

Univerzális programozás

Így neveld a programozód!

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

Copyright © 2019 Dr. Bátfai Norbert

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

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

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

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

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

COLLABORATORS

	<i>TITLE :</i> Univerzális programozás		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Bátfai, Norbert, Bátfai, Mátyás, Bátfai, Nándor, Ács Bátfai, Margaréta	2019. május 17.	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
0.0.1	2019-02-12	Az iniciális dokumentum szerkezetének kialakítása.	nbatfai
0.0.2	2019-02-14	Inciális feladatlisták összeállítása.	nbatfai
0.0.3	2019-02-16	Feladatlisták folytatása. Feltöltés a BHAX csatorna https://gitlab.com/nbatfai/bhax repójába.	nbatfai
0.0.4	2019-02-19	A Brun tételes feladat kidolgozása.	nbatfai

Ajánlás

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

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

Tartalomjegyzék

I. Bevezetés	1
1. Vízió	2
1.1. Mi a programozás?	2
1.2. Milyen doksikat olvassak el?	2
1.3. Milyen filmeket nézzek meg?	2
II. Tematikus feladatok	4
2. Helló, Turing!	6
2.1. Végtelen ciklus	6
2.2. Lefagyott, nem fagyott, akkor most mi van?	7
2.3. Változók értékének felcserélése	9
2.4. Labdapattogás	9
2.5. Szóhossz és a Linus Torvalds féle BogomIPS	11
2.6. Helló, Google!	11
2.7. A Monty Hall probléma	13
2.8. 100 éves a Brun tétel	14
3. Helló, Chomsky!	18
3.1. Decimálisból unárisba átváltó Turing gép	18
3.2. Az $a^n b^n c^n$ nyelv nem környezetfüggetlen	19
3.3. Hivatkozási nyelv	20
3.4. Saját lexikális elemző	21
3.5. l33t.1	22
3.6. A források olvasása	22
3.7. Logikus	23
3.8. Deklaráció	24

4. Helló, Caesar!	27
4.1. double ** háromszögmátrix	27
4.2. C EXOR titkosító	29
4.3. Java EXOR titkosító	30
4.4. C EXOR törő	33
4.5. Neurális OR, AND és EXOR kapu	36
4.6. Hiba-visszaterjesztéses perceptron	37
5. Helló, Mandelbrot!	39
5.1. A Mandelbrot halmaz	39
5.2. A Mandelbrot halmaz a std::complex osztállyal	42
5.3. Biomorfok	45
5.4. A Mandelbrot halmaz CUDA megvalósítása	49
5.5. Mandelbrot nagyító és utazó C++ nyelven	49
5.6. Mandelbrot nagyító és utazó Java nyelven	49
6. Helló, Welch!	54
6.1. Első osztályom	54
6.2. LZW	57
6.3. Fabejárás	62
6.4. Tag a gyökér	64
6.5. Mutató a gyökér	66
6.6. Mozgató szemantika	67
7. Helló, Conway!	69
7.1. Hangyaszimulációk	69
7.2. Java életjáték	70
7.3. Qt C++ életjáték	77
7.4. BrainB Benchmark	79
8. Helló, Schwarzenegger!	81
8.1. Szoftmax Py MNIST	81
8.2. Szoftmax R MNIST	87
8.3. Mély MNIST	87
8.4. Deep dream	93
8.5. Minecraft-MALMÖ	93

9. Helló, Chaitin!	95
9.1. Iteratív és rekurzív faktoriális Lisp-ben	95
9.2. Weizenbaum Eliza programja	95
9.3. Gimp Scheme Script-fu: króm effekt	96
9.4. Gimp Scheme Script-fu: név mandala	99
9.5. Lambda	102
9.6. Omega	103
10. Helló, Gutenberg!	104
10.1. Programozási alapfogalmak	104
10.2. Programozás bevezetés	105
10.3. Programozás	105
III. Második felvonás	107
11. Helló, Arroway!	109
11.1. A BPP algoritmus Java megvalósítása	109
11.2. Java osztályok a Pi-ben	109
IV. Irodalomjegyzék	110
11.3. Általános	111
11.4. C	111
11.5. C++	111
11.6. Lisp	111

Ábrák jegyzéke

2.1. A B_2 konstans közelítése	17
4.1. A double ** háromszögmátrix a memóriában	29

Előszó

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

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

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

Hogyan forgasd

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

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

Milyen nyelven nyomjuk?

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

Hogyan nyomjuk?

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

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

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

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



A DocBook XML 5.1 új neked?

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

I. rész

Bevezetés

1. fejezet

Vízió

1.1. Mi a programozás?

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

1.2. Milyen doksikat olvassak el?

- Kezd ezzel: <http://esr.fsf.hu/hacker-howto.html>!
- Olvasgasd aztán a kézikönyv lapjait, kezd a **man man** parancs kiadásával. A C programozásban a 3-as szintű lapokat fogod nézegetni, például az első feladat kapcsán ezt a **man 3 sleep** lapot
- C kapcsán a [**KERNIGHANRITCHIE**] könyv adott részei.
- C++ kapcsán a [**BMECPP**] könyv adott részei.
- Az igazi kockák persze csemegéznek a C nyelvi szabvány **ISO/IEC 9899:2017** kódcsipeteiből is.
- Amiből viszont a legeslegjobban lehet tanulni, az a **The GNU C Reference Manual**, mert gcc specifikus és programozókra van hangolva: szinte csak 1-2 lényegi mondat és apró, lényegi kódcsipetek! Aki pdf-ben jobban szereti olvasni: <https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.pdf>
- Az R kódok olvasása kis általános tapasztalat után automatikusan, erőfeszítés nélkül menni fog. A Python nincs ennyire a spektrum magától értetődő végén, ezért ahhoz olvasd el a [**BMECPP**] könyv - 20 oldalas gyorstalpaló részét.

1.3. Milyen filmeket nézzek meg?

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

- , , benne a bemutatása.
- , , benne a bemutatása.
- , , benne a bemutatása.
- , , benne a bemutatása.
- , , benne a bemutatása.
- , , benne a bemutatása.

DRAFT

II. rész

Tematikus feladatok

**Bátf41 Haxor Stream**

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

DRAFT

2. fejezet

Helló, Turing!

2.1. Végtelen ciklus

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

Megoldás videó: <https://youtu.be/lvmi6tyz-nI>

Megoldás forrása: [bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/infty-f.c](#), [bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozo/Turing/infty-w.c](#).

Rengetek fajta végtelen ciklus van. Szükség van rájuk pár fontosabb feladatban mint mondjuk egy szerveren akarunk állapotfolyamatokat lementeni es csinalunk egy végtelenciklust es beleirjuk hogy mentse le a szerver állapotát + sleep mondjuk 5 percig maskor meg bugok felvedzése vagy maga a végtelen ciklus a bugos.

Egy mag 100 százalékban:

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

    return 0;
}
```

még egy csak mert mért ne

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

    return 0;
}
```


Ajánlott a `for(;;)` formát használni mert az minden c verzióba lefut és ha valaki más olvassa a programodat akkor látja hogy végtelen ciklust akartál és nem csak elírtad. Érdekeség hogy a fordító a `while-t` és `for-t` ugyanúgy fordítja.

Egy mag 0 százalékbán:

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

    return 0;
}
```

Minden mag 100 százalékbán:

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

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

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

Megoldás videó:

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

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

```
    else
        return false;
}

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

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

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

akár önmagára

```
T100 (T100)
false
```

ezt a kimenetet adja.

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

```
Program T1000
{

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

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

    main(Input Q)
    {
        Lefagy2(Q)
    }
}
```

Mit for kiírni erre a T1000 (T1000) futtatásra?

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

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

Tanulságok, tapasztalatok, magyarázat...

A feladatnak nincs megoldása mert: ha a program tartalmaz végtelen ciklus akkor lefagy = true és akkor lefagy2 is true lesz ha meg a programban nincs végtelen ciklus akkor lefagy = false és akkor lefagy 2 egy végtelen ciklussá alakul át.

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

```
#include <stdio.h>

int main()
{
    int a = 2;
    int b = 1;

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

    printf(a);
    printf(b);
    return 0;
}
```

Megoldás forrása:

E feladat megoldása véleményem szerint egyértelmű.

2.4. Labdapattogás

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

A bitzero megnezi mekkora az ablakunk. A void rajzol vár két bemenetet amivel állítgatja a magasságot és a szélességet gyakorlatilag ez rajzolja az O-t. A main-be jön egy clear parancs ami megtisztítja a terminál ablakot. A vx és vy változókkal és a bitzero-ba kapott számokkal számoljuk ki az O pozícióját számolja. Usleep pedig hogy ne pattogjon túl gyorsan Sleep és a Usleep közötti különbség a Usleep microsecond-ba kéri az időt Sleep másodpercbe.

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

```
#include <stdio.h>
#include <stdlib.h>
char bitzero(char x) {
    int i;
    char bitt = x&0x1;
    for (i=0; i<8; i++) {
        bitt |= (x>>i)&1;
    }
    return 1-bitt;
}

void rajzol(char width, char height) {
    int i;
    /*magasság*/
    for (i=1; i<=height; i++) {
        printf("\n");
    }
    /*szélesség*/
    for (i=1; i<=width; i++) {
        printf(" ");
    }
    printf("O\n");
}

int main() {
    char x=1, y=1, vx=1, vy=1;
    while(1) {
        system("clear");
        vx-=2*bitzero(79-x); //balra pattanjon
        vx+=2*bitzero(x);    //
        vy-=2*bitzero(24-y); //lefele
        vy+=2*bitzero(y);    //
        x+=vx;
        y+=vy;
        //printf("X: %d Y: %d \n", x, y); //Koordináták
        //printf("Vx: %d Vy: %d", vx, vy); //Velocity
        rajzol(x,y);
        usleep(100000);
    }
    return 0;
}
```

Tanulságok, tapasztalatok, magyarázat... Egy alap program amiben a terminálban egy "o" betűt pattogtat while ciklussal.

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. Használd ugyanazt a while ciklus fejet, amit Linus Torvalds a BogomIPS rutinjában!

Megoldás videó: https://youtu.be/9KnMqrkj_kU, <https://youtu.be/KRZlt1ZJ3qk>, .

Megoldás forrása: [bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/bogomips.c](https://github.com/bhax/thematic_tutorials/blob/master/bhax_textbook_IgyNeveldaProgramozod/Turing/bogomips.c)

Tanulságok, tapasztalatok, magyarázat...

MIPS az egy rövidítés teljes valójába Millions of Instructions Per Second ami annyit tesz ki milliónyi utasítások/másodperc. Működése mögött a titok hogy van egy ciklus amit úgy kalibrál be a program hogy folyamatosan menjen bármi féle akadás nélkül es ezzel a processor sebességéhez igazodik.

```
int a = 1; //kettes számrendszerben: 00000000 00000000 00000000  ←
00000001
while (a != 0){
    a <<= 1; //itt léptetjük eggyel:00000000 00000000  ←
    00000000 00000010
    // újra: 00000000 00000000 00000000 00000100
    //...
}
```

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:

Tanulságok, tapasztalatok, magyarázat... A PageRank képlete: $PR(A) = (1-d) + d (PR(T1)/C(T1) + \dots + PR(Tn)/C(Tn))$ A: az aktuális oldal T: oldalak amik az a-ra hoz vannak linkelve PR(A): a oldal pagerank száma PR(Tn): azok oldalak pagerank száma amik az a-ra mutatnak C(Tn): t oldalon az összes link ami a-ra mutatnak d: 0 és 1 közé esik

Elsőnek is a kapcsolatokat a mátrixba tároljuk. Sorok és oszlopok metszetébe láthatjuk milyen kapcsolat van az oldalak között. Ezt a mátrixot adjuk a pagerank() - nak függvénynek. A tömbben tároljuk az oldalak értékét, és a PR-ben tároljuk el a mátrixszorzás eredményét. A mátrixszorzást a L és PRv tömbökkel hajtjuk végre. Ezek a tömbök összeszorzásával kapunk egy 4x1 oszlopvektort ami a 4 oldalunk pagerankja lesz.

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

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

    int i;

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

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

    int i;
    double osszeg=0;

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

    return sqrt(osszeg);
}

void
pagerank(double T[4][4]){
    double PR[4] = { 0.0, 0.0, 0.0, 0.0 };
    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] = PR[i] + T[i][j]*PRv[j];
            }
        }

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

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

```
    kiir (PR, 4);
}

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

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

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

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

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

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

    printf("\n");

    return 0;
}
```

2.7. 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

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

A program véletlen eseteket generál. Meghatározzuk a kísérletek számát majd a játékos vektorban véletlenszerűen megadjuk mit választ esetenként. A vektorok mérete a kísérletek számával egyenlő.

Egy for ciklussal végigmegyünk a kísérletek száman és megnézzuk hogy eltalálta-e az eredményt. Ha nem találta el, a műsorvezető kap pontot. Ha nyer, kiértékeljük hogy hányszor nyert volna változtatás nélkül. Egyéb esetekben változtatással nyert volna. Megszámoljuk a vektorokon az eseményeket majd kiiratjuk az eredményeket..

2.8. 100 éves a Brun tétel

Írj R szimulációt a Brun tétel demonstrálására!

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

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

A természetes számok építőelemei a prímszámok. Abban az értelemben, hogy minden természetes szám előállítható prímszámok szorzataként. Például $12=2*2*3$, vagy például $33=3*11$.

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

Két egymást követő páratlan prím között a legkisebb távolság a 2, a legnagyobb távolság viszont bármilyen nagy lehet! Ez utóbbit könnyű bebizonyítani. Legyen n egy tetszőlegesen nagy szám. Akkor szorozzuk össze $n+1$ -ig a számokat, azaz számoljuk ki az $1*2*3*\dots*(n-1)*n*(n+1)$ szorzatot, aminek a neve $(n+1)$ faktoriális, jele $(n+1)!$.

Majd vizsgáljuk meg az a sorozatot:

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

- $(n+1)!+2=1*2*3*\dots*(n-1)*n*(n+1)+2$, azaz $2*$ valamennyi $+2$, 2 többszöröse, így ami osztható kettővel
- $(n+1)!+3=1*2*3*\dots*(n-1)*n*(n+1)+3$, azaz $3*$ valamennyi $+3$, ami osztható hárommal
- ...
- $(n+1)!+(n-1)=1*2*3*\dots*(n-1)*n*(n+1)+(n-1)$, azaz $(n-1)*$ valamennyi $+(n-1)$, ami osztható $(n-1)$ -el
- $(n+1)!+n=1*2*3*\dots*(n-1)*n*(n+1)+n$, azaz $n*$ valamennyi $+n$, ami osztható n -el
- $(n+1)!+(n+1)=1*2*3*\dots*(n-1)*n*(n+1)+(n+1)$, azaz $(n+1)*$ valamennyi $+(n+1)$, ami osztható $(n+1)$ -el

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

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

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

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

```
# Copyright (C) 2019 Dr. Norbert Bátfai, nbatfai@gmail.com
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program. If not, see <http://www.gnu.org/licenses/>

library(matlab)

stp <- function(x) {

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

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

Soronként értelmezzük ezt a programot:

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

Kiszámolja a megadott számig a prímeket.

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

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

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

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

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

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

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

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

Kivette a `primes`-ből a párok első tagját.

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

A párok második tagját az első tagok kettő hozzáadásával képezzük.

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

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

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

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

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

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

```
> 1/t1primes+1/t2primes
[1] 0.53333333 0.3428571 0.1678322
```

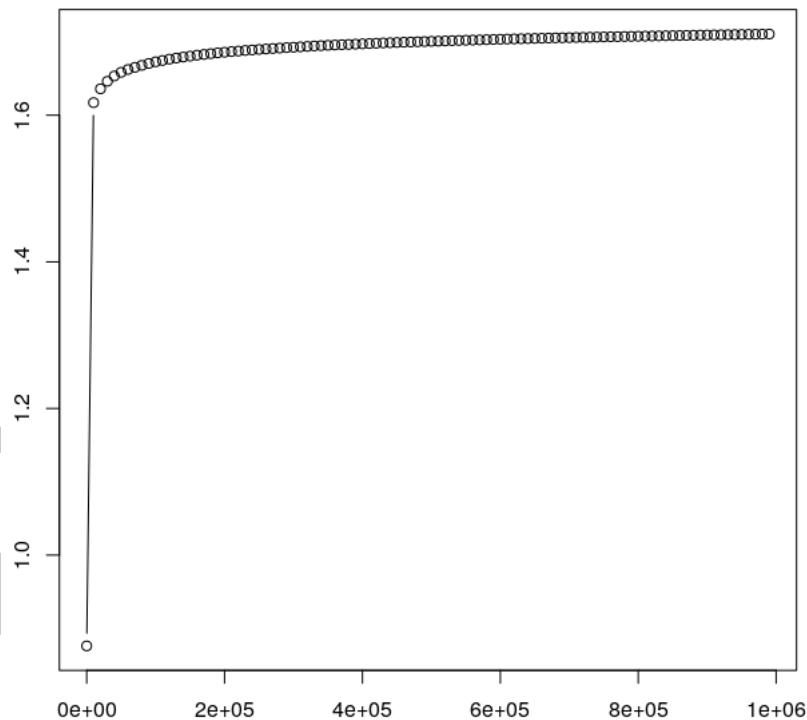
Nincs más dolgunk, mint ezeket a törteket összeadni a `sum` függvénnyel.

