Univerzális programozás

Egy programozós könyv Tony Stark tollából.

Ed. Dékány Róbert, Debrecen, 2019. február 25, v. 0.0.5

KÖZREMÜKÖDTEK			
	CÍM : Univerzális programozás		
HOZZÁJÁRULÁS	*NÉV*	*DÁTUM*	*ALÁÍRÁS*
ÍRTA	Dékány Róbert Zsolt	2019. május 7.	

VERZIÓTÖRTÉNET			
VERZIÓ	DÁTUM	LEÍRÁS	NÉV
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	Aktualizálás, javítások.	nbatfai

Ajánlás

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

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

Tartalomjegyzék

Ábrák jegyzéke

Előszó

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

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

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

Hogyan forgasd

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

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

Milyen nyelven nyomjuk?

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

Hogyan nyomjuk?

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

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

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

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

A DocBook XML 5.1 új neked?
Ez esetben forgasd a https://tdg.docbook.org/tdg/5.1/ könyvet, a végén találod az informatikai szövegek jelölésére használható gazdag „API” elemenkénti bemutatását.

I. rész

Bevezetés

1. fejezet

Vízió

1.1. Mi a programozás?

1.2. Milyen doksikat olvassak el?

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

1.3. Milyen filmeket nézzek meg?

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

II. rész

Tematikus feladatok

Bátf41 Haxor Stream
A feladatokkal kapcsolatos élő adásokat sugároz a https://www.twitch.tv/nbatfai csatorna, melynek permanens archívuma a https://www.youtube.com/c/nbatfai csatornán található.

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!

Végtelen ciklus, egy olyan "állatfajta" a programozás világában, amivel a kisgyerekeket riogatják az iskolákban. Jobb elkerülni, de akarva vagy akaratlanul néha bele-belefutunk. Ha még nem láttál végtelen ciklust, íme egy C nyelven:

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

    return 0;
}
```

A fenti kódot a `gcc inf.c -o inf` parancsal fordítjuk, majd `./inf` -el futtatjuk. Ha szépen lefutott a `top` parancs segítségével megtudjuk nézni a CPU állapotát! A futtatott programunk sorában tisztán látszik, hogy a CPU 100%-on pörög.

Ha egy olyan végtelen ciklust szeretnénk, ahol a CPU terheltsége a 0%-hoz közeli, akkor a `sleep(seconds)` függvényt kell használunk a ciklusmagban. Minden egyes ciklus lefutásnál a programszál "altatva" van a sleep paraméteréül megadott x másodpercig, így elérve a 0% CPU állapotot.

```
#include <unistd.h>

int main ()
{
    for (;;)
    {
        sleep (1);
    }

    return 0;
```

```
    }
```

Minden magot 100%-on terhelni többszálaszított végtelen ciklussal tudunk. Ehhez a `#pragma omp parallel` utasítást kell alkalmaznunk a for ciklusra, majd a `gcc teszt-feladat.c -o teszt -fopenmp` parancssal fordítjuk. A `top` utasítással le tudjuk ellenőrizni, hogy valóban minden mag 100%-on pörög.

```
    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 vlgtelen ciklus:

```
Program T100
{

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

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

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

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

akár önmagára

```
T100(T100)
false
```

ezt a kimenetet adja.

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

```
Program T1000
{

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

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

  main(Input Q)
  {
    Lefagy2(Q)
  }

}
```

Mit for kiírni erre a `T1000(T1000)` futtatásra?

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

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

2.3. Változók értékének felcserélése

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

Számtalan olyan feladat vagy probléma létezik, ahol két változók kell felcserélni majd ezekkel dolgozni tovább. Programozókként az egyszerű s nagyszerű elvnek megfelelően 2 különböző módszert nézünk meg a változók felcserélésére.

I. Segédváltozós csere:

