyer)/length(valtoztatesnyer)
length(nemvaltoztatesnyer)+length(valtoztatesnyer)

}
```



2.2. ábra. Monty Hall Forrás:Wikipedia:

## 3. fejezet

# Helló, Chomsky!

### 3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfjával megadva írd meg ezt a gépet!

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Turingfeladat>

Tehát a feladatban decimalis szamrendszerből kell átváltanunk Unarisba. Tehát első részben érdemes tisztázni az alapfogalmakat ebben az esetben is. A decimalis szamrendszer másnéven 10es számrendszer az a szamrendszer amelyet a minden napokban használunk ez "Default" szamrendszer a minden napokban vele menyem szerint habar lehet valaki valahol nem ezt használja. De azért nagyobbsegben szerintem igen. Az unáris azaz az egyes szamrendszer azaz az egyes szamrendszert ritkábban használjuk, alkalmazzuk de attól függetlenül hogy ismeretlennek hangzik mindenki talalkozott mar vele valószínűleg ha nem is magas szinten alkalmazta viszont ezt a palcikas szamolás módszerrel élete során 1-1 alkalommal biztosan hasznalta. Ez a legegyszerubb szamrendszer fontos hangsúlyozni hogy kizárolag természetes számok alkalmazására lehet használni! Tehát például törteket nem tudunk vele leírni. Lassuk is a kódot tehát Gyakorlatilag eben a szamrendszerben pálcikák, vonalak segítségvel tudunk leírni számokat! (Egy |, kettő ||, öt ||||, 10 ||||| |||||) programunk ugy mukodik hogy bekérjük a számot jelen esetünkben ez az "a" változóba kerül. A program megvizsgálja hogy tényleg szám-e. Erre azért van szükség mert nem megfelelő bementei adatok esetében a program nem képes a megfelelő lefutásra. Ha a program és a bemeneti adatok megfelelőek akkor a ciklus annyiszor fut le amekkora a bekert számunk azaz amennyi a valtozo értéke. A programunk minden lefutás után ír egy l-t az outputra azaz a megadott kiemelten. Megfelelő lefutás esetén a palcikák száma megegyezik a valtozóban szereplő szám értékével tehát ha a mi számunk a hármas volt akkor az outputon ||| ennek kellett megjelennie.

```
// nem saját kód
#include <stdio.h>
#include <iostream>

using namespace std;
int main()

{
```

```
int a=0
cout << "írjon be egy számot : "<< '\n';
cin >> a ;
for (int i =0 ; i<a , i++ ) {
cout << "I" ;

}
cout<< '\n';

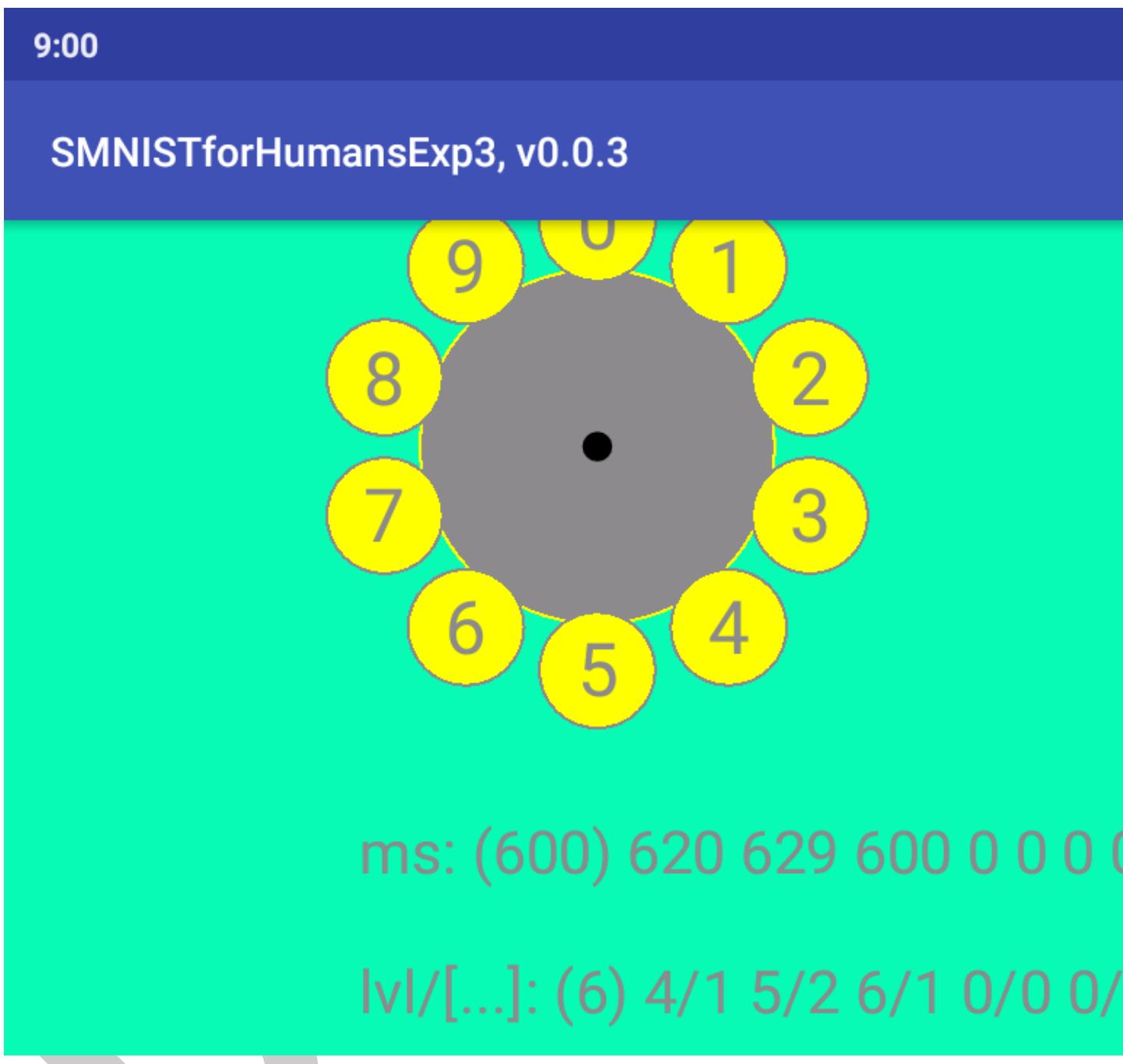
}
```

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

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

Megoldás videó:

Megoldás forrása:



3.1. ábra. 1.

### 3.3. Hivatkozási nyelv

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

Megoldás videó:

<https://github.com/Samate99>Hello-world/blob/master/Prog1/C89.c>

A BNF az egy ugynevezett metszaniaxis amely a kulonbozo nyelvtanok leírásához használnak illetve alkalmazzak. Fontos megjegyezni hogy talan ez a metaszinaxis a legyakoribb amint használnak különboző programozási nyelvek nyelvtanainak megfogalmazásához , leírására , igazolására . Tehat a feladat "feladat" része az hogy a C89 és C99 között kell talánunk különbségeket . Fontos letisztázni hogy nagy mértékben kotodnek egymashoz hisz ez a 2 nyelv 2 különbőző változata. Az egyik a már napjainkra kissé elavult a masikat pedig jelenleg is használjuk. Tehat a lenyeg hogy mindenki a C nyelv egy változata . Ami példánkban van egy kis különbség a 2 szabvány között !A probléma a for cikluson belüli deklaráció , tehát korábbi "C" verizóba minden fontos volt deklarálni a ciklus előtt hogy a program megfelelően lefusson . Ezzel ellentétben az új kiadásban már akár a cikluson belül is dekláralhatunk nem fogja befolyasolni a program felutását .

### 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:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Chomsky/lx.1>

Tehát az elsőre bonyolultnak tűnő lexikálásról lesz szó tehát az a kérdés hogy mi is fog történi. Ha megfelelő a lefutás . A lefutás után a %-jelek között található kódból a lexel létre fog nekünk hozni egy C programot . Egy olyan programot amely a beírt szöveget egy adott szempont szerint vizsgálja .A mi esetünkben valós számokat keres , rögzíti öket illet összeadja őket A program kezdetekor a realnumbers változót nagyon fontos hogy deklárljuk 0 kezdőértékkal . Ha más értéket adunk meg akkor nem valós adatokat fog megadni a program. Ezutan a digit definíáljuk ! Ez lényegében ez egy sablon lesz aminek a segítségével fogja atnezni , és vizsgálni a kesobb szamaram megadott szöveget . Tehát ennek a segítségével próbálja kiszurni szamunkra a valós számokat Itt megadjuk neki hogy milyen karaktercsoporthoz mit adjon vissza tehát .Tehát a mi esetünkben hogy mi az alltalanos leírása egy valós számnak. Ha a később megadott szövegben talál a sablonnak megfelelő adatot akkor realnumbers változó értéke megnő 1-el . Tehát ha talál 5 dolgot amely megfelel az álltalunk megadott sablonnak akkor a realnumbers értéke 5-vel fog megnőni. és kiiratjuk ! Végül kiemelném a nagyon fontos yylex() függvény alkalmazását , ennek a segítségével indítjuk el a lexikálást tehát ha ezt nem alkalmazzuk a programunk nem fog megfelelően lefutni .

```
// nem saját kód
#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;
}
```

### 3.5. l33t.l

Lexelj össze egy l33t ciphert!

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Chomsky/l33t.l>

Ezt programot az előzőhez hasonlóan a lexikális elemző generátor segítségével fogjuk elkészíteni. Tehat itt is lexelni fogunk. Tehat ebben az esetben is a lexel segítségével fogunk létrehozni egy C kódot. A program első felében létrehozunk egy ugynevezett strukturát ami tartalmaz 2 nagyon fontos részt. Tehat elsőről tartalmazza a cserélendő karaktert illetve tartalmaz 4 karaktert amik közül véletlenszerűen fogunk választani és ezeket a karaktereket fogjuk berakni a cserélendő karakterek helyére ! A program masodik részében talalható egy random generator ,ami a megadott intervallumban general különböző számokat szamunkat és ezek kiértékelésével foglalkozik. A randomgeneratorban generált random szám segítségével fogja eldöntheti a programunk hogy melyik cserélendő karaktert megfelelő csere karakterre cserelje a program felutásra során . szóval összegezve a program a random generált szám segítségével kiválaszt egy karaktert az egyik strukturából majd ezt a karaktert a megfelelő helyre a másik strukturába athelyezi és így keveri össze a szöveget. Végül pedig az előző feladathoz hasonlóan itt is kiemelném hogy használnunk kell az yylex() függvényt a program megfelelő működéséhez mivel ha ezt elhanyagoljuk akkor a program nem fog lefutni.

```
// nem saját kód
#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"}}

};

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

```

```
    if(r<91)
        printf("%s", 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());
    ylex();
    return 0;
}
```

## 3.6. A források olvasása

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

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

- iii. 

```
for(i=0; i<5; i++)
```
- iv. 

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

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

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

```
printf("%d %d", f(a), a);
```
- viii. 

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

Megoldás forrása:

Megoldás videó:

1. Hat a Sigin jel figyelje volt tehat nem volt figyelmen kivül hagyva akkor az adott jelkezelő függvény kezelje ,de ha figyelmen kivül volt hagyva akkor Ő is hagyja figyelmen kivül.
2. Hat a masodik kód nem tul nehez egy ciklusrol beszelhetünk ami 4ig megy . A ciklus addig fog lefutni még az i értéke kisebb mint 5. Ha igaz a feltétel noveli az I értékét .
3. Szerintem ez a 2 kód ugyanaz azzal a kivetellel hogy az I vegső értéke nem 5 lesz mint az előző esetben hanem 4.
4. A ciklus során elemeket kepezünk , minden lefutás után egyre nagyobb és nagyobb elemeket kapunk és ezeket beletoltjuk a Tomb megfelelő elemébe
- 5.A következő kódban is egy ciklussal talalkozhatunk itt addig megy a folyamat még az I kisebb mint n illetve a d eleme megegyezik az s elemével
6. Ebben az esetben kiiratunk 2 számot amelyet aez F függvény határoz meg
7. Itt is kiiratunk 2 erteket az egyik az F(a) értéke lesz a masik pedig az 'a' értéke
8. Ebben a helyzetben is kiiratunk 2 erteket ebben az esetben az F nagy mertekben meghatarozza az a értékét
- .

## 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 \text{ prim})) \leftrightarrow $
```

```
$ (\exists y \forall x (x \text{ prim}) \supset (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](https://gitlab.com/nbatfai/bhax/blob/master/attention_raising/MatLog_LaTeX)

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

1 Az elsőben véleményem szerint az olvasható hogy a prímszámok száma végtelen , Azaz Végtelen sok prímszám van. . A másodikban ugyanugy a prímszámok számáról van szó , Itt arra utal hogy a Ikerprímek száma végtelen . Ez termeszetesen igaz hisz ha a primszamok száma végtelen akkor az ikerprímek szama is vegtelen A harmadik illetve negyedik esetben csak a "nyers" kiolvasás az eltérő minden kettőtől ugy lehet fordítani hogy "véges sok prímszám van"

### 3.8. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

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

Mit vezetnek be a programba a következő nevek?

- `int a;`
- `int *b = &a;`
- `int &r = a;`
- `int c[5];`
- `int (&tr)[5] = c;`

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Egy változót A néven

Egy változóra mutatót

A valtozó azaz a referenciaját

A kovetkező peldaban egy öt elemből álló tömbel talakozhatunk.

Az előző feladatban talalható tömb referenciaját lathatjuk.

Az öt elemből álló tömbre mutatót lathatunk.

Egy függvényre mutatót lathatunk .

Egy függvényre mutatót függvényt.

Egy olyan mutató függvényt amely visszaad egy C-t és kap 2 valtozot az a-t és a b-t

Egy Függvényre mutató függvény mutatót ami az elozohoz hasonloan visszad egy értéket azaz egy valtozót és visszaad 2 valtozot .

## 4. fejezet

# Helló, Caesar!

### 4.1. int \*\*\* háromszögmátrix

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Ceaser/matrix.c>

Tehát a feladatban egy haromszög Matrixot kell készítenünk. Mátrixokkal már mindenki talalkozott a többség talán már gimnáziumban de Diszkrét Matematika óran biztos mindenki talalkozott velük . Illetve csodálhatta meg őket. A Háromszög mátrix egy olyan specialis , negyzetes matrix aminek az osszes eleme a főatló felett vagy a főatló alatt 0 . Az programnak egy ilyen ennek megfelelő matrixot kell hogy letrehozni illetve használni , . A matrixunk a ebben az esetben 5 sorból fog állni , és mivel már korábban jeleztem hogy ez egy negyzetes mátrix így a sorok / oszlopok számának egyenlőnek kell lennie Tehát 5 sorból és 5 oszloból kell állni. A megfelelő konyvtarakat ebben a programban is erdemes megfelelően includeolni mert anelkül nem valószínű hogy megfelelő eredményre jutunk. Az oldalak és sorok elkészítéséhez egyerettemű hogy pointereket és ciklusokat kell aklamaznunk. Deklarálnunk kell az nr-t ami a mi esetünkben 5 lesz . Az nr gyakorlatilag a sorok számát határozza meg a ez a mi esetünkben 5. Ezt fontos hogy a mainen belül vegyük fel . Ahogy a lenti kódon lathatjuk több ciklust is alkalmaznunk kell pontosan For ciklusokat . A kódban lathatjuk hogy a tm értéket többször is ki fogjuk iratni . Lathatjuk hogy több Ciklus is NR-ig azaz 5ig mennek és utana lep ki . Fontos kiemelni a malloc függvényt ennek a segítségével foglalunk le memóriat . Lathatjuk hogy ha ez nem sikerül akkor a program -1es hibakóddal kilep . Láthatjuk hogy a for ciklus segítsével lekrealjuk a matrixut . A program végén felszabadítjuk a Malloc függvénytel foglalt memóriat majd befejezzük a programot

```
// nem sajat kód
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    int nr = 5;
```

```
double **tm;

printf("%p\n", &tm);
if ((tm = (double **) malloc (nr * sizeof (double *))) == NULL)

{
    return -1;
}

printf("%p\n", tm);
for (int i = 0; i < nr; ++i)

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

printf("%p\n", tm[0]);
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)

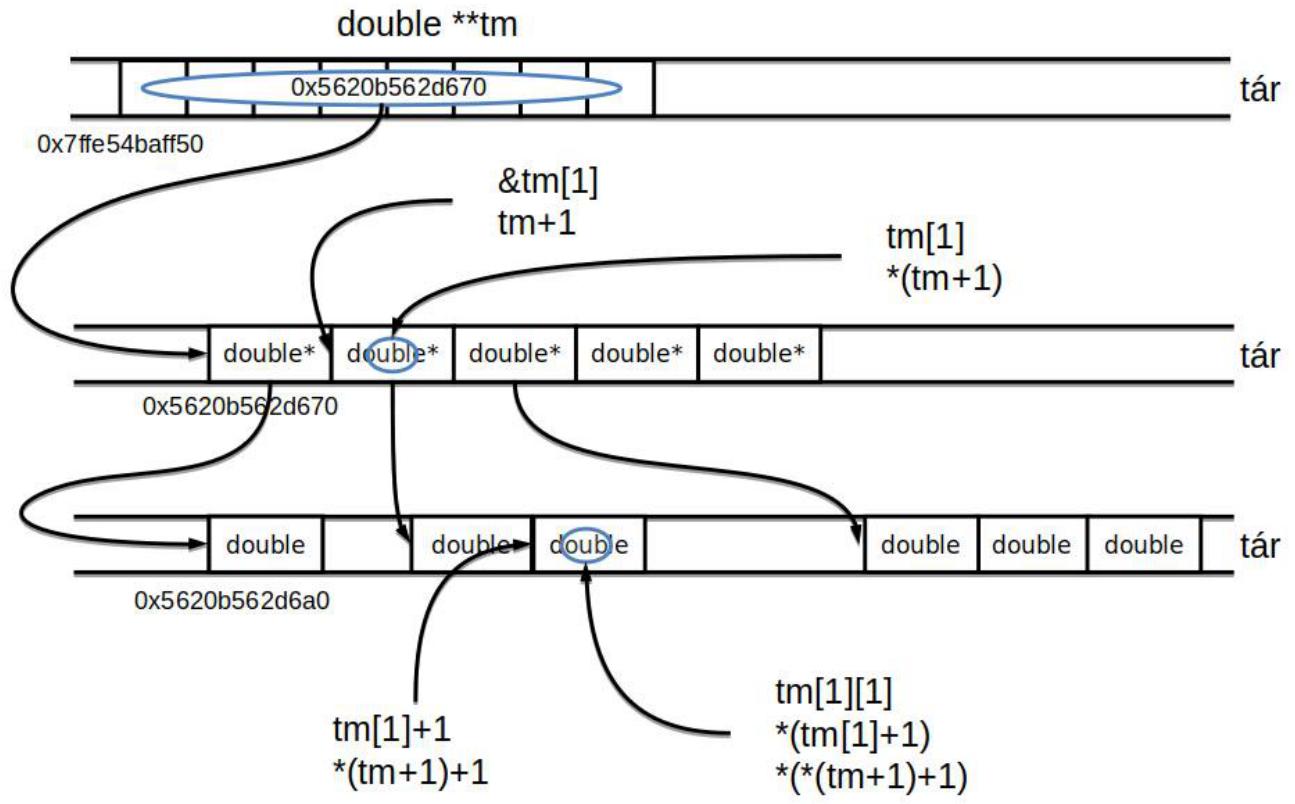
{
    for (int j = 0; j < i + 1; ++j)
        printf ("%f, ", tm[i][j]);
    printf ("\n");
}
```

```

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

    return 0;
}
}

```



4.1. ábra. Forrás:Batfai Norbert

## 4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/EC>

Tehát a feladatunk egy C exor titkosító létrehozása . A feladatban meg kell adnunk 2 nagyon fontos dolgot egy szöveget amit a program letitkosít és egy kulcsot amiből a karaktereket nyeri. A program az előre megadott kulcsban szereplő karakterekkel exorozza , keveri össze a szöveget . Ennek a segítsével hozzuk létre az exorozott szöveget. Nagyon érdekes hogy egyes kulcsok megadasaval milyen érdekes kódolt szövegek jönnek létre . A programhoz szükség van 2 TXT állományra az egyikbe a tiszta szöveget kell tennünk, a másikba pedig kesobb fog kerülni a titkos szöveg. A programhoz includeolunk kell a megfelelő

könyvtárakat . Fontos a mainbe beleírnunk hogy argumentumokkal dolgozunk fontos definialni a kulcs méretet illetve a buffer méretet . Illetve a megfelelő dolgokat deklarálnunk kell. Beolvassuk a szöveget illetve a kulcsot . Utana a kódban lathatjuk ahogy egy ciklus segítségével exorozza össze a kulcs karaktereit a szöveg karaktereivel és hozza létre a titkos szöveget .