```
sum(rtlplust2)
```

```
> sum(rtlplust2)
[1] 1.044023
```

A következő ábra azt mutatja, hogy a szumma értéke, hogyan nő, egy határértékhez tart, a B_2 Brun konstanshoz. Ezt ezzel a csipettel rajzoltuk ki, ahol először a fenti számítást 13-ig végezzük, majd 10013, majd 20013-ig, egészen 990013-ig, azaz közel 1 millióig. Vegyük észre, hogy az ábra első köre, a 13 értékhez tartozó 1.044023.

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



2.1. ábra. A B_2 konstans közelítése



Werkfilm

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

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfiával megadva írd meg ezt a gépet!

Megoldás forrása: [link](#)

Tanulságok, tapasztalatok, magyarázat...

Turing gép atyja angol származása Alan Turing matematikus az 1936 évekbe megjelent cikkében már elkezdte a turing gép alapjait.

Turing a számítógépek fejlesztésének máig tartó folyamatát indították el.

A Turing-gépűgynevezett absztrakt automata.

Létrehozunk egy main függvényt benne pedig két db int típusú változot az egyik neve b ebben fogom tárolni a számot.

A másik a számláló aminek már értéke nulla.

cin parancsal tudjuk eltárolni a b értéket amit a kijelzőn adja meg aki programot futatja.

Cout kiíratunk mondatokat.Lérte kell hozni egy forciklust ami anyiszor fog lefutni ahány értéket megadtunk a b változóban.

cout kirtunk 1 -eseket és megszámloljuk b értéket és ha 5 egyes van akkor tesz egy szóközt a program.

A program végén pedig ki íratjuk az egyeseket.

```
[
#include <iostream>
using namespace std;

int main()
{
    int b;
    int szamlalo = 0;
    cout<<"Add meg egy decimalis szamot!\n";
    cin >> b;
    cout<<"Unárisban:\n";
```

```
for (int i = 0; i < b; ++i)
{
    cout<<"1";
    ++szamlalo;
    if (szamlalo % 5 == 0)
        cout<<" ";
}
cout<<'\n';
return 0;
}
```

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

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

Megoldás videó:

Megoldás forrása: [link](#)

Tanulságok, tapasztalatok, magyarázat... A forrásomban van részletezve a feladat.

1. pelda

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

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

S (S → aXbc)
aXbc (Xb → bX)
abXc (Xc → Ybcc)
abYbcc (bY → Yb)
aYbbcc (aY → aaX)
aaXbbcc (Xb → bX)
aabXbcc (Xb → bX)
aabbXcc (Xc → Ybcc)
aabbYbcc (bY → Yb)
aabbYbbcc (bY → Yb)
aaYbbbcc (aY → aa)
aaabbcc

2. pelda

```
A, B, C "változók"
a, b, c "konstansok"
A -> aAB, A -> aC, CB -> bCc, cB -> Bc, C -> bc
S-ből indulunk ki
-----
A (A -> aAB)
aAB (A -> aC)
aaCB (CB -> bCc)
aabCc (C -> bc)
aabbcc
-----
A (A -> aAB)
aAB (A -> aAB)
aaABB (A -> aAB)
aaaABBB (A -> aC)
aaaaCBBB (CB -> bCc)
aaaabCcBB (cB -> Bc)
aaaabCBcB (cB -> Bc)
aaaabCBBc (CB -> bCc)
aaaabbCcBc (cB -> Bc)
aaaabbCBcc (CB -> bCc)
aaaabbbCccc (C -> bc)
aaaabbbbcccc
```

3.3. Hivatkozási nyelv

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

Megoldás forrása: [link](#)

Könyv fogalma: A c program nyelv különbséget tesz az utasítások és a definíciók között, hogy az utasítás csak végrehajtható kódot tárol, ezzel szemben a definíció azonosítót generál.

Program nyelveket nem hiába nevezzük "nyelvnek" mert olyan mint egy normális nyelv fejlődik új funkciókat tulajdonságokat kap mint ebbe a feladatban is.c89 és a c99 verziók is ilyenek.Ez a példaprogram azért nem fut le c89-be mert c99-ben implementálták a kommenteket így c89 nem ismeri fel így hát hibaüzenetet dob be.

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

```
int main ()
{
    // Print string to screen.
    printf ("Hello World\n");
    return 0;
}
```

3.4. Saját lexikális elemző

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

Megoldás forrása: [link](#)

Tanulságok, tapasztalatok, magyarázat... A lexikális elemző feladata, hogy a forrásnyelvű program lexikális egységeit felismerje, azaz meghatározza a forrásnyelvű kód szimbólumok szövegét és típusát. Az egységek definíciója reguláris kifejezésekkel adható meg. Fontos hogy nem adható meg Chomsky 3-as nyelvosztállyal mert külön válik. A lexer segítségével tudunk lexikális szabályokból elemző kódot csinálni. A kód a `szamok_szama` változóba számolja a számokat. Ahhoz, hogy kiírja a számok számát, a Ctrl+D billentyű-kombinációt kell használnunk.

Lex forráskód 3 részből áll, az első a definíciós rész, amely lényegében bármilyen C forrást tartalmazhat. A második rész a szabályoknak van fenntartva. Ez 2 részből áll, reguláris kifejezésekből, és az azokhoz tartozó C utasításokból. Tehát, ha a program futása során a bemenetként kapott string illeszkedik valamelyik reguláris kifejezésre, akkor végrehajtja a hozzá tartozó utasítást. A harmadik rész pedig egy C-kód, amely lényegében a lexikális elemzőt hívja meg.

```
%{
#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: [link](#)

Tanulságok, tapasztalatok, magyarázat...

Elsőnek nézzük meg, hogy mi is az a leet nyelv. Ennek a nyelvnek az a lényege, hogy a szavakban lévő bizonyos betűket, számokkal, vagy egyéb más karakterekkel helyettesítjük. Az egyes betűket közmegállapodás szerinti karakterekre cserélhetjük. Erről a teljes listát google-ön megtalálod.

A program fordítása

```
lex -o l337d1c7.c l337d1c7.l
```

```
gcc l337d1c7.c -o l337d1c7 -lfl
```

```
./l337d1c7]]>
```

3.6. A források olvasása

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

```
if(signal(SIGINT, jelkezelolo)==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 jelkezelolo 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 megváránézésre, elkapja valamelyiket esetleg a splint vagy a frama?

i.

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

ii.

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

for ciklus ami 0-tól indul 5-ig megy és az i értékét növeli egyel az összes iterációban ezt a ++i prefixnek nevezük.

iii.

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

for ciklus ami 0-tól indul 5-ig megy és az i értékét növeli egyel az összes iterációban ezt a i++ postfixnek nevezük. Ez a különbség a két csipet között. Ám

iv.

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

for ciklus ami 0-tól indul 5-ig megy és tomb i-edik elemét felülírja az i++ értékére. De ez a ciklus hibát fog kiadni.

v.

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

A for ciklus 0-tól n-ig fut, majd s-pointer hozzá rendeli magát d-pointerhez és utána az i értékét egyel növeli.

vi.

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

Ez is hibás kód, mivel az f függvény két int-et kap, de azok kiértékelésének sorrendje nincs meghatározva.

vii.

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

Kiíratunk két db decimális számértéket az egyiket 'f(a)' függvény adja vissza, a másik pedig az 'a' változó értéke.

viii.

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

Ez ugyanaz mint az előző csipet a különbség ,hogy az f függvénynek van referenciája. Kiíratunk két db decimális számértéket az egyiket 'f(a)' függvény adja vissza, a másik pedig az 'a' változó értéke.

3.7. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

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

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

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

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

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

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

1. Minden számnál van nagyobb prím szám.
2. Minden számnál létezik nagyobb ikerprím számpáros.
3. Van olyan szám amelynél minden prím szám kisebb.
4. Van olyan szám aminél bármely nagyobb szám nem prím.

3.8. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

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

Mit vezetnek be a programba a következő nevek?

- ```
int a;
```

Egész típusú változó.

- ```
int *b = &a;
```

Egészre mutató mutató.

- ```
int &r = a;
```

Egészre történő referencia.

- ```
int c[5];
```

Egész tömb.

- ```
int (&tr)[5] = c;
```

Egész tömbre történő hivatkozás.

- ```
int *d[5];
```

Egész tömbre mutató mutató.

- ```
int *h ();
```

Egészre mutatót visszaadó függvény.

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

Egészre mutató mutatót visszaadó függvényre mutató mutató.

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

Egészre visszaadó és egészet kapó függvény mutatót visszaadó, két egészet kapó függvény mutató.

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

Egészre mutató egészet kapó függvény mutatót visszaadó 2 egészet kapó függvény.

```
#include <iostream>
int* fakt(int szam){
    static int a = 1;
    if (szam < 2)
        return &a;
    while (szam>1){
        a = a*szam;
        --szam;
    }
    return &a;
}

int* sum(int egyik, int masik){
    static int sum = egyik + masik;
    return &sum;
}

int szorzat(int egyik, int masik){
    return egyik*masik;
}

int osztas(int egyik, int masik){
    return egyik/masik;
}

int (*pfgv (int a)) (int,int){
    if (a){
        return &szorzat;
    }
    else
        return &osztas;
}
```

```
int main()
{
    int a = 10;
    int b = 5;
    int* pa = &a;
    int& ra = a;
    int tomb[a];
    int(& rtomb)[a] = tomb;
    int* ptomb[2];
    ptomb[0] = &a;
    ptomb[1] = &b;
    int* fakt_a = fakt(a);
    int(*psum)(int,int) = &sum;
    int (*p_pfgv)(int valami)(int, int) = &pfgv;
    std::cout<<"a és b szorzata "<<(pfgv(1))(a,b)<<std::endl;
    std::cout<<"a és b hányadosa "<<(pfgv(0))(a,b)<<std::endl;
    std::cout<<"a és b hányadosa "<<(p_pfgv(0))(a,b)<<std::endl;
    //int* pfakt_b = (*pfakt)(b);
    std::cout<<"a értéke "<<a<<'\\t'<<"a! értékes "<<*fakt_a<<std::endl;
    std::cout<<"a és b összege "<<*psum(a,b)<<std::endl;
}
```

Megoldás videó:

Megoldás forrása: [link](#)

Tanulságok, tapasztalatok, magyarázat... A forrásba le vannak írva meglátásaim a feladathoz.

4. fejezet

Helló, Caesar!

4.1. double ** háromszögmátrix

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

Megoldás videó: <https://youtu.be/1MRTuKwRsB0>, <https://youtu.be/RKbX5-EWpzA>.

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

Ahhoz hogy megértsük a feladatot kell tudnunk mi az a Mátrix. A mátrixoknak 2 attribútumjuk van hány sora és hány oszlopa van. Egy t sorból és y oszlopból álló mátrixot txy mátrixnak neveznek. Most már tudjuk mi az a mátrix de mi a háromszögmátrix? A háromszögmátrix az egy négyzet mátrix melyen a sorai és oszlopainak a száma megegyezik és a főátló alatt vagy felett csak 0 van. Ebben a feladatban 3 sorunk lesz így 6 db elemünk lesz. A `malloc` függvény képes lefoglalni a dinamikus területen egy, a paramétereként kapott méretű területet. Ezt követően vagy a lefoglalt terület kezdőcímét, vagy `NULL`-t ad vissza (hiba esetén). A `man 3 malloc` parancs terminálba való beírásával megkaphatjuk a függvény pontos szintaktikai leírását. Egyetlen paramétere (`size_t size`) azt mondja meg, hogy hány bájtot szeretnénk a memóriában lefoglalni. A `free` függvény felszabadítja a lefoglalt területet. Manuáljából (`man 3 free`) megtudhatjuk, hogy bemeneti paramétere a memóriát azonosító pointer (`ptr`).

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

int
main ()
{
    int nr = 5; //hány sor legyen a kiadott 3 szög
    double **tm;

    printf("%p\n", &tm); //memória foglalás kiírása

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

```
printf("%p\n", tm); //memória foglalás kiiratása

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

printf("%p\n", tm[0]); //memória foglalás kiiratása

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

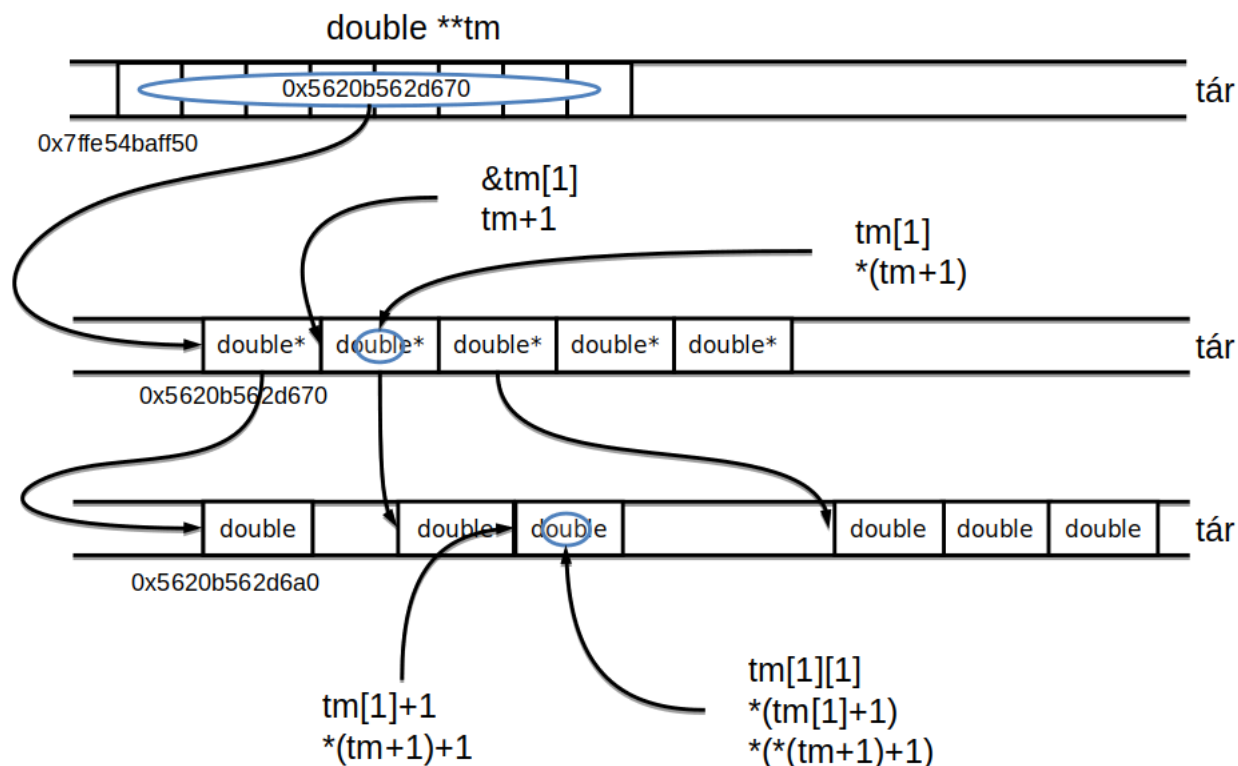
tm[3][0] = 42.0;
(* (tm + 3)) [1] = 43.0; // mi van, ha itt hiányzik a külső ()
*(tm[3] + 2) = 44.0;
* (* (tm + 3) + 3) = 45.0;

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

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

free (tm);

return 0;
}
}
```



4.1. ábra. A double ** háromszögmátrix a memóriában

Tanulságok, tapasztalatok, magyarázat...

4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás videó:

Megoldás forrása: A feladat megoldásához az udprog repository-t használtam

Ez a feladat titkosításról lesz szó. Ez úgy fut le hogy megadunk egy "tisztá szöveget" és ez a program átkódolja egy olvashatatlan fájlá. Futtatásához meg kell adnunk egy tisztá szöveget nevét és a cél fájl nevét és típusát. A titkosítás módja az EXOR logikai művelet.

Elsőnek a kulcs méret és a buffer méretének maximumát konstansban tároljuk el, ezek nem módosíthatóak érdekessége hogy nem program futtatásakor történik meg a behelyettesítés, hanem a már a fordítás alatt. Az argc-vel adjuk át az argumentumok számát, és az argumentumokra mutató mutatókat pedig az argv tömbben tároljuk el. A main()-en belül deklarálunk két tömböt, egyikbe a kulcsot tároljuk, a másikban pedig a beolvasott karaktereket. Definiálunk számlálókat, melyek segítségével bejárjuk majd a kulcs tömböt, és számoljuk a beolvasott bájtokat. A kulcsméretet a strncpy függvényel kapjuk meg. A while cikluson feltétele addig lesz igaz, ameddig a read parancs beolvassa a megadott mennyiségű bájtokat. A read 3 argumentumot kap. A beolvasott bájtok számát adja vissza. Ezután pedig végigmegyünk elemenként a bufferben eltárolt karaktereken és össze EXOR-ozzuk a kulcs tömb megfelelő elemével. Végezetül pedig kiírjuk a buffer tartalmát a standard outputra.

A program futtatásához ezt a paracsot kell használni: `./e 56789012 asd.txt asd.szoveg more asd.szoveg`