```
#include <stdio.h>

int main ()
{
    int a = 99; int b = 45;

    int c = a;
    a = b;
    b = c;

    printf("A értéke: %d\n", a);
    printf("B értéke: %d\n", b);

    return 0;
}
```

Rém egyszerű a történet. C segédváltozónak értékül adjuk magát az A értékét, majd az A értékét egyenlővé teszük a B-vel. Végül B értékét egyenlő lesz C értékével.

II. Segédváltozó nélkül egy kis matekkal:

```
#include <stdio.h>

int main ()
{
    int a = 99; int b = 45;

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

    printf("A értéke: %d\n", a);
    printf("B értéke: %d\n", b);

    return 0;
}
```

Egy kis összeadással és kivonással egyszerűen megoldható a két változó felcserélése, aki nem hiszi számolja ki.

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

2.4. Labdapattogás

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

Labda pattogtatásához elsőként a bejárandó területet kell definiálnunk a programban. A bejárandó terület a terminál ablakunk X és Y koordinátái fogják meghatározni. Ehhez felveszünk két konstans változót a `#define {VÁLTOZÓNÉV}` kulcsszó segítségével. Ez után szükség van egy `positionPrinting(int x, int y)` függvényre, ami a paraméterében átadott X és Y koordináták felhasználásával előállítja a labdakimenetet.

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

#define WIDTH 78
#define HEIGHT 22

int positionPrinting(int x, int y)
{
    int i;

    for(i=0; i<x; i++)
        printf("\n");

    for(i=0; i<y; i++)
        printf(" ");

    printf("&#x26bd;\n");

    return 0;
}
```

Ha mindez megvan elkészíthetjük a `main` metódusunkat. Itt két hosszú egész típusú változót kell inicializálnunk. Ezekben tároljuk az aktuális X és Y koordinátákat. Majd írunk egy végtelen ciklust. A ciklus magjában először minden egyes lefutásnál töröljük a képernyőt a `system("clear")` függvényel. Ezután kirajzolhatjuk a labdánkat az előbb elkészített `positionPrinting(abs(HEIGHT-(x++%(HEIGHT*2))),` függvény segítségével. Majd az `usleep(55000)` függvényel "altatjuk". Ezzel implementáljuk le a látszólag folyamatos labdamozgást.

```
int main()
{
    long int x=0;

    long int y=0;

    while(1)
    {
        system("clear");

        positionPrinting(abs(HEIGHT-(x++%(HEIGHT*2))),abs(WIDTH-(y++%( ↩
            WIDTH*2))));

        usleep(55000);
    }

    return 0;
}
```

Ezen program után rádöbbenhetünk, hogy elég egy console és bármiféle egyszerűbb játékot leimplementálhatunk benne. Következőképp el tudnék képzelni egy console-os amőba vagy akasztófa játékot. Ha lesz rá időm meg is csinálom. :)

Videó: https://bhaxor.blog.hu/2018/08/28/labdapattogas

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!

Az alábbi Linus Torvalds féle BogoMIPS egy program, ami segít a hibakeresésben illetve ellenőrizhető a számítógép számítási teljesítménye. Ellenőrizve az egymillió utasítás per másodperc végrehajtását végül visszaad egy indexszámot, ami jellemzi a processzorteljesítményt.

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

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

int main (void)
{
```

```
        unsigned long long int loops_per_sec = 1;
        unsigned long long int ticks;

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

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

            printf ("%llu %llu\n", ticks, loops_per_sec);