```
// nem sajat kód
#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);
    }
}
```

## 4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/exor.java>

Gyakorlatilag ugyanugy mukodik mint a C program. Megadunk egy kulcsot és egy tiszta szöveget , a kulcs karaktereit összeexorozzuk a szöveg karaktereivel és ebből kapunk egy olvashatatlan titkos szöveget. A program itt is ugyanugy mukodik meg kell adnunk az alap adatokat , deklaralnunk kell . Bekerül a szöveg a bufferbe . Utana kovetkezik while ciklus itt a kulcs segítségével elkészítjük a kódolt, exorozott szöveget ! Illetve lathatjuk hogy ebben az esetben bekerült egy Try-Catch a kódba. Fontos kiemelni hogy mivel ebben az esetben egy masik nyelvben dolgozink a programnak más formatumban kell tarolunk tehát jelen esetben .java lesz a fajlformatum

```
// nem sajat kód
public class ExorTitkosito {

    public ExorTitkosito(String kulcssSzoveg,
                          java.io.InputStream bejovoCsatorna,
                          java.io.OutputStream kimenocsatorna)
                          throws java.io.IOException {

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

        while((olvasottBajtok =
               bejovoCsatorna.read(buffer)) != -1) {
            for(int i=0; i<olvasottBajtok; ++i) {

                buffer[i] = (byte)(buffer[i] ^ kulcs[kulcsIndex]);
                kulcsIndex = (kulcsIndex+1) % kulcs.length;
            }
            kimenocsatorna.write(buffer, 0, olvasottBajtok);
        }
    }

    public static void main(String[] args) {
        try {
            new ExorTitkosito(args[0], System.in, System.out);

        } catch(java.io.IOException e) {
            e.printStackTrace();
        }
    }
}
```

## 4.4. C EXOR törő

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

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/TC>

A feladatunk egy C exor Torő lesz . a különboző jelszó /kulcs feltörések nagyon erdekesek illetve nagyon erdeklik az embereket . Több féle feltörési módszer létezik van melyik random próbálgaat jelszavakat / kulcsokat , van amelyik valamilyen algoritmus alapjan próbálgaat a kulcsokat/ jelszavakat , vagy van a mindenki szamara ismert bruteforce modszer ebben az esetben a program az osszes lehetséges variációt / konbinációt kipróbalja es így próbálja feltörni a dolgokat. . Amint megtalalja a kulcs segítségével visszaalakítja a szöveget az eredeti állapotába . Nagyon fontos hogy elore meg kell adnunk hogy hánny karakterből áll a kulcs.Ahogy lathatjuk a programban miutan beolvassuk a szöveget egy behívott függvény segítésével és megadtuk a kulcs méretét a program elkezdi generalni a kulcsokat és elkezdi próbálgnai a szövegen . Az exor addig próbálja feltörni a szöveget ameddig vagy le nem alltjuk vagy nem sikerül neki. Ez a módszer rengeteg erőforrást illetve időt igényel emiatt sajnos sok időbe is tellhet egy olyan meretű kulcs megtalalása. A program az exor\_tores segítésével ellenorzi hogy sikeres volt e a müvelet vagy haladhat tovább a kovetkezőre Ha sikerült akkor a program vagy a standard outputra vagy az elorre definált helyre kiírja a titkos szoveget

```
// nem sajat kód
#define MAX_TITKOS 4096

#define OLVASAS_BUFFER 256

#define KULCS_MERET 5

#define _GNU_SOURCE

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

double
atlagos_szohossz (const char *titkos, int titkos_meret)
{
    int sz = 0;
    for (int i = 0; i < titkos_meret; ++i)
        if (titkos[i] == ' ')
            ++sz;

    return (double) titkos_meret / sz;
}
```

```
int
tiszta_lehet (const char *titkos, int titkos_meret)
{

    // a tiszta szöveg valszeg tartalmazza a gyakori magyar szavakat
    // illetve az átlagos szóhossz vizsgálatával csökkentjük a
    // potenciális töréseket

    double szohossz = atlagos_szohossz (titkos, titkos_meret);

    return szohossz > 6.0 && szohossz < 9.0

        && strcasestr (titkos, "hogy") && strcasestr (titkos, "nem")
        && strcasestr (titkos, "az") && strcasestr (titkos, "ha");
}

void
exor (const char kulcs[], int kulcs_meret, char titkos[], int titkos_meret)

{
    int kulcs_index = 0;

    for (int i = 0; i < titkos_meret; ++i)

    {

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

    }

}

int
exor_tores (const char kulcs[], int kulcs_meret, char titkos[],
            int titkos_meret)
{
    exor (kulcs, kulcs_meret, titkos, titkos_meret);
    return tiszta_lehet (titkos, titkos_meret);
}

int
main (void)
{
    char kulcs[KULCS_MERET];
    char titkos[MAX_TITKOS];
```

```
char *p = titkos;
int olvasott_bajtok;

// titkos fajt berantasa
while ((olvasott_bajtok =
    read (0, (void *) p,
    (p - titkos + OLVASAS_BUFFER <
    MAX_TITKOS) ? OLVASAS_BUFFER : titkos + MAX_TITKOS - p)))
{
    p += olvasott_bajtok;
    // maradek hely nullazasa a titkos bufferben
    for (int i = 0; i < MAX_TITKOS - (p - titkos); ++i)
        titkos[p - titkos + i] = '\0';

    // osszes kulcs eloallitasa
    for (char ii = 'A'; ii <= 'Z'; ++ii)
        for (char ji = 'A'; ji <= 'Z'; ++ji)
            for (char ki = 'A'; ki <= 'Z'; ++ki)
    for (char li = 'A'; li <= 'Z'; ++li)
        for (char mi = 'A'; mi <= 'Z'; ++mi)
    {
        kulcs[0] = ii;
        kulcs[1] = ji;
        kulcs[2] = ki;
        kulcs[3] = li;
        kulcs[4] = mi;

        if (exor_tores (kulcs, KULCS_MERET, titkos, p - titkos))
            printf
("Kulcs: [%c%c%c%c]\nTiszta szoveg: [%s]\n",
ii, ji, ki, li, mi, titkos);

        // ujra EXOR-ozunk, igy nem kell egy masodik buffer
        exor (kulcs, KULCS_MERET, titkos, p - titkos);
    }
    return 0;
}
```

## 4.5. Neurális OR, AND és EXOR kapu

R

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

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

A Neuralis halók nagyon erdekesek . Nezzük is meg a kódot . A kódban azt lathatjuk hogy a neuralis halók bementei adatokat kapnak és logikai értékeket adnak vissza . Ennek a programnak a segítségével nagyon egyszeruen tudunk előallítani rendkívül bonyolult logika kódokat . Tudhatjuk hogy véges számu osszesen 7 DB neuralis kapuról beszélhetünk . ezek az AND OR NOT NAND NOR EXOR EXNOR A kódot olvasva lathatjuk hogy mi ezek közül harmat használunk az OR-t and AND-et és a Exor-t.

Program T100

```
# 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/>
#
# https://youtu.be/Koyw6IH5ScQ

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.operand)

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])

}
```

## 4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/Ceaser/perceptron.cpp>

Szóval a feladatunk hasonló az előzőhez itt is hasonló temakorben fogunk foglalkozni . Gyakorlatilag a program egy gépi tanulással kapcsolatos algoritmus . Tehát a feladatunk nagy mértékben kötődik az előzőhez , akar beszélhetünk arról is hogy az előző egy változata . Az első kérdés talan az lehet az olvasóba hogy mi is a perceptron. A perceptron egy tanulással kapcsolatos algoritmus . Kiemelném hogy a perceptron

Létrejött rengeteg embernek koszonhetjük de az első változatanak elkeszítője Perceptron Rosenbalt volt . Aki ki nem találnak a nevadó is volt . A program eleg bonyolult de szerintem mindenki számára latható hogy korülbelül 3 részre oszthatjuk a kódot . Lathatunk egy részt amelyben ahogy latjuk fogadjuk az adatokat lathatunk egy második részt amely képes lesz összegezni az adatokat illetve eltarolni illetve van egy harmadik rész amely képes döntéseket hozni és lathatjuk hogy képes kezeli az adatokat.

```
// nem saját kód
#include <iostream>
#include "ql.hpp"
#include <png++/png.hpp>

int main (int argc, char **argv)
{
    png::image<png::rgb_pixel> png_image (argv[1]);
    int size = png_image.get_width() * png_image.get_height();
    Perceptron* p = new Perceptron(3, size, 256, 1);
    double* image = new double[size];

    for (int i{0}; i<png_image.get_width(); ++i)
        for (int j{0}; j<png_image.get_height(); ++j)
            image[i * png_image.get_width() + j] = png_image[i][j].red;
    double value = (*p)(image);
    std::cout << value << std::endl;

    delete p;
    delete [] image;
}
```



## 5. fejezet

# Helló, Mandelbrot!

### 5.1. A Mandelbrot halmaz

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/mandelpngt.c%2B%2B>

Első lépésként itt is fontos tisztázni hogy mi is az ugynevezett Mandelbrot halmaz . Ez gyakorlatilag egy a komplex szamsíkon talált halamaz . Felfedezését Benoit Mandelbrotnak köszönhetjük aki 1980ban találta meg . Tehetjük fel a kérdést hogy miért is fontosak számunkra a komplex szamsíkon talalható számok ! Hát azért mert rengeteg olyan matematika pelda es feladat van amelyekre egyéb számhalmazokban nem tudnánk választ vagy pontos választ adni. Ez az a számhalmaz ahol az értelmezhetetlen feladatok értelmezhetők es szinte minden problemat tudunk orvosolni. Mielott belekezdenünk a kódok világát böngészni fontos kiemelni hogy a program megfelelő futtatásához a linbpng es a linbpng++ illetve a png++ telepítése nem elhanyagolható ! Tehát ezek nelkul az eredményünk nem lesz valós ! Tehát a kódban lathatjuk hogy előtte meg kell adnunk a képünk méretét jelen esetben ez egy 600x600as kép lesz. Tehát a programban létrehozunk egy rácsot amelyen kesobb különboző pontokat veszünk fel. A programban megfigyelhetünk 2 ciklust amelynek jelentős szerepe lesz a programban. Ha jól megfigyeljük lathatjuk hogy ez a 2 ciklus megy végig és számolja ki a megfelelő irányokba . Ezután felfedezhetünk egy While ciklust is amely a rácspontok ellenorzsere szolgál . Ennek a segítséggel állapítja meg a program hogy az előbb kiszamolt racspont rajta van e az előtte definált szamsíkon. Ha rajta van akkor természetesen lementi az adatokat ha nincs rajta akkor pedig nyilvan továbbhalad . Ezzel a módszerrel folyamatosan számolja ki a racs egyre több és több pontját . Miutan ezzel vegzett és feltöltötte a kep pixeleit a program a már előtte megadott helyre létrehozza számunkra ezt a csodás képet. Ezt a halmazt úgy tudjuk létrehozni hogy egy 4 oldalhosszúságú négyzetben lefektetünk egy rácsot . Kiszámoljuk hogy a racs pontjai mely komplex számoknak felelnek meg .

Program T100

```
/ mandelpngt.c++
// 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 <iostream>
#include "png++/png.hpp"
#include <sys/types.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 elértük, 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;

}

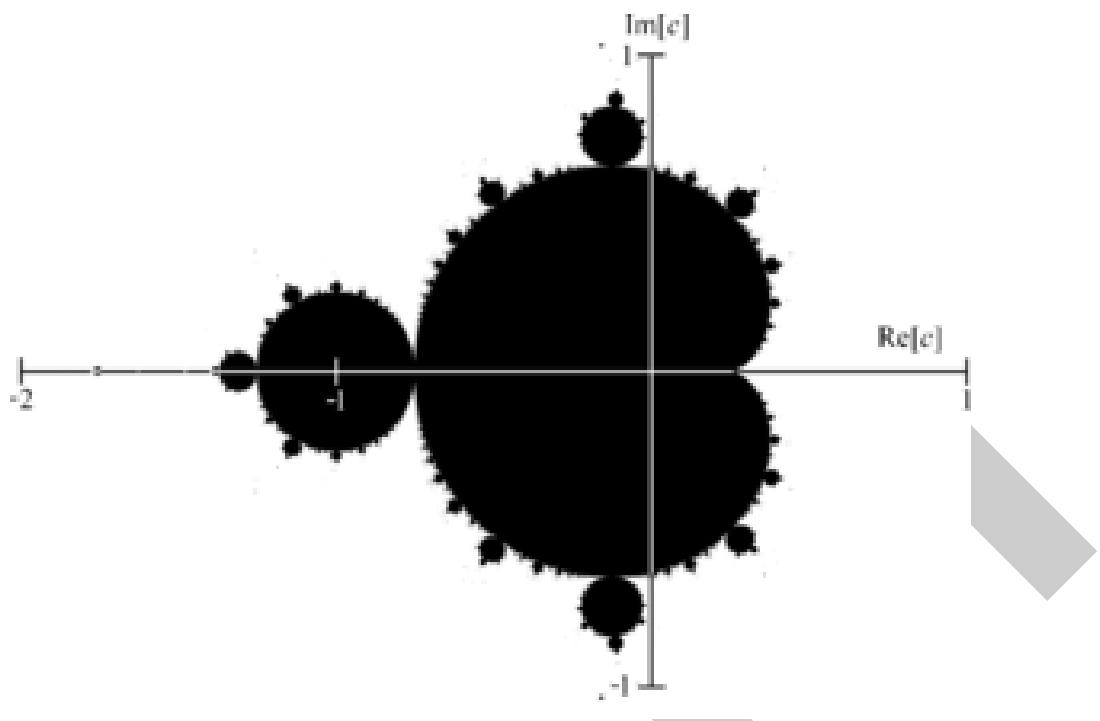
int
main (int argc, char *argv[])
{
    if (argc != 2)
    {
        std::cout << "Hasznalat: ./mandelpng fajlnev";
        return -1;
    }

    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;
}

}
```



5.1. ábra. Mandelbrot Forrás :Wikipedia:

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

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/mandel.cpp>

Tehát az előző feladatban már beszéltünk a Mandelbrot halmazról illetve a komplex szamsíkról . Itt külön nem emelném ki a történetét illetve a komplex szamsík és a halmaz jelentőségét. A programunk egyszerűen kifogja számolni az előzőhöz hasonlóan a mandelbrot halmazt viszont ebben az esetben az std:complex osztály fogja segítségük hívni a feladathoz . Tehát itt is kiemelném hogy ehez a programhoz is kellenek a már fentebb említett könyvtárak illetve egyéb eszközöt telepítése hisz nélküük ez a program se fog megfelelően lefutni. Szóval mivel itt külön az std:complex osztályt kell használnunk erdemessé válik ezt importálnunk a programba. Ahogy az előző feladatban ebben is előre meg kell adnunk azaz definialunk kell a kép adatik A program hasonló képpen mukodik mint az előző program tehát ebben az esetben is for ciklusok segítségével megy végig a pontokon , egy while ciklussal ellenorzi magat illetve tolta fel a jónak talált pontokat. Fontos kiemelni hogy a programnak van egy része amely folyamatosan kiírja számunkra hogy hany %nál jár a programunk. Végül de nem utolsó sorban szeretnem kiemelni hogy az előzővel ellentében az itt hasznalunk szinezést emiatt ez a csodás kép egy színes kép lesz. Ha minden jól csináltunk minden megfelelően leírtunk , deklaráltunk akkor a program lefutása utan itt is egy képpel lehetünk gazdagabbak valamelyik meghajtón .

```
Program T100
// 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 ←
0.79850575693437919611028519284457924366
// ./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 -1 --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 << "Hasznalat: ./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 < iteracionsHatar )
        {

            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;

}
}

```

### 5.3. Biomorfok

Megoldás videó: <https://youtu.be/IJMbqRzY76E>

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

Tehetjük fel hogy honnan is erednek a biomorfok . Hát gyakorlatilag egy Hiba , tévedés miatt találtak rá ezekre mint egyéb más dolgokra. Clifford Pickover volt az személy aki megpróbálta a Julia halmazokat kirajzolatni sajnos az alatta megírt program több helyen is hibákkal rendelkezik de szerencséjére ez hozta meg számára sikert hisz ezzel a programmal tudta kirajzolatni a biomorfok első "csapatát" Tehát itt is rajzolatni fogunk valamit kiemelném hogy itt is fontos a megfelelő dolgok telepítése hisz ezek nélkül megint rossz eredményhez juthatunk . Kiemelném a julia halmazt hisz ez egy a Mandelbrot halmazhoz hasonló halmaz azzal a kivetellel hogy ami a mandelbrót állandó az a Julia halmazban inkább változó . Tehát itt is egy szamolás fog következni az előző példához hasonlóan itt is rácspontokat fog a program számunkra szamolni . Kisebb-Nagyobb eltérésekkel s at fogjuk adni illetve ki fogjuk szamolni az adatokat illetve fel fogjuk tolteni a racspontokat . Miutan feltolttuk a racspontokat és megadtuk a valtozókat illetve a képünk méretét itt is egy PNG formátumú alkotással lehetünk boldogabbak .

```

Program T100
// 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 -1 --line-numbers=1 --left-footer=" ↵  
BATF41 HAXOR STR34M" --right-footer="https://bhaxor.blog.hu/" --pro= ↵  
color  
// ps2pdf 3.1.3.cpp.pdf 3.1.3.cpp.pdf.pdf  
//  
// 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;
    }

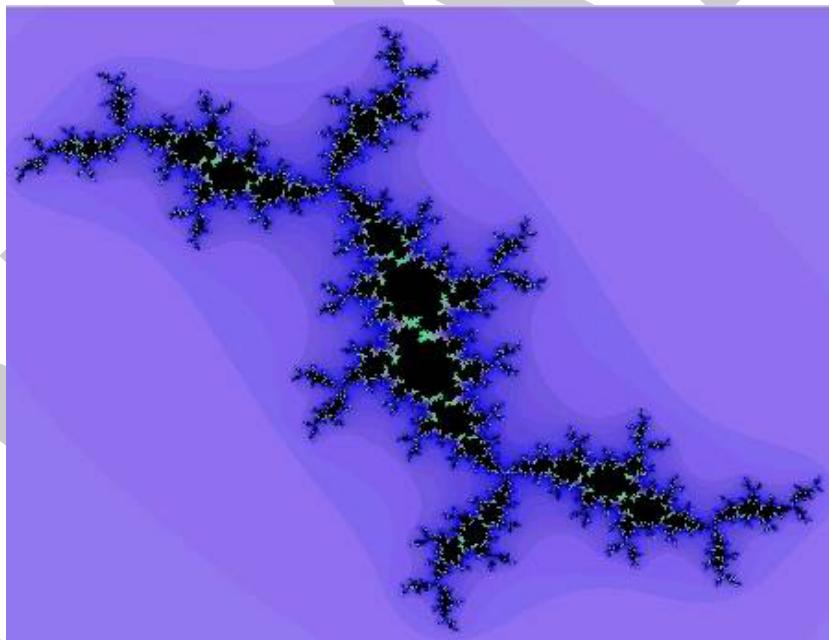
    kep.set_pixel ( x, y,
                    png::rgb_pixel ( (iteracio*20)%255, (iteracio ←
                        *40)%255, (iteracio*60)%255 ) );
}

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;

}

}
```



5.2. ábra. Julia halmaz Forrás :[www.t-es-t.h](http://www.t-es-t.h)

## 5.4. A Mandelbrot halmaz CUDA megvalósítása

Megoldás Forrása : <https://github.com/Samate99>Hello-world/blob/master/Prog1/cuda>

Hát ehez a feladathoz szükségünk lesz egy megfelelő grafikus kartyahoz amit az Nvidiank kell legyartani ellenkezo esetben nemhiszem hogy kepesek leszünk elvegezni a muveleteket Ebben az esetben is a Mandelbrot halmazt fogjuk szamolni viszont a egy elég érdekes számítási platformon, Tehat ahoz hogy a program lefusson ujabb Generaciós NVIDIA kartyara lesz szükségünk amely kepes a CUDA rendszer kezeléséhez Ahoz hogy ez mukodjon még ilyen korulmenyek között érdemes ellatogatni az NVIDIA oldalara ahol rengeteg uj informacióval és egyéb szoftverrel lehetünk gazdagabbak illetve szerintem érdemes 1-2 egyéb oldalról is szarmazó cikket elolvasni a CUDA lehetőségei kapcsan. Maga a Cuda egy elég érdekes az NVIDIA alltal létrehozott platform. Nepszeruseget annal koszonheti hogy ezzel nagyon nagy mertekben tudjuk kihasználni a videokartyank erőforrásait . Tehat a grafikus kartyank legtobb magyat szalat kiváló százalékban leszünk képesek kihasználni. Fontos kiemelni hogy ez a program hasonló elven mukodik . Itt is feltoltjuk a racspontokat a megfelelő szamításokkal illetve itt is kirajzoltatunk egy képet a számítógépünkre

Program T100

```
// mandelpngc_60x60_100.cu
// Copyright (C) 2019
// Norbert Bátfa, batfa.norbert@inf.unideb.hu
//
// This program is free software: you can redistribute it and/or modify
// at 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/types.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 < 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;

}

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);
}

void
cudamandel (int kepadat [MERET] [MERET])

{
```

```
int *device_kepadat;
cudaMalloc ((void **) &device_kepadat, MERET * MERET * sizeof (int));

// dim3 grid (MERET, MERET);
// mandelkernel <<< grid, 1 >>> (device_kepadat);

dim3 grid (MERET / 10, MERET / 10);
dim3 tgrid (10, 10);
mandelkernel <<< grid, tgrid >>> (device_kepadat);

cudaMemcpy (kepadat, device_kepadat,
            MERET * MERET * sizeof (int), cudaMemcpyDeviceToHost);
cudaFree (device_kepadat);

}

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

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

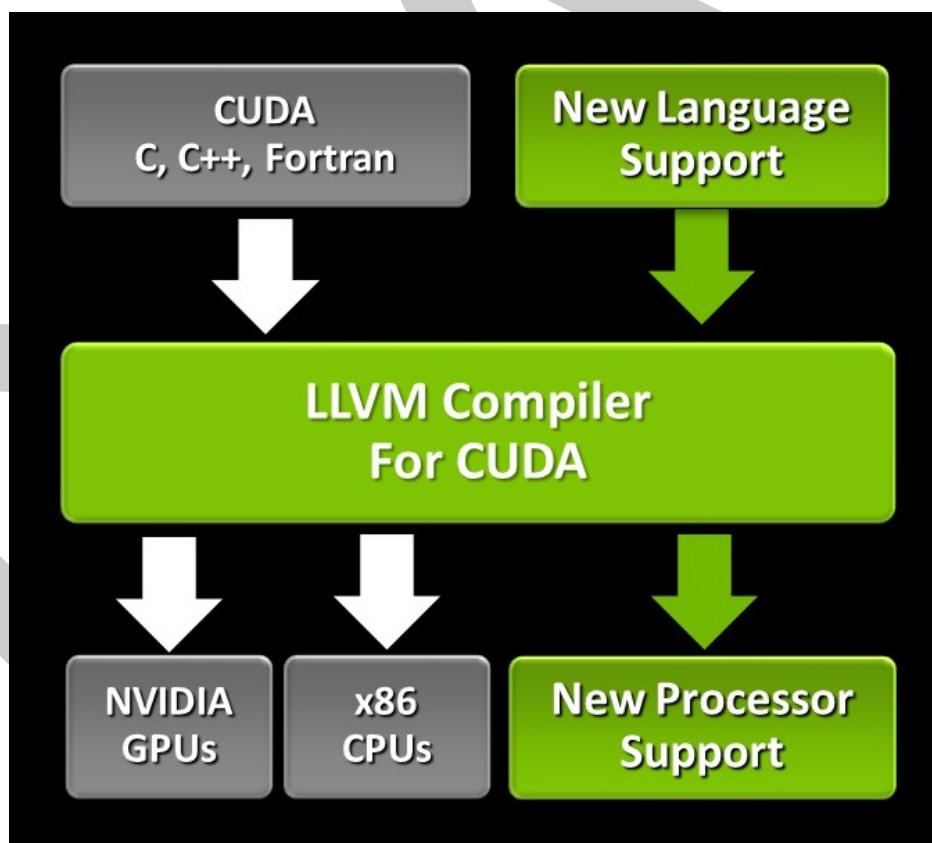
if (argc != 2)
{
    std::cout << "Hasznalat: ./mandelpngc fajlnev";
    return -1;
}

int kepadat [MERET] [MERET];
cudamandel (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;  
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;  
  
}  
}
```



5.3. ábra. CUDA Forrás: NVIDIA Developer

## 5.5. Mandelbrot nagyító és utazó C++ nyelven

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

Megoldás forrása: <https://sourceforge.net/p/udprog/code/ci/master/tree/source/labor/Qt/Frak/>

Megoldás videó:

A feladat megfelelő megoldásához szükségünk lesz több csomag telepítésére többek között a libqt4-dev csomagra is. Ahogy már ismerjek ezt folyamatot a qmake parancsal letre kell hoznunk a makefilet és mint ahogy eddig is tettük a Makefileos feladatok esetében le kell futtatnunk. Ha a telepítések és a futtatások sikeresek voltak 4 abalkot fogunk kapni. Mind a 4 képen ugyanaz a teknikai eljárás használata van, csak a részletek eltérőek. A képekkel kapcsolatos részben azonban a következőket kell előre meghallgatni: A mandelbrot halmaz kirajzolásában van szerepe van emlyik van amelyik csak osztályokat tartalmaz viszont a megfelelő működéshez az összesre van szükségünk.

## 5.6. Mandelbrot nagyító és utazó Java nyelven

<https://github.com/Samate99>Hello-world/blob/master/Prog1/nagyito.java>

Az előző feladathoz nagyon hasonló feladatot kaptunk. Az előzőhez hasonlóan ez is ki fog számunkra rajzolni egy képet. Illetve ez a kép is egy Mandelbrot halmaz lesz. Ebben az esetben is fontos a megfelelő fajlok, csomagok telepítése mert a program megfelelő futásához elengedhetetlenek. Magát a halmazt szinte ugyanúgy hozza létre mint a c++ esetében az eltérő változás a Zoomban teknikai eljárás használata. Az interaktív lesz. Az eger segítségével tudunk kijelölni részeket majd ezekből a részekből kapunk új képet. Az előzőhez hasonlóan itt is egy pillanatkeppel lehetünk gazdagabbak, de az előzővel ellentétben nem arról van szó hogy előre megadott képeket kapunk hanem mi választhatjuk ki hogy a mandelbrot halmaz eppen melyik szegletére vagyunk kíváncsiak.

Program T100

```
* MandelbrotHalmazNagyító.java
*
* DIGIT 2005, Javat tanítok
* Bátfai Norbert, nbatfai@inf.unideb.hu
*
*/
/***
 * A Mandelbrot halmazt nagyító és kirajzoló osztály.
 *
 * @author Bátfai Norbert, nbatfai@inf.unideb.hu
 * @version 0.0.1
*/

```