```
#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://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/ch01.html#exor_titkosito

Mint az előző feladatban megint az a feladatunk hogy egy szöveges fájlt titkosítsunk az EXOR algoritmussal , azonban most Java környezetben. A `Titkosito.java` mint az előző feladatban meg kell adnunk egy

tiszta szöveget csak most be kell gépelnünk konzolba ezt a kulso while függvény olvassa. A belső for ciklusban húzzuk rá a szövegre a kulcsot kulcsIndex változó segítségével, majd program átkódolja a beirt szövegünket. Az eredmény a buffer tömbben keletkezik, amit végül a kimenetre írunk.

```
$ more Titkosito.java
public class Titkosito {

    public Titkosito(String kulcsSzöveg,
        java.io.InputStream bejövőCsatorna,
        java.io.OutputStream kimenőCsatorna)
        throws java.io.IOException {

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

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

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

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

            }

            kimenőCsatorna.write(buffer, 0, olvasottBájtok);

        }

    }

    public static void main(String[] args) {

        try {

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

        } catch(java.io.IOException e) {

            e.printStackTrace();

        }

    }

}

public class Titkosito {
```

```
public Titkosito(String kulcsSzöveg,
    java.io.InputStream bejövőCsatorna,
    java.io.OutputStream kimenőCsatorna)
    throws java.io.IOException {

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

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

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

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

        }

        kimenőCsatorna.write(buffer, 0, olvasottBájtok);

    }

}

public static void main(String[] args) {

    try {

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

    } catch(java.io.IOException e) {

        e.printStackTrace();

    }

}

}
```

Futtatása:

```
$ javac Titkosito.java
$ java Titkosito pelda > titkositott.szoveg
első sor
második sor
```

A szöveg titkosítva lett, dekódolni hasonló paranccsal tudunk, azonban figyeljünk arra, hogy ezúttal a kacsacsőrnek balra kell mutatnia. Ezek után outputként megkapjuk az eredeti, jól értelmezhető szöveget.

```
$ java Titkosito pelda < titkositott.szoveg  
első sor  
második sor
```

4.4. C EXOR törő

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

Megoldás videó:

Megoldás forrása: https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0063_01_parhuzamos_prog_linux/

Feladatunk most hogy az előző feladatokban titkosított szövegeket visszadekódoljuk úgy hogy nem tudjuk az exor kulcsát csak azt tudjuk hogy a kulcs 8 karakter hosszú és csak számokat tartalmaz.

Ismét konstansokat létrehozásával kezdünk, ebből a kulcs_meret érdekes, mert itt megadjuk hogy a kulcs csak 8 karakter hosszú így ez a program nem a leghatékonyabb a való életben. Az atlagos_szohossz azzal kiszámítjuk a bemenet átlagos szóhosszát ennek odaadunk egy tömböt és ennek a tömbnek a méretét. Ezután bejárjuk egy for ciklussal és minden elem után növeljük az sz értékét eggyel. Returnként pedig tömbünk méretét elosztjuk az sz változóval. A tiszta_lehet átlagos szóhossz segítségével ellenőrzi hogy a beadot szöveg tiszta szöveg e. Az exor függvény ugyan azt csinálja az exor titkosító mivel ha egy exorozott forrást újra exorottuk akkor megkapjuk a tiszta szöveget. Az exor_tores 0 vagy 1-et ad vissza attól függően hogy tiszta e a szöveg vagy nem. Ezek után jön a main ebben csinálunk egy kulcs[] tömböt és egy titkos tömböt létrehozunk egy mutatót ami a titkos tömbre mutat és behozzuk egy olvasott_bajtok számlálót. A while ciklussal addig megyünk míg a buffer meg nem telik vagy a bemenet végére érünk. Következő for ciklussal kinullázzuk a buffer üres helyeit és összeállítjuk a lehetséges kulcsokat. Utánna végig futtatjuk az összes lehetséges kulcsunkat a for ciklusokon és meghívjuk az exor törőt ami eldönti hogy a kikódolt szöveg tiszta e vagy nem. Ha az eredmény igaz lesz azt kiírja egy txt fájlba.

Futtatási parancs:

```
$ gcc t.c -o t -std=c99  
$ ./t titkos.szoveg |grep 12344567
```

```
#define MAX_TITKOS 4096  
#define OLVASAS_BUFFER 256  
#define KULCS_MERET 8  
#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
tisztalehet (const char *titkos, int titkos_meret)
{
    // a tiszta szoveg valszeg tartalmazza a gyakori magyar szavakat
    // illetve az átlagos szóhossz vizsgálatával csökkentjük a
    // potenciális töréseket

    double szohossz = atlagos_szo_hossz (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 tisztalehet (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 (int ii = '0'; ii <= '9'; ++ii)
        for (int ji = '0'; ji <= '9'; ++ji)
            for (int ki = '0'; ki <= '9'; ++ki)
                for (int li = '0'; li <= '9'; ++li)
                    for (int mi = '0'; mi <= '9'; ++mi)
                        for (int ni = '0'; ni <= '9'; ++ni)
                            for (int oi = '0'; oi <= '9'; ++oi)
                                for (int pi = '0'; pi <= '9'; ++pi)
                                    {
                                        kulcs[0] = ii;
                                        kulcs[1] = ji;
                                        kulcs[2] = ki;
                                        kulcs[3] = li;
                                        kulcs[4] = mi;
                                        kulcs[5] = ni;
                                        kulcs[6] = oi;
                                        kulcs[7] = pi;

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

                                        // ujra EXOR-ozunk, így 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

Tanulságok, tapasztalatok, magyarázat... A neuron az elektromos jelek összegyűjtéséért, feldolgozásáért és szétterjesztéséért felelős agysejt. Az agy információfeldolgozó kapacitása elsősorban neuronok hálózatából alakult ki. Neurális hálónak nevezzük azt a párhuzamos működésre képes információfeldolgozó eszközt, amely nagyszámú, hasonló típusú elem összekapcsolt rendszeréből áll. Továbbá jellemzője az is, hogy rendelkezik tanulási algoritmussal és képes előhívni a megtanult információt. Éppen ezért a neurális kapu szimulálásához az R programozási nyelvet fogjuk használni. a1 és a2 értékeket tartalmaz, az OR pedig a logikai VAGY műveletet jelöli. A program az általunk meghatározott szabályok alapján elkezd tanulni. A `compute` parancs segítségével tudjuk leellenőrizni, hogy a megfelelő eredményeket kaptuk-e vagy sem. A logikai ÉS művelet (AND) betanítása is hasonló módon történik. Az EXOR műveletnél azonban csak többretegű neuronokkal lehetséges a tanítás (`hidden = 2`).

```
$ more neuralis.r  
library(neuralnet)  
  
a1    <- c(0,1,0,1)  
a2    <- c(0,0,1,1)  
OR    <- c(0,1,1,1)  
  
or.data <- data.frame(a1, a2, OR)  
  
nn.or <- neuralnet(OR~a1+a2, or.data, hidden=0, linear.output=FALSE, ←  
    stepmax = 1e+07, threshold = 0.000001)  
  
plot(nn.or)  
  
compute(nn.or, or.data[,1:2])  
  
a1    <- c(0,1,0,1)  
a2    <- c(0,0,1,1)  
OR    <- c(0,1,1,1)  
AND   <- c(0,0,0,1)  
  
orand.data <- data.frame(a1, a2, OR, AND)
```

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

Erre a feladatra felhasználnám az egyik passzomat amit SMNISTforHUMANS lvl9 eléréséért kaptam.

Hiba-visszaterjesztést a többrétegű perceptronoknál alkalmazunk(MLP), amelyek tartalmaznak rejtett rétegeket is.

Erre a módszerre azért van szükség, mert az olyan hálókból ahol van hidden layer a bemenetből nem rögtön a kimenet következik. Így nem lehet rögtön a kimenetre kapott hiba alapján kiszámolni a súlyokat.

Ilyenkor rétegenként visszafele haladva számoljuk ki az egyes rétegekhez tartozó súlyokat, míg vissza nem érünk a bemenethez.

DRAFT

5. fejezet

Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

Írj olyan C programot, amely kiszámolja a Mandelbrot halmazt!

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

Megoldás forrása: [bhax/attention_raising/CUDA/mandelpngt.c++](https://github.com/bhax/attention_raising/CUDA/mandelpngt.c++) nevű állománya.

A Mandelbrot halmazt 1980-ban találta meg Benoit Mandelbrot a komplex számsíkon. Komplex számok azok a számok, amelyek körében válaszolni lehet az olyan egyébként értelmezhetetlen kérdésekre, hogy melyik az a két szám, amelyet összeszorozva -9 -et kapunk, mert ez a szám például a $3i$ komplex szám.

```
// mandelpngt.c++
// Copyright (C) 2019
// Norbert Bátfaí, 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 Páternosztér/PAEP
// 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/times.h>  
  
#define MERET 600  
#define ITER_HAT 32000  
  
void  
mandel (int kepadat[MERET][MERET]) {  
  
    // MÉRÜNK IDŐT (PP 64)  
    clock_t delta = clock ();  
    // MÉRÜNK IDŐT (PP 66)  
    struct tms tmsbuf1, tmsbuf2;  
    times (&tmsbuf1);  
  
    // számítás adatai  
    float a = -2.0, b = .7, c = -1.35, d = 1.35;  
    int szelesseg = MERET, magassag = MERET, iteraciosHatar = ITER_HAT;  
  
    // a számítás  
    float dx = (b - a) / szelesseg;  
    float dy = (d - c) / magassag;  
    float reC, imC, reZ, imZ, ujreZ, ujimZ;  
    // Hány iterációt csináltunk?  
    int iteracio = 0;  
    // Végigzongorázzuk a szélesség x magasság rácsot:  
    for (int j = 0; j < magassag; ++j)  
    {  
        //sor = j;  
        for (int k = 0; k < szelesseg; ++k)  
        {  
            // c = (reC, imC) a rács csomópontjainak  
            // megfelelő komplex szám  
            reC = a + k * dx;  
            imC = d - j * dy;  
            // z_0 = 0 = (reZ, imZ)  
            reZ = 0;  
            imZ = 0;  
            iteracio = 0;  
            // z_{n+1} = z_n * z_n + c iterációk  
            // számítása, amíg |z_n| < 2 vagy még  
            // nem értük el a 255 iterációt, ha  
            // viszont elértük, akkor úgy vesszük,  
            // hogy a kiindulási 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
            ujureZ = reZ * reZ - imZ * imZ + reC;
            ujimZ = 2 * reZ * imZ + imC;
            reZ = ujureZ;
            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;  
}
```

Mandelbrot halmaz komplex számsíkon

A Mandelbrot halmazt 1980-ban találta meg Benoit Mandelbrot a komplex számsíkon. Komplex számok azok a számok, amelyek körében válaszolni lehet az olyan egyébként értelmezhetetlen kérdésekre, hogy melyik az a két szám, amelyet összeszorozva -9-et kapunk, mert ez a szám például a $3i$ komplex szám.

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

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

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

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

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

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

Megoldás forrása:

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

```
// Verzio: 3.1.2.cpp
// Forditas:
// g++ 3.1.2.cpp -lpng -O3 -o 3.1.2
// Futtatas:
// ./3.1.2 mandel.png 1920 1080 2040 ↵
// -0.01947381057309366392260585598705802112818 ↵
// -0.0194738105725413418456426484226540196687 ↵
// 0.7985057569338268601555341774655971676111 ↵
// 0.798505756934379196110285192844457924366
// ./3.1.2 mandel.png 1920 1080 1020 ↵
// 0.4127655418209589255340574709407519549131 ↵
// 0.4127655418245818053080142817634623497725 ↵
// 0.2135387051768746491386963270997512154281 ↵
// 0.2135387051804975289126531379224616102874
// Nyomtatas:
// a2ps 3.1.2.cpp -o 3.1.2.cpp.pdf -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 Bاتفai, 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 < iteraciosHatar )
{
    z_n = z_n * z_n + c;

    ++iteracio;
}

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

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

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

5.3. Biomorfok

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

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

Maradunk a Mandelbrot-halmaz témakörénél, azonban ezen belül inkább a Júlia-halmazokon lesz a hangsúly. Az előző feladatban c változóként szerepelt és ehhez számoltunk z értékeket. Most azonban c konstansként szerepel és a rácsbejárás z -vel történik. Itt már érezhető a részhalmaz-kapcsolat. A biomorfokra (a Julia halmazokat rajzoló bug-os programjával) rátaláló Clifford Pickover azt hitte természeti törvényre bukkan: https://www.emis.de/journals/TJNSA/includes/files/articles/Vol9_Iss5_2305--2315_Biomorphs_via_modif

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

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

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

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

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

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

    ++iteracio;
}
```

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

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

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

A bimorfos algoritmus pontos megismeréséhez ezt a cikket olvastam: https://www.emis.de/journals/TJNSA/includes/files/articles/Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf. korábbi **Mandelbrot halmazt** kiszámoló forrásunkat módosítjuk. Viszont a program változóinak elnevezését összhangba hozzuk a közlemény jelöléseivel: így könnyebben megérthető.

```
// 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
// Nyomtatás:
```



```
// a2ps 3.1.3.cpp -o 3.1.3.cpp.pdf -l --line-numbers=1 --left-footer=" ←  
BATF41 HAXOR STR34M" --right-footer="https://bhaxor.blog.hu/" --pro= ←  
color  
//  
// BHAX Biomorphs  
// Copyright (C) 2019  
// Norbert Batfai, batfai.norbert@inf.unideb.hu  
//  
// This program is free software: you can redistribute it and/or modify  
// it under the terms of the GNU General Public License as published by  
// the Free Software Foundation, either version 3 of the License, or  
// (at your option) any later version.  
//  
// This program is distributed in the hope that it will be useful,  
// but WITHOUT ANY WARRANTY; without even the implied warranty of  
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
// GNU General Public License for more details.  
//  
// You should have received a copy of the GNU General Public License  
// along with this program. If not, see <https://www.gnu.org/licenses/>.  
//  
// Version history  
//  
// https://youtu.be/IJMbRzY76E  
// 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;
}
```

5.4. A Mandelbrot halmaz CUDA megvalósítása

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

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

5.5. Mandelbrot nagyító és utazó C++ nyelven

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

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

Megoldás forrása: [bhax/tree/master/attention_raising/Mandelbrot/Zoom](https://github.com/bhax/tree/master/attention_raising/Mandelbrot/Zoom)

A program a QT GUI-t használja, ennek segítségével tudjuk elkészíteni a Mandelbrot halmazra nagyító programunkat. A QT az egyik leggyakrabban használt grafikus interfésze a C++-nak.

Fordítás: 4 fájlnak kell a mappánkban lennie. Elsőnek a mappánkra kell nyitni egy terminált és be kell írunk hogy `qmake -project`. Ez létrehoz egy *.pro fájlt. Ebbe a fájlba be kell írni hogy: `QT += widgets` sort. Ezek után futtatni kell a `qmake *.pro`. Ha minden sikerült akkor kapunk kell egy Makefile -t, ez fogja majd futtatni a programot. Be írjuk a make parancsot, amely létrehoz egy bináris fájlt. Ezekután már csak futtatni kell a bináris fájlt.

Ha részletesebb képet akarunk akkor "n" billentyűt kell lenyomnunk mert ekkor a látható területet újraszámolja a program.

5.6. Mandelbrot nagyító és utazó Java nyelven

Az előző feladatban látott C++ nagyítót most Java-ban implementáljuk. A fordításhoz, futtatáshoz szükséges: `sudo apt-get install openjdk-8-jdk`.

Futtatása:

```
laci@ubuntu:~/Asztal/java$ javac MandelbrotHalmazNagyító.java
```

```
laci@ubuntu:~/Asztal/java$ java MandelbrotHalmazNagyító
```

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

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

```
* MandelbrotHalmazNagyító.java
*
* DIGIT 2005, Javat tanítók
* 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 nygítani 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 űs 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(++pillanatfelvételSzámláló);
sb.append("_");
// A fájl nevébe bele vesszük, 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 eltérő kiszámolt szám.