```
public class MandelbrotHalmazNagyító extends MandelbrotHalmaz {
```

```
/** A nagyítandó kijelölt területet bal felső sarka. */
private int x, y;
/** A nagyítandó kijelölt terület szélessége és magassága. */
private int mx, my;
/** Létrehoz egy a Mandelbrot halmazt a komplex síkon
 * [a,b]x[c,d] tartománya felett kiszámoló és nyígtani tudó
 * <code>MandelbrotHalmazNagyító</code> objektumot.
 *
 * @param a           a [a,b]x[c,d] tartomány a koordinátája.
 * @param b           a [a,b]x[c,d] tartomány b koordinátája.
 * @param c           a [a,b]x[c,d] tartomány c koordinátája.
 * @param d           a [a,b]x[c,d] tartomány d koordinátája.
 * @param szélesség   a halmazt tartalmazó tömb szélessége.
 * @param iterációsHatár a számítás pontossága.
 */
public MandelbrotHalmazNagyító(double a, double b, double c, double d,
                                  int szélesség, int iterációsHatár) {
    // Az ős osztály konstruktorának hívása
    super(a, b, c, d, szélesség, iterációsHatár);

    setTitle("A Mandelbrot halmaz nagyításai");

    // Egér kattintó elemények feldolgozása:

    addMouseListener(new java.awt.event.MouseAdapter() {

        // Egér kattintással jelöljük ki a nagyítandó területet
        // bal felső sarkát:

        public void mousePressed(java.awt.event.MouseEvent m) {

            // A nagyítandó kijelölt területet bal felső sarka:

            x = m.getX();
            y = m.getY();

            mx = 0;
            my = 0;

            repaint();
        }
    });
}
```

```
// Vonzolva kijelölünk egy területet...
// Ha felengedjük, akkor a kijelölt terület

// újraszámítása indul:

public void mouseReleased(java.awt.event.MouseEvent m) {
    double dx = (MandelbrotHalmazNagyító.this.b
                  - MandelbrotHalmazNagyító.this.a)
                /MandelbrotHalmazNagyító.this.szélesség;

    double dy = (MandelbrotHalmazNagyító.this.d
                  - MandelbrotHalmazNagyító.this.c)
                /MandelbrotHalmazNagyító.this.magasság;
    // Az új Mandelbrot nagyító objektum elkészítése:

    new MandelbrotHalmazNagyító(MandelbrotHalmazNagyító.this.a +
        x*dx,
        MandelbrotHalmazNagyító.this.a+x*dx+mx*dx,
        MandelbrotHalmazNagyító.this.d-y*dy-my*dy,
        MandelbrotHalmazNagyító.this.d-y*dy,
        600,
        MandelbrotHalmazNagyító.this.iterációsHatár);

}

} );

// Egér mozgás események feldolgozása:

addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {

    // Vonszolással jelöljük ki a négyzetet:
    public void mouseDragged(java.awt.event.MouseEvent m) {
        // A nagyítandó kijelölt terület szélessége és magassága:

        mx = m.getX() - x;
        my = m.getY() - y;
        repaint();
    }
});

}

/***
 * Pillanatfelvételek készítése.
 */

```

```
public void pillanatfelvétel() {
    // Az elmentendő kép elkészítése:
    java.awt.image.BufferedImage mentKép =
        new java.awt.image.BufferedImage(szélesség, magasság,
            java.awt.image.BufferedImage.TYPE_INT_RGB);

    java.awt.Graphics g = mentKép.getGraphics();
    g.drawImage(kép, 0, 0, this);
    g.setColor(java.awt.Color.BLUE);

    g.drawString("a=" + a, 10, 15);
    g.drawString("b=" + b, 10, 30);
    g.drawString("c=" + c, 10, 45);
    g.drawString("d=" + d, 10, 60);
    g.drawString("n=" + iterációsHatár, 10, 75);

    if(számításFut) {
        g.setColor(java.awt.Color.RED);
        g.drawLine(0, sor, getWidth(), sor);
    }
    g.setColor(java.awt.Color.GREEN);
    g.drawRect(x, y, mx, my);
    g.dispose();
    // A pillanatfelvétel képfájl nevének képzése:

    StringBuffer sb = new StringBuffer();
    sb = sb.delete(0, sb.length());
    sb.append("MandelbrotHalmazNagyitas_");
    sb.append(++pillanatfelvételszámLálo);
    sb.append("_");

    // A fájl nevébe belelevesszük, hogy melyik tartományban
    // találtuk a halmazt:
    sb.append(a);
    sb.append("_");
    sb.append(b);
    sb.append("_");
    sb.append(c);
    sb.append("_");
    sb.append(d);
    sb.append(".png");
    // png formátumú képet mentünk
    try {
        javax.imageio.ImageIO.write(mentKép, "png",
```

```
        new java.io.File(sb.toString())));
    } catch(java.io.IOException e) {
        e.printStackTrace();
    }
}

/***
 * A nagyítandó kijelölt területet jelző négyzet kirajzolása.
 */
public void paint(java.awt.Graphics g) {
    // A Mandelbrot halmaz kirajzolása
    g.drawImage(kép, 0, 0, this);

    // Ha éppen fut a számítás, akkor egy vörös
    // vonallal jelöljük, hogy melyik sorban tart:
    if(számításFut) {
        g.setColor(java.awt.Color.RED);
        g.drawLine(0, sor, getWidth(), sor);
    }
    // A jelző négyzet kirajzolása:
    g.setColor(java.awt.Color.GREEN);
    g.drawRect(x, y, mx, my);
}

/***
 * Példányosít egy Mandelbrot halmazt nagyító obektumot.
 */
public static void main(String[] args) {

    // A kiinduló halmazt a komplex sík [-2.0, .7]x[-1.35, 1.35]
    // tartományában keressük egy 600x600-as hálóval és az
    // aktuális nagyítási pontossággal:
    new MandelbrotHalmazNagyító(-2.0, .7, -1.35, 1.35, 600, 255);

}
}
```

## 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 video:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/polargen.java>

Tehát első ránézésre a polartranszformációs algoritmus talan illesztőnek tünhet viszont ha jobban beleolvassunk a programkódban akkor azért megnyugodhatunk. A polártranszformációval fogunk random számokat generálni amelyeket fel fogunk használni a program különboző részein! A program több nyelven is megtalálható jelen esetben en egy java nyelven írt kódot mutatok be. Szóval kezdjük is el nézni a programunkat szerintem ami egyből feltűnik a kódot olvasók számára hogy rendelkezünk egy PolarGen Classal aminek talán az a specifikus tulajdonsága hogy 2 reszből all Beszélhetünk egy Privat illetve egy publikus részről is. De akkor lessük is meg hogy hogyan dolgozik a program. A program megnezi hogy van e tarolt ertek ezután 2 felé válik a történet ha nem rendelkezünk akkor szamalonk megfelelő mennyiségű értéket. Az egyiket visszaadjuk a programunknak a másikra még később szükségünk lesz emiatt azt eltaroljuk. Tehát utana vegzunk par műveletet es egy ciklus segítségével kiiratjuk az adott szamu erteket, eredményt.

```
Program T100
import java.util.Random;
import java.io.*;
import java.lang.Math;
public class PolarGen {

    public final static int RAND_MAX = 32767;
    private static boolean bExists;
    private double dValue;
    static Random cRandomGenerator = new Random();

    public PolarGen() {
        bExists = false;
```

```
cRandomGenerator.setSeed(20);
};

public double PolarGet() {
    if (!bExists)
    {
        double u1, u2, v1, v2, w;
        do{
            u1 = cRandomGenerator.nextInt (RAND_MAX) / (RAND_MAX + 1.0); //innen ←
            œl jön az algoritmus
            u2 = cRandomGenerator.nextInt (RAND_MAX) / (RAND_MAX + 1.0);
            v1 = 2 * u1 - 1;
            v2 = 2 * u2 - 1;
            w = v1 * v1 + v2 * v2;
        }
        while (w > 1);
        double r = Math.sqrt ((-2 * Math.log (w)) / w);

        dValue = r * v2;
        bExists = !bExists;

        return r * v1; //idáig tart az algoritmus
    }
    else

    {
        bExists = !bExists; //ha van korábbi random érték, akkor azt adja ←
        vissza
        return dValue;
    }
};

public static void main(String args[])
{
    PolarGen cPolarGen = new PolarGen();
    double dEredmeny = cPolarGen.PolarGet();
    System.out.println(dEredmeny);

}
}
```



## 6.2. LZW

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

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/lzw.c>

Mi is az a binfa . Eddigi Egyetemi palyafutásom során talán a "legrémisztőbb" dolog . Még a bevprog idején

találkoztunk elsőnek hisz a bevprog védés nagyrészében rajta kellett kisebb változtatásokat létrehozni. . Talan feladata elsőre fel sem tünt de gyarkolatilag ez egy eleg jól mukodő veszteségmentes tomorítesi algoritmus . Nagyon meglepodtem mikor kiderült hogy ezt az algoritmust nagyon széles körben használjak szerte az informatika különbőző "ágazataiban" . Maga a binfa Terry Welchhez kothető ő volt aki egy előző úgynevezett LZ78as algoritmus továbbfejlesztéseként publikálta . Tehát mint fentebb említettem ez egy tomorítesi eljarás amelynek során a kódoló csak a szóbeli idexet kului ami egy nagyon ritka és erdekes dolog . Kiemelném hogy ebben az esetben a folyamat dinamikus A binfa megfelelő elkészítéséhez és futtatásához érdemes különbőző könytárákra includeolni . A futtatáshoz szükség lesz egy befilera illetve egy kifilera .

## 6.3. Fabejárás

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

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Fabe.cpp>

A binaris fák bejárásaiban alltalaban 3 módot v -in a -pre és a posztordet különböztetünk meg . A 3 mód között igazaból felépítésbeli eltérések talalhatóak Tehát gyakorlatilag a gyökér és ágak elhelyezkedésétől függ hogy éppen milyen módról beszélhetünk.ú Kiemelném hogy a program lefutásában illetve "viselkedésében" lényeges valtozas nem fedezhető fel. Inorder esetben a gyökér középen van az ágai pedig felette illetve alatta Preorder nél a gyökér felül van az ágai alatta postordernél a gyökér alul van ágai pedig felette.

Program T100

```
/inorder
void kiir (Csomopont * elem, std::ostream & os)
{ if (elem != NULL)
{
    ++melyseg;
    for (int i = 0; i < melyseg; ++i)
os << "___";
kiir ( elem->nullasGyermek (), os);
os << elem->getBetu () << "(" << melyseg - 1 << ")" << std::
endl;
kiir ( elem->egyesGyermek (), os);
--melyseg; }

}

//preorder
void kiir (Csmopont * elem, std::ostream & os)
{ if (elem != NULL)
{
    ++melyseg;
    for (int i = 0; i < melyseg; ++i)
os << "___";
os << elem->getBetu () << "(" << melyseg - 1 << ")" << std::
endl;
kiir ( elem->nullasGyermek (), os);
```

```
kiir ( elem->egyesGyermek (), os);
--melyseg; } }

//postorder
void kiir (Csomopont * elem, std::ostream & os)
{ if (elem != NULL)
{
    ++melyseg;
    for (int i = 0; i < melyseg; ++i) os
<< "----";
    kiir ( elem->nullasGyermek (), os);
    kiir ( elem->egyesGyermek (), os);
    os << elem->getBetu () << "(" << melyseg - 1 << ")" << std::
    endl;
    --melyseg; }
}

}
```

## 6.4. Tag a gyökér

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

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/lwzbinaf.cpp>

Na tehat ebben a reszben a feladatunk nagyon hasonló az előzőkhöz . Lényegében a mar fentebb említett C nyelven megírt binfa egy C++ változatáról beszélhetünk. A program mivel ugyanaz csak egy mas programozói nyelvben lett leírva ugyanazt csinalja mint a C nyelven írt binfa ugyanugy a veszteségmentes tomorításban van fontos szerepe. Erdemes megneznunk magat a kód felépítését. Ahogy lathatjuk a programnak 2 része van rendelkezünk egy privat illetve egy public résszel is . A fában átalakításokra is lesz szükség ebben az esetben a gyökér csomópont változásán meg át . Megfigyelhetjük a csomópontok épülését a fában . A programban továbbra lathatjuk a tagfüggvény tulterhelését . Utana megnezzük hogy az esetben eppen mivel kell dolgoznunk ketto lehetoseg van vagy 0-as vagy 1essel gyakorlatilag minden részben ugyanaz zajlik le. Nezzük meg az 1-essel szemleltetve. Megvizsgáljuk hogy van 1 es ha van akkor arra haladunk tovább ha nincs akkor letrehozunk egy egyest. A C++ ba áthelyezett program esetén is fontos a befile illetve a kifile használata futtatáskor.

## 6.5. Mutató a gyökér

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

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/lwzbinaf.cpp>

Tehát a feladatunkban továbbra is a binfat kell jobban tanulmanyoznunk illetve a binfaban kell különboző módosításokat létrehoznunk . A feladatunk hogy a gyökér ne kompozícióban legyen hanem aggregációban

. Ez a gyakorlatban azt jelenti hogy a fa egy részét át kell dolgoznunk illetve hogy megfelelő kapcsolat legyen azaz aggregaciós érdemes mutatókat használni . Viszont ha mutatókat használunk az több hibát is előhozhatunk kezdhetünk azzal hogy gyakorlatilag más szintaktikai elemeket kell alkalmaznunk illetve átjavítanunk hogy lefusson a program . A legelső lépés az lenne hogy a gyokeret a megfelelő szabalyok szerint atírjuk pointerre .

## 6.6. Mozgató szemantika

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

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/Prog1/z3a9.cpp>

A binfa témakör utolsó állomására értünk ebben a peldában a feladatunk a mozgató szemantika viselkedésének megvizsgálása lesz. A mozgató szemantika esetében arról beszélhetünk hogy letre kell hoznunk egy Operatort Az ő feladata lesz hogy a program lefutása során lemásolja magát . Természetesen egy adott metódus szerint történnek az események. A ujjepules irányat a rekurzív függvény határozza meg . A programban az 1es és a 0as gyermekek vizsgálatát ő fogja elvégezni . Ennek segítsével dönti el a program hogy merre haladjon . A program egyszeruen új csomopontokat hoz létre és lemasolja magát . Tehát maga az std::move nem vegez mozgatást

## 7. fejezet

# Helló, Conway!

### 7.1. Hangyszimulációk

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

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

Megoldás forrása: <https://github.com/Samate99>Hello-world/tree/master/Prog1/Myrmecologist>

Ez a szimuláció egy nagyon erdekes muvelet . Vegrehatjasahoz viszont több egyéb csomag , program telepítésére van szükség ilyen peladaul a libqt4 csomag amelyet telepítenünk kell . Miutan a telepítések lezajlottak kiemelném a qmake használatat , ennek segítségével fogjuk megkapni a makefilet ami a program futtatásához elengedhetetlen. Nézzük is meg a kódot ahogy lathatjuk a main függénnyen belül több helyen adatokat kell megadnunk illetve deklarálnunk kell . Lathatjuk hogy a program legalább 3 elkülöníthető osztállyal /resszel rendelekezik . Ant , Antwim és Anthread lesz ez a 3 osztály. Nézzük is meg őket egyesével nézzük meg hogy mi miért felelős . Tehát az Antwin lesz az osztály amelynek feladata a megjelenítés ebben tudjuk megadni az ablak tulajdonságait , méreteit , szélességét pixeleinek szamat stb.... Az Ant maga a virtualis "hangyak" tulajdonságait fogja tartalmazni . Tehát ebben a részben szerepelnek a hangyaink mozgasat , koordinálasat rögzítő beállítások . Itt lathatjuk a Hangyak mereteit , illetve mozgasuk elorre meghatarozott irányat . Vegul nezzük meg az Anthreadet hogy mit is tartalmaz . Az Anthread eleg sok dolgot tartalmaz . Igy első korben kiemelném hogy tartalmazza a szimulációban résztvevő hangyak számát, illetve tartalmazza a hangyak utvonalait , továbbá ebben a részben adhatjuk meg a feromon a feromon párolgásának mértékét és egyéb feromon adatokat amelynek segítségével képesek leszünk a hangyak "irányítására" Kiemelném hogy a programban több előre definált bind is található ilyen például a P amely feltételezésem szerint az angol PAUSE szóból eredhet . Lenyomásával termesztesn meg tudjuk állítani a folyamatot szüneteltetni tudjuk. A következő ilyen a Q lesz ez az angol QUIT szóból ered gondolom . Ennek lenyomásával kitudunk lejni a folyamatból.

### 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/Samate99>Hello-world/blob/master/Prog1/Sejtautomata.java>

A feladatunk az Élet jatekanak bemutatása .Maga az élet jateka John Hortony Conway kezei között született meg . Ő alkotta meg eme csodás világnak egyszerű szabalyait . Tehát hogy is épül fel ez a világ nezzük is meg . Ha jobban megnezzük a kódöt eszrevesszük hogy sejteket tartalmaznak szerintem mindenki szamara feltunó lehet hogy nem egyszerű sejtekéről van szó . A sejtek tulajdonságokkal rendelkeznek . Szamszerint kettővel Beszélhetünk élő és halott sejtekéről . Itt jönnek képben a fentebb már említett uriember nevéhez fűződő szabályok . Tehát egy sejt akkor gondolok , vélünk élő sejtnek ha van adott számú szomszedja ami esetünkben 2 szomszédnak kell lennie ahoz hogy a sejt eletben maradjon Ha a sejtünk nem rendelkezik szomszeddal akkor "elpusztul". De nem örökre Ha valamilyen módon kesobb urja szomszédokat fog kapni a sejtünk ujraéled . Tehát a kepernyőn pixeleket fogunk feldezeni . A 2 sarokból indulnak el a folyamatok és atlósan töltik meg a képet. A program futása során random sejt csoportok "agyu-golyók" fognak eletben maradni illetve meghalni , vegül ezekből kerülnek ki a kepeken latható alakzatok , mintak Fontos kiemelni hogy ebben a programban is több Bindről beszélhetünk Ilyen például az S az N vagy a G, Az S lenyomásával egy pillanatképpel lehetünk gazdagabbak az eseményről . Az N lenyomásával tudjuk noveli a sejtek méretét vegül pedig az I-G gombok segítségével fogunk tudni a folyamat lefolyásának gyorsaságára hatni.

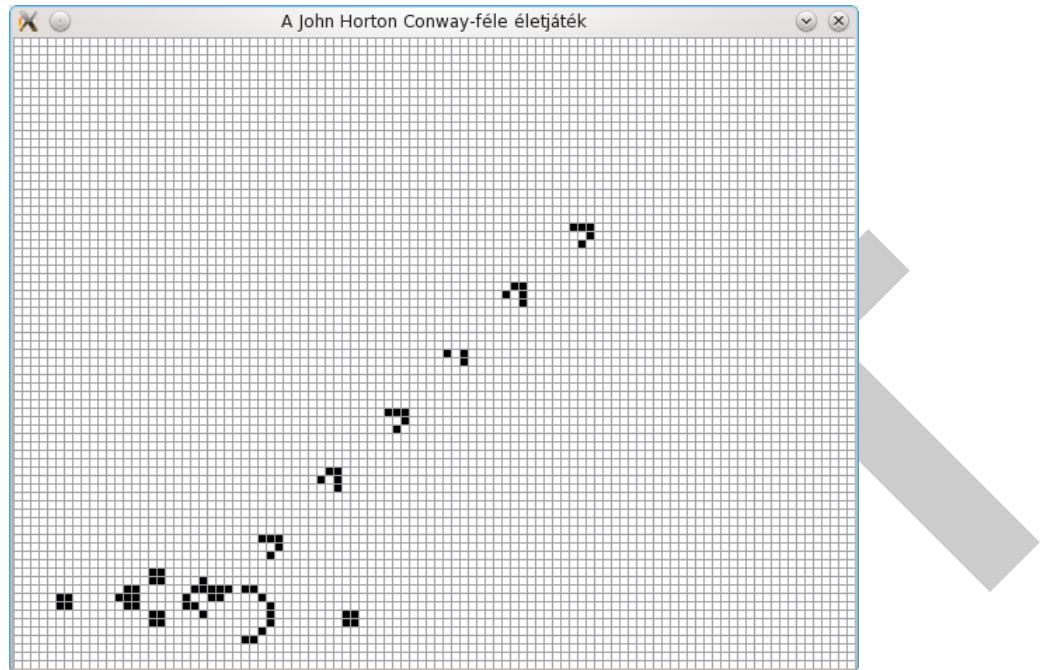
### 7.3. Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/tree/master/Prog1/sejtautomata>

Ez a pelda szinte teljesen ugyanaz legalábbis a feladat . Egyetlen valtozást az jelenti hogy ebben az esetben egy masik nyelvben kell dolgoznunk . A mi esetünkben ez a nyelv a nem a java hanem a c++ lesz . A program megfelelő mukodéséhez , lefutásához , lefordításához különböző szoftverek előtelepítésére . Itt is alkalmaznunk kell a qmake parancsot amelyel le tudjuk generalni az itt is eleg fontos makefilet. Ettol kezdve a program uyanugy mukodik mint a java kód . Tehát uyanugy megvannak sejtjeink uyanugy elindulnak a pixeleken uyanugy elhallnak illetve felélednek és így hozzá létre a dolgokat .



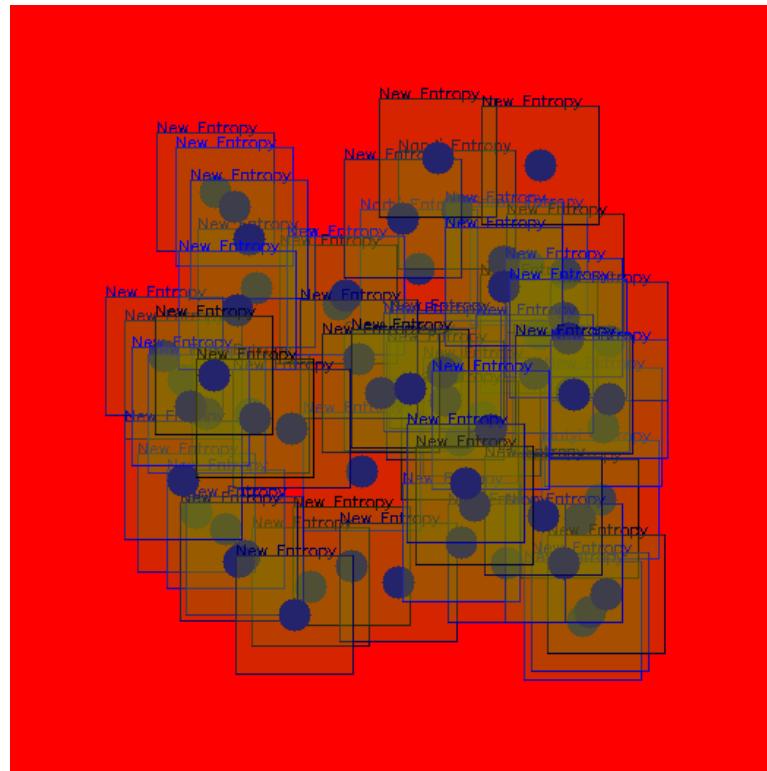
7.1. ábra. John Horton Conway-féle életjáték Forrás : Batfai Norbert

## 7.4. BrainB Benchmark

Megoldás videó:

<https://github.com/Samate99>Hello-world/tree/master/Prog1/brainB>

A BrainB Bechnmark talan szamomra a legerdekesebb program nagyon szeretem az esportot jelenleg is erősen csogozok es probalok benne eredményket elerni . Habar ez a program nem feltetlenül a cs-seknek szól inkabb a lolosok , dotasok szamara lehet hasznos . Tudjuk hogy ez a program a tehetséges esport jatekosok felfedezésére illetve a jelenleg is aktív jatekosok tesztelésére felmérése lehet hasznos . Megmondom oszinten nagyon erdekelne hogy egy Profi mezőnybe jatszó esportoló milyen eredményeket lenne képes elerni a jatekban. A programban az eger segítésével kell egy bizonyos pontban tartani az egeret miközben egyeb tenyezek jelen esetben negyzetek hatraltatnak . Ha jól haladunk a jatek szerint akkor egyre gyorsabb lesz illetve a negyzetek szama megnő ha elveszítjük "karakterünket" akkor lassul a mozgás . A program megfelelő használatahoz telepítenünk kell egyéb programokat peldaúl a OpenCV-t illetve a libqt-t Ha ezt telepítettük mar csak el kell indítani a programot es elkezdhetjük tesztelni a kepessegeinket es akar statisztikakkal is alatamaszthatjuk a bennünk rejlő potencialt .



7.2. ábra. BrainB Benchmark : Batfai Norbert

## 8. fejezet

# Helló, Schwarzenegger!

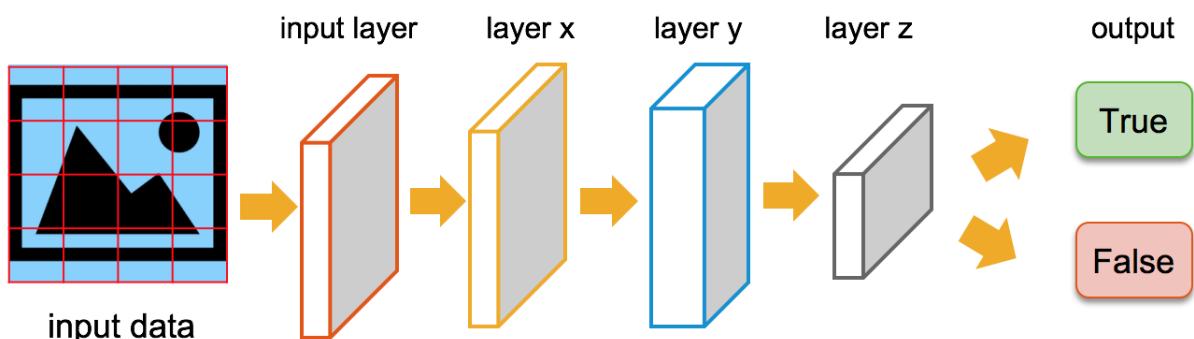
### 8.1. Szoftmax Py MNIST

aa Python

Megoldás videó: <https://github.com/tensorflow/tensorflow/releases/tag/v0.9.0>

Megoldás forrása:

Ez a program szerintem a konyv egyik legérdekesebb programja mukodéséhez 2 elengedhetetlen dolog telepítésére van szükség egyfelől szükség lesz egy adatbazisra amit a tensorflowal érhetünk ell illetve egy fejlesztői környezetre ami a mi esetünkben a Python lesz . Talan ez az a nyelv amelyel eddigi tanulmányaim során a legkevesebbet foglalkoztam de szobatársamnak hála szerencsére ezzel is sikerült talalkoznom . Miutan telepítettük ezt a 2 dolgot kapunk egy adatbazist , Egy adazbazist ami rengeteg képet tartalmaz .Itt tehetjük fel a kérdést hogy mire is fogjuk használni ezt a rengeteg képet. Ezekkel a képekkel fogjuk megtanítani elemzni a programunkat . A program az adatbazisból rengeteg képpel talalkozik , ezzel megtanulja felismerni a kepeket . Rengeteg keppel fogjuk megtanítani 1-1 dologra és további képekkel fogjuk ellenorizni hogy sikerült-e a megfelelő "oktatas" . Ha minden jól csináltunk és sikerült a dolog akkor ha az előző képekez hasonló . Tehát ugyanazt a dolgot abrazoló képet fogunk mutatni a programunk szamara a program képes lesz arra hogy felismerje es a megfelelő kategóriahoz sorolja a kepet .



8.1. ábra. TensorFlow Forrás:SAP Blogs

## 8.2. Szoftmax R MNIST (Passz)

R

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 8.3. Minecraft-MALMÖT(Passz)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

## 9. fejezet

# Helló, Chaitin!

### 9.1. Iteratív és rekurzív faktoriális Lisp-ben(Passz)

Megoldás videó:

Megoldás forrása:

### 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](https://youtu.be/OKdAkI_c7Sc)

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

A program megfelelő működéséhez érdemes a GIMP megfelelő változatát használni. Jelen esetben én a Gimp 2.10es verzióját használtam. Tehát mit is kell csinalunk... Azt kell tennünk hogy a programunkban meg kell adnunk az alap feltételeket tehát a betutipust a szöveget a kép méretét a szint es sok egyéb informaciót. Tudni kell hogy maga a GIMP az egyik legnepszerubb szoftver illetve hogy nagyon felhasználó barát. Rendkívül jól kezeli a scripteket , kulon mappa is van rá hogy kifejezetten ide rakjuk be őket . Tehát ez a feladatunk most is . Gyakorlatilag a programunk forráskódjat be kell másolnunk a program forráskódjak a GIMP "Scripts" mappájába . Ezutan mar csak par kattintásra van szükségünk hogy elkészítsük a képet Gyakorlatilag a program úgy működik hogy a program forráskódját be kell másolnunk a Gimp "scripts" mappájába. A programunkban meg kell adnunk a szöveget , a szöveg betűtipust, a szöveg/kép méretet , a szint a szöveg szinet és egyéb információkat.