Megoldás videó:

Megoldás forrása:

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

Az objektumorientált programozás (OOP) gyakorlatilag átalakította a régen ismert programozás fogalmát mivel segít a programkód olvashatóságán megértésén sebességén és megbízhatóságában. Objektumorientált program, objektumokból és osztályokból áll tehát a valós világ elemeinek programozási modelljeiből építi fel a programot. A C++ és a Java és még sok más is objektumorientált programozási nyelv.

A feladat által előírt polártranszformációs algoritmus segítségével random számokat tudunk generálni. Maga az algoritmus annyira eltrejedd, hogy a Java random szám generátor függvénye (pl: `(int)(Math.random()*50+1)`) is ezt használja.

Java-ban:

```
public class PolárGenerátor {  
  
    boolean nincsTárolt = true;  
    double tárolt;  
  
    public PolárGenerátor() {  
  
        nincsTárolt = true;  
  
    }  
}
```



```
public double következő() {  
  
    if(nincsTárolt) {  
  
        double u1, u2, v1, v2, w;  
        do {  
            u1 = Math.random();  
            u2 = Math.random();  
  
            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);  
  
        tárolt = r*v2;  
        nincsTárolt = !nincsTárolt;  
  
        return r*v1;  
  
    } else {  
        nincsTárolt = !nincsTárolt;  
        return tárolt;  
    }  
}  
  
public static void main(String[] args) {  
  
    PolárGenerátor g = new PolárGenerátor();  
  
    for(int i=0; i<10; ++i)  
        System.out.println(g.következő());  
  
}  
  
}
```

C++-ban:

```
#include <cstdlib>  
#include <cmath>  
#include <ctime>  
#include <iostream>  
class PolarGen  
{  
public:
```

```
PolarGen ()
{
    nincsTarolt = true;
    std::srand (std::time (NULL));
}
~PolarGen ()
{
}
double kovetkezo ()
{
    if (nincsTarolt)
    {
        double u1, u2, v1, v2, w;
        do
        {
            u1 = std::rand () / (RAND_MAX + 1.0);
            u2 = std::rand () / (RAND_MAX + 1.0);
            v1 = 2 * u1 - 1;
            v2 = 2 * u2 - 1;
            w = v1 * v1 + v2 * v2;
        }
        while (w > 1);
        double r = std::sqrt ((-2 * std::log (w)) / w);
        tarolt = r * v2;
        nincsTarolt = !nincsTarolt;
        return r * v1;
    }
    else
    {
        nincsTarolt = !nincsTarolt;
        return tarolt;
    }
}
private:
    bool nincsTarolt;
    double tarolt;
};
int main (int argc, char **argv)
{
    PolarGen pg;
    for (int i = 0; i < 10; ++i)
        std::cout << pg.kovetkezo () << std::endl;
    return 0;
}
```

Futtatása:

```
laci@ubuntu:~/Desktop$ g++ polargen.cpp -o p
laci@ubuntu:~/Desktop$ ./p
```

6.2. LZW

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

Megoldás videó:

Megoldás forrása:

Az LZW (Lempel-Ziv-Welch) algoritmus az egy veszteségmentes tömörítő eljárás, melyet Terry Welch publikálta 1984-ben. Maga az algoritmus teljesen elterjedt sok manaságbeli tömörítő használja a formátuma a gif de más tömörítő program is használja mint pl: zip,gzip. Az algoritmus lényege az hogy a bemeneti egyesekből és nullákból egy bináris fát épít. Mondjuk ezzel nem érünk el sokat de ezután jön a fontos rész Úgy építi fel a fát, hogy ellenőrzi, hogy van-e már 0-ás vagy 1-es gyermek, ha nincs akkor létrehoz egyet, és visszaugrik a gyökérre. Ha van, akkor a 0-és vagy 1-es gyermekre lép, és addig halad lefele a fában, ameddig nem talál egy olyan részfat, ahol létre kellene hozni egy új gyermeket, a létrehozás után visszaugrik a gyökérre.

Első lépésként létrehozunk egy struktúrát ezzel egy új típust definiálva, amely 3 részből áll, egy értékből, és a gyermekeire mutató mutatókból. Az `uj_elem()` függvény segítségével foglalunk helyet a BINFA típusú változóknak, majd visszaadunk egy erre a területre mutató pointert. Ezután deklaráljuk a szükséges függvényeket, melyeket a későbbiekben majd definiálunk. Nézzük a maint elsőnek létrehozuk a gyökeret. Az értékét beállítjuk a '/' jelre ezután a 2 mutatót null értékre állítjuk és a fa mutatót a gyökérre állítom. A while ciklusban alkotjuk meg a binfánkat. A standard inputról olvassuk a bemenetet, bitenként. A main függvény végén íratjuk ki a binfát, és számolunk ki hozzá néhány érdekes adatot.

```
// z.c
//
// LZW fa építő
// Programozó Páternoszter
//
// Copyright (C) 2011, Bátfai Norbert, nbatfai@inf.unideb.hu, 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/>.
//
```

```
// Ez a program szabad szoftver; terjeszthető illetve módosítható a
// Free Software Foundation által kiadott GNU General Public License
// dokumentumában leírtak; akár a licenc 3-as, akár (tetszőleges) későbbi
// változata szerint.
//
// Ez a program abban a reményben kerül közreadásra, hogy hasznos lesz,
// de minden egyéb GARANCIA NÉLKÜL, az ELADHATÓSÁGRA vagy VALAMELY CÉLRA
// VALÓ ALKALMAZHATÓSÁGRA való származtatott garanciát is beleértve.
// További részleteket a GNU General Public License tartalmaz.
//
// A felhasználónak a programmal együtt meg kell kapnia a GNU General
// Public License egy példányát; ha mégsem kapta meg, akkor
// tekintse meg a <http://www.gnu.org/licenses/> oldalon.
//
//
// Version history:
//
// 0.0.1, http://progpater.blog.hu/2011/02/19/gyonyor\_a\_tomor
// 0.0.2, csomópontok mutatóinak NULLázása (nem fejtette meg senki :)
// 0.0.3, http://progpater.blog.hu/2011/03/05/ ←
//      labormeres_otthon_avagy_hogyan_dolgozok_fel_egy_pedat
//

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>

typedef struct binfa
{
    int ertekek;
    struct binfa *bal_nulla;
    struct binfa *jobb_egy;
} BINFA, *BINFA_PTR;

BINFA_PTR
uj_elem ()
{
    BINFA_PTR p;

    if ((p = (BINFA_PTR) malloc (sizeof (BINFA))) == NULL)
    {
        perror ("memoria");
        exit (EXIT_FAILURE);
    }
    return p;
}

extern void kiir (BINFA_PTR elem);
```

```
extern void ratlag (BINFA_PTR elem);
extern void rszoras (BINFA_PTR elem);
extern void szabadit (BINFA_PTR elem);

int
main (int argc, char **argv)
{
    char b;

    BINFA_PTR gyoker = uj_elem ();
    gyoker->ertek = '/';
    gyoker->bal_nulla = gyoker->jobb_egy = NULL;
    BINFA_PTR fa = gyoker;

    while (read (0, (void *) &b, 1))
    {
        //      write (1, &b, 1);
        if (b == '0')
        {
            if (fa->bal_nulla == NULL)
            {
                fa->bal_nulla = uj_elem ();
                fa->bal_nulla->ertek = 0;
                fa->bal_nulla->bal_nulla = fa->bal_nulla->jobb_egy = NULL;
                fa = gyoker;
            }
            else
            {
                fa = fa->bal_nulla;
            }
        }
        else
        {
            if (fa->jobb_egy == NULL)
            {
                fa->jobb_egy = uj_elem ();
                fa->jobb_egy->ertek = 1;
                fa->jobb_egy->bal_nulla = fa->jobb_egy->jobb_egy = NULL;
                fa = gyoker;
            }
            else
            {
                fa = fa->jobb_egy;
            }
        }
    }

    printf ("\n");
    kiir (gyoker);
}
```

```
extern int max_melyseg, atlagosszeg, melyseg, atlagdb;
extern double szorasosszeg, atlag;

printf ("melyseg=%d\n", max_melyseg-1);

/* Átlagos ághossz kiszámítása */
atlagosszeg = 0;
melyseg = 0;
atlagdb = 0;
ratlag (gyoker);
// atlag = atlagosszeg / atlagdb;
// (int) / (int) "elromlik", ezért casoljuk
// K&R tudatlansági védelem miatt a sok () :)
atlag = ((double)atlagosszeg) / atlagdb;

/* Ághosszak szórásának kiszámítása */
atlagosszeg = 0;
melyseg = 0;
atlagdb = 0;
szorasosszeg = 0.0;

rszoras (gyoker);

double szoras = 0.0;

if (atlagdb - 1 > 0)
    szoras = sqrt( szorasosszeg / (atlagdb - 1));
else
    szoras = sqrt (szorasosszeg);

printf ("atlag=%f\nszoras=%f\n", atlag, szoras);

szabadit (gyoker);
}

// a Javacska ONE projekt Hetedik Szem/TudatSzamitas.java mintajara
// http://sourceforge.net/projects/javacska/
// az atlag() hivasakor is inicializalni kell oket, a
// a rekurziv bejaras használja
int atlagosszeg = 0, melyseg = 0, atlagdb = 0;

void
ratlag (BINFA_PTR fa)
{
    if (fa != NULL)
```

```
{
    ++melyseg;
    ratlag (fa->jobb_egy);
    ratlag (fa->bal_nulla);
    --melyseg;

    if (fa->jobb_egy == NULL && fa->bal_nulla == NULL)
{
    ++atlagdb;
    atlagosszeg += melyseg;
}

}

}

// a Javacska ONE projekt Hetedik Szem/TudatSzamitas.java mintajara
// http://sourceforge.net/projects/javacska/
// az atlag() hivasakor is inicializalni kell oket, a
// a rekurziv bejaras használja
double szorasosszeg = 0.0, atlag = 0.0;

void
rszoras (BINFA_PTR fa)
{
    if (fa != NULL)
    {
        ++melyseg;
        rszoras (fa->jobb_egy);
        rszoras (fa->bal_nulla);
        --melyseg;

        if (fa->jobb_egy == NULL && fa->bal_nulla == NULL)
        {

            ++atlagdb;
            szorasosszeg += ((melyseg - atlag) * (melyseg - atlag));

        }

    }

}

//static int melyseg = 0;
int max_melyseg = 0;
```

```
void
kiir (BINFA_PTR elem)
{
    if (elem != NULL)
    {
        ++melyseg;
        if (melyseg > max_melyseg)
            max_melyseg = melyseg;
        kiir (elem->jobb_egy);
        // ez a postorder bejáráshoz képest
        // 1-el nagyobb mélység, ezért -1
        for (int i = 0; i < melyseg; ++i)
            printf ("---");
        printf ("%c(%d)\n", elem->ertek < 2 ? '0' + elem->ertek : elem->ertek ←
            ,
            melyseg-1);
        kiir (elem->bal_nulla);
        --melyseg;
    }
}

void
szabadit (BINFA_PTR elem)
{
    if (elem != NULL)
    {
        szabadit (elem->jobb_egy);
        szabadit (elem->bal_nulla);
        free (elem);
    }
}
```

6.3. Fabejárás

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

Megoldás videó:

Egy fa szerkezetű fájl 3 módon lehet bejárni inorder, preorder, és posztorderrel ha van egy alap fánk mely (1,2,3,4,5)

```
Inorder (Left, Root, Right) : 4 2 5 1 3
Preorder (Root, Left, Right) : 1 2 4 5 3
Postorder (Left, Right, Root) : 4 5 2 3 1
```


Preorder

```
void
kiir (BINFA_PTR elem)
{
    if (elem != NULL)
    {
        ++melyseg;
        if (melyseg > max_melyseg)
            max_melyseg = melyseg;
        kiir (elem->jobb_egy);
        // ez a postorder bejáráshoz képest
        // 1-el nagyobb mélység, ezért -1
        kiir (elem->bal_nulla);
        for (int i = 0; i < melyseg; ++i)
            printf ("---");
        printf ("%c(%d)\n", elem->ertek < 2 ? '0' + elem->ertek : elem->ertek ←
            ,
            melyseg-1);
        --melyseg;
    }
}
```

Posztorder

```
void
kiir (BINFA_PTR elem)
{
    if (elem != NULL)
    {
        ++melyseg;
        if (melyseg > max_melyseg)
            max_melyseg = melyseg;
        for (int i = 0; i < melyseg; ++i)
            printf ("---");
        printf ("%c(%d)\n", elem->ertek < 2 ? '0' + elem->ertek : elem->ertek ←
            ,
            melyseg-1);
        kiir (elem->jobb_egy);
        // ez a postorder bejáráshoz képest
        // 1-el nagyobb mélység, ezért -1
        kiir (elem->bal_nulla);
        --melyseg;
    }
}
```

```
}
```

6.4. Tag a gyökér

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

Megoldás videó:

Megoldás forrása:

Az első újdonság az osztály lesz, mely lényegében a C forrásban megismert struktúrának a továbbgondolása. Az osztályban már nem csak változók egy csoportját tudjuk együtt kezelni, hanem írhatunk bele függvényeket is. Az osztályt egy konstruktorral kezdjük ami csak annyit csinál, hogy a fa mutatót a gyökér elem memóriacímére állítjuk utána meg a szabadít függvényt hívjuk. A következőben a void operatorban látjuk a C++ egyik érdekességét ezt hívjuk operátor túlterhelésnek, melynek segítségével tudjuk módosítani, hogy egy operátor mit csináljon ez abban segít hogy a saját típusainkat tudja kezelni. A továbbiban ugyanúgy mint a C-s feladatnál feltölti a fát a leírt módszerrel. Utána a kiír függvény nem kap paramétert, csak meghívja a másik kiír függvényt, és a melysegértékét 0-ra állítja. A get* függvényre azért lesz szükségünk mert a változók amiket használni akarunk azok privát részben találhatóak.

```
class LZWBinFa
{
public:
    LZWBinFa (char b = '/') : betu (b), balNulla (NULL), jobbEgy (NULL) ←
    {};
    ~LZWBinFa () {};
    void operator<<(char b)
{
    if (b == '0')
    {
        // van '0'-s gyermeke az aktuális csomópontnak?
        if (!fa->nullasGyermeke ()) // ha nincs, csinálunk
        {
            Csomopont *uj = new Csomopont ('0');
            fa->ujNullasGyermeke (uj);
            fa = &gyoker;
        }
        else // ha van, arra lépünk
        {
            fa = fa->nullasGyermeke ();
        }
    }
    else
    {
        if (!fa->egyenesGyermeke ())
        {
```

```
        Csomopont *uj = new Csomopont ('1');
        fa->ujEgyesGyermekek (uj);
        fa = &gyoker;
    }
    else
    {
        fa = fa->egyesGyermekek ();
    }
}

class Csomopont
{
public:
    Csomopont (char b = '/'):betu (b), balNulla (0), jobbEgy (0) {};
    ~Csomopont () {};
    Csomopont *nullasGyermekek () {
        return balNulla;
    }
    Csomopont *egyesGyermekek ()
    {
        return jobbEgy;
    }
    void ujNullasGyermekek (Csomopont * gy)
    {
        balNulla = gy;
    }
    void ujEgyesGyermekek (Csomopont * gy)
    {
        jobbEgy = gy;
    }
private:
    friend class LZWBinFa;
    char betu;
    Csomopont *balNulla;
    Csomopont *jobbEgy;
    Csomopont (const Csomopont &);
    Csomopont &operator=(const Csomopont &);
};

if (b == '0')
{
    if (!fa->nullasGyermekek ())
    {
        Csomopont *uj = new Csomopont ('0');
        fa->ujNullasGyermekek (uj);
        fa = &gyoker;
    }
}

int main ()
{
    char b;
```

```
LZWBinFa binFa;
while (std::cin >> b)
{
    binFa << b;
}
binFa.kiir ();
binFa.szabadit ();
return 0;
}
```

6.5. Mutató a gyökér

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

Megoldás videó:

Megoldás forrása:

Az LZWBinfa protected részében megadott gyökeret Csomópont -hoz kell írni egy *-ot.

Aztán a konstruktornak nem a gyökeret kell odaadni, hanem ezt a gyökér mutatót, és ezt fogjuk a fá-ban tárolni, mint aktuális elem.

Ezek után le kell cserélni a gyökér referenciákat csak simán gyökérre mert már nem egy elem hanem egy mutató.

```
class LZWBinFa
{
public:

    LZWBinFa ():fa (&gyoker)
    {
    }
    ~LZWBinFa ()
    {
        szabadit (gyoker.egyesGyermekek ());
        szabadit (gyoker.nullasGyermekek ());
    }
}
```

```
protected:
    Csomopont gyoker;
};
```

```
public:
```

```
LZWBinFa ()
{
    gyoker = new Csomopont();
    fa = gyoker;
}
~LZWBinFa ()
{
    szabadit (gyoker->egyenesGyermekek ());
    szabadit (gyoker->nullasGyermekek ());
    delete gyoker;
}
```

```
protected:
    Csomopont *gyoker;
};
```

6.6. Mozzgató szemantika

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

Megoldás videó:

Megoldás forrása:

Erre a feladatra felhasználnám a 2. passzomat amit SMNISTforHUMANS lvl9 eléréséért kaptam.

A feladat megoldásához az osztály elején meg kell adnunk két további publikus függvényt.

```
LZWBinFa ( LZWBinFa && regi ){
    std::cout << "LZWBinFa move ctor" << std::endl;

    gyoker.ujEgyenesGyermekek ( regi.gyoker.egyenesGyermekek() );
    gyoker.ujNullasGyermekek ( regi.gyoker.nullasGyermekek() );

    regi.gyoker.ujEgyenesGyermekek ( nullptr );
    regi.gyoker.ujNullasGyermekek ( nullptr );
}

LZWBinFa& operator = (LZWBinFa &&& regi){
    if (this == &regi)
        return *this;

    gyoker.ujEgyenesGyermekek ( regi.gyoker.egyenesGyermekek() );
```

```
gyoker.ujNullasGyermeke ( regi.gyoker.nullasGyermeke() );  
  
regi.gyoker.ujEgyesGyermeke ( nullptr );  
regi.gyoker.ujNullasGyermeke ( nullptr );  
  
return *this;  
}
```

Ezen függvények segítségével, már tudjuk mozgatni az objektumainkat szabályosan.

7. fejezet

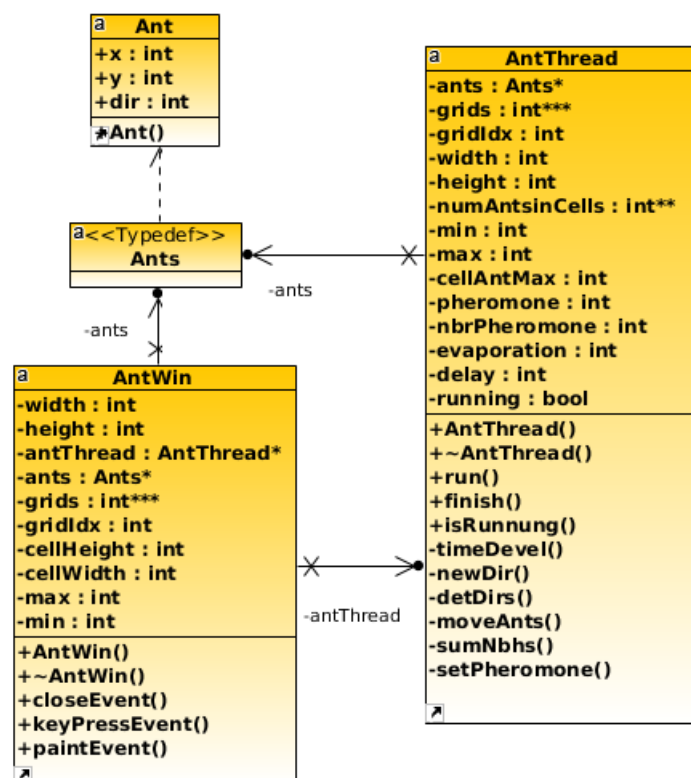
Helló, Conway!

7.1. Hangyaszimulációk

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

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

Megoldás forrása:

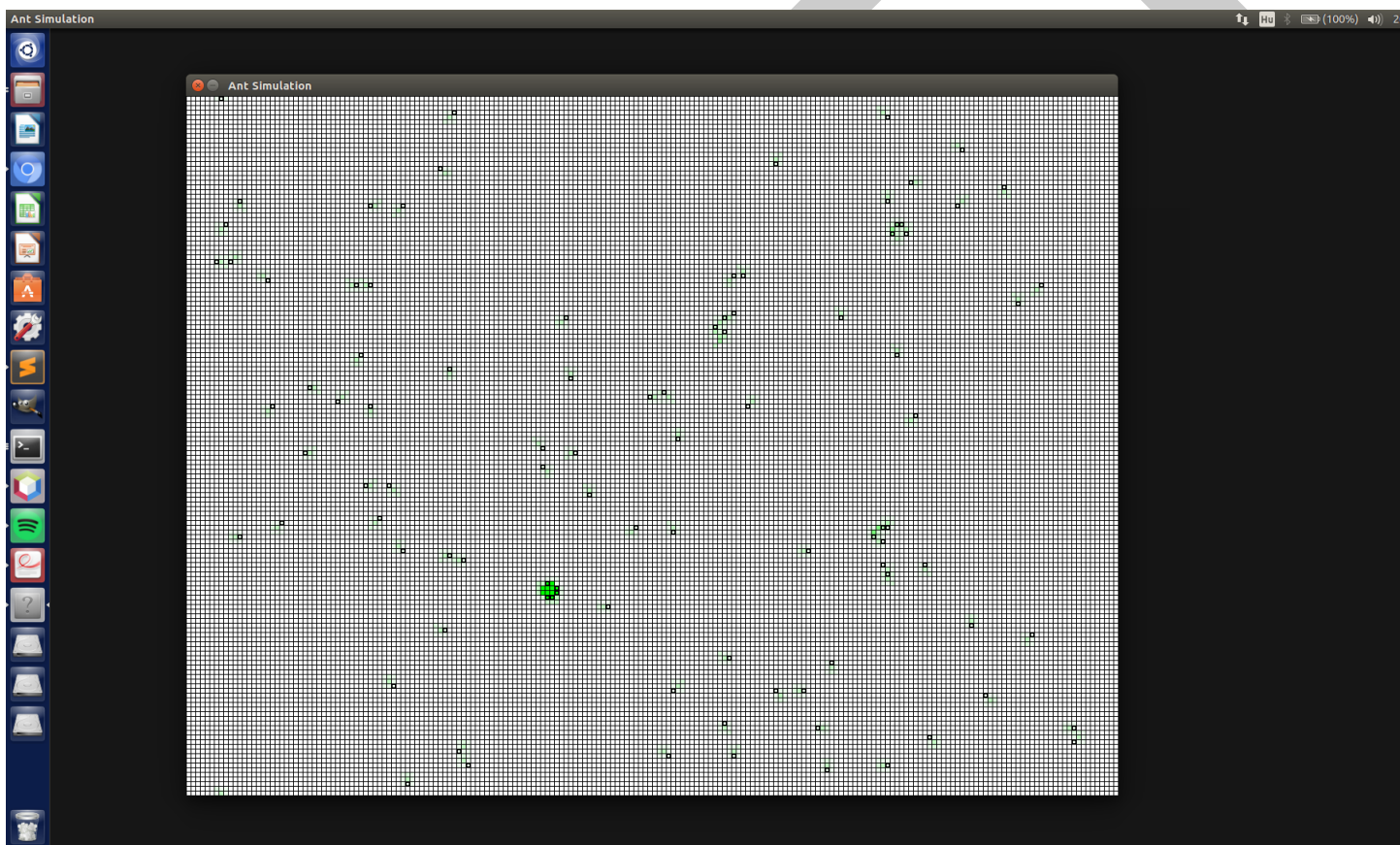


Fordításkor QT-t használunk qmake-et használjuk. Q gombal kilépünk P gombal megállítuk a programot.

A program fordítása és futtatása:

```
qmake myrmecologist.pro
make
./myrmecologist -w 250 -m 150 -n 400 -t 10 -p 5 -f 80 -d 0 -a 255 -i 3 -s 3 ↵
-c 22
```

Az Ant osztályba a hangyáknak vannak benne a tulajdonságaik . Az x és y a koordinátái, a dir pedig az irány amibe tart. Az ants egy vektor, ami a hangyákat tárolja. A feladatot kezdem azzal hogy mik van az AntWin osztályban ablak szélessége és magassága van benne pixeleken, cellák szélessége, magassága amiben a hangyák vannak. Az antThread a számításokat végzi. A grids a két rácsot jelenti, a gridldx pedig a a két rácspont közül egyet tárol. A closeEvent() mint nevéből tippelhető bezárja a programot úgy hogy meghívja a finish metódust. keyPressEvent() megint mint nevéből tippelhető billentyűzet inputot figyel, a paintEvent() meg a hangyák kiszínezésére figyel.



7.2. Java életjáték

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

Megoldás videó:

Megoldás forrása:

Az életjátékot John Conway, a Cambridge Egyetem matematikusa találta ki. A "játékos" vagyis nekünk annyi a szerepünk hogy meg kell adnunk egy kezdőalakzatot és ezután hátradőlni és nézni a műsort. Matematikai szempontból sejtautomatának nevezzük Conway játékát, melyben a négyzetrácsokat celláknak

és a korongokat sejteknek nevezzük. A sejt túléli a kört, ha két vagy három szomszédja van. A sejt elpusztul, ha kettőnél kevesebb (elszigetelődés), vagy háromnál több (túlnépesedés) szomszédja van. Új sejt születik minden olyan cellában, melynek környezetében pontosan három sejt található. Ezek a szabályok a `időFejlődés()` függvényben kerülnek bevezetésre. A sejtterbe "élőlényeket" helyezünk, ez a sikló-ágyú. Adott irányban siklókat lő ki. Az `addKeyListener`, `addMouseListener`, `addMouseMotionListener` függvényekben meghatározzuk a egér és billentyű kombinációk feladatait

```
Pár érdekes gomb funkció
-s Screenshotoz
-n Sejt méret növelés
-k Sejt méret csökkentés
-g Szimuláció gyorsítása
-l Szimuláció lassítása
Ha az egeret egy halott sejtre rávisszük akkor feléled :)
```

```
public class Sejtautomata extends java.awt.Frame implements Runnable {
    public static final boolean ÉLŐ = true;
    public static final boolean HALOTT = false;
    protected boolean [][][] rácsek = new boolean [2][][];
    protected boolean [][] rác;
    protected int rácIndex = 0;
    protected int cellaSzélesség = 20;
    protected int cellaMagasság = 20;
    protected int szélesség = 20;
    protected int magasság = 10;
    protected int várakozás = 1000;
    private java.awt.Robot robot;
    private boolean pillanatfelvétel = false;
    private static int pillanatfelvételSzámláló = 0;

    public Sejtautomata(int szélesség, int magasság) {
        this.szélesség = szélesség;
        this.magasság = magasság;
        rácsek[0] = new boolean[magasság][szélesség];
        rácsek[1] = new boolean[magasság][szélesség];
        rácIndex = 0;
        rác = rácsek[rácIndex];
        for(int i=0; i<rác.length; ++i)
            for(int j=0; j<rác[0].length; ++j)
                rác[i][j] = HALOTT;
        siklóKilövő(rác, 5, 60);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                setVisible(false);
                System.exit(0);
            }
        });
    }
}
```

```
addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent e) {
        if(e.getKeyCode() == java.awt.event.KeyEvent.VK_K) {
            cellaSzélesség /= 2;
            cellaMagasság /= 2;
            setSize(Sejtautomata.this.szélesség*cellaSzélesség,
                Sejtautomata.this.magasság*cellaMagasság);
            validate();
        } else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_N) {
            cellaSzélesség *= 2;
            cellaMagasság *= 2;
            setSize(Sejtautomata.this.szélesség*cellaSzélesség,
                Sejtautomata.this.magasság*cellaMagasság);
            validate();
        } else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_S)
            pillanatfelvétel = !pillanatfelvétel;
        else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_G)
            várakozás /= 2;
        else if(e.getKeyCode() == java.awt.event.KeyEvent.VK_L)
            várakozás *= 2;
        repaint();
    }
});

addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent m) {
        int x = m.getX()/cellaSzélesség;
        int y = m.getY()/cellaMagasság;
        rácsok[rácsIndex][y][x] = !rácsok[rácsIndex][y][x];
        repaint();
    }
});

addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
    public void mouseDragged(java.awt.event.MouseEvent m) {
        int x = m.getX()/cellaSzélesség;
        int y = m.getY()/cellaMagasság;
        rácsok[rácsIndex][y][x] = ÉLŐ;
        repaint();
    }
});

cellaSzélesség = 10;
cellaMagasság = 10;
try {
    robot = new java.awt.Robot(
        java.awt.GraphicsEnvironment.
            getLocalGraphicsEnvironment().
            getDefaultScreenDevice());
} catch(java.awt.AWTException e) {
```

```
        e.printStackTrace();
    }

    setTitle("Sejtautomata");
    setResizable(false);
    setSize(szélesség*cellaSzélesség,
            magasság*cellaMagasság);
    setVisible(true);
    new Thread(this).start();
}

public void paint(java.awt.Graphics g) {
    boolean [][] rács = rácsok[rácsIndex];
    for(int i=0; i<rács.length; ++i) {
        for(int j=0; j<rács[0].length; ++j) {
            if(rács[i][j] == ÉLŐ)
                g.setColor(java.awt.Color.BLACK);
            else
                g.setColor(java.awt.Color.WHITE);
            g.fillRect(j*cellaSzélesség, i*cellaMagasság,
                    cellaSzélesség, cellaMagasság);
            g.setColor(java.awt.Color.LIGHT_GRAY);
            g.drawRect(j*cellaSzélesség, i*cellaMagasság,
                    cellaSzélesség, cellaMagasság);
        }
    }
}

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

public int szomszédokSzáma(boolean [][] rács,
        int sor, int oszlop, boolean állapot) {
    int állapotúSzomszéd = 0;
    for(int i=-1; i<2; ++i)
        for(int j=-1; j<2; ++j)
            if(!((i==0) && (j==0))) {
                int o = oszlop + j;
                if(o < 0)
                    o = szélesség-1;
                else if(o >= szélesség)
                    o = 0;

                int s = sor + i;
```

```
        if(s < 0)
            s = magasság-1;
        else if(s >= magasság)
            s = 0;

        if(rács[s][o] == állapot)
            ++állapotúSzomszéd;
        }

        return állapotúSzomszéd;
    }

    public void időFejlődés() {

        boolean [][] rácsElőtte = rácsok[rácsIndex];
        boolean [][] rácsUtána = rácsok[(rácsIndex+1)%2];

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

                int élők = szomszédokSzáma(rácsElőtte, i, j, ÉLŐ);

                if(rácsElőtte[i][j] == ÉLŐ) {

                    if(élők==2 || élők==3)
                        rácsUtána[i][j] = ÉLŐ;
                    else
                        rácsUtána[i][j] = HALOTT;
                } else {

                    if(élők==3)
                        rácsUtána[i][j] = ÉLŐ;
                    else
                        rácsUtána[i][j] = HALOTT;
                }
            }
        }

        rácsIndex = (rácsIndex+1)%2;
    }

    public void run() {

        while(true) {
            try {
                Thread.sleep(várákozás);
            } catch (InterruptedException e) {}

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

```
}

public void sikló(boolean [][] rács, int x, int y) {

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

}

public void siklóKilövő(boolean [][] rács, int x, int y) {

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

    rács[y+ 3][x+ 13] = ÉLŐ;

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

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

    rács[y+ 6][x+ 11] = ÉLŐ;
    rács[y+ 6][x+ 15] = ÉLŐ;
    rács[y+ 6][x+ 16] = ÉLŐ;
    rács[y+ 6][x+ 22] = ÉLŐ;
    rács[y+ 6][x+ 23] = ÉLŐ;
    rács[y+ 6][x+ 24] = ÉLŐ;
    rács[y+ 6][x+ 25] = ÉLŐ;

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

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

```
        rács[y+ 8][x+ 35] = ÉLŐ;

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

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

        rács[y+ 11][x+ 25] = ÉLŐ;

    }

    public void pillanatfelvétel(java.awt.image.BufferedImage felvetel) {
        StringBuffer sb = new StringBuffer();
        sb = sb.delete(0, sb.length());
        sb.append("sejtautomata");
        sb.append(++pillanatfelvételSzámláló);
        sb.append(".png");
        try {
            javax.imageio.ImageIO.write(felvetel, "png",
                new java.io.File(sb.toString()));
        } catch (java.io.IOException e) {
            e.printStackTrace();
        }
    }

    public void update(java.awt.Graphics g) {
        paint(g);
    }

    public static void main(String[] args) {
        new Sejtautomata(100, 75);
    }
}
```

A program fordítása és futtatása:

```
$ javac Sejtautomata.java
$ java Sejtautomata
```

7.3. Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A szabály implementálása ugyanúgy mint javaban: sejt túléli a kört, ha két vagy három szomszédja van. A sejt elpusztul, ha kettőnél kevesebb (elszigetelődés), vagy háromnál több (túlnépesedés) szomszédja van. Új sejt születik minden olyan cellában, melynek környezetében pontosan három sejt található.

```
void SejtSzal::idoFejlodes() {
    bool **racsElotte = racsok[racsIndex];
    bool **racsUtana = racsok[(racsIndex+1)%2];
    for(int i=0; i<magassag; ++i) { // sorok
        for(int j=0; j<szelesseg; ++j) { // oszlopok
            int elok = szomszedokSzama(racsElotte, i, j, SejtAblak::ELO);
            if(racsElotte[i][j] == SejtAblak::ELO) {

                if(elok==2 || elok==3)
                    racsUtana[i][j] = SejtAblak::ELO;
                else
                    racsUtana[i][j] = SejtAblak::HALOTT;
            } else {

                if(elok==3)
                    racsUtana[i][j] = SejtAblak::ELO;
                else
                    racsUtana[i][j] = SejtAblak::HALOTT;
            }
        }
    }
    racsIndex = (racsIndex+1)%2;
}
```

Magát a konkrét szabályokat az idoFejlodes eljárás tartatja be, itt dől el, hogy egy sejt milyen állapotba kerül. Elsőnek meghatározzuk, hogy melyik rácsból váltunk melyikbe. Majd ezalapján a racsUtana rácsba betöltjük a sejtállapotokat. Ahhoz, hogy ezt megtegyük, tudnunk kell, hogy az egyes sejteknek a racsElotte rácsban milyen szomszédai voltak. Ehhez meghívjuk az imént taglalt szomszedokSzama függvényt.