```
; bhax_chrome3.scm
;
; BHAX-Chrome creates a chrome effect on a given text.
; Copyright (C) 2019
; Norbert Bátfai, batfai.norbert@inf.unideb.hu
; Nándor Bátfai, batfai.nandi@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 <https://www.gnu.org/licenses/>.
;
; Version history
;
; This Scheme code is partially based on the Gimp tutorial
; http://penguinpeta.com/b2evo/index.php?p=351
; (the interactive steps of this tutorial are written in Scheme)
;
; https://bhaxor.blog.hu/2019/01/10/ ←
; a_gimp_lisp_hackelese_a_scheme_programozasi_nyelv
;

(define (color-curve)
  (let*
    (
      (tomb (cons-array 8 'byte))
    )
    (aset tomb 0 0)
    (aset tomb 1 0)
    (aset tomb 2 50)
    (aset tomb 3 190)
    (aset tomb 4 110)
    (aset tomb 5 20)
    (aset tomb 6 200)
    (aset tomb 7 190)
  tomb)
)

; (color-curve)

(define (elem x lista)
  (if (= x 1) (car lista) (elem (- x 1) (cdr lista) ) ) )

(define (text-wh text font fontsize)
(let*
  (
    (text-width 1)
```

```
(text-height 1)
)

(set! text-width (car (gimp-text-get-extents-fontname text fontsize ←
    PIXELS font)))
(set! text-height (elem 2 (gimp-text-get-extents-fontname text ←
    fontsize PIXELS font)))

(list text-width text-height)
)
)

; (text-width "alma" "Sans" 100)

(define (script-fu-bhax-chrome text font fontsize width height color ←
    gradient)
(let*
    (
        (image (car (gimp-image-new width height 0)))
        (layer (car (gimp-layer-new image width height RGB-IMAGE "bg" 100 ←
            LAYER-MODE-NORMAL-LEGACY)))
        (textfs)
        (text-width (car (text-wh text font fontsize)))
        (text-height (elem 2 (text-wh text font fontsize)))
        (layer2)
    )
    ;step 1
    (gimp-image-insert-layer image layer 0 0)
    (gimp-context-set-foreground '(0 0 0))
    (gimp-drawable-fill layer FILL-FOREGROUND)
    (gimp-context-set-foreground '(255 255 255))

    (set! textfs (car (gimp-text-layer-new image text font fontsize PIXELS) ←
        ))
    (gimp-image-insert-layer image textfs 0 0)
    (gimp-layer-set-offsets textfs (- (/ width 2) (/ text-width 2)) (- (/ ←
        height 2) (/ text-height 2)))

    (set! layer (car (gimp-image-merge-down image textfs CLIP-TO-BOTTOM- ←
        LAYER)))
    ;step 2
    (plug-in-gauss-iir RUN-INTERACTIVE image layer 15 TRUE TRUE)

    ;step 3
    (gimp-drawable-levels layer HISTOGRAM-VALUE .11 .42 TRUE 1 0 1 TRUE)

    ;step 4
    (plug-in-gauss-iir RUN-INTERACTIVE image layer 2 TRUE TRUE)
```

```
;step 5
(gimp-image-select-color image CHANNEL-OP-REPLACE layer '(0 0 0))
(gimp-selection-invert image)

;step 6
(set! layer2 (car (gimp-layer-new image width height RGB-IMAGE "2" 100 ←
    LAYER-MODE-NORMAL-LEGACY)))
(gimp-image-insert-layer image layer2 0 0)

;step 7
(gimp-context-set-gradient gradient)
(gimp-edit-blend layer2 BLEND-CUSTOM LAYER-MODE-NORMAL-LEGACY GRADIENT-←
    LINEAR 100 0 REPEAT-NONE
    FALSE TRUE 5 .1 TRUE width (/ height 3) width (- height (/ height ←
        3)))
    )

;step 8
(plug-in-bump-map RUN-NONINTERACTIVE image layer2 layer 120 25 7 5 5 0 ←
    0 TRUE FALSE 2)

;step 9
(gimp-curves-spline layer2 HISTOGRAM-VALUE 8 (color-curve))

(gimp-display-new image)
(gimp-image-clean-all image)
)
)

;(script-fu-bhax-chrome "Bátf41 Haxor" "Sans" 120 1000 1000 '(255 0 0) "←
    Crown molding")

(script-fu-register "script-fu-bhax-chrome"
    "Chrome3"
    "Creates a chrome effect on a given text."
    "Norbert Bátfai"
    "Copyright 2019, Norbert Bátfai"
    "January 19, 2019"
    ""
    SF-STRING      "Text"      "Bátf41 Haxor"
    SF-FONT        "Font"       "Sans"
    SF-ADJUSTMENT  "Font size" '(100 1 1000 1 10 0 1)
    SF-VALUE        "Width"     "1000"
    SF-VALUE        "Height"    "1000"
    SF-COLOR        "Color"     '(255 0 0)
    SF-GRADIENT    "Gradient"  "Crown molding"
)
(script-fu-menu-register "script-fu-bhax-chrome"
    "<Image>/File/Create/BHAX"
)
```

{

### 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\\_hackelete\\_a\\_scheme\\_programozasi\\_nyelv](https://bhaxor.blog.hu/2019/01/10/a_gimp_lisp_hackelete_a_scheme_programozasi_nyelv)

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

Tehát az előző feladathoz hasonlóan ujra a GIMPel kell alkotnunk jelen helyzetben egy manadalát fogunk készíteni . Szerintem elsőnek fontos tisztázni a mandala jelentését a mandala egy kep amelyekkel a hindu es buddhista vallásban világszemléletben ábrázolnak dolgokat értek ezalatt isteneket vagy barmiféle szamukra szent vagy fontos dolgot. A mostani esetben is fontos a GIMP megfelelő verzióját használni a biztos lefutás érdekében . A GIMP szerencsénkre nagyon könnyedén használja a scripteket , saját mappával rendelkezik hogy még könnyebben tudjunk dolgozni vele. Tehát a scriptunket az előző feladathoz hasonlóan itt is bele kell raktunk a a GIMP "Scripts" mappájába Mivel a GIMP egy csodás program és a sripteket könnyen és gördülékenyen kezeli nincs is más dolgunk csak a scripts mappába be kell más Az elozo feladathoz hasonlóan itt is meg kell adnunk még a programban az adakat. Tehát meg kell adnunk a Szöveget ,a szöveg méretét méretet ,a kep azaz a mandala méretét , a szöveg betűtípusát , színt a szinkonbinaciót , es egyeb adatokat. Ha ezt megfelelően adtuk meg es minden klappol akkor a program futása utan egy sajat mandalával lehetünk gazdagabbak .

```
; bhax_mandala9.scm
;
; BHAX-Mandala creates a mandala from a text box.
; 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
;
; This Scheme code is partially based on the Python code
; Pat625_Mandala_With_Your_Name.py by Tin Tran, which is released under ←
; the GNU GPL v3, see
```

```
; https://gimplearn.net/viewtopic.php/Pat625-Mandala-With-Your-Name-Script ←
; -for-GIMP?t=269&p=976
;
; https://bhaxor.blog.hu/2019/01/10/ ←
; a_gimp_lisp_hackelese_a_scheme_programozasi_nyelv
;

(define (elem x lista)

  (if (= x 1) (car lista) (elem (- x 1) (cdr lista) ) )

)

(define (text-width text font fontsize)
(let*
  (
    (
      (text-width 1)
    )
    (set! text-width (car (gimp-text-get-extents-fontname text fontsize ←
      PIXELS font))) 

    text-width
  )
)

(define (text-wh text font fontsize)
(let*
  (
    (
      (text-width 1)
      (text-height 1)
    )
    ;;;
    (set! text-width (car (gimp-text-get-extents-fontname text fontsize ←
      PIXELS font)))
    ;;; ved ki a lista 2. elemét
    (set! text-height (elem 2 (gimp-text-get-extents-fontname text ←
      fontsize PIXELS font)))
    ;;

    (list text-width text-height)
  )
)

; (text-width "alma" "Sans" 100)

(define (script-fu-bhax-mandala text text2 font fontsize width height color ←
  gradient)
(let*
  (
    
```

```
(image (car (gimp-image-new width height 0)))
  (layer (car (gimp-layer-new image width height RGB-IMAGE "bg" 100 ←
    LAYER-MODE-NORMAL-LEGACY)))
    (textfs)
    (text-layer)
    (text-width (text-width text font fontsize))
    ;;
    (text2-width (car (text-wh text2 font fontsize)))
    (text2-height (elem 2 (text-wh text2 font fontsize)))
    ;;
    (textfs-width)
    (textfs-height)
    (gradient-layer)
  )
(gimp-image-insert-layer image layer 0 0)

(gimp-context-set-foreground '(0 255 0))
(gimp-drawable-fill layer FILL-FOREGROUND)
(gimp-image-undo-disable image)

(gimp-context-set-foreground color)

(set! textfs (car (gimp-text-layer-new image text font fontsize PIXELS) ←
  ))
(gimp-image-insert-layer image textfs 0 -1)
(gimp-layer-set-offsets textfs (- (/ width 2) (/ text-width 2)) (/ ←
  height 2))
(gimp-layer-resize-to-image-size textfs)

(set! text-layer (car (gimp-layer-new-from-drawable textfs image)))
(gimp-image-insert-layer image text-layer 0 -1)
(gimp-item-transform-rotate-simple text-layer ROTATE-180 TRUE 0 0)
(set! textfs (car (gimp-image-merge-down image text-layer CLIP-TO-BOTTOM ←
  -LAYER)))

(set! text-layer (car (gimp-layer-new-from-drawable textfs image)))
(gimp-image-insert-layer image text-layer 0 -1)
(gimp-item-transform-rotate text-layer (/ *pi* 2) TRUE 0 0)
(set! textfs (car (gimp-image-merge-down image text-layer CLIP-TO-BOTTOM ←
  -LAYER)))

(set! text-layer (car (gimp-layer-new-from-drawable textfs image)))
(gimp-image-insert-layer image text-layer 0 -1)
(gimp-item-transform-rotate text-layer (/ *pi* 4) TRUE 0 0)
(set! textfs (car (gimp-image-merge-down image text-layer CLIP-TO-BOTTOM ←
  -LAYER)))

(set! text-layer (car (gimp-layer-new-from-drawable textfs image)))
(gimp-image-insert-layer image text-layer 0 -1)
```

```
(gimp-item-transform-rotate text-layer (/ *pix* 6) TRUE 0 0)
(set! textfs (car(gimp-image-merge-down image text-layer CLIP-TO-BOTTOM ←
-LAYER)))

(plug-in-autocrop-layer RUN-NONINTERACTIVE image textfs)
(set! textfs-width (+ (car(gimp-drawable-width textfs)) 100))
(set! textfs-height (+ (car(gimp-drawable-height textfs)) 100))

(gimp-layer-resize-to-image-size textfs)

(gimp-image-select-ellipse image CHANNEL-OP-REPLACE (- (- (/ width 2) ←
(/ textfs-width 2)) 18)
(- (- (/ height 2) (/ textfs-height 2)) 18) (+ textfs-width 36) (+ ←
textfs-height 36))
(plug-in-sel2path RUN-NONINTERACTIVE image textfs)

(gimp-context-set-brush-size 22)
(gimp-edit-stroke textfs)

(set! textfs-width (- textfs-width 70))
(set! textfs-height (- textfs-height 70))

(gimp-image-select-ellipse image CHANNEL-OP-REPLACE (- (- (/ width 2) ←
(/ textfs-width 2)) 18)
(- (- (/ height 2) (/ textfs-height 2)) 18) (+ textfs-width 36) (+ ←
textfs-height 36))
(plug-in-sel2path RUN-NONINTERACTIVE image textfs)

(gimp-context-set-brush-size 8)
(gimp-edit-stroke textfs)

(set! gradient-layer (car (gimp-layer-new image width height RGB-IMAGE ←
"gradient" 100 LAYER-MODE-NORMAL-LEGACY)))

(gimp-image-insert-layer image gradient-layer 0 -1)
(gimp-image-select-item image CHANNEL-OP-REPLACE textfs)
(gimp-context-set-gradient gradient)
(gimp-edit-blend gradient-layer BLEND-CUSTOM LAYER-MODE-NORMAL-LEGACY ←
GRADIENT-RADIAL 100 0
REPEAT-TRIANGULAR FALSE TRUE 5 .1 TRUE (/ width 2) (/ height 2) (+ (+ (/ ←
width 2) (/ textfs-width 2)) 8) (/ height 2))

(plug-in-sel2path RUN-NONINTERACTIVE image textfs)

(set! textfs (car (gimp-text-layer-new image text2 font fontsize PIXELS ←
)))
(gimp-image-insert-layer image textfs 0 -1)
(gimp-message (number->string text2-height))
(gimp-layer-set-offsets textfs (- (/ width 2) (/ text2-width 2)) (- (/ ←
height 2) (/ text2-height 2)))
```

```
; (gimp-selection-none image)
; (gimp-image-flatten image)

(gimp-display-new image)
(gimp-image-clean-all image)
)
)

; (script-fu-bhax-mandala "Bátfai Norbert" "BHAX" "Ruge Boogie" 120 1920 ←
 1080 '(255 0 0) "Shadows 3")

(script-fu-register "script-fu-bhax-mandala"
  "Mandala9"
  "Creates a mandala from a text box."
  "Norbert Bátfai"
  "Copyright 2019, Norbert Bátfai"
  "January 9, 2019"
  ""
  SF-STRING      "Text"      "Bátf41 Haxor"
  SF-STRING      "Text2"     "BHAX"
  SF-FONT        "Font"      "Sans"
  SF-ADJUSTMENT   "Font size" '(100 1 1000 1 10 0 1)
  SF-VALUE        "Width"     "1000"
  SF-VALUE        "Height"    "1000"
  SF-COLOR        "Color"     '(255 0 0)
  SF-GRADIENT    "Gradient"  "Deep Sea"
)
(script-fu-menu-register "script-fu-bhax-mandala"
  "<Image>/File/Create/BHAX"
)
}
```



# 10. fejezet

## Helló, Gutenberg!

### 10.1. Programozási alapfogalmak

[?]

Tehát eljutottunk a konyunk utolsó részéhez . Ebben a részben a feladatunk egy olvasónapló kidolgozása lesz. Az olvasó napló 3 könyvből fog összetevődni. Az első köny a méltán híres Juhasz Istvan kezei közül kikerült alkotás lesz amely a Magas Szintű programozási nyelvek 1 címet viseli. Kiemelném Juhasz Istvan munkassagát hisz meghatározó alakja volt a Debreceni Egyetemi informatika oktatásban illetve A diákok a mai napig emlegetik feltöltött jegyzeteit használjak , hasznosítjak ami nem sok mamár nem tanító oktatóról mondható el véleményem szerint. A masodik alkotás Benedek Zoltán: Szoftverfejlesztés C++ nyelven című könyv lesz részemrol ez is egy igen erdekes olvasmany volt hasonlóan a harmadik kincshez ami Brian W.Kernighan és Dennis M. Ritchie . The C programming lanugage nevet viseli. Tehát akkor kezdjük is feldolgozni a könyveket egyesével . Tehát az első a Juhasz Istvan féle könyv ez egy oldalas alkotás . Az alkotásban több részt különböztethetünk meg . A mu első felében megismerkedhetünk az alapfogalmakkal . A programozási nyelvek 3 szintjevel a gépi nyelvel az assaembly nyelvel és az alltalunk ismert és használt magas szintű programozási nyelvekkel . Tehát nezzük is meg miben kulonboznek ezek a nyelvek mire jók mire használjuk vagy eppen nem használjuk oket. A gépi nyelv az a processzor számára érthető nyelv . minden processzor gépi nyelvel rendelkezik . Ez egy rendkívül bonyolult illetve összetett nyelv . Emiatt nem is tanuljuk . Viszont rendkívül sokat foglalkozunk vele hisz az összes Magas szintű nyelven megírt program vagy ugynevezett forrásprogram erre a nyelvre kerül lefordításra . Tehát amikor mi lefordítunk egy programot abban az esetben az alltalunk leírt magas szintű nyelv atkerül gépi nyelvé . Es így képes lefutni . A szovegben olvashatunk a fordításról tudjuk hogy ez egy nagyon fontos feladat hisz ez kódolja át a masik nyelvre. Fontos kiemelni hogy a program vegignezi hogy mit használtunk es hibakat keres a kódolasunkban ez azért fontos mert így konnyeden megkapjuk a hibakat , kiemelném hogy a program egy apró hiba esetén is visszadobja a kódot tehát ha csak egy kis hiba van akkor se fordul le. Ez azért van mert a program egészét nézi nem pedig sorokra vagy részekre bontva fordítja a programot. Az alkotásban továbbá olvashatunk a kulonbozo magasszintű programozasi nyelvek különböző szabalyairól illetve alap elgondolásainkról. Folytatásként akkor nezzük meg a programozási nyelvek alapelemeit .Ki nem találná mi is a legkisebb építő elem egy programban . Ez az építő elem nem más mint a karakterek . Mik is azok a karakterek. A karaktereként elnevezve elég sok minden tartunk számon kezdve a betűkkel es számokkal ugynevezett specialis karakterekkel. Specialis karakter néven értem a vesszőt nyilakat kulonböző zárójeleket . Betükkel is kiemelném hisz rengeteg beturol beszélhetünk hisz barmely nyelv betuit hasznalhatjuk a kódunkban tehát karakterek minősülnek a kínai íras elemei a ciril betük és természetesen az alltalunk ismert betük . Fontos kiemelni

nem minden fordító és programnyelv képes minden karaktert megfelelően kezelni több program is megkuzzd az ékezetekkel és egyéb karakterekkel . Kivételt élveznek ezalól a kommentbe írt dolgok mivel ezek az kód olvasójához szolnak , nincs közül a fordításhoz nem fordítja la őket a program . Fontos hogy vannak olyan szovegek karakterek szavak amelyeknek az eppen írt nyelvben rendelkeznek kulonbozó tulajdonsaggal. Tehát előre megadott dolgokat hívnak eletbe ilyen peldaul az IF a for a while vagy akar a case. elég sok eszközöt használunk és többet is kiemelnék illetve kiemel a könyv . Ilyenek peldaul a kontstansok amelyek segítségével egy fix értéket rakhattuk a kódba vagy ilyenek a nevesített konstansok amelyek segítségével ugyanugy értékeket tudunk megadni . Ezt a megadási módot deklarásának nevezzük itt érdemes szót ejteni az adattipusokról . Eleg sok adattipus van amelyeket a megfelelő modon kell használnunk beszélhetünk int double char vagy akar a tombi is , és egyéb tipusokról . Ezek előre megszabalyozzák hogy mire fogjuk használni az adatot . Ezeken felül a könyv szerint egy program írása kozben rengeteg dologgal talalkozhatunk .Találkozhatunk ugynevezett kifejezésekkel ilyenek peldaul az operandusok operatorok Ezekhez a részhez tartoznak a zarójelek egy része amelyeket rengetegszer használunk egy progarm írásánál. A következő nagy rész az utasítások része lesz . Ebben a részben találkozhatunk az utasítások 2 csoportjával a deklarációval es a vegrehajtásos utasítással. Nos akkor mi is ak kulonbség ekozott a 2 utasítási rendszer között . Az első esetében a fordításkor használt fordítóprogrammal való szerepe miatt fontos. Ez a fajta utasítás befolyásolja a fordítórprogram fordítását. A masodik viszont a kódolás során nyilvanul meg ezekkel az utasításokkal mar mindannyian találkoztunk ilyenek peldaul az ertekadó utasítások és ciklusszervező , az egyéb utasítások.

## 10.2. Programozás bevezetés

[KERNIGHANRITCHIE]

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

A következő számunkra fontos konyv A Brian W.Kernighan és Dennis M. Ritchie . The C programming lanugage nevet viseli.Ebben a konyvben a C nyelv sajátosságaival fogunk megismerkedni.Így kezdésként fontos kiemelni hogy a C nyelv maga rendkívül sok specialis tulajdonsággal rendelkezik nezzük meg akart a vesszőt ha egy sort vagy több sort egy vessző követ akkor alltalaban annak elvalasztasi funkciót adunk vele menyem szerint. A C nyelvben való kódoláskor nem érdemes ilyet használni mivel könnyen lefordulási problémakkal találkozhatunk . Mivel ha egy sort vesszővel zarunk az itt utasításnak minősül . Ebben a nyelvben kifejezzen fontos a kapcsos zarójelek megfelelő használata mert ez is könnyen hibahoz vezethet. Tovább fontos kiemelni hogy több dologban megegyezik a többi nyelvek ugyanugy használhatunk megfelelő ciklusokat ugyanugy Ifel tudunk választ kérni / adni ugyanugy tudunk Switchet alkalmazni hogy elágazásokat vezessünk be. A konyvben külön szóba kerülnek a ciklusok illetve a ciklusok alkalmazása . Hisz tudjuk hogy a különböző ciklusok között (while, do-while , for) akkora kulonbség nincs tehát ugyanugy feltételt kell megadni. Talán ami fontosabbnak tekinthető hogy van olyan fajta ciklus amely 1x mindenfeleplekken lefut illetve van olyan amelynek ha nincs megfelelő feltétel egyszer sem fut le. A C++ ban mar megtanult cout helyett Cben a printF függvényt kell használnunk hogy a megfelelő dolgokat kírassuk az outputra. Így összességében a C nyelv egy nagyon érdekes nyelv amely talan az egyik nehezebb de vele menyem szerint ez ízlések és pofonok hogy eppen ki miben szeret és akar vagy eppen miben köteles programozni.

## 10.3. Programozás

[BMECPP]

A harmadik és egyben utolsó könyv a Benedek Zoltán: Szoftverfejlesztés C++ nyelven címet viseli Az alkotás fő témája a c++ nyelv amely egy nagyon fontos nyelvé nőtte ki magát napjainkra . Tehat a Benedek Zoltán Féle "BME-S" Könyvben rengeteg C++ információval lehetünk gazdagabbak . A konyvben megismerhetjük a Függvényparaméterek megfelelő mukodését meglathatjuk , Megismerhetjük hogy mikre eppen mit ad vissza. Megismerhetjük hogy a c++ nyelvben a függvényeket mi azonosítja tehát megtudjuk hogy ez a dolog a nevük és az argumentumlistajuk. Tehát a C kóddal ellentétben C++ ban lehet 2 függvény azonos névvel ha az argumentumlistajuk különbozo. Megismerkedhetünk a c++ memóriakezelési képességeivel . Azaz hogy ebben az esetben ezt a feladatot Operátorok látják el Cvel ellentétben ahol függvény végzik ugyanezt a dolgot . A c++ rengeteg kiváló tulajdonságát lathatjuk meg a konyvben ilyen peldaúl a hibakezelése ami sokkal atlathatóbb és talan konnyebb is mint a Cnek

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. OO szemlélet

A módosított polártranszformációs normális generátor beprogramozása Java nyelven. Mutassunk rá, hogy a mi természetes saját megoldásunk (az algoritmus egyszerre két normálist állít elő, kell egy példánytag, amely a nem visszaadottat tárolja és egy logikai tag, hogy van-e tárolt vagy futtatni kell az algot.) és az OpenJDK, Oracle JDK-ban a Sun által adott OO szervezés ua.! <https://arato.inf.unideb.hu/batfai.norbert/UDPROG> (16-22 fólia) Ugyanezt írjuk meg C++ nyelven is! (lásd még UDPROG repó: source/labor/polargen)

Megoldás videó:

Megoldás forrása:

Tehát ha megnezzük a programunk úgy mukodik hogy megnezi hogy van e már előre tárolt értékünk . Termesztesen ha van tarolt érték akkor a program azzal dolgozik ha nincs akkor pedig készít egyet . Vagyis pontosabban kettőt erteket készít szamunka a metódus amelyek közül az egyiket letarolja a masodikat meg visszaadja. és azzal vegzi el a számításokat

```
import java.util.Random;
import java.io.*;
import java.lang.Math;

public class PolarGen {

    public final static int RAND_MAX = 32767;
    private static boolean bExists;
    private double dValue;

    static Random cRandomGenerator = new Random();

    public PolarGen() {

        bExists = false;
        cRandomGenerator.setSeed(20);
    }

    public double PolarGet() {
```

```
if (!bExists)
{
    double u1, u2, v1, v2, w;

    do {

        u1 = cRandomGenerator.nextInt (RAND_MAX) / (RAND_MAX + 1.0);
        u2 = cRandomGenerator.nextInt (RAND_MAX) / (RAND_MAX + 1.0);
        v1 = 2 * u1 - 1;
        v2 = 2 * u2 - 1;
        w = v1 * v1 + v2 * v2;
    }

    while (w > 1);
    double r = Math.sqrt ((-2 * Math.log (w)) / w);
    dValue = r * v2;
    bExists = !bExists;
    return r * v1;
}
else

{
    bExists = !bExists;
    return dValue;
}

};

public static void main(String args[]) {
    PolarGen cPolarGen = new PolarGen();
    double dEredmeny = cPolarGen.PolarGet();
    System.out.println(dEredmeny);
}
```

}

### C++ valtozat

```
#include<cmath>
#include <cstdlib>
#include<ctime>

using namespace std;

class PolarGen {

public:
    PolarGen () {
```

```
nincsTarolt = true;
srand (time (NULL));
}

~PolarGen () {
}

double kovetkezo ();

private:

    bool nincsTarolt;
    double tarolt;

};

double PolarGen::kovetkezo() {

    if (nincsTarolt) {
        double u1, u2, v1, v2, w;
        do {
            u1 = rand() / (RAND_MAX + 1.0);
            u2 = rand() / (RAND_MAX + 1.0);
            v1 = 2 * u1 - 1;
            v2 = 2 * u2 -1;
            w = v1 * v1 + v2 * v2;

        } while ( w > 1);
        double r = sqrt ((-2* log(w) / w));
        tarolt = r * v2;
        nincsTarolt = !nincsTarolt;
    }

    return r * v1;
}

else {
    nincsTarolt = !nincsTarolt;
    return tarolt;
}

}

int main(int argc, char** argv) {
    PolarGen pg;
    for (int i = 0; i < 10; i++)
        std::cout<<pg.kovetkezo()<<std::endl;
    return 0;
}
```

## 11.2. Homokozó

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

Megoldás videó:

Megoldás forrása: <https://github.com/Samate99>Hello-world/blob/master/LZWBInFa.java>

A feladatunk hogy átírjuk a binfát java nyelvre . Gyakorlatilag át kell írnunk c kódban szereplő pointereket illetve referenciákat nem másra mint javas referenciára illetve át kell írnunk java szintaxisnak megfelelően a kódot ..

## 11.3. „Gagyi”

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

Megoldás videó:

Megoldás forrása:

A szöveg alatt láthatunk 2 darab kódot láthatunk . Amelyekben szinte teljesen ugyanolyanok. Ezekeben csak 2 érték az x, és a t érték különböző. Az első esetben láthatjuk hogy az x és a t érték jóval nagyobb mint a masodik esetben. A két szám nagyobb értéke azt generalja hogy ebben az esetben a ciklus feltétele igaz lesz ez pedig előidéz szamunkra egy végtelen ciklust. Ez a folyamat alap integer osztálynak köszönhető ami alapból egy 2 érték közötti tartomány amely nem mas mint -128 és 127 között talalható . De mivel ez a 2 érték kívül esik az adott tartományon emiatt a ciklus feltétele igaz lesz. Ezzel szemben ha megnezzük a masodik esetet ahol a 2 szám bennevan ebben az osztályban mint ahogy látható akkor természetesen a a ciklusunk feltétele hamis lesz es így nem nem következik ugyanaz be mint az első esetben tehat a ciklus feltétele nem igaz hanem hamis lesz tehat nem jön létre egy végtelen ciklus.

```
public class gagyi {
    public static void main(String args[]) {
        Integer x = 2001 ;
        Integer t = 2001 ;
        System.out.print("A " + "ciklus " + "kezdete\n");
        while(x<=t && x>=t && x!=t)
        {}
        System.out.println("A ciklus vége");
    }
}
```

```
public class gagyi {
    public static void main(String args[])
}
```

```
{  
Integer x = -21;  
Integer t = 21;  
System.out.print("A "+ "ciklus +"+"kezdete\n");  
while(x<=t && x>=t && x!=t)  
{ }  
System.out.println("A ciklus vége");  
}  
}
```

## 11.4. Yoda

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

Megoldás video:

Megoldás forrása:

A yoda conditions alatt egy megfordított összehasonlítást értünk . Tehát ha egy alap esetet nezunk akkor valtozot hasonlítjuk . De itt a megvan fordítva és a valtozóhoz hasonlítunk . Igazaból ez végülis programozási szempontból mindegy hisz minden esetben elvileg mukodni fog a program.. De ha egy olyan valtozott hasonlítunk melynek erteke null akkor hibát kapunk. Nem is akarmilyet a java.lang.Nullpointer.ex hibát ami a feladatunk.

Ennek a kódnak a segítségével idézhetjük elő a java.lang.nullpointer.ex hibát . Pontosabban a 2. if esetében történik a hiba , mivel ott a valtozonk erteke nem más mint null így hibát kapunk

```
public class Yoda {  
  
    public static void main(String[] args) {  
        String text = null;  
  
        if ("text".equals(texttext)) {  
        }  
        System.out.println("text".equals(text));  
  
        texttext = null;  
        if (text.equals("text")) {  
        }  
        System.out.println("text".equals(text));  
  
    }  
}
```

## 11.5. Kódolás from scratch

Induljunk ki ebből a tudományos közleményből: <http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/bbpalg.pdf> és csak ezt tanulmányozva írjuk meg Java nyelven a BBP algoritmus megvalósítását! Ha megakadsz, de csak végső esetben: [https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitokjavat/apbs02.html#pi\\_jegyei](https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitokjavat/apbs02.html#pi_jegyei) (mert ha csak lemosolod, akkor pont az a fejlesztői élmény marad ki, melyet szeretném, ha átélnél).

Megoldás videó:

Megoldás forrása:

A BBP algoritmus ki nem találnak a BBP formulát használja melyet Simon Plouffe nevéhez köthetünk. Maga az algoritmus a Pi számjegyeinek meghatarozására alkamas. Továbbá ami még egyedibbé és érdekesebbé teszi a programot az nem mas mint az hogy nem feltétlenül az első helyről kaphatjuk meg a pi számjegyeit hanem egy alltalunk kivalaszott pozíciótól is lekérhetjük a számjegyeit.

package bpp;

```
* PiBBP.java

public class PiBBP {
    public PiBBP(int d) {

        double d16Pi = 0.0d;
        double d16S1t = d16Sj(d, 1);
        double d16S4t = d16Sj(d, 4);
        double d16S5t = d16Sj(d, 5);
        double d16S6t = d16Sj(d, 6);

        d16Pi = 4.0d*d16S1t - 2.0d*d16S4t - d16S5t - d16S6t;
        d16Pi = d16Pi - StrictMath.floor(d16Pi);

        StringBuffer sb = new StringBuffer();
        Character hexaJegyek[] = {'A', 'B', 'C', 'D', 'E', 'F'};

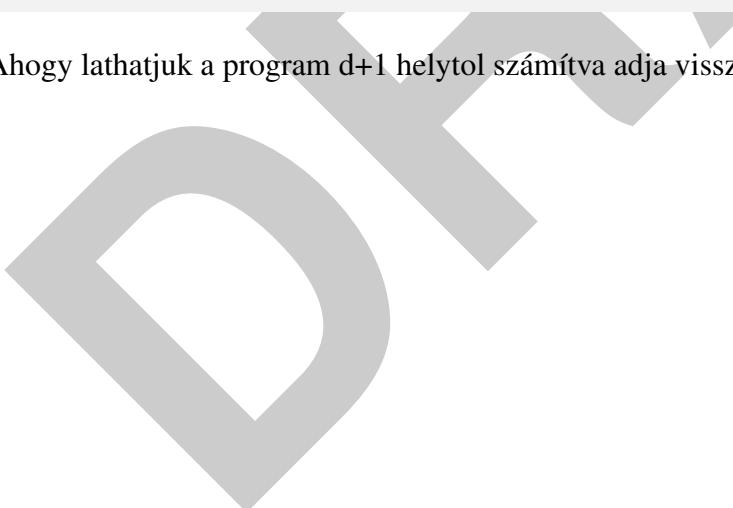
        while(d16Pi != 0.0d) {
            int jegy = (int)StrictMath.floor(16.0d*d16Pi);
            if(jegy<10)
                sb.append(jegy);
            else
                sb.append(hexaJegyek[jegy-10]);
            d16Pi = (16.0d*d16Pi) - StrictMath.floor(16.0d*d16Pi);
        }
        d16PiHexaJegyek = sb.toString();
    }

    public double d16Sj(int d, int j) {

        double d16Sj = 0.0d;
        for(int k=0; k<=d; ++k)
            d16Sj += (double)n16modk(d-k, 8*k + j) / (double)(8*k + j);
```

```
    return d16Sj - StrictMath.floor(d16Sj);
}
public long n16modk(int n, int k) {
    int t = 1;
    while(t <= n)
        t *= 2;
    long r = 1;
    while(true) {
        if(n >= t) {
            r = (16*r) % k;
            n = n - t;
        }
        t = t/2;
        if(t < 1)
            break;
        r = (r*r) % k;
    }
    return r;
}
public String toString() {
    return d16PiHexaJegyek;
}
public static void main(String args[]) {
    System.out.print(new PiBBP(1000000));
}
}
```

Ahogy lathatjuk a program d+1 helytol számítva adja vissza a Pi értékeit tehát.



## 12. fejezet

# Helló, Liskov!

### 12.1. Liskov helyettesítés sértése

Írunk olyan OO, leforduló Java és C++ kódcsipetet, amely megséríti a Liskov elvet! Mutassunk rá a megoldásra: jobb OO tervezés. [https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_1.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf) (93-99 fólia) (számos példa szerepel az elv megsértésére az UDPROG repóban, lásd pl. source/binom/Batfai-Barki/madarak/)

Megoldás videó:

Megoldás forrása:

Tutor : Veress Máté

Tehát maga a liskov helyettesítési elv vagy röviden csak LSP a S.O.L.I.D elvek talán legerdekebb tagja. Ma az S.O.L.I.D Robert C. Martin nevéhez fűződik aki a clean code egyik vezéralakja. Maga az elv lenyege hogy minden alap osztály legyen helyettesíthatő azt alt azaz a leszármazott osztályával (öröklödés). Tehát ha peldaúl nem volt ki semmitse az alap osztály akkor azt alt osztály se valtson ki semmit vagyis a program mukodese nem változik. Ez azért fontos mert azáltal minden olyan helyre ahol az alap osztály használható az alt osztály is használható. A feladatunk hogy készítsünk egy olyan programot amely sérti a liskov elvet. Tehát ahogy a kódban lathatjuk van egy Madarak osztalyunk. Az osztály rendelkezik egy repülés nevű funkcióval. létrehozunk még 2 alosztályt . Ezek a sas illetve a pingvin nevet viselik. Ahogy lathatjuk minden alosztály örökli a repules()-t. Ezzel nem is lenne baj ha csak a sast nezzük viszont ha mar a pingvint nezzük akkor igen. Emiatt logikailag a programunk nem mukodik megfelelően.

```
#include <stdio.h>
class Madarak {
    public:
        char repules() { printf("Repülök\n"); };
};
class Sas : public Madarak {};
class Pingvin : public Madarak {};
int main()
{ Sas sas;
    Pingvin pingvin;
    sas.repules();
    pingvin.repules(); }
```

```
}
```

Ezert kell letrehoznunk egy új alosztályt a madarakon belül a repulomadarak osztalyat . Majd ezt az osztályt kell ellátunk a repules funkcióval. Ezáltal képesek vagyunk arra hogy a sas rendelekzzent repules() funkcióval de a pingvin nem.

```
include <stdio.h>
class Madarak {
};
class RepuloMadarak : public Madarak {
public:
char repules() { printf("Repülök\n"); }
}
class Sas : public RepuloMadarak {};
class Pingvin : public Madarak {};

int main() { Sas sas;
Pingvin pingvin;
sas.repules();
pingvin.repules(); }

}
```

így a programunk logikai szempontból tökeletes . Viszont hibát fogunk kapni mivel a pingvin osztálynak nincs repules funkciója.

## 12.2. Szülő-gyerek

irjunk Szülő-gyerek Java és C++ osztálydefiníciót, amelyben demonstrálni tudjuk, hogy az ősön keresztül csak az ős üzenetei küldhetőek! [https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_1.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf) (98. fólia)4

Megoldás videó:

Megoldás forrása:

Tutor : Veress Máté

Tehát nezzük meg a programunk. Láthatjuk hogy a kódban rendelkezünk egy téglalap névű osztállyal amely rendelkezik egy meret névű funkcióval és a meret tartalmaz 2 értéket , ezek nem masak mint szelesség és a magasság . Láthatjuk hogy a téglalap osztály rendelkezik egy alosztállyal ami a negyzet nevet viseli. Ez az alosztály rendelkezik egy terület névű függvénytel. A program futtatása során hibát tapasztalhatunk ez a "teglalap.terulet()" köszönhető . Mivel a téglalap osztály nem rendelkezik ilyen "teglalap.terulet()" függvénytel csak az a negyzet névű alosztálya.

C++ Kód

```
#include <stdio.h>
class Teglalap {
public: int meret()
```

```
{ int height, width;
int meret[2] = {height, width};
return meret[2]; } };

class Negyzet : public Teglalap {

public:
int terulet(int height, int width)
{ return height*width; }
};

int main() {
Teglalap teglalap;
Negyzet negyzet;
negyzet.meret();
printf("%d", negyzet.terulet(10,14));
teglalap.meret();
teglalap.terulet();
}
```

### Java kód

```
class Teglalap {
private int height;
private int width;

public int meret() {
    int[] meret = new int[]{height, width};
    return meret[1];
}

class Negyzet extends Teglalap {
    public int terulet(int height, int width){
        return height * width; } }

public class Main {
public static void main(String[] args)
{ Teglalap teglalap = new Teglalap();
    Negyzet negyzet = new Negyzet();
    System.out.println(negyzet.terulet(1,1));
    System.out.println(teglalap.terulet(1,1)); //itt a hiba } }
```

## 12.3. Anti OO

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

Megoldás videó:

Megoldás forrása:

A feladatunk nem mas mint az elozo csokorbol mar megismert BBP alogirtmus hasznalata. A PI-nek kell a  $10^6$   $10^7$   $10^8$  darab jegyet kiszamolnunk illetve meg kell mernunk hogy melyik nyelv ( C , C++ , Java , C# mekkora ) sebességek képes kiszámolni a szamjegyeket. A program mukodes szempontjabol ugy nez ki hogy a cikluson belul talalhato D valtozot átírva tudjuk a megvaltoztatni hogy meddig írja ki a PI szamjegyeit. Minnel több nulla kerül utana annál több számjegyet fog szamunkra kiirni. Amint lefuttattuk a programot több eredmennnyel lehetunk gazdagabbak . Nekunk a masodik ertekre van szükségünk amely nem mas mint a mérési idő.