Sikló:

```
void SejtAblak::siklo(bool **racs, int x, int y) {

    racs[y+ 0][x+ 2] = ELO;
    racs[y+ 1][x+ 1] = ELO;
    racs[y+ 2][x+ 1] = ELO;
    racs[y+ 2][x+ 2] = ELO;
    racs[y+ 2][x+ 3] = ELO;
```

```
}
```

Siklókilövő

```
void SejtAblak::sikloKilovo(bool **racs, int x, int y) {  
  
    racs[y+ 6][x+ 0] = ELO;  
    racs[y+ 6][x+ 1] = ELO;  
    racs[y+ 7][x+ 0] = ELO;  
    racs[y+ 7][x+ 1] = ELO;  
  
    racs[y+ 3][x+ 13] = ELO;  
  
    racs[y+ 4][x+ 12] = ELO;  
    racs[y+ 4][x+ 14] = ELO;  
  
    racs[y+ 5][x+ 11] = ELO;  
    racs[y+ 5][x+ 15] = ELO;  
    racs[y+ 5][x+ 16] = ELO;  
    racs[y+ 5][x+ 25] = ELO;  
  
    racs[y+ 6][x+ 11] = ELO;  
    racs[y+ 6][x+ 15] = ELO;  
    racs[y+ 6][x+ 16] = ELO;  
    racs[y+ 6][x+ 22] = ELO;  
    racs[y+ 6][x+ 23] = ELO;  
    racs[y+ 6][x+ 24] = ELO;  
    racs[y+ 6][x+ 25] = ELO;  
  
    racs[y+ 7][x+ 11] = ELO;  
    racs[y+ 7][x+ 15] = ELO;  
    racs[y+ 7][x+ 16] = ELO;  
    racs[y+ 7][x+ 21] = ELO;  
    racs[y+ 7][x+ 22] = ELO;  
    racs[y+ 7][x+ 23] = ELO;  
    racs[y+ 7][x+ 24] = ELO;  
  
    racs[y+ 8][x+ 12] = ELO;  
    racs[y+ 8][x+ 14] = ELO;  
    racs[y+ 8][x+ 21] = ELO;  
    racs[y+ 8][x+ 24] = ELO;  
    racs[y+ 8][x+ 34] = ELO;  
    racs[y+ 8][x+ 35] = ELO;  
  
    racs[y+ 9][x+ 13] = ELO;  
    racs[y+ 9][x+ 21] = ELO;  
    racs[y+ 9][x+ 22] = ELO;  
    racs[y+ 9][x+ 23] = ELO;  
    racs[y+ 9][x+ 24] = ELO;  
    racs[y+ 9][x+ 34] = ELO;  
    racs[y+ 9][x+ 35] = ELO;  
}
```



```
racs[y+ 10][x+ 22] = ELO;
racs[y+ 10][x+ 23] = ELO;
racs[y+ 10][x+ 24] = ELO;
racs[y+ 10][x+ 25] = ELO;

racs[y+ 11][x+ 25] = ELO;

}
```

A siklokilovo függvény pedig előre állítja racs mutató által mutatott rács megfelelő tagjait.

```
#include <QApplication>
#include "sejtablak.h"
#include <QDesktopWidget>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    SejtAblak w(100, 75);
    w.show();

    return a.exec();
}
```

Futtatása:

```
qmake -project
qmake sejtautomata_c++.pro
make
./sejtautomata_c++
```

7.4. BrainB Benchmark

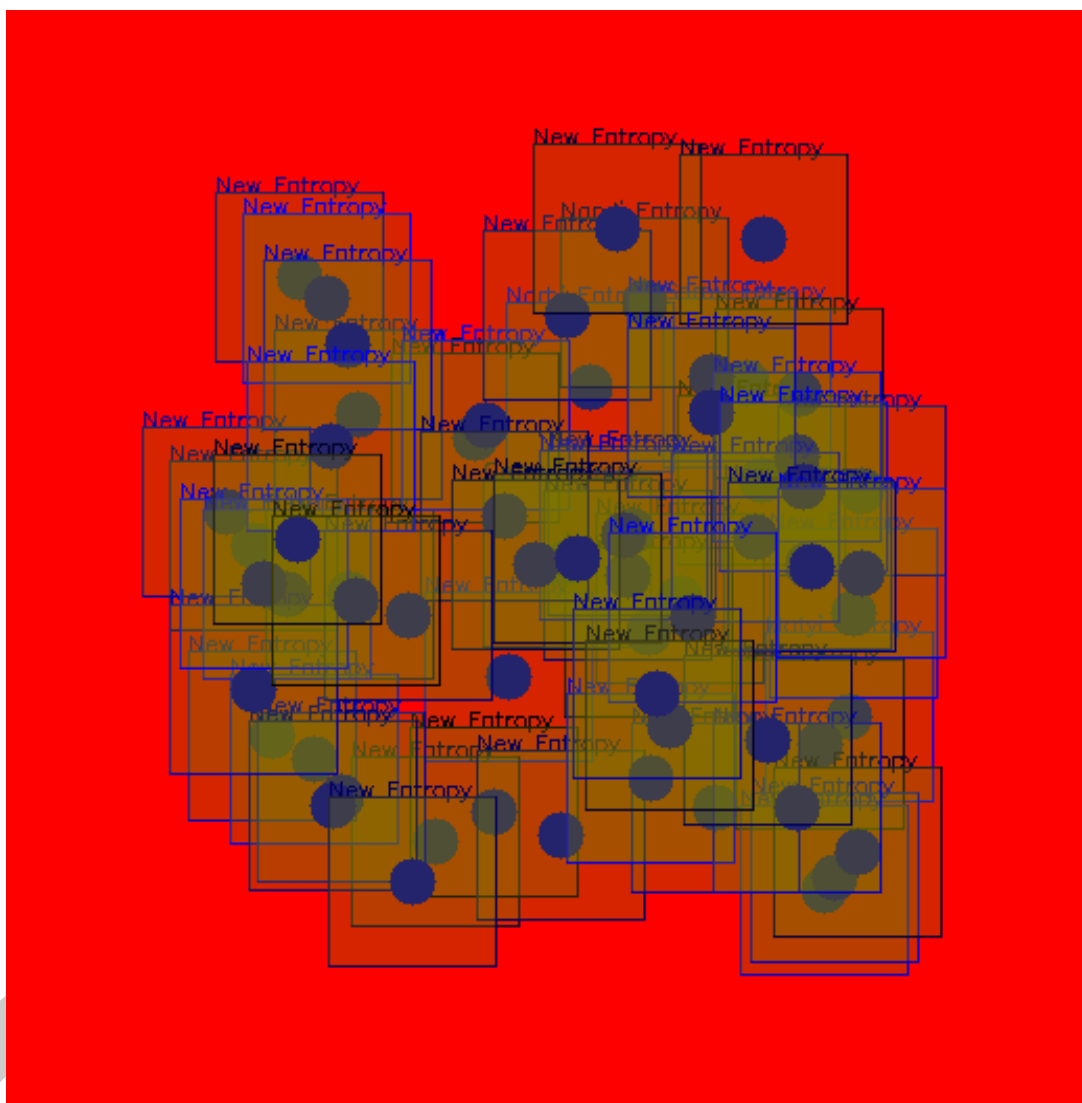
Megoldás videó:

Megoldás forrása: https://github.com/lacikaq9/bhax/tree/master/bhax-master/attention_raising/brainb

Tanulságok, tapasztalatok, magyarázat... A BrainB Benchmark e-sportolók tesztelésére / megtalálására íródott a játék gyakorlatilag a reflexet és az égerkövetési skillt követi. A mérőprogram arra a játékelményre, jelenségre épül, mely során a játékos átmenetileg elveszíti a karakterét az intenzív, komplex pillanatokban. Ezekben a pillanatokban direkt túl sok vizuális effektet hoz be a benchmark hogy megnehezítse a játékos dolgát. Ebben a programban a 'Samu Entropy' karaktert (dobozt) kell figyelniünk: csak akkor sikerül mérés ha egy másodpercig rajta tudjuk tartani az egerünket a kék pögyre. Az értékek növekedésével egyre több doboz (vizuális elem) jelenik meg a képernyőn, a dobozok pedig egyre gyorsabban kezdenek el mozogni. Ez a benchmark addig megy míg a játékos 1 másodperc felett tudja tartani az égerkövetés ha a játékos túllép akkor veszít.

Futtatáshoz:

```
brainB.pro  
make  
./BrainB
```



8. fejezet

Helló, Schwarzenegger!

8.1. Szoftmax Py MNIST

Python

Megoldás videó:

Megoldás forrása:

a felépítés és a futtatása: <https://www.youtube.com/watch?v=wQ8BIBpya2k>

a környezet felépítése windowsban: https://github.com/jeffheaton/t81_558_deep_learning Amerikai deep learning tanár

Az MNIST egy olyan program ami a Tensorflow alapot használja. A tensorflow egy mesterséges intelligenciához használt dolog gyakorlatilag a tensorflow keras segítségével próbál szimulálni egy emberi agyat amelynek saját neurális hálózata van amivel saját magát tudja tanítani. Ebben a feladatban a program azt tanulja meg hogy vannak kezzelírt számok 0-9 ig és a gép feladata hogy egyre nagyobb pontossággal eltalálja. Minél több ideig engedjük futtatni a programot egyre pontosabb lesz a számok felismerésével

A feladatot windowson oldottam meg ennek a telepítése: Elsőnek is szükségünk van egy python alapra ha ez nincs meg en a miniconda javasolom mert nem tart sok ideig a telepítése és könnyű használni ha ezt megtettük. A windows keresősávba be kell írunk anaconda command prompt és ebben a command prompt-ban telepítünk már fontos dolgot

```
conda install jupyter, conda install scipy, pip install --upgrade sklearn, pip install --upgrade pandas, pip install --upgrade pandas-datareader, pip install --upgrade matplotlib, pip install --upgrade pillow, pip install --upgrade requests, pip install --upgrade h5py, pip install --upgrade pyyaml, pip install --upgrade psutil, pip install --upgrade tensorflow, pip install --upgrade keras,
```

Ha ezekkel megvagyunk az anaconda promptba beírjuk hogy jupyter notebook ez létrehoz egy miniwebserver-t és egy virtuális felületet a munkához

A program tensorflowal és keras-al

```
import tensorflow as tf # deep learning library. Tensors are just multi-  
dimensional arrays | itt hívjuk be a tensorflowot tf néven
```

```
mnist = tf.keras.datasets.mnist # mnist is a dataset of 28x28 images of ↵
    handwritten digits and their labels | mnist maga az adatbázis amibe 28 ↵
    x28 kepek vannak
(x_train, y_train), (x_test, y_test) = mnist.load_data() # unpacks images ↵
    to x_train/x_test and labels to y_train/y_test | kicsomagoljuk a kepeket

x_train = tf.keras.utils.normalize(x_train, axis=1) # scales data between ↵
    0 and 1 | számokat atalítjuk 0-255 ról 0-1 ig
x_test = tf.keras.utils.normalize(x_test, axis=1) # scales data between 0 ↵
    and 1 | számokat atalítjuk 0-255 ról 0-1 ig

model = tf.keras.models.Sequential() # a basic feed-forward model | ez a " ↵
    taplalasi mód"
model.add(tf.keras.layers.Flatten()) # takes our 28x28 and makes it 1x784 ↵
    | ezzel kilapositjuk ad adatforrast 28x28 ról 1x784
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu)) # a simple ↵
    fully-connected layer, 128 units, relu activation
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu)) # a simple ↵
    fully-connected layer, 128 units, relu activation
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax)) # our ↵
    output layer. 10 units for 10 classes. Softmax for probability ↵
    distribution

model.compile(optimizer='adam', # Good default optimizer to start with
              loss='sparse_categorical_crossentropy', # how will we ↵
              calculate our "error." Neural network aims to minimize ↵
              loss.
              metrics=['accuracy']) # what to track

model.fit(x_train, y_train, epochs=3) # train the model | ezeket kezdjuk a ↵
    tanulást

val_loss, val_acc = model.evaluate(x_test, y_test) # evaluate the out of ↵
    sample data with model | a hiba % számoljuk ezzel
print(val_loss) # model's loss (error) | hiba érték
print(val_acc) # model's accuracy | pontosság értéke
```

Pythonban

```
# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
```

```

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ↵
=====

"""Builds the MNIST network.
Implements the inference/loss/training pattern for model building.
1. inference() - Builds the model as far as required for running the ↵
    network
forward to make predictions.
2. loss() - Adds to the inference model the layers required to generate ↵
    loss.
3. training() - Adds to the loss model the Ops required to generate and
apply gradients.
This file is used by the various "fully_connected_*.py" files and not meant ↵
    to
be run.
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import math

import tensorflow as tf

# The MNIST dataset has 10 classes, representing the digits 0 through 9.
NUM_CLASSES = 10

# The MNIST images are always 28x28 pixels.
IMAGE_SIZE = 28
IMAGE_PIXELS = IMAGE_SIZE * IMAGE_SIZE

def inference(images, hidden1_units, hidden2_units):
    """Build the MNIST model up to where it may be used for inference.
    Args:
        images: Images placeholder, from inputs().
        hidden1_units: Size of the first hidden layer.
        hidden2_units: Size of the second hidden layer.
    Returns:
        softmax_linear: Output tensor with the computed logits.
    """
    # Hidden 1
    with tf.name_scope('hidden1'):
        weights = tf.Variable(
            tf.truncated_normal([IMAGE_PIXELS, hidden1_units],
                                stddev=1.0 / math.sqrt(float(IMAGE_PIXELS))),