```
/*
 * PiBBPBench.java
 *
 * DIGIT 2005, Javat tanítok
 * Bátfai Norbert, nbatfai@inf.unideb.hu
 */
/**
 * A PiBBP.java-ból kivettük az "objektumorientáltságot", így kaptuk
 * ezt az osztályt.
 *
 * (A PiBBP osztály a BBP (Bailey–Borwein–Plouffe) algoritmust a Pi hexa
 * jegyeinek számolását végző osztály. A könnyebb olvahatóság
 * kedvéért a változó és metódus neveket megpróbáltuk az algoritmust
 * bemutató [BBP ALGORITMUS] David H. Bailey: The BBP Algorithm for Pi.
 * cikk jelöléseihez.)
 *
 * @author Bátfai Norbert, nbatfai@inf.unideb.hu
 * @version 0.0.1
 */

public class PiBBPBench {

    /**
     * BBP algoritmus a Pi-hez, a [BBP ALGORITMUS] David H. Bailey: The
     * BBP Algorithm for Pi. alapján a {16^d Sj} részlet kiszámítása.
     *
     * @param d a d+1. hexa jegytől számoljuk a hexa jegyeket
     * @param j Sj indexe
     */
    public static double d16Sj(int d, int j) {
        double d16Sj = 0.0d;
        for(int k=0; k<=d; ++k)

            d16Sj += (double)n16modk(d-k, 8*k + j) / (double)(8*k + j);
        /* (bekapcsolva a sorozat elejen az első utáni jegyekben növeli pl.
           a pontosságot.)
        for(int k=d+1; k<=2*d; ++k)
            d16Sj += Math.pow(16.0d, d-k) / (double)(8*k + j);
    }
}
```

```
        return d16Sj - Math.floor(d16Sj);
    }
/***
 * Bináris hatványozás mod k, a 16^n mod k kiszámítása.
 *
 * @param n   kitevő
 * @param k   modulus
 */
public static long n16modk(int n, int k) {

    int t = 1;
    while(t <= n)
        t *= 2;
    long r = 1;

    while(true) {

        if(n >= t) {
            r = (16*r) % k;
            n = n - t;
        }
        t = t/2;
        if(t < 1)

            break;
        r = (r*r) % k;
    }

    return r;
}

/***
 * A [BBP ALGORITMUS] David H. Bailey: The
 * BBP Algorithm for Pi. alapján a
 * {16^d Pi} = {4*{16^d S1} - 2*{16^d S4} - {16^d S5} - {16^d S6}}
 * kiszámítása, a {} a törtrészt jelöli. A Pi hexa kifejtésében a
 * d+1. hexa jegytől
 */
public static void main(String args[]) {

    double d16Pi = 0.0d;
    double d16S1t = 0.0d;
    double d16S4t = 0.0d;
    double d16S5t = 0.0d;
    double d16S6t = 0.0d;

    int jegy = 0;

    long delta = System.currentTimeMillis();
```

```

for(int d=1000000; d<1000001; ++d) {

    d16Pi = 0.0d;
    d16S1t = d16Sj(d, 1);
    d16S4t = d16Sj(d, 4);
    d16S5t = d16Sj(d, 5);
    d16S6t = d16Sj(d, 6);

    d16Pi = 4.0d*d16S1t - 2.0d*d16S4t - d16S5t - d16S6t;

    d16Pi = d16Pi - Math.floor(d16Pi);
    jegy = (int) Math.floor(16.0d*d16Pi);

}

System.out.println(jegy);
delta = System.currentTimeMillis() - delta;
System.out.println(delta/1000.0);
}
}

```

Ha összehasonlítjuk a futási időket . Láthatjuk hogy gyakorlatilag a java a leggyorsabb illetve hogy a c++ és c elegendő közel a futási időket produkál.

## 12.4. Hello, Android!

Élesszük fel az SMNIST for Humans projektet! <https://gitlab.com/nbatfai/smnist/tree/master/forHumans/SMNIST>  
Apró módosításokat eszközölj benne, pl. színvilág.

Megoldás videó:

Megoldás forrása:

SMNIST egy olyan program amelyben pontok jelennek meg a 1000ms-enként kepernyőn és az a feladatunk hogy a pontok mennyiségeinek megfelelő számot nyomjuk meg. Természetesen mindezt egy adott időn belül kell megtennünk . A sebességünket a program természetesen megmutaja ms pontosággal. Amint a pontok számával a háttér színe is megváltozik. A programban több színt különíthetünk el . Ez amiatt fontos mert minnen nagyobb a színtünk annál több pont kerül a kepernyőre tehát annal nehezebb értelmezünk . A feladatunk hogy a programban egy apró változtatást vigyük véghet pár lépésen ebben az esetben a színt változtatunk meg. Ahogy láthatjuk alapból van egy belső és 9 körök . A nagy kör színe szürke ebben fekete pontok jelennek. A kis körök sárgák és bennük a számok szürkén jelennek meg . Az SMNIST.SURFACE.view.java-t megnevezve a 354. sorban találhatjuk a színekért felelős részt Tehát az egész 4 kisebb részre van szedve. Ezekben tudjuk allítani a programban szereplő színeket , attudjuk írni a pontok a körök vagy akár a számok színet is ahogy láthatjuk a meretüket és az elhelyezkedésüket is tudjuk alakítani

```
private void cinit(android.content.Context context) {
```

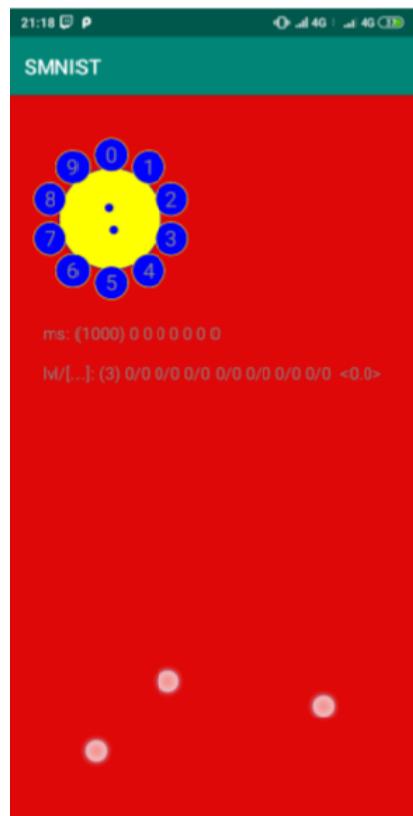
```
textPaint.setColor(android.graphics.Color.YELLOW);
textPaint.setStyle(android.graphics.Paint.Style.FILL_AND_STROKE);
textPaint.setAntiAlias(true);
textPaint.setTextAlign(android.graphics.Paint.Align.CENTER);
textPaint.setTextSize(50);

msgPaint.setColor(android.graphics.Color.YELLOW);
msgPaint.setStyle(android.graphics.Paint.Style.FILL_AND_STROKE);
msgPaint.setAntiAlias(true);
msgPaint.setTextAlign(android.graphics.Paint.Align.LEFT);
msgPaint.setTextSize(40);

dotPaint.setColor(android.graphics.Color.YELLOW);
dotPaint.setStyle(android.graphics.Paint.Style.FILL_AND_STROKE);
dotPaint.setAntiAlias(true);
dotPaint.setTextAlign(android.graphics.Paint.Align.CENTER);
dotPaint.setTextSize(50);

borderPaint.setStrokeWidth(2);
borderPaint.setColor(android.graphics.Color.YELLOW);
fillPaint.setStyle(android.graphics.Paint.Style.FILL);
fillPaint.setColor(android.graphics.Color.YELLOW);
```

DPRY



### 12.1. ábra. Nem saját kep

# 13. fejezet

## Helló, Mandelbrot!

### 13.1. Reverse engineering UML osztálydiagram

UML osztálydiagram rajzolása az első védési C++ programhoz. Az osztálydiagramot a forrásokból generáljuk (pl. Argo UML, Umbrello, Eclipse UML) Mutassunk rá a kompozíció és aggregáció kapcsolatára a forráskódban és a diagramon, lásd még: [https://youtu.be/Td\\_nIERIEOs](https://youtu.be/Td_nIERIEOs). <https://arato.inf.unideb.hu/batfai.norbert/UD/> (28-32 fólia)

Megoldás videó:

Megoldás forrása:

Tehát a Helló, Mandelbrot csokkrunk első feladata nem más hoz mint az első védési programunkhoz azaz a LZWbinfa c++os verziojához kotheto. Feladtunk hogy egy ugynevet UML Diagramot készítsünk a bin-faból. Az UML azaz a Unidified modeling language rövidítése . Az UML egy alltanaos célú modellező nyelv amelyet széles körben használnak . Peldaul üzleti elemzők rendszertervezők és szoftvermérnökök is. Az UML alkalmas programrendserek vizualis megvalósítására. A feladat megoldásához az Umbrella nevezetű programot használtam. Itt nincs is mas dolgunk mint hogy beimportáljuk a forrást .Ezt a Code , Code Importing Wizard menu segítségével tehetjük meg. Ebből a programunk el is készíti számunkra a diagramot Semmifele változtatást nem kell csinalunkk. Az importálás után az alul latható sárga négyzetet kapunk.

```

LZWBinFa
- fa : Csomopont*
- melyseg : int
- atlagosszeg : int
- atlagdb : int
- szorasosszeg : double
# gyoker : Csomopont
# maxMelyseg : int
# atlag : double
# szoras : double
+ LZWBinFa() «constructor»
+ ~ LZWBinFa() «destructor»
+ operator <<(b : char)
+ kiir()
+ getMelyseg() : int
+ getAtlag() : double
+ getSzoras() : double
+ operator <<(os : std::ostream&, bf : LZWBinFa&) : std::ostream& «friend»
+ kiir(os : std::ostream&)
- LZWBinFa( : const LZWBinFa&) «constructor»
- operator =( : const LZWBinFa&) : LZWBinFa&
- kiir(elem : Csomopont*, os : std::ostream&)
- szabadit(elem : Csomopont*)
# rmelyseg(elem : Csomopont*)
# ratlag(elem : Csomopont*)
# rszoras(elem : Csomopont*)

```

13.1. ábra. Forrás : Sajat kep

## 13.2. Forward engineering UML osztálydiagram

UML-ben tervezünk osztályokat és generálunk belőle forrást!

Megoldás videó:

Megoldás forrása:

A masodik feladatban . Az előzőhez hasonloan az UML osztálydiagrammokkal kell foglalkoznunk. Viszont itt az előzővel ellentetben meg kell fordítanunk a folyamatot . Tehát itt az UML diagrammból kell letrehoznunk egy forráskódot. Letrekel hoznunk egy új classt egy ures sarga negyszoget. Ezen belül tudunk letrehozni különbozo dolgokat. En szemely szerint letrehoztam a ProgClasst ahol felvettetem 3 változót Ebbol tudunk forrást generalni ezt egyszeruen az elozohez hasonló helyen tehetjük meg a Code Code Generation Wizard helyen tudjuk itt ki kell valasztanunk hogy hova mentsen illtve ki kell valasztanunk a nyelvet. Ha ezt legeneraljuk 2 fajlt kapunk egy headert(.h) es egy forrás (.cpp)fajlt . Termeszesesen c++ nyelven A diagramrol es a forráskódrol a szöveg alatt lathatunk kepeket.

```

ProgClass
- Prog : int
- Esport : int
- Nemtudom : string

```

13.2. ábra. Forrás : Sajat kep

```

#include "ProgClass.h"
// Constructors/Destructors
// 
```

```
ProgClass::ProgClass () {  
    initAttributes();  
}  
ProgClass::~ProgClass () { }  
//  
// Methods  
//  
// Accessor methods  
//  
  
// Other methods  
//  
void ProgClass::initAttributes () {  
}
```

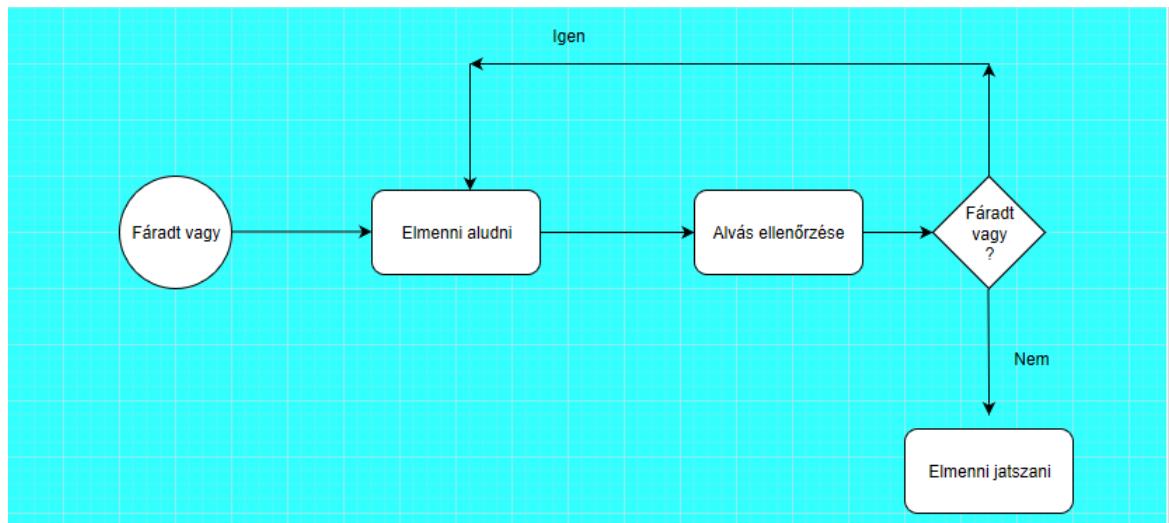
### 13.3. BPMN

Rajzolunk le egy tevékenységet BPMN-ben! <https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog> (34-47 fólia)

Megoldás videó:

Megoldás forrása:

A feladatunk hogy lerajzoljunk egy BPMN Tevékenységet. Amely nem mas mint egy adott folyamat grafi-kus ábrázolás. Ahogy a nevéről "Business Process Model and Notation" ered alltalaban üzlet folyamatok , modellek abrazolására használják. a Rajzot a draw.io használva készítettem. Itt könnyen lehet BPMN szeruen abrazolni. Itt van egy alap kezdes a faradsag erre tortenik egy folyamat az hogy "Elmenni aludni" . Ezutan kovetkezik egy ellenőrzés nyilvan ha felkeltünk kiderül hogy sikerült e kialudnunk magunkat . Itt ketté válik a történet de akár válthatna több fele is Tehát a faradtak vagyunk azaz igaz a feltelel akkor vissza-fekszünk pihenni ez mindaddig tortenik még a feltelel hamis nem lesz ha hamis leszt azaz nem leszunk faradtak akkor elmegyünk most csinalni



13.3. ábra. Forrás : Sajat kep

DRAFT

## 14. fejezet

# Helló, Chomsky!

### 14.1. Encoding

Fordítsuk le és futtassuk a Javat tanítok könyv MandelbrotHalmazNagyító.java forrását úgy, hogy a fájl nevekben és a forrásokban is meghagyjuk az ékezes betűket! <https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/adatok.html>

Megoldás videó:

Megoldás forrása:

A javaban a C++/C-vel ellentetben van lehetőségünk ekezes betük használatara . A java nyelv egyik nagy előnye hogy jóval több karakterkészletet képes kezelní . Ezáltal a mi magyar betűinkat is simán használhatjuk , gond nélkül . A mandelbrot halmaz is magyar karakterkészletet felhasználva készült .Ebből kifolyólag A hagyományos fordítás során hibát tapasztalhatunk. A jelen esetben az ISO-8859-2es karakterkészletet kell használnunk hogy a fordítás sikeres legyen ehez nincs más dolgunk mint az -encoding kapcsoló használata. ezzel tudjuk beállítani hogy milyen karakterkészletet használunk. A program ugyanaz mint a prog len mar sokak számára megismert program . Tehát kapunk egy mandelbrot halmazt amelyben az eger segítséggel tudunk részeket kijelölni illetve ezeket a kijelölt részeket kinagyítani.



## 14.2. Paszigráfia Rapszódia OpenGL full screen vizualizáció

Lásd vis\_prel\_para.pdf! Apró módosításokat eszközölj benne, pl. színvilág, textúrázás, a szintek jobb elkülönítése, kézreállóbb irányítás.

Megoldás videó:

Megoldás forrása:

A feladatunkban paszgrafiai rapszódia , vagyis az ugynevezett PaRaval fogunk foglalkozni ez nem mas mint egy mesterséges nyelv kialakítására törekvő kezdeményezés . Celjanem mas mint az ugynevezett homunkulusz és a mesterséges homunkulusz közötti kapcsolat.A programban a para vizualizacios lehetőségeit nezzuk meg. A program az SMNISThez hasonloan pöttyökkel dolgozunk csak itt eppen nem a szamosaggon hanem a "pöttyök" elhelyezkedésén van a hangsúly ! A programban kulonbozo forgatható kockakkal es ezekben elhelyezkedő négyzetekkel fogunk dolgozni. A mi feladatunk hogy ezeket alakítsuk at kicsit. Peldaul ahoz hogy az ablakunk meretet noveljuk a w és a h valtozo értéket kell átírnunk. Szineket a 3 alapszinből tudunk generalni tehat a piros kék zöld szinekből tudunk letrehozni. A billentyuzet gombajait pedig a keyboard reszben tudjuk atalakítani

```
glColor3f ( 0.180f, 0.202f, 0.191f );
```

```
int w = 1280;
int h = 768;
```

## 14.3. Perceptron osztály

Dolgozzuk be egy külön projektbe a projekt Perceptron osztályát! Lásd <https://youtu.be/XpBnR31BRJY>

Megoldás videó:

Megoldás forrása:

Ebben a feladatban a perceptron osztálytal fogunk dolgozni . A feladat szoros osszefuggesben van a mandelbrot halmazzal hiszelsőnek a mandelbort programunk segítségével kell létrehoznunk egy mandelbort halmazt abrazolo kepet . Fontos kiemelni hogy a kepnek PNG formatumunak kell lennie . így az elejen kiemelnem hogy a Mandelbrot halmaz Benoit Mandelbrot lengyel szarmazasú Francia-Amerika matematikus nevéhez kothető. A mandelbrot halmazt ő emelte a koztudatba ezert is kapta róla a nevet. A program megfelelő futásához elsőnek el kell vegezni a megfelelő telepítéséket . ilyen peldaul a libpng konyvtar is . A program futása utan egy keppel leszunk gazdagabbak . A program a mar letrehozott PNG kep alapan fogja az uj kepet elkeszíteni.Mereteit a Get\_width() és a get\_height() befolyasollja . a program futása során 2 forciklus segítsével végigmegy a kep szélességén és magassagan beallítva annak mereteit . A beimportalt kep piros reszeit a lefoglalt tarba helyezi el majd ennek a segítésével készíti el a kepet a program.

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

using namespace std;
```

```
using namespace png;

int main ( int argc, char *argv[] )
{
image <rgb_pixel> png_image (argv[1]);
int size = png_image.get_width() * png_image.get_height();

Perceptron* p = new Perceptron(3, size, 256, 1);

double* image = new double[size];

for (int i{0}; i < png_image.get_width(); i++)
for (int j{0}; j < png_image.get_height(); j++)
image[i * png_image.get_width() + j] = png_image[i][j].red;

double value = (*p)(image);
cout << " " << value << endl;

delete p; delete [] image
```

DRAFT

# 15. fejezet

## Helló, Stroustrup!

### 15.1. JDK osztályok

Írunk olyan Boost C++ programot (indulj ki például a fénykardból) amely kilistázza a JDK összes osztályát (miután kicsomagoltuk az src.zip állományt, arra ráengedve)!

Megoldás videó:

Megoldás forrása:

Tehát a feladatunk egy boost program írása C++ban amely kilistazza a jdk osszes osztályát . Ahoz hogy egy ilyen programot írunk szükségünk lesz egy ugynevezett boost könyvtárra . Ahoz hogy ezt includolni tudjuk elsonek le kell toltenunk . Amint ezt megtettük ki kell csomagolnunk és mappájában talalható bootstrap.bat fajlt kell futtatnunk . Ez előkészíti szamunkra a boost konyvtarat. Ha windowson dolgozunk peldaul visual studioban meg kell tennünk a megfelelő beallításokat nyilvan egy c++ projektet kell beallítanunk illetve nyivan ahogy fentebb emíltettem mar inculeoldnunk kell a boost könyvtárat. A JDK eléréshez elsonek a JDK scr.zipet kell kibontatnunk a java konyvtaraban. Ha ezt elvegeztük kezdhetünk a programma foglalkozni nyilvan includeolnunk kell a boost konyvatarakat. lathatunk egy stringet amely utan egy repeat funkció következik nyilvan ez a folyamat annyiszor jatszodik le mint az n erteke tehát amekkorat megadunk szamara. Ezutan következik a main itt egyrészt jdk osztály eleresi utja ahogy lathatjuk. masreszt lathatunk egy IF fugvenyt ennek a segítségével minden .javara vegzodo de nem a kivetelek koze tartozo kiiratunk a konzolba . Ahogy lathatjuk több jelet is lathatunk a konzolba. A + száma mapparendszerben való elhelyezkedese hatarozza meg . | pedig a következonek vizsgalt mappa neve.

```
include <iostream>
#include <sstream>
#include "boost\filesystem.hpp"
#include "boost\algorithm\string.hpp"
#include "boost\algorithm\string\predicate.hpp"

using namespace std;
using namespace boost;
using namespace boost::filesystem;

string repeat(int n, string what) {
```

```
ostringstream os;
for (int i = 0; i < n; i++)
os << what;
return os.str();
}

int main()
{ string pa = "C:\\Program Files\\Java\\jdk1.8.0_221\\src";
path apk_path(pa); long long count = 0;
recursive_directory_iterator end;
for (recursive_directory_iterator i(apk_path); i != end; i++)
{ const path cp = (*i);
string name = cp.string();
vector < string > strs;
split(strs, name, is_any_of("\\\""));
string fName = strs[strs.size() - 1];
if (fName.compare("module-info.java") == -1 && fName.compare("package
← -
←info.java") == -1) {

if (ends_with(fName, "java")) {
cout << " " << repeat(strs.size() - 7, "+") << " " << fName. ←
substr←
(0, fName.size() - 5) << endl;
count++;
}
else {
cout << " | " << fName << " >" << endl;
}
} cout << "JDK osztályok száma: " << count << endl;
}
```

## 15.2. Hibásan implementált RSA törése

Készítünk betű gyakoriság alapú törést egy hibásan implementált RSA kódoló: <https://arato.inf.unideb.hu/batfai.r> (71-73 fólia) által készített titkos szövegen.

Megoldás videó:

Megoldás forrása:

A következő a hibásan implementált RSA Törés feladat lesz. Szerintem ennel a feladatnal erdemes elsonek az RSAról illetve az RSA algoritmusról beszélnünk. Maga az RSA egy eleg híres és viszonylag titkosító eljárás. Maga a neve az RSA Ron Rivest, Adi Shamir és Len Adleman nevéről adódik. Ők hárman hoztak létre 1978ban. Működése alapján fermat-tétellel van szoros összefüggésben. A titkosítás során az RSA esetében 2 kulcsról egy nyílt és egy titkos kulcsal találkozhatunk. Ez a megfelelő biztonságú titkosításhoz szükséges. A feladatban ebben az esetben egy hibás RSAra van szükségünk ezért nem szöveget titkosítunk hanem karaktereket.

Ahogy a kódban lathatjuk rsa\_chiper osztályban történik meg a kódolás. Ahogy lathatjuk a példa szövegünk a This is a Perfect string lesz. 2 ciklus segítségével történik meg a kódolás ahogy fentebb már említettem betűről betűre.

```

public class rsa_chiper
{ public static void main(String[] args)
{ int bitlength = 2100;
SecureRandom random = new SecureRandom();

BigInteger p = BigInteger.probablePrime(bitlength/2, random);
BigInteger q = BigInteger.probablePrime(bitlength/2, random);

BigInteger publicKey = new BigInteger("65537");
BigInteger modulus = p.multiply(q);

String str = "this is a perfect string".toUpperCase();
System.out.println("Eredeti: " + str);
byte[] out = new byte[str.length()];
for (int i = 0; i < str.length(); i++) {
char c = str.charAt(i);
if (c == ' ')
out[i] = (byte)c;
else
out[i] = new BigInteger(new byte[] {(byte)c}).modPow(publicKey,
modulus).byteValue(); }

String encoded = new String(out);
System.out.println("Kódolt:" + encoded);

Decode de = new Decode(encoded);

System.out.println("Visszafejtett: " + de.getDecoded());
}

```

```

private void loadFreqList()
{ BufferedReader reader;
try
{ reader = new BufferedReader(new FileReader("freq.txt"));
String line;
while((line = reader.readLine()) != null) { String[] args = line.split("\t" );
char c = args[0].charAt(0);
int num = Integer.parseInt(args[1]);
this.charRank.put(c, num);
}
} catch (Exception e)
{ System.out.println("Error when loading list ->" + e.getMessage());
}
}

```

nyilvan ahoz hogy ezt visszafejtsük többek között szükségünk lesz egy txt allomanyra amely tartalmazni fogja a karakterek gyakorisagat , amely nagyon fontos szerepet fog jatszani . Itt lathatjuk a tabla beolvasast

. Itt dönti el a program hogy mihez mit rendel hozza , itt vizsgalja meg a szoveget termesztesen ha a program egy spacet talakozik nem vizsgal semmit egyszeruen át lép rajta tovabb halad. Ha talalt karatker benne van akkor a txtben szereplő listában minden egyel noveli az értéket ha nincs benne a listaban akkor pedig hozzaadja a karaktert 1-es kezdőértékkel.

```
ublic Decode(String str) {  
this.charRank = new HashMap<Character,  
Integer>(); thisdecoded = str;  
  
this.loadFreqList();  
HashMap<Character, Integer> frequency = new HashMap<Character, Integer <  
>();  
for (int i = 0; i < str.length(); i++) {  
char c = str.charAt(i);  
if (c != ' ')  
if(frequency.containsKey(c))  
frequency.put(c, frequency.get(c) + 1);  
else frequency.put(c, 1); }  
while (frequency.size() > 0) {  
int mi = 0;  
char c = 0;  
for (Entry<Character, Integer> e : frequency.entrySet()) {  
if (mi < e.getValue()) {  
mi = e.getValue();  
c = e.getKey();  
}  
}  
thisdecoded = thisdecoded.replace(c, this.nextFreq());  
frequency.remove(c); }
```

Decodolas soran lathatjuk hogy a program a gyakorisag vizsgalata miatt nem feltetlenül pontos hisz egy nem megfelelő tabla eseten nem megfelelo dolgokat kapunk.

### 15.3. Változó argumentumszámú ctor

Készítsünk olyan példát, amely egy képet tesz az alábbi projekt Perceptron osztályának bemenetére és a Perceptron ne egy értéket, hanem egy ugyanakkora méretű „képet” adjon vissza. (Lásd még a 4 hét/Perceptron osztály feladatot is.)

Megoldás videó:

Megoldás forrása:

Ebben a feladatunkban nem massal mint a Helló ,Casear fejezetből már megismert perceptronnal kell dolgoznunk. Ebben a helyzetben ezzel a programmal egy képet atalkított képet szeretnénk visszakapni. Alul latható mar az átalíktott kód. Fontos kiemelni hogy a perceptron függvényt érdemes nem 1 értékre meghívni hogy eszlelhessük a különbséget . Mit is lathatunk benne Egyszerű a visszatérési érték megváltozik mostmar double \* így már több értékkel tér vissza a függvény. A main függvényen belül atírára kerültek a meghívások. Továbbá lathatjuk 2 ciklus segítségével vegigmegyünk a kép adatain amelyek nem masik mint

a meretei hossza és szélessége illetve a pixelek száma . Ahogy lathatjuk a vannak piros és zöld részek ezek nem mast mint az eredeti kep mereit tartalmazza a harmadik szín a kek az pedig nem mast mint a kiszamolt értékeket tartalmazza . Ezek segítségével vagyunk képesek létrehozni az uj kepunket.

```
double* operat<or> ( double image [] )  
{  
    /* ... */  
    for(int i = 0; i < n_units[n_layers - 1]; i++)  
        image[i] = units[n_layers - 1][i];  
    return image;  
}  
  
int main ( int argc, char *argv[] )  
{  
    image <rgb_pixel> png_image (argv[1]);  
    image <rgb_pixel> new_png_image(png_image.get_width(), png_image. ←  
    get_height());  
    int size = png_image.get_width() * png_image.get_height();  
  
    Perceptron* p = new Perceptron(3, size, 256, size);  
  
    double* image = new double[size];  
  
    for (int i{0}; i < png_image.get_width(); i++)  
        for (int j{0}; j < png_image.get_height(); j++)  
            image[i * png_image.get_width() + j] = png_image[i][j].red;  
    double* value = (*p)(image);  
    for (int i = 0; i < png_image.get_width(); i++)  
        for (int j = 0; j < png_image.get_height(); j++) {  
            new_png_image[i][j].red = png_image[i][j].red;  
            new_png_image[i][j].green = png_image[i][j].green;  
            new_png_image[i][j].blue = value[i * png_image.get_width() + j];  
        }  
  
    new_png_image.write("uj_mandel.png");  
    delete p;  
    delete [] image;  
}
```

# 16. fejezet

## Helló, Gödel!

### 16.1. Gengszterek

Gengszterek rendezése lambdával a Robotautó Világbajnokságban <https://youtu.be/DL6iQwPx1Yw> (8:05-től)

Megoldás videó:

Megoldás forrása:

Tehát a feladatunk a gengszterek rendezéses lesz lambdával, nem mashol mint a robotautó világbajnokságban. Ahogy lathatjuk a programban elsőként az adatok beolvasása ezt a std::sscanf függvény végzi el számunkra. maguk az az adatok egy vektorba kerülnek. Nevszerint a gansters vektorba . Nyilvan a programban majd eztkell rendeznunk. Miutan megadtuk az elejet és a végét kezdődhet a rendezés a fentebb már említett lambda segítségével. A program kiszámolja a 2 gangster közötti távolsagot és ez lapjan rendezi őket.

```
struct SmartCar
{
    int id;
    unsigned from;
    unsigned to;
    int step;
};

typedef SmartCar Gangster;
std::vector<Gangster> gangsters ( boost::asio::ip::tcp::socket & socket ←
    ,
    int id, osmium::unsigned_object_id_type cop );

}

int idd {0};
unsigned f, t, s;
int n {0};
int nn {0};
```

```

    std::vector<Gangster> gangsters;

}

while ( std::sscanf ( data+nn, "<OK %d %u %u %u>%n", &idd, &f, &t, &s, &n
) == 4 )
{
nn += n;
gangsters.push_back ( Gangster {idd, f, t, s} );
}
std::sort ( gangsters.begin(), gangsters.end(), [this, cop] ( Gangster x,
Gangster y )
{
return dst ( cop, x.to ) < dst ( cop, y.to ); }
);

}

```

## 16.2. Alternatív Tabella rendezése

Mutassuk be a [https://progpater.blog.hu/2011/03/11/alternativ\\_tabella](https://progpater.blog.hu/2011/03/11/alternativ_tabella) a programban a java.lang Interface Comparable T szerepét!

Megoldás videó:

Megoldás forrása:

Tehát a program elkezdése előtt erdemessé frissíteni az adatokat hiszen fociról beszélünk ezen belül e csodás NB1ről ahol par ev alatt jópar csapat cserelődött illetve "stadion" sőt raadasul meg bajnokság letszama is megcsappant hisz 16ról 12csapatra esett a letszam de sajnos a fujpest meg bent van... Szóval ha végeztünk a frissítéssel . Fel kell töltenünk a kódban szereplő kereszttáblát. 1,2 ,3 0 értékekkel . Ezek a győzelmet/-vereséget/dontetlen jelentik megkülönboztetve az ideges es hazai pályát A csapatok rendezése a "Csapat" osztályban történik . Az ertekeket pedig a compareto függvény rendezi illetve ez is hasonlíta össze a csapatokat . Az osszehasonlítás során a függvény ertekeit ad vissza ez lehet -1 , 0 és 1 . Ez az ertek befolyásolja a sorrendben elfoglalt helyet. .

```

public static void main(String[] args) {
int[][] kereszt = {
{ 0, 1, 1, 3, 1, 1, 2, 1, 1, 1, 2, 3 },
{ 1, 0, 1, 1, 2, 1, 2, 2, 1, 1, 2, 3 },
{ 1, 1, 0, 3, 1, 2, 2, 1, 2, 2, 3, 3 },
{ 1, 2, 1, 0, 1, 1, 1, 2, 1, 1, 2 },
{ 3, 3, 2, 1, 0, 3, 3, 3, 1, 2, 3 },
{ 3, 1, 2, 3, 1, 0, 3, 1, 2, 1, 3, 2 },
{ 3, 2, 1, 3, 1, 2, 0, 3, 1, 1, 2, 1 },
{ 2, 3, 1, 3, 1, 3, 2, 0, 3, 1, 1, 3 },
{ 2, 1, 3, 3, 2, 1, 1, 0, 1, 2, 3 },
{ 1, 3, 1, 1, 1, 2, 1, 3, 2, 0, 3, 1 },

```

```
{ 2, 1, 1, 2, 1, 1, 2, 3, 2, 1, 0, 1 },
{ 1, 2, 3, 1, 1, 1, 3, 2, 3, 1, 0 } };

};

int[][] pontotSzerez = new int[kereszt.length][kereszt.length];

for (int i = 0; i < pontotSzerez.length; ++i) {
    for (int j = 0; j < pontotSzerez[i].length; ++j) {
        pontotSzerez[i][j] = 0;
    }
}

for (int i = 0; i < pontotSzerez.length; ++i) {
    for (int j = 0; j < pontotSzerez[i].length; ++j) {

        if (kereszt[i][j] == 1) { // zold
            ++pontotSzerez[i][j];
        } else if (kereszt[i][j] == 2) { // sarga
            ++pontotSzerez[i][j];
            ++pontotSzerez[j][i];
        } else if (kereszt[i][j] == 3) { // piros
            ++pontotSzerez[j][i];
        } else if (kereszt[i][j] == 0) { // őres
            ;
        } else {
            System.out.println("Luke, zavart trzek az ervben... ");
        }
    }
}

System.out.println("A x ontot-szerez y-től matrix");
nyom(pontotSzerez);

System.out.println("\nSor is oszlop osszegekkel");

nyom2(pontotSzerez);

int oszlopOsszeg[] = new int[pontotSzerez.length];
```

```
for (int i = 0; i < pontotSzerez.length; ++i) {

    int o = 0;
    for (int j = 0; j < pontotSzerez[i].length; ++j) {
        o += pontotSzerez[j][i];
    }
    oszlopOsszeg[i] = o;
}

System.out.println("\nA \"link\" matrix");

nyom3(pontotSzerez, oszlopOsszeg);

}

public static void nyom(int[][] k) {
    for (int i = 0; i < k.length; ++i) {
        System.out.println();
        for (int j = 0; j < k[i].length; ++j) {
            System.out.print(k[i][j] + ", ");
        }
    }
}

public static void nyom2(int[][] k) {

    for (int i = 0; i < k.length; ++i) {
        System.out.println();
        int o = 0;
        for (int j = 0; j < k[i].length; ++j) {
            System.out.print(k[i][j] + ", ");
            o += k[i][j];
        }
        System.out.print(" " + o);
    }

    System.out.println();
    System.out.println();

    for (int i = 0; i < k.length; ++i) {

        int o = 0;
        for (int j = 0; j < k[i].length; ++j) {
            o += k[j][i];
        }
        System.out.print(o + " ");
    }

}
```

```

public static void nyom3(int[][] k, int[] oszlopOsszeg) {
    for (int i = 0; i < k.length; ++i) {
        System.out.println();
        System.out.print("{");
        for (int j = 0; j < k[i].length; ++j) {

            if (oszlopOsszeg[j] != 0.0) {
                System.out.print(k[i][j] * (1.0 / oszlopOsszeg[j]) + ", ");
            } else {
                System.out.print(k[i][j] + ", ");
            }
        }
        System.out.print("}, ");
    }
}
}

```

## 16.3. GIMP Scheme hack

Ha az előző félévben nem dolgoztad fel a témat (például a mandalás vagy a króm szöveges dobozosat) akkor itt az alkalom!

Megoldás videó:

Megoldás forrása:

Tehát feladatunkban elsőkent a scriptet be kell masolnunk a GIMP script mappajában. Mert enekül nem fogja erzekelni . Ha bemasoltuk lathatjuk hogy a kódban nagyon sok lehetőségünk van. Tehát kedvünkre alakíthatjuk a szöveget alakíthatjuk a magát a szöveget akár a színet színátmennetett illetve ugyanerre kepesek vagyunk a hatterrel is. A kódban lathatjuk a dolgokat tehát h ebben az esetben a szövegünk feher színu lesz és fekete a hatter . Továbbá lathatjuk h rengeteg dolgot alakíthatunk tehát van lehetőség a szöveg elmosására és különboző efektek használatara

```

(define (color-curve)
  (let* (
    (tomb (cons-array 8 'byte))
  )
    (aset tomb 0 0)
    (aset tomb 1 0)
    (aset tomb 2 50)
    (aset tomb 3 190)
    (aset tomb 4 110)
    (aset tomb 5 20)
    (aset tomb 6 200)
    (aset tomb 7 190)
  )
)

```

```
tomb)
)

; (color-curve)

(define (elem x lista)

  (if (= x 1) (car lista) (elem (- x 1) (cdr lista) ) )

)

(define (text-wh text font fontsize)
(let*
  (
    (text-width 1)
    (text-height 1)
  )

  (set! text-width (car (gimp-text-get-extents-fontname text fontsize ←
    PIXELS font)))
  (set! text-height (elem 2 (gimp-text-get-extents-fontname text ←
    fontsize PIXELS font)))

  (list text-width text-height)
  )
)

; (text-width "alma" "Sans" 100)

(define (script-fu-bhax-chrome text font fontsize width height color ←
  gradient)
(let*
  (
    (image (car (gimp-image-new width height 0)))
    (layer (car (gimp-layer-new image width height RGB-IMAGE "bg" 100 ←
      LAYER-MODE-NORMAL-LEGACY)))
    (textfs)
    (text-width (car (text-wh text font fontsize)))
    (text-height (elem 2 (text-wh text font fontsize)))
    (layer2)
  )

; step 1
(gimp-image-insert-layer image layer 0 0)
(gimp-context-set-foreground '(0 0 0))
(gimp-drawable-fill layer FILL-FOREGROUND )
(gimp-context-set-foreground '(255 255 255))

(set! textfs (car (gimp-text-layer-new image text font fontsize PIXELS) ←
  ))
```

```
(gimp-image-insert-layer image textfs 0 0)
(gimp-layer-set-offsets textfs (- (/ width 2) (/ text-width 2)) (- (/ ←
height 2) (/ text-height 2)))

(set! layer (car(gimp-image-merge-down image textfs CLIP-TO-BOTTOM- ←
LAYER)))

;step 2
(plug-in-gauss-iir RUN-INTERACTIVE image layer 15 TRUE TRUE)

;step 3
(gimp-drawable-levels layer HISTOGRAM-VALUE .11 .42 TRUE 1 0 1 TRUE)

;step 4
(plug-in-gauss-iir RUN-INTERACTIVE image layer 2 TRUE TRUE)

;step 5
(gimp-image-select-color image CHANNEL-OP-REPLACE layer '(0 0 0))
(gimp-selection-invert image)

;step 6
(set! layer2 (car (gimp-layer-new image width height RGB-IMAGE "2" 100 ←
LAYER-MODE-NORMAL-LEGACY)))
(gimp-image-insert-layer image layer2 0 0)

;step 7
(gimp-context-set-gradient gradient)
(gimp-edit-blend layer2 BLEND-CUSTOM LAYER-MODE-NORMAL-LEGACY GRADIENT- ←
LINEAR 100 0 REPEAT-NONE
    FALSE TRUE 5 .1 TRUE width (/ height 3) width (- height (/ height ←
3)))

;step 8
(plug-in-bump-map RUN-NONINTERACTIVE image layer2 layer 120 25 7 5 5 0 ←
0 TRUE FALSE 2)

;step 9
(gimp-curves-spline layer2 HISTOGRAM-VALUE 8 (color-curve))

(gimp-display-new image)
(gimp-image-clean-all image)
)
)

;(script-fu-bhax-chrome "Bátf41 Haxor" "Sans" 120 1000 1000 '(255 0 0) " ←
Crown molding")

(script-fu-register "script-fu-bhax-chrome"
  "Chrome3"
  "Creates a chrome effect on a given text."
```

```
"Norbert Bátfai"
"Copyright 2019, Norbert Bátfai"
"January 19, 2019"
""

SF-STRING      "Text"      "Bátf41 Haxor"
SF-FONT        "Font"       "Sans"
SF-ADJUSTMENT "Font size" '(100 1 1000 1 10 0 1)
SF-VALUE       "Width"     "1000"
SF-VALUE       "Height"    "1000"
SF-COLOR       "Color"     '(255 0 0)
SF-GRADIENT    "Gradient"  "Crown molding"
)

(script-fu-menu-register "script-fu-bhax-chrome"
 "<Image>/File/Create/BHAX"

}
```

DRAFT

## 17. fejezet

### Helló, !

#### 17.1. OOCWC Boost ASIO hálózatkezelése

Mutassunk rá a scanf szerepére és használatára! <https://github.com/nbatfai/robocaremulator/blob/master/justine/ro>

Megoldás videó:

Megoldás forrása:

A következő feladatban az OOCW Boost ASIO hálózatkezelésről és azon belül is a sancf szereperől fogunk beszélni. A programban tcpre jönnek a csomagok a amelyeket a scanf kezel . A snacf egy fajlkezelő függvény melynek az a szerepe hogy beolvasás az eppen adott inputból . Illetve ezt a amit kapunk adatot eltaroljuk a megfelelő helyre az adott input tipusanak alapjan.

```
std::sscanf(yytext, "<pos %d %u %u", &m_id, &from, &to)  
}
```

Az inputbol kiiratjuk a megfelelő pozíciót amiket az m\_id a from és a to referenciaértékéből kapunk meg, Az első decimalis ( %d) a masik kettő( %u %u) pedig unsigned éréket jelöl

#### 17.2. SamuCam

Mutassunk rá a webcam (pl. Androidos mobilod) kezelésére ebben a projektben: <https://github.com/nbatfai/Samu>

Megoldás videó:

Megoldás forrása:

Samucam ! Tehát ahoz hogy ezzel tudjunk dolgozni telepítésekre van szükségünk. Telepítenünk kell a Qt-t , majd egy parancsot kiadni amely ezutan general szamunkra egy maket-et hogy tudjuk futtatni a samucamot. A programunk futtatása egy specialis parancsal törtenik hisz ahoz hogy mukodjon a camunk . Peldaul ha IP kamerát akarunk használni --ip kapcsolra van szükségünk. Nezzük meg a kódot pontosabban a SamuCam.cpp . A programban a videotream alltal megnyitja a streamet az eltarolt ipn keresztt . Ezutan lathatjuk hogy meg tudjuk adni a stream mereteit szelességet , magassagat . Illetve be tudjuk az fpst is 10re. Ha a program sikeresen fut akkor minden egyes képet eltarol majd ezek után ezeken fogja keresni az arcokat. A talált arcokat eltarolja és vegül ezekból készít el minden.

```
#include "SamuCam.h"

SamuCam::SamuCam ( std::string videoStream, int width = 176, int height = ←
    144 )
    : videoStream ( videoStream ), width ( width ), height ( height )
{
    openVideoStream();
}

SamuCam::~SamuCam ()

{
}

void SamuCam::openVideoStream()

{
    videoCapture.open ( videoStream );

    videoCapture.set ( CV_CAP_PROP_FRAME_WIDTH, width );
    videoCapture.set ( CV_CAP_PROP_FRAME_HEIGHT, height );
    videoCapture.set ( CV_CAP_PROP_FPS, 10 );

}

void SamuCam::run()

{
    cv::CascadeClassifier faceClassifier;
    std::string faceXML = "lbpcascade_frontalface.xml"; // https://github.com ←
        /Itseez/opencv/tree/master/data/lbpcascades
    if ( !faceClassifier.load ( faceXML ) )

    {
        qDebug() << "error: cannot found" << faceXML.c_str();
        return;
    }

    cv::Mat frame;
    while ( videoCapture.isOpened() )
```

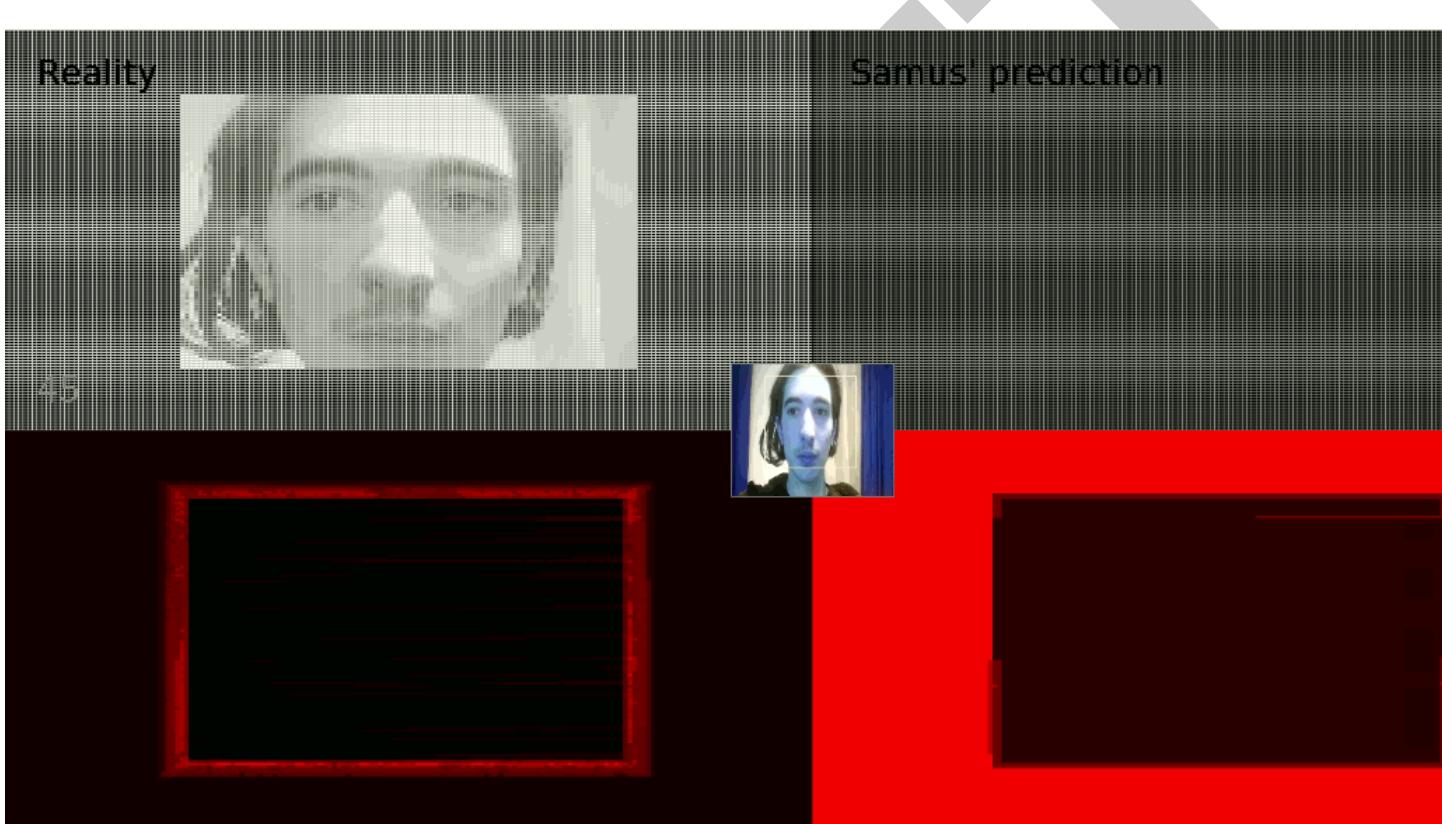
```
{  
    QThread::msleep ( 50 );  
    while ( videoCapture.read ( frame ) )  
  
    {  
  
        if ( !frame.empty() )  
        {  
  
            cv::resize ( frame, frame, cv::Size ( 176, 144 ), 0, 0, cv::INTER_CUBIC );  
            std::vector<cv::Rect> faces;  
            cv::Mat grayFrame;  
  
            cv::cvtColor ( frame, grayFrame, cv::COLOR_BGR2GRAY );  
            cv::equalizeHist ( grayFrame, grayFrame );  
  
            faceClassifier.detectMultiScale ( grayFrame, faces, 1.1, 4,  
                0, cv::Size ( 60, 60 ) );  
  
            if ( faces.size() > 0 )  
            {  
  
                cv::Mat onlyFace = frame ( faces[0] ).clone();  
  
                QImage* face = new QImage ( onlyFace.data,  
                    onlyFace.cols,  
                    onlyFace.rows,  
                    onlyFace.step,  
                    QImage::Format_RGB888 );  
  
                cv::Point x ( faces[0].x-1, faces[0].y-1 );  
                cv::Point y ( faces[0].x + faces[0].width+2, faces[0].y +  
                    faces[0].height+2 );  
                cv::rectangle ( frame, x, y, cv::Scalar ( 240, 230, 200 ) );  
  
                emit faceChanged ( face );  
            }  
            QImage* webcam = new QImage ( frame.data,  
                frame.cols,  
                frame.rows,  
                frame.step,  
                QImage::Format_RGB888 );  
  
            emit webcamChanged ( webcam );  
        }  
  
        QThread::msleep ( 80 );  
    }  
}
```

```
        }

        if ( ! videoCapture.isOpened() )
        {
            openVideoStream();
        }

    }

}
```



17.1. ábra. Forrás : Sajat kép

### 17.3. BrainB

Mutassuk be a Qt slot-signal mechanizmust ebben a projektben: <https://github.com/nbatfai/esporttalents-search>

Megoldás videó:

Megoldás forrása:

TA BrainB nemmas mint egy Benchmark azaz egy mérés . Amely nem mast mint a kulonbozo jatekokat probalja szimulalni és lemerni hogy mennyire vagyunk kepesek figyelni a karakterunkre egy tomegben .