```

```
        name='weights')
    biases = tf.Variable(tf.zeros([hidden1_units]),
                        name='biases')
    hidden1 = tf.nn.relu(tf.matmul(images, weights) + biases)
# Hidden 2
with tf.name_scope('hidden2'):
    weights = tf.Variable(
        tf.truncated_normal([hidden1_units, hidden2_units],
                            stddev=1.0 / math.sqrt(float(hidden1_units))),
        name='weights')
    biases = tf.Variable(tf.zeros([hidden2_units]),
                        name='biases')
    hidden2 = tf.nn.relu(tf.matmul(hidden1, weights) + biases)
# Linear
with tf.name_scope('softmax_linear'):
    weights = tf.Variable(
        tf.truncated_normal([hidden2_units, NUM_CLASSES],
                            stddev=1.0 / math.sqrt(float(hidden2_units))),
        name='weights')
    biases = tf.Variable(tf.zeros([NUM_CLASSES]),
                        name='biases')
    logits = tf.matmul(hidden2, weights) + biases
return logits

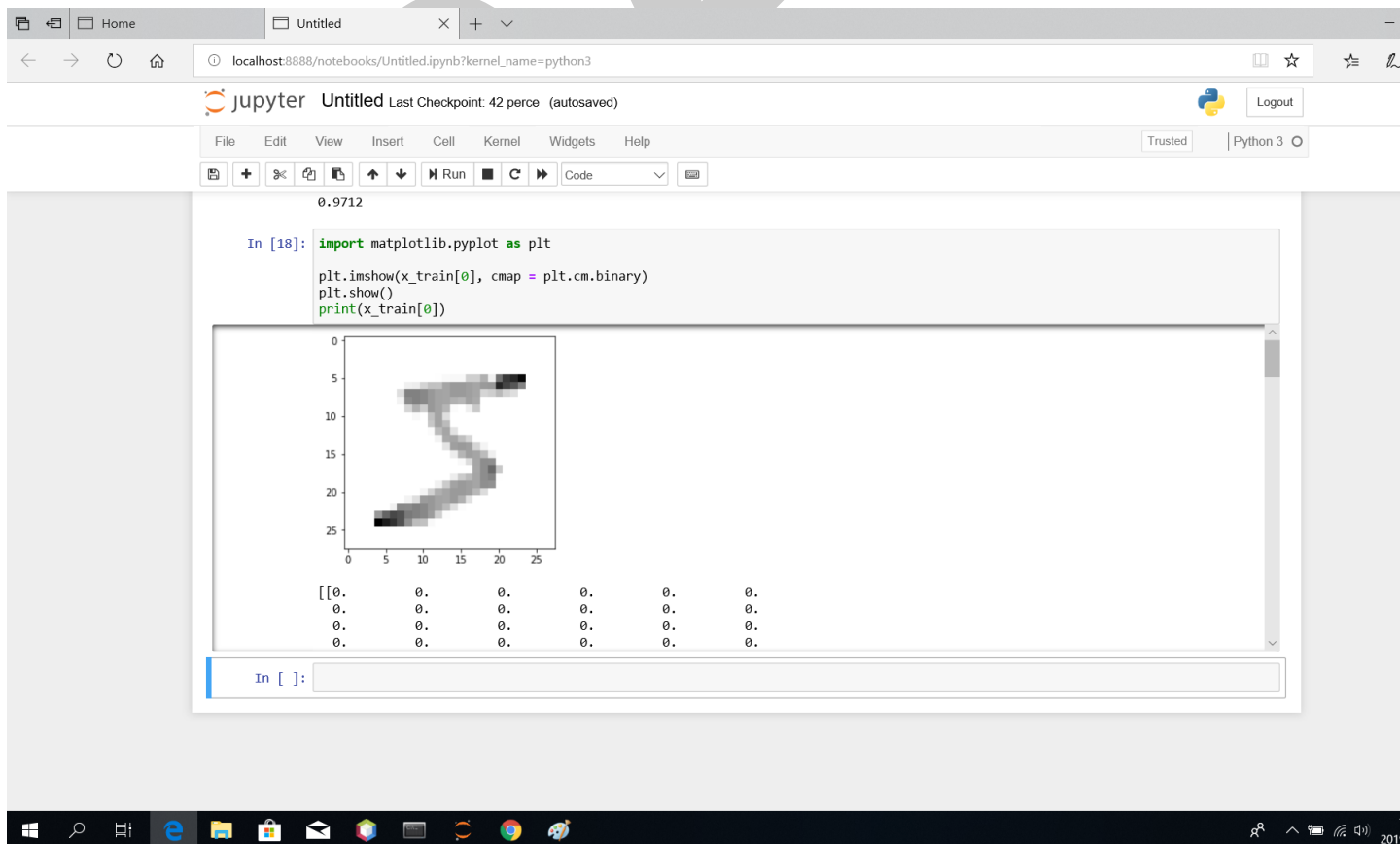
def loss(logits, labels):
    """Calculates the loss from the logits and the labels.
    Args:
        logits: Logits tensor, float - [batch_size, NUM_CLASSES].
        labels: Labels tensor, int32 - [batch_size].
    Returns:
        loss: Loss tensor of type float.
    """
    labels = tf.to_int64(labels)
    return tf.losses.sparse_softmax_cross_entropy(labels=labels, logits= ↵
        logits)

def training(loss, learning_rate):
    """Sets up the training Ops.
    Creates a summarizer to track the loss over time in TensorBoard.
    Creates an optimizer and applies the gradients to all trainable variables ↵
    .
    The Op returned by this function is what must be passed to the
    'sess.run()' call to cause the model to train.
    Args:
        loss: Loss tensor, from loss().
        learning_rate: The learning rate to use for gradient descent.
    Returns:
        train_op: The Op for training.
```

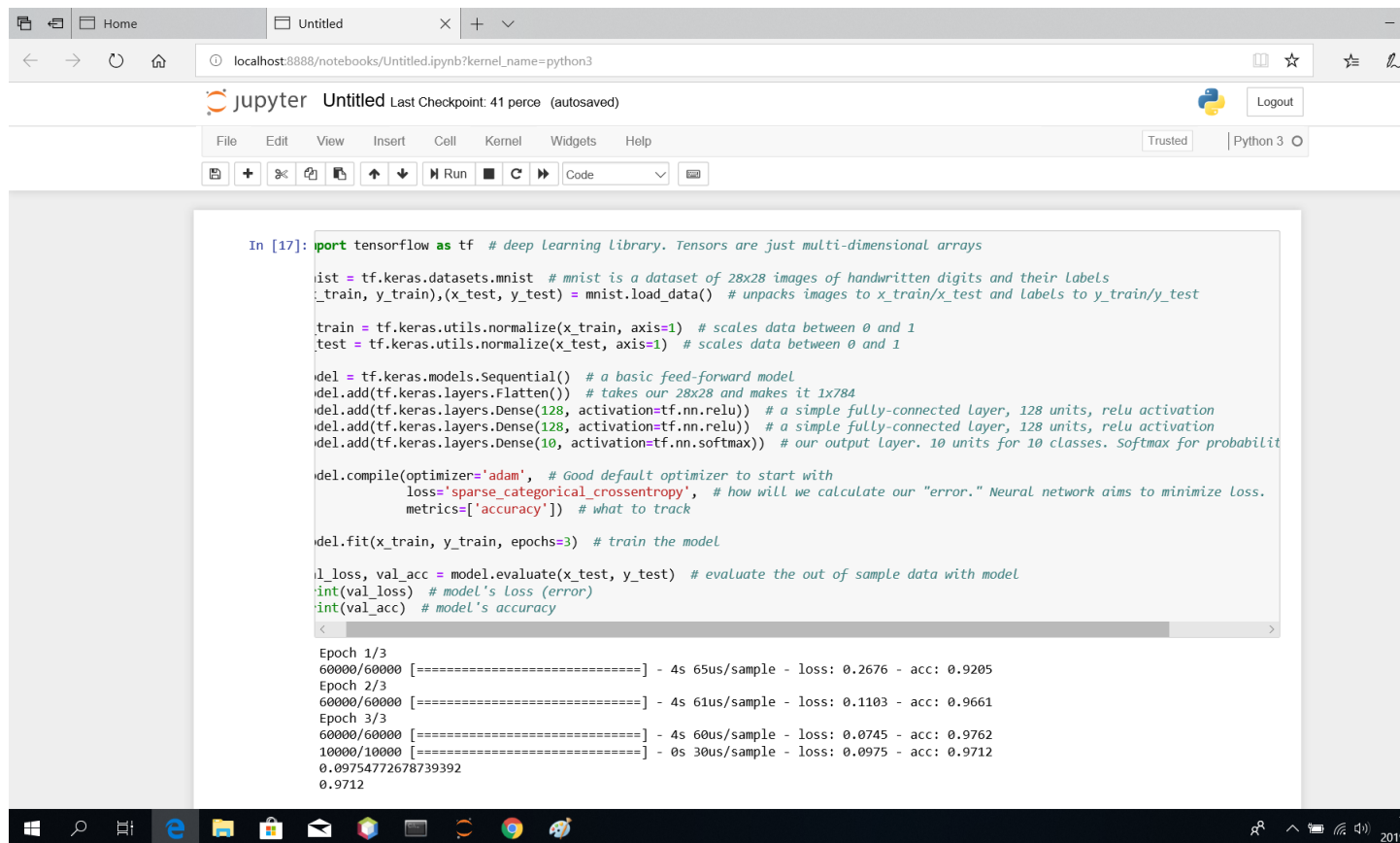
```
"""
# Add a scalar summary for the snapshot loss.
tf.summary.scalar('loss', loss)
# Create the gradient descent optimizer with the given learning rate.
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
# Create a variable to track the global step.
global_step = tf.Variable(0, name='global_step', trainable=False)
# Use the optimizer to apply the gradients that minimize the loss
# (and also increment the global step counter) as a single training step.
train_op = optimizer.minimize(loss, global_step=global_step)
return train_op

def evaluation(logits, labels):
    """Evaluate the quality of the logits at predicting the label.
    Args:
        logits: Logits tensor, float - [batch_size, NUM_CLASSES].
        labels: Labels tensor, int32 - [batch_size], with values in the
            range [0, NUM_CLASSES).
    Returns:
        A scalar int32 tensor with the number of examples (out of batch_size)
        that were predicted correctly.
    """
    # For a classifier model, we can use the in_top_k Op.
    # It returns a bool tensor with shape [batch_size] that is true for
    # the examples where the label is in the top k (here k=1)
    # of all logits for that example.
    correct = tf.nn.in_top_k(logits, labels, 1)
    # Return the number of true entries.
    return tf.reduce_sum(tf.cast(correct, tf.int32))
```

Az adatbázis



A training



```
In [17]: import tensorflow as tf # deep learning library. Tensors are just multi-dimensional arrays

x_train, y_train, x_test, y_test = mnist.load_data() # mnist is a dataset of 28x28 images of handwritten digits and their labels
# unpacks images to x_train/x_test and labels to y_train/y_test

x_train = tf.keras.utils.normalize(x_train, axis=1) # scales data between 0 and 1
x_test = tf.keras.utils.normalize(x_test, axis=1) # scales data between 0 and 1

model = tf.keras.models.Sequential() # a basic feed-forward model
model.add(tf.keras.layers.Flatten()) # takes our 28x28 and makes it 1x784
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu)) # a simple fully-connected layer, 128 units, relu activation
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu)) # a simple fully-connected layer, 128 units, relu activation
model.add(tf.keras.layers.Dense(10, activation=tf.nn.softmax)) # our output layer. 10 units for 10 classes. Softmax for probabilities

model.compile(optimizer='adam', # Good default optimizer to start with
              loss='sparse_categorical_crossentropy', # how will we calculate our "error." Neural network aims to minimize loss.
              metrics=['accuracy']) # what to track

model.fit(x_train, y_train, epochs=3) # train the model

_, loss, val_acc = model.evaluate(x_test, y_test) # evaluate the out of sample data with model
print(val_loss) # model's loss (error)
print(val_acc) # model's accuracy

Epoch 1/3
60000/60000 [=====] - 4s 65us/sample - loss: 0.2676 - acc: 0.9205
Epoch 2/3
60000/60000 [=====] - 4s 61us/sample - loss: 0.1103 - acc: 0.9661
Epoch 3/3
60000/60000 [=====] - 4s 60us/sample - loss: 0.0745 - acc: 0.9762
10000/10000 [=====] - 0s 30us/sample - loss: 0.0975 - acc: 0.9712
0.09754772678739392
0.9712
```

8.2. Szoftmax R MNIST

R

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.3. Mély MNIST

Python

Megoldás videó:

Megoldás forrása:

A mély MNIST ugyanolyan mint az MNIST csak training közbe jobban javul a pontossága és létrehoz egy gráfot a fejlődéséről amit nekem a C:\Users\asus\AppData\Local\Temp\ mappában teszi sajnos arra nem jöttem rá hogyis nyissam meg ezt a fájlt de legalább megvan.

A feladat amúgy nagyon megdolgoztatja a gépet mint a screenshotomról is látható hogy a 4 magos 8 szás gépemen 100% -osan leterheli és a memóit is nagyon szereti

```
Extracting /tmp/tensorflow/mnist/input_data/train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data/t10k-labels-idx1-ubyte.gz
Saving graph to: C:\Users\asus\AppData\Local\Temp\tmpopltx9xf
step 0, training accuracy 0.04
step 100, training accuracy 0.9
step 200, training accuracy 0.86
step 300, training accuracy 0.94
step 400, training accuracy 0.9
step 500, training accuracy 0.98
step 600, training accuracy 0.96
step 700, training accuracy 0.94
step 800, training accuracy 0.94
step 900, training accuracy 1
step 1000, training accuracy 1
step 1100, training accuracy 0.96
step 1200, training accuracy 0.98
step 1300, training accuracy 0.94
step 1400, training accuracy 0.96
step 1500, training accuracy 0.96
step 1600, training accuracy 0.94
step 1700, training accuracy 1
step 1800, training accuracy 1
step 1900, training accuracy 0.98
step 2000, training accuracy 0.98
step 2100, training accuracy 0.98
step 2200, training accuracy 0.98
step 2300, training accuracy 0.98
step 2400, training accuracy 0.98
step 2500, training accuracy 0.98
step 2600, training accuracy 0.98
step 2700, training accuracy 0.96
step 2800, training accuracy 1
step 2900, training accuracy 1
step 3000, training accuracy 1
step 3100, training accuracy 0.94
step 3200, training accuracy 1
step 3300, training accuracy 1
step 3400, training accuracy 0.94
step 3500, training accuracy 0.98
step 3600, training accuracy 1
step 3700, training accuracy 0.98
step 3800, training accuracy 0.94
step 3900, training accuracy 0.9
step 4000, training accuracy 0.98
step 4100, training accuracy 0.98
step 4200, training accuracy 1
```

```
step 4300, training accuracy 0.96
step 4400, training accuracy 1
```

```

# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ↵
=====

"""A deep MNIST classifier using convolutional layers.
See extensive documentation at
https://www.tensorflow.org/get_started/mnist/pros
"""
# Disable linter warnings to maintain consistency with tutorial.
# pylint: disable=invalid-name
# pylint: disable=g-bad-import-order

from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import sys
import tempfile

from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf

FLAGS = None

def deepnn(x):
    """deepnn builds the graph for a deep net for classifying digits.
    Args:
        x: an input tensor with the dimensions (N_examples, 784), where 784 is ↵
```

```
    the
    number of pixels in a standard MNIST image.
Returns:
    A tuple (y, keep_prob). y is a tensor of shape (N_examples, 10), with ↵
    values
    equal to the logits of classifying the digit into one of 10 classes ( ↵
    the
    digits 0-9). keep_prob is a scalar placeholder for the probability of
    dropout.
"""
# Reshape to use within a convolutional neural net.
# Last dimension is for "features" - there is only one here, since images ↵
    are
# grayscale -- it would be 3 for an RGB image, 4 for RGBA, etc.
with tf.name_scope('reshape'):
    x_image = tf.reshape(x, [-1, 28, 28, 1])

# First convolutional layer - maps one grayscale image to 32 feature maps ↵
    .
with tf.name_scope('conv1'):
    W_conv1 = weight_variable([5, 5, 1, 32])
    b_conv1 = bias_variable([32])
    h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)

# Pooling layer - downsamples by 2X.
with tf.name_scope('pool1'):
    h_pool1 = max_pool_2x2(h_conv1)

# Second convolutional layer -- maps 32 feature maps to 64.
with tf.name_scope('conv2'):
    W_conv2 = weight_variable([5, 5, 32, 64])
    b_conv2 = bias_variable([64])
    h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)

# Second pooling layer.
with tf.name_scope('pool2'):
    h_pool2 = max_pool_2x2(h_conv2)

# Fully connected layer 1 -- after 2 round of downsampling, our 28x28 ↵
    image
# is down to 7x7x64 feature maps -- maps this to 1024 features.
with tf.name_scope('fc1'):
    W_fc1 = weight_variable([7 * 7 * 64, 1024])
    b_fc1 = bias_variable([1024])

    h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
    h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

# Dropout - controls the complexity of the model, prevents co-adaptation ↵
    of
```

```
# features.
with tf.name_scope('dropout'):
    keep_prob = tf.placeholder(tf.float32)
    h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

# Map the 1024 features to 10 classes, one for each digit
with tf.name_scope('fc2'):
    W_fc2 = weight_variable([1024, 10])
    b_fc2 = bias_variable([10])

    y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
    return y_conv, keep_prob

def conv2d(x, W):
    """conv2d returns a 2d convolution layer with full stride."""
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
    """max_pool_2x2 downsamples a feature map by 2X."""
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1], padding='SAME')

def weight_variable(shape):
    """weight_variable generates a weight variable of a given shape."""
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    """bias_variable generates a bias variable of a given shape."""
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

def main(_):
    # Import data
    mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)

    # Create the model
    x = tf.placeholder(tf.float32, [None, 784])

    # Define loss and optimizer
    y_ = tf.placeholder(tf.float32, [None, 10])

    # Build the graph for the deep net
    y_conv, keep_prob = deepnn(x)
```

```

with tf.name_scope('loss'):
    cross_entropy = tf.nn.softmax_cross_entropy_with_logits(labels=y_,
                                                            logits=y_conv)
cross_entropy = tf.reduce_mean(cross_entropy)

with tf.name_scope('adam_optimizer'):
    train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)

with tf.name_scope('accuracy'):
    correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
    correct_prediction = tf.cast(correct_prediction, tf.float32)
    accuracy = tf.reduce_mean(correct_prediction)

graph_location = tempfile.mkdtemp()
print('Saving graph to: %s' % graph_location)
train_writer = tf.summary.FileWriter(graph_location)
train_writer.add_graph(tf.get_default_graph())

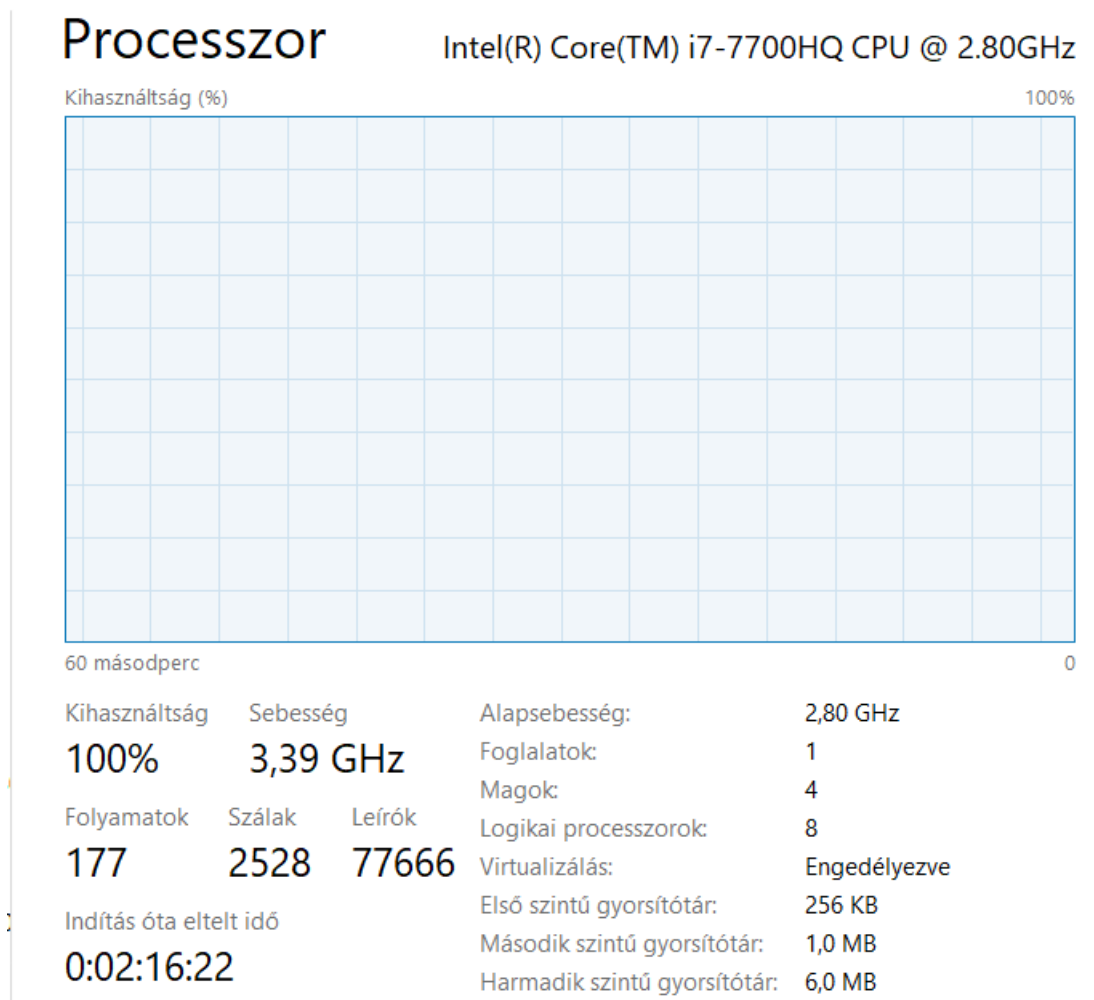
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(20000):
        batch = mnist.train.next_batch(50)
        if i % 100 == 0:
            train_accuracy = accuracy.eval(feed_dict={
                x: batch[0], y_: batch[1], keep_prob: 1.0})
            print('step %d, training accuracy %g' % (i, train_accuracy))
            train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

    print('test accuracy %g' % accuracy.eval(feed_dict={
        x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--data_dir', type=str,
                        default='/tmp/tensorflow/mnist/input_data',
                        help='Directory for storing input data')
    FLAGS, unparsed = parser.parse_known_args()
    tf.app.run(main=main, argv=[sys.argv[0]] + unparsed)