Ez egy eleg gyakori dolg hisz egy MMORPGben vagy MOBAban , RTSben vagy egyéb ehez hasonló jatekokban gyakran előfordulhat hogy a spellek es a rengeteg karakter , esemény miatt elveszítjuk a karakterünket és ezáltal előnytelen pozícióba kerülünk vagy eppen az ellenfelet nem talaljuk ... Szóval ezzel a programmal tudjuk mérni az egyeni fokuszáló képességünket. 10 percig kell a megfelelő négyzetet követnunk az egérrel . A feladatunk pontosan a Qt slot-signal mechanizmusának bemutatása Maga a QT a slot-signal az objektumok kozotti kommunikaciért felel. Ez nem mas mint mikor valamit változtatunk az egyik objektumba ezzel a mechanizmussal jelezük a többi objektumnak. Több ilyen funkció is található ilyen például az emit vagy eppen a connect

ilyeneket találhatunk több fajlban is például a BrainBWin.cppben connect ( brainBThread, SIGNAL ( heroesChanged ( QImage, int, int ) ),

connect ( brainBThread, SIGNAL ( heroesChanged ( QImage, int, int ) ),

vagy a brainBthread.cppben

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

```
#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;

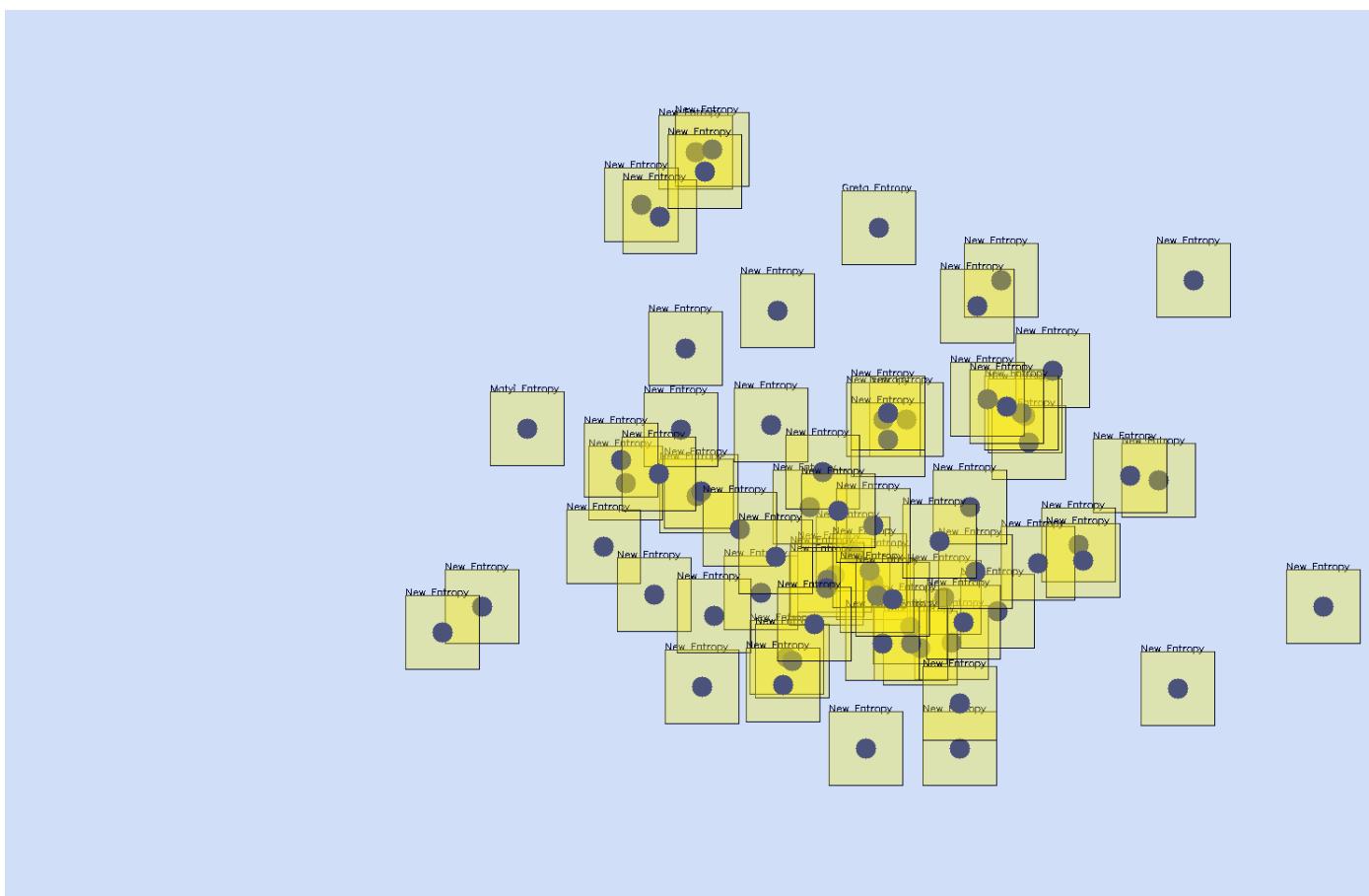
    qpainter.drawPixmap ( xs, ys, pixmap );
    qpainter.drawText ( 10, 20, "Press and hold the mouse button on the ←
                        center of Samu Entropy" );

    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 );\n\n    if ( brainBThread->get_paused() ) {\n        QString pausedstr = "PAUSED (" + QString::number ( ←\n            brainBThread->get_nofPaused() ) + ")";\n\n        qpainter.drawText ( 210, 40, pausedstr );\n    }\n    qpainter.end();\n}\n\nvoid BrainBWin::mousePressEvent ( QMouseEvent *event )\n{\n    brainBThread->set_paused ( false );\n}\n\nvoid BrainBWin::mouseReleaseEvent ( QMouseEvent *event )\n{\n    //brainBThread->set_paused(true);\n}\n\nvoid BrainBWin::mouseMoveEvent ( QMouseEvent *event )\n{\n    start = true;\n\n    mouse_x = event->pos().x() -xs - 60;\n    //mouse_y = event->pos().y() - yshift - 60;\n    mouse_y = event->pos().y() - ys - 60;\n}\n\nvoid BrainBWin::keyPressEvent ( QKeyEvent *event )\n{\n    if ( event->key() == Qt::Key_S ) {\n        save ( brainBThread->getT() );\n    } else if ( event->key() == Qt::Key_P ) {\n        brainBThread->pause();\n    } else if ( event->key() == Qt::Key_Q || event->key() == Qt::←\n        Key_Escape ) {\n        close();\n    }\n}\n\nBrainBWin::~BrainBWin()\n{
```

{



17.2. ábra. BrainB

## 18. fejezet

# Helló, Schwarzenegger!

### 18.1. Port scan

Mutassunk rá ebben a port szkennelő forrásban a kivételkezelés szerepére! <https://www.tankonyvtar.hu/hu/tartalom/tanitok-javat/ch01.html#id527287>

Megoldás videó:

Megoldás forrása:

Tehát feladatunk a Port scan forrásban lévő kivételkezelések megvizsgálása majd bemutatása. Ahogy láthatjuk a kód javaban született . Szerintem elsonek erdemessé beszélni magáról a kivettelekről . A kivetel nem más mint egy esemény amely a program futása során keletkezik és megszakítja az utasítások végrehajtásának normális folyamatát .A kivetteleket többféle módon is kezelhetjük akár eldobhatjuk oket (Throw ) vagy elkapthatjuk őket (catch) Maga a kivetelkezelés 3 blokk segítségével történik ez a három blokk nem más mint a TryaBlokk a catch blokk illetve a Finally block. Az első lépés nem más mint a kód elhelyezése a megfelelő helyen tehát a tryBlokkban Miutan ezzel megvagyunk és szeretnénk a bekovetkezett kivetelt lekezelni szükségünk van egy Catch blokkra amit hozzá kell kapcsolnunk a Try blokkunkhoz. A catch maga egy kivetelkezelő , rendelkezik egy paraméterrel , tehát azon tipusu kivetelt képes kezelní amelyen paraméterrel rendelkezik. A kivetelkezelők többet is képesek tenni mint egy hibauzenet képesek akár hibajavításra is. Végül a Finally blokk kovetkezik mely rengeteg szereppel rendelkezik gyakorlatilag a "takarító " szerepkört látja be . Peldaul használhatjuk fajlok bezárására amelyekre már nincs szükségünk.

```
public class KapuSzkenner {  
    public static void main(String[] args) {  
        for(int i=0; i<1024; ++i)  
  
            try {  
                java.net.Socket socket = new java.net.Socket(args[0], i);  
                System.out.println(i + " figyeli");  
                socket.close();  
            } catch (Exception e) {  
                System.out.println(i + " nem figyeli");  
            }  
    }  
}
```

}

A kódban a fentebb említett blokkok közül kettő található egy Try es ebben egy catch . Lathatjuk hogy a tryal probaljuk a kívételt illetve kiiratjuk az it + egy üzenetet ami nem mas mint a "Figyeli" Illetve lathatjuk a catch reszt amelyel kezeljuk a kivetelt nyilvan ha idekerülünk itt is van egy üzenet hasonlóan az előzőhöz kiiratjuk az i értéket illetve a "Nem figyeli " üzenetet

## 18.2. AOP

Szőj bele egy átszövő vonatkozást az első védési programod Java átiratába! (Sztenderd védési feladat volt korábban.)

Megoldás videó:

Megoldás forrása:

Tehát az AOP vagyis Aspektus Orientált programozás nem mas mint egy folyamat amelynek a segítségével könnyebb lesz a kód olvashatosága. Maga az AOP az OOP egy továbbfejlesztett változata. Az en megoldasomban azt lhathatjuk hogy egy AOP scriptbe kerül bele a binfa java valtozata . Ahogy lathatjuk a kódban az események a Meghívásoknal fognak keletkezni tehát a program a kiir függvény meghívására reagál . Meghívás előtt kiirja hogy meghívjuk miutan meghívtuk pedig kiirja hogy meghívtuk ezt a before es after figyelő segítséggel csinalja . illetve ezeken felül a progrtam termeszetesen írja hogy a futás során hányszor hívjuk meg illetve a ha a futás során keletkeztek hibák azokról is kapunk üzenetet. A kódban lathatjuk hogy a fentebb említett számolásra van 2 darab int valtozonk .

```
public aspect kiirMeghiv {  
    public pointcut fgv(): call(public void LZWBInFa.Csomopont.kiir());  
    int bef = 0;  
    int aft = 0;  
    before(): fgv() {  
        bef++;  
        System.out.println("kiir indul");  
    }  
    after(): fgv() {  
        aft++;  
        System.out.println("kiir lefutott");  
    }  
}
```

## 18.3. Junit teszt

A [https://progpater.blog.hu/2011/03/05/labormeres\\_otthon\\_avagy\\_hogyan\\_dolgozok\\_fel\\_egy\\_pedat\\_poszt\\_kézzel\\_számított\\_mélységet\\_és\\_szórását\\_dolgozd\\_be\\_egy\\_Junit\\_tesztbe](https://progpater.blog.hu/2011/03/05/labormeres_otthon_avagy_hogyan_dolgozok_fel_egy_pedat_poszt_kézzel_számított_mélységet_és_szórását_dolgozd_be_egy_Junit_tesztbe) (sztenderd védési feladat volt korábban).

Megoldás videó:

Megoldás forrása:

Következő feladatunk nem más mint a Junit teszt lesz . Ebben a feladatban a posztban kézzel számított mélységet és szórasat kell egy Junit tesztben megcsinalnunk. A junit teszt nem más mint egy egységeszt keretrendszer eredetileg javához de mostmar portolasok utjan a legtobb nyelven elérhető . A feladatunk hogy ezt a tesztet össze rakjuk a binfa védési programba és a kapott illetve előre kiszamolt értékeket össze kell hasonlítani . A "@Test" használataval állítjuk be hogy merre készítse a tesztet . A feladatban fel kell vennünk a 01111001001001000111 egy változóba majd vegig kell mennünk minden karakteren , vegül visszadunk a fanak ezutan kell kiszamolnunk a megfelelő értékeket amelyek nem lesznek masok mint a getAtlag a getSzoras és a getMelyseg . Ha megvagyunk a szamolassal a mar meglevő értékekkel kell összehasonlítanunk . Nyilvan itt kiderül hogy a teszt helyes volt e vagy nem

```
LZWBinFA binfa = new LZWBinFa();
String value = "01111001001001000111";

@Test
public void atlagTest() {
    Univerzális programozás 137 / 143

    for(int i = 0; i < value.length(); i++)
        binfa.push_back(str.charAt(i));
    double atlag = binfa.getAtlag();
    assertEquals(2.75, atlag, 0.001);
}

@Test
public void szorasTest() {
    for(int i = 0; i < value.length(); i++)
        binfa.push_back(str.charAt(i));

    double szoras = binfa.getSzoras();
    assertEquals(0.9574271077563381, szoras, 0.001);
}

@Test
public void melysegTest(){ for(int i = 0; i < value.length(); i++)
    binfa.push_back(str.charAt(i));

    double melyseg = binfa.getMelyseg();
    assertEquals(1040, melyseg, 0.001);
}
```

# 19. fejezet

## Helló, Calvin!

### 19.1. MNIST

Az alap feladat megoldása, +saját kézzel rajzolt képet is ismerjen fel, [https://progpater.blog.hu/2016/11/13/hello\\_sbol](https://progpater.blog.hu/2016/11/13/hello_sbol) Háttérként ezt vetítsük le: <https://prezi.com/0u8ncvvoabcr/no-programming-programming/>

Megoldás videó:

Megoldás forrása:

Az MNIST egy nagyon erdekes program amely a fejezet többi feladatahoz hasonloan képes felismerni valamit, ebben az esetben a valami a rajzolt számokat takarja . A megfelelő futáshoz többek között szükségünk lesz a tensorflow könytárra is. A programnal minél több futtatásra van szüksége így képes "tanulni". Tehát a futások számával nő a program pontossága . Maga a progam 28x28 as kepekkel dolgozik . Tehát ekkora felbontasú kepeket kell szereznunk vagy eppen készítenünk .

```
for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
    if i % 100 == 0:
        print(i/10, "%")
}
```

Tehát a programnak több része van az első a tanulas resze itt a program iterációkat vegez szamszerint ezret es ezt százalékos formában jelzi a felhasználónak tehát hogy a tanulas eppen hogy áll.

```
img = mnist.test.images[23]
image = img

matplotlib.pyplot.imshow(image.reshape(28, 28), cmap=matplotlib.pyplot.cm.binary)

matplotlib.pyplot.savefig("4.png")
matplotlib.pyplot.show()

classification = sess.run(tf.argmax(y, 1), feed_dict={x: [image]})
```

```
    print("<!-- Ezt a halozat ennek ismeri fel: ", classification[0])<br/>}
```

Ha a tanulas folyamatat befejezük elkezdhetünk a programnak kepeket adni

## 19.2. CIFAR-10

Az alap feladat megoldása, +saját fotót is ismerjen fel, [https://progpater.blog.hu/2016/12/10/hello\\_samu\\_a\\_cifar-10\\_tf\\_tutorial\\_peldabol](https://progpater.blog.hu/2016/12/10/hello_samu_a_cifar-10_tf_tutorial_peldabol)

Megoldás videó:

Megoldás forrása:

Tehát a CIFAR-10 ez ne mmas mint a többi programhoz hasonloan egy TensorFlowos pelda . Viszont a MNITSel ellentetben itt nem csak szamokat lesz képes felismerni a programunk. Tehát itt nem csak szamokat hanem fotokat es a fotokon szereplő dolgokat képes felismerni a program. A program az 32x32 pixel méretű képekkel dolgozik megfelelően . Nyilvan mas nem megfelelő felbontású kepekkal is mukodhet csak termeszetesen nem megfelelően. Maga a CIFAR-10 is rendelkezik rengeteg alap keppel . Viszont termeszetesen mi is adhatunk neki saját kepeket a tanulashoz. A programot a megfelelő mukodeshez , felismeréshez rengetegszer kell futtatnunk hiosz így futtatásról futtatásra okosabb lesz es képes lesz felismerni a kepunket.

Mivel a program binaris fajlokkal képes dolgozni emiatt a kepeinket is ilyen formaba kell hoznunk hisz kizárolag csak ilyen fajlokkal dolgozik a program. Ehez nincs is mas dolgunk mint a feladat leírásában található blog elolvása es az ott található program hasznalata .

```
from PIL import Image  
import numpy as np  
  
im = Image.open('images.jpeg')  
im = (np.array(im))  
  
r = im[:, :, 0].flatten()  
g = im[:, :, 1].flatten()  
b = im[:, :, 2].flatten()  
label = [1]  
  
out = np.array(list(label) + list(r) + list(g) + list(b), np.uint8)  
out.tofile("out.bin")  
}
```

## 19.3. Android telefonra a TF objektum detektálója

Telepítük fel, próbáljuk ki!

Megoldás videó:

Megoldás forrása:

Tehát a feladatban a tensorflowt fogjuk használni, Maga a tensorflow egy nagyon érdekes program amely többek között képes objektumfelismerésre mi ezt fogjuk kihasználni. A programot egyszerűen meg lehet szerezni a például a Google play segítségével képesek vagyunk egyszerűen letölteni és használni. A program a kamerán keresztül képes tárgyakat felismerni. A felismerés után egy színes negyszöget rajzol majd kiirja alá hogy a program alapjan mi latható a képen. A program könnyedén képes mindenfele tárgyat, embert állatot felismerni kisebb-nagyobb elteressel. Szerintem egy par-perces használat után már lathatjuk hogy több esetben is nem megfelelő eredményt ad a program. Tehát például az xbox360ot egernek :D az egerem dobozat pedig laptopnak erzekelte illetve a mikrofonomat repülőnek. Illetve a szekrény mellett nem a szekrényt vette eszre hanem kiírta hogy person.

Ezek a kapon egy kiflit érzékelt macskának

DRAFT



mondjuk szerintem így nez ki egy macska...

DRAFT



# 20. fejezet

## Bernes Lee

### 20.1. C++ és Java összehasonlítás

C++: Benedek Zoltán, Levendovszky Tihamér Szoftverfejlesztés C++ nyelven

Java: Nyékyné Dr. Gaizler Judit et al. Java 2 útikalauz programozóknak 5.0 I-II

Megoldás videó:

Megoldás forrása:

java es a c/c++ talan korunk legmeghatározobb program nyelvei . A java c/c++ sok mindenben hasonlít egymásra és sok mindenben különbozik is. A javában a biztonság nagyon nagy szerepe van emiatt sok helyen peldaúl a biztonságot nem pedig a gyorsaságot tarották szem előtt. Maga a java hírnevet eredetileg a WWW oldalokon betoltott szerepenek köszönheti. java egy platform és op.rendszer független nyelv . Nincsenek benne pointerek illetve többszörös öröklés sem illetve velemenem szerint sokkal objektumorientáltabb mint a c++ , a java nyelvet szoktak teljesen objektumorientáltnak is nevezni. Ezzel ellentétben a c++ egy platformfüggő nyelv itt is beszélhetünk objektumorientáltsagról de mégse olyan mértékben mint a java esetében . Futási szempontbol is vannak különbözők ahogy már fentebb említettem a c/c++ és a java között mivel a java a forráskódot a java virtualis gép önálló iterpreterkent fog ertelmezni. Ez biztonsági szempontból előny viszont sebesség szempontjából hátrány. A javában egy konstans felvételéhez a Final kulcsszóra van szüksegünk ezzel ellentétben a c++/cben a const szót kell alkalmaznunk. Több dolog is van a c++ban mint peldaúl a javába ilyen peldaúl az operátortulterheles ilyen a goto ami az ugrásokhoz fontos az utóbbi foleg biztonsági szempontból nem használja a java. Megjegyzéseket igazából minden nyelvben használhatunk ebben nincs eltérés. a . Referencia szempontjából hasonló a történet hiszen minden a c/c++-ba minden a javába van referencia Minden nyelvben találkozhatunk az alapvető változókkal ilyen peldaúl char , short vagy az int. Ha a karakterkészletet megnezzük eleg nagy a különböző a java hiszen a legtöbb nyelvel ellentétben nem 8 bites rendszert használ tehát akar a magyart is használhatjuk még a c/c++-ba erre nincs lehetőség

### 20.2. Python

Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba. Gyors prototípusfejlesztés Python és Java nyelven (35-51 oldal), a kijelölt oldalakból élmény-olvasónapló

Megoldás videó:

Megoldás forrása:

A python egy általános célú programozási nyelv . létrejöttét Guido Van Rossumnak köszönhetjük. Maga a nyelv 1990ben született meg , és azóta is töretlen sikerrel bír. A python maga egy viszonylag nepszerű nyelv hisz rengeteg kiváló tulajdonsággal rendelkezik . Ilyen például az is hogy más nyelveken megírt modulokkal is kiválóan képes együtt dolgozni, vagy nincs szükség fordítási fázisra hisz az értelmezőnek elegendő a Python forrását megadni . Kiemelném hogy a python legelszerűbb felhasználása az ha prototípusok vagy tesztelesi esetekben használjuk. Hisz a legtöbb nyelvel ellentetbe a hibakezelése és a hibák javítása egyszerűbb. A nyelv szintaxisa behúzásos ami röviden azt takarja hogy a programba lévő állításokat az azonos szintű behúzásokkal csoportosítjuk , tehát nem használunk kulcsszavakat vagy egyéb eszközököt. Egy résznek a veget egy kisebb behúzású sor jelzi . Fontos kiemelni hogy a sor első utasítása nem lehet behúzott. A nyelvben egy utasítás a sor végéig tart , ha több sorba szeretnék dolgozni a /-t kell használnunk A ptyhonba beépített értemlező a sorokat tokenekre bontja ez lehet akár azonosító , kulcsszó operátor delimitér , literál. a nyelven belül megkülönboztetjük a kis és nagy betűket illetve a nyelv tartalmaz különbozo előre lefoglalt kulcsszavakat . Ilyen például az and , a break a a def es egy csomó másik szó is. A nyelvben minden adatot objektumokkal reprezentálnak . Az hogy egy adattal miféle műveletet tudunk elvégezni a típus határozza meg . Amely ebben a nyelvben adott , tehát automatikusan vegzi a nyelv . Maguk az adattípusok lehetnek számok sztringek , ennesek , listák és szótárak. A számok lehetnek akár egészek , komplexek decimálisok oktálisak és egyéb mások. A sztringeket idézőjelek , aposztrofok vagy az u betű segítségével tudjuk felvenni.A pythonban a változokat objektumok referenciajaként tudjuk értelmezni. értéket az = segítségével tudunk adni számukra illetve a del segítségével vagyunk képesek törölni őket. A nyelvben 2fele változót különboztetünk meg beszélhetünk globális és lokális változóról is . Ha a változót a függvényen belül vesszük fel akkor lokális változóként tekintünk rá ha kívül van es a global szót írjuk elő akkor természetesen globálisról beszélhetünk . A nyelv rengeteg eszközzel dolgozik . Ilyen eszköz például a print amivel kiiratni tudunk , ilyen eszköz az elágazás amit az if segítségével tudunk használni. A nyelvben használhatunk különféle ciklusokat is természetes . A nyelvben továbbá találkozhatunk címkekkel és ugrásokkal a címkeket a label kulcsival helyezhetünk el és a gotoval pedig ugorhatunk rájuk. A python továbbá támogatja a függvények használatát is amelyeket a def kulcsszóval definíálhatunk . Osztály és objektum szempontjából a python a megszokott objektumorientáltságot támogatja Tehát az osztályok lehetnek atríbutumok tehát objektumok függetlenek. Az osztályok képesek mas osztályból is örökölni. Modulok szempontjából a Python viszonylag sok szabványos modult tartalmaz . Ezek például a mobilokhoz történő fejlesztések megkönnyítése miatt fontosak ilyenek például a halózat vagy a kamera kezelés. Ha a kivételkezelés szempontjából nézzük meg a pythonot akkor egy egyszerű megoldást találhatunk. A nyelvben a try kulcsszó után lévő részben kell lennie a tesztelni kivánt résznek illetve a rész után az expect kulcsszónak kell kerülnie . Erdemes az else agat is használni illetve a a finally kulcsszót is használhatjuk a kivételkezelésnél.

## IV. rész

### Irodalomjegyzék

DRAFT

## 20.3. Általános

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

## 20.4. C

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

## 20.5. C++

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

## 20.6. Lisp

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

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