```

Név	Állapot	100% Processzor	62% Memória	0% Lemez	0% Hálózat	0% GPU
Python (4)		93,1%	1 058,8 MB	0 MB/s	0 Mb/s	0%
Python		93,1%	988,8 MB	0 MB/s	0 Mb/s	0%
Python		0%	58,4 MB	0 MB/s	0 Mb/s	0%
Python		0%	6,7 MB	0 MB/s	0 Mb/s	0%
Konzolablak-kezelő		0%	4,9 MB	0 MB/s	0 Mb/s	0%



8.4. Deep dream

Keras

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.5. Minecraft-MALMÖ

Megoldás videó:

Megoldás forrása:

Minecraft MALMO egy minecraft amibe lehet python c, c++, és java kóddal irányítani a karaktert és alakítani a világot főleg AI fejlesztésre használják. Ezeken kívül készíthtünk mission-öket az elobb felsorolt programozási nyelveken.

Windows-on való telepítése:

Először 3 dolog kell: JDK 8.0 telepítése ,Python 3.6 telepít, zip letöltése és kicsomagolása

2 módszer van a letöltésre

Githubról letöltjük az egész mappát

Vagy powershellbe bizonyos parancsokkal

2-es módszernél telepítéséhez ezeket kell beírni

```
Set-ExecutionPolicy -Scope CurrentUser Unrestricted | ↔  
Engedélyezzük a powershellnek külsős progok telepítését  
cd ahova kicsomagoltad\Malmo-0.35.6-Windows-64bit_Python3.6\ ↔  
scripts  
.\malmo_install.ps1
```

Ezzel kész is minecraft elindításához ezt kell beírni

```
cd ahol ki lett csomagolva\Malmo-0.37.0-Windows-64 ↔  
bit_withBoost_Python3.6\Minecraft  
.\launchclient
```

Ezzel el is indul a minecraft nem kell várni hogy a számláló 100% üssön 95% már indul és sose éri el a 100-at

És mondjuk egy missiont akarunk indítani python nyelvben

```
cd ahol ki lett csomagolva\Malmo-0.37.0-Windows-64 ↔  
bit_withBoost_Python3.6\Python_Examples  
python tutorial_1.py
```

Ezzel letesz minket a játék egy szerver alapu világba és 10 másodpercig egyhelyben állunk ennél vannak izgalmasabb küldetések is de csak eddig jutottam

Pár alap parancs

```
!! az 1 az truera állítja így folyamatosan csinálja a 0 falsera ↔  
azzal leállítjuk.  
agent_host.sendCommand("turn -0.5") | fordulás  
agent_host.sendCommand("move 1") | előre mozgás  
agent_host.sendCommand("jump 1") | ugrás  
agent_host.sendCommand("pitch 1") | sötétség  
time.sleep(1) | alvás  
agent_host.sendCommand("attack 1") | támadás
```


9. fejezet

Helló, Chaitin!

9.1. Iteratív és rekurzív faktoriális Lisp-ben

Megoldás videó:

Megoldás forrása:

A lisp az alapból egy információ feldolgozó nyelvnek szánták de az évek alatt inkább mesterséges intelligenciai algoritmusok írására. Ma a két fegyerjetebb lisp verziók azok a Common Lisp és a Scheme. A Common Lisp az egy multiparadigmás nyelv ami miatt inkrementális és evolúciós fejlesztésre használják. A Scheme pedig egy leegyszerűsített lisp amelyből mindent "fölösleges" tulajdonságát próbálták kivenni és emiatt a nyelv tanításra tökéletes.

Elsőnek is az iteratív jelentése az hogy ismétlődő ezért is használtam a do paracsot

```
> (define (fak n) (do ((i 1 (+ 1 i)) (num 1 (* i num))) ((> i n) num)))
fak
> (fak 7)
5040
```

A rekurzív pedig olyan hogy a program addig hívja meg magát ameddig nem kapja meg a problémára a választ

```
> (define (fakk n) (if (< n 1) 1 (* n (fakk (- n 1)))))
fakk
> (fakk 7)
5040
```

9.2. Weizenbaum Eliza programja

Éleszd fel Weizenbaum Eliza programját!

Megoldás videó:

Megoldás forrása:

9.3. Gimp Scheme Script-fu: króm effekt

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

Megoldás videó: https://youtu.be/OKdAkI_c7Sc

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

A Gimp egy primitív képszerkesztő program amelyben lehet scripteket írni előző feladatban beszélt Lisp nyelven azon belül is a Scheme nyelven. Tanárúr által kreált skriptet csak be kell másolni a GIMP 2\share\gimp2.0\scripts nevű mappába indítani a gimpet ráklikkelünk hogy fájl -> létrehozás -> bhax -> Chrome rá klikkelve beállíthatjuk: szöveget, betűtípus, betűtípus méretét, kép magassága, kép szélessége, a színt, és hogy milyen effecttel legyen chromosítva. Ezek után létrehozás és a script legenerál egy képet amin a megadott szövegnek chrome mintája van.

```
(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"
)
```

9.4. Gimp Scheme Script-fu: név mandala

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

Megoldás videó: https://bhaxor.blog.hu/2019/01/10/a_gimp_lisp_hackelese_a_scheme_programozasi_nyelv

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

Mint az előző feladatban ennek a script fajlat is be kell tenni a GIMP 2\share\gimp\2.0\scripts nevű mappába ugyanúgy fájl -> létrehozás -> bhax -> Mandala itt is betudjuk állítani a: szöveg 1 , szöveg 2, betűtípus , betűméret, szélesség , magasság , szín (ez a minta háttérszíne) , effect (maga a háttérnek és a körnek a színe) ha ezek beírtuk létrehozás és a script generál egy képet amin lesz egy szöveg középen és körülötte lesz egy 2 kör meg a szöveg mögött egy fura minta.

```
(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 (/ *pi* 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"
)
```

9.5. Lambda

Hasonlítsd össze a következő programokat!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9.6. Omega

Megoldás videó:

Megoldás forrása:

DRAFT

10. fejezet

Helló, Gutenberg!

10.1. Programozási alapfogalmak

[?]

Első rátekintésre láttam hogy ez a könyv lesz a legszárazabb a 3 közül és igazam is lett. Mint a cím sugalja az ebben a részben bevezet minket a könyv programozási nyelvi szintek. Bevezet hogy a számítógép hogy látja a kódunkat és hogy futtatja le és még kitér az imperatív és deklaratív nyelvekre.

Ez a hét a programozási nyelv alapeszközeit, alapfogalmait mutatja be. Karakterkészleteket olvashatunk azon belül címkék stb. Ez után a lexikális egyégekről van szó. Olvashatunk a többkarakteres szimbólumokról, a szimbolikus nevekről, a címkékről, megjegyzésekről és iterálokról, rengeteg példával. Majd a hét vége felé belevágunk az adattípusok világába. Azután meg tanulmányozzuk az alapelemek minden féle nyelvekben

Kifejezésekről olvashatunk ezen a héten amely C programnyelvben mutatja be

Utasításokról van szó 9 fajta van: 1. Értékadó utasítás 2. Üres utasítás 3. Ugró utasítás 4. Elágaztató utasítások (Kétirányú elágaztató utasítás (feltételes utasítás), Többirányú elágaztató utasítás) 5. Ciklusszervező utasítások (Feltételes ciklus, Elírt lépésszámú ciklus, Felsorolásos ciklus, Végtelen ciklus, Összetett ciklus) 6. Hívó utasítás 7. Vezérlésátadó utasítások 8. I/O utasítások 9. Egyéb utasítások A hét ezeket bőven kifejti példákkal több fajta programozási nyelveken.

Programok Szerkezet az 5. hét mondandója. Itt tanulunk az alprogramokról erről ugrunk a Hívási lánc, rekurzió témakörre. Utánna kitérünk Másodlagos belépési pontok

A hatodik hét, ami egy kihétten rövid hét, az absztrakt adattípusokról szól, amik megvalósítják a bezárást vagy információ rejtést.

A hetedik hét a csomagokról szól. Főbb elemei: típus változó nevesített konstans saját kivétel alprogram csomag. Plusz a könyv leírja ADA-ban a komplementálását

A 8. hét az ADA fordítáráról szól ezt 2 részre töri: Pragmák meg Fordítási egységek ezek a: alprogram specifikáció , alprogram törzs , csomag specifikáció , csomag törzs , fordítási alegység , valamint ezek tetszőleges kombinációja szól

A 9. hét a kivételkezelésről szól. Leginkább a PL/I kivételkezelésről és az ada kivételkezelésről.

10. hét a generikus programozást fejti ki ADA programozási nyelven

11.hét témája a párhuzamos programozás ami a felépüli gépek szekvenciálisak: a processzor a programnak megfelelő sorrendben hajtja végre az utasításokat elemi lépésenként. Pár beszélt téma

Véleményem a könyvről: Túl száraz.

10.2. Programozás bevezetés

[KERNIGHANRITCHIE]

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

Könyv első része szóval az alapokat fekteti le mint pl: változók, állandók, aritmetika, függvények, vezérlési szerkezetek, alap be/ki-vitel. Ezeknek az ismertetésével az olvasó kicsit belekóstol a programozás világába ez után mint minden tutorial ami csak létezik jön a hello world példa bár itt Hello mindenkinek nevezzük. Persze itt nem állunk le második feladat az egy Celsius és Fahrenheites táblázatot kiíró program. Harmadik feladatban már meg is kapjuk a klasszikus for ciklust. Ezek után kapunk egy szimbólikus állandókat. 5 alhétben karakterekkel való műveleteket látjuk pl: beolvasás, sorok számlálása. Utánna látunk egy rész tömbökkel való dolgok. Utolsó előtti részbe kapunk egy hatványozási részt. Végén befejezve beletekitünk az argumentumok világába

A C programnyelvre az jellemző, hogy kifejezetten kevés változótípust használ. int, double, char, float típusok jellemzőek.

Az állandóknak van típusa és minősítője.

A változókat mindig deklarálnunk kell. Ha C nyelvben akarunk változót deklarálni akkor a változóknév első karakterének betűnek kell lennie.

Egy olyan kifejezést hogy $v = v + 241$ helyettesíthetünk $v += 241$ -el. $v +=$ jelkombinációt értékadó operátornak nevezzük.

A feltételt az if segítségével határozzuk meg. Az else az if hamis ágának utasítását adja meg, használata nem kötelező. Többszörös feltétel esetén az if-else egymásba ágyazása a megoldás, illetve a switch-case. Amennyiben egyik case sem hajtott végre, úgy a default ág fut le, ez azonban nem kötelező. A switch-case-t a break utasítással zárjuk le.

A két leggyakoribb ciklus a for és a while. Ezek a típusú ciklusok addig futnak amíg false-t vagyis hamist nem kapnak.

A do-while kap 2 utasítást do(csinald) while(amig) ez program minden alkalommal legalább 1x lefut.

A break parancsal megállítja a programot és continue -val megkezdhetjük a következő iterációs lépés.

A goto utasítással megadott címekre ugorhatunk és onna folytatjuk. Nem kedvelt megoldás.

10.3. Programozás

[BMECPP]

17. oldal objektumok és osztályok címmel startol ahol el is kezdi magyarázni hogy mik azok az objektumorientált nyelvek alapelveit. Ezekután rátérünk az egységbe záras c++-ban a ahol elmagyarázza mi az a tagváltozók és mik a tagfüggvények. Átbeszéli mik azok a konstruktorok és a destruktorkok. A következő

részt tökre szerettem mert sokat segített végre megértettem hogy kell dinamikus memóriát kezelni na meg persze a másoló konstruktort elmagyarázta és példáknak hála megértettem hogy is kell őket használni. 44. oldaltól kitérünk a Friend függvények és osztályokra pontosan leírja példákkal teljesen érthető. 47 oldaltól tagváltozókat beszél át a könyv utána a statikus tagokat és végül a beágyazott definiókat

Operátorok és túlterhelésük. C++ előzményekkel kezdődik, és beszél az operátorokról általában is. Függvénytaxis és a túlterhelés a következő, itt is sok sok példa és kódcsipetke melyek sokat segítenek az elméleti rész felfogásában.

utolsó rész

A C nyelvben lehetetlen mivel úgy lett megírva de van egy kiskapu azt kell csinálni hogy a változóra mutató mutatót adunk át. C++-ban ez lehetséges, a referencia szerinti paraméterátadással

Összesség:

A három könyv közül ezt találom a leghasznosabban mert minden elméleti részhez van egy kis c kód részlet és amikor azokon magyarázza el bármilyen nehéznek tűnő elméleti részt levezet a példákon keresztül

III. rész

Második felvonás

DRAFT

**Bátf41 Haxor Stream**

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

DRAFT

11. fejezet

Helló, Arroway!

11.1. A BPP algoritmus Java megvalósítása

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

11.2. Java osztályok a Pi-ben

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

IV. rész

Irodalomjegyzék

11.3. Általános

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

11.4. C

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

11.5. C++

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

11.6. Lisp

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

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

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