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

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

                return 0;
            }
        }

        printf ("failed\n");

        return -1;
    }
```

2.6. Helló, Google!

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

Középiskolában a (leg)népszerűbbnek számít az az ember, akinek sok barátja és ismerőse van. Kissé közhely de igaz. Ezen példát továbbgondolva rájöhetünk a hasonlóságra, ha az interneten található weboldalak népszerűségét figyeljük meg. Népszerű, ha az adott weboldara sok-sok link mutat. Ezen népszerűségi rangsorolást a Google-féle PageRank algoritmus végzi.

Az, hogy milyen "népszerű" egy adott oldal a PageRank algoritmus által hozzárendelt érték adja meg. Minél nagyobb annál népszerűbb. Továbbiakban tisztába kell lenni azzal, hogy egy weboldara kétféle link definiált. Vannak a rámutató (pointing) linkek, illetve a kimenő (outgoing) linkek. Pointing link az adott weboldara mutató linkek. Az outgoing linkek azon weboldal, ahova a weboldalunk mutat.

$$PR(h_2) = \sum_{h \in B(h_2)} \frac{PR(h)}{N(h)}$$

2.1. ábra. PageRank algoritmus

Elsőkörbe megírjuk a megjelenítő `kiir` függvényt. A paramétereként megkapott double vektoron végigiterál, majd kiiírja a PageRank értékeket az output-ra.

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

void kiir (vector<double> tomb)
{
    int i;
    for (i=0; i < tomb.size(); i++)
    printf("PageRank [%d]: %lf\n", i, tomb[i]);
}
```

`tavolsag` függvényt a paraméterben megkapott két vectorból számol egy közelítő távolságértéket. Ezt az értéket fogjuk vizsgálni a továbbiakban. Ha ez az érték közelíti vagy nagyobb a 0.00001 akkor nincs értelme tovább iterálni az a PR értékeket.

```
double tavolsag(vector<double> pagerank,vector<double>  ↩
   pagerank_temp)
{
    double tav = 0.0;
    int i;
    for(i=0;i < pagerank.size();i++)
    tav +=abs(pagerank[i] - pagerank_temp[i]);
    return tav;
}
```

A main függvényben történik a program lényegi része. Elsőként deklarálunk egy 4x4-es mátrixot. Ebbe lesz eltárolva a mutató linkek szerinti értékek amikkel dolgozni fogunk. Egy PR vektorban fogjuk eltárolni az épp aktuális PageRank értékeket. A PRv vektorban az egyes iterációknál meghatározott PageRank értékek. Alap esetben mind 1/4, mert 4 weblapunk van. Egy végtelen ciklusban fogjuk végrehajtani az egyes

iterációs lépéseket. Minden iterációban végigmegyünk a PR illetve a PRv elemeken és mátrixszorzással kiszámoltatjuk a PageRank értékeket. Ezeket a lépéseket addig ismételgetjük, amíg a `tavolsag` függvény vissza nem tér a megfelelő értékkel. A végén pedig a `kiír` -el kiiratunk.

```
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}
};

vector<double> PR = {0.0, 0.0, 0.0, 0.0};
vector<double> PRv = {1.0 / 4.0, 1.0 / 4.0, 1.0 / 4.0, 1.0 /  ↩
   4.0};

long int i,j,h;
i=0; j=0; h=5;

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

    for (i=0;i<4;i++)
    {
        double temp=0;

        for (j=0;j<4;j++)
            temp+=L[i][j]*PR[j];

        PRv[i]=temp;
    }

    if ( tavolsag(PR,PRv) < 0.00001)
    break;
}

kiir (PR);

return 0;

}
```

2.7. 100 éves a Brun tétel

Írj R szimulációt a Brun tétel demonstrálására!

```
#   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")
```

Kidolgozás később!

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

2.8. A Monty Hall probléma

Írj R szimulációt a Monty Hall problémára!

```
#   An illustration written in R for the Monty Hall Problem
#   Copyright (C) 2019  Dr. Norbert Bátfai, nbatfai@gmail.com
#
#   This program is free software: you can redistribute it and/or ↩
   modify
#   it under the terms of the GNU General Public License as published ↩
   by
#   the Free Software Foundation, either version 3 of the License, or
#   (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program.  If not, see <http://www.gnu.org/licenses/ ↩
   >
#
#   https://bhaxor.blog.hu/2019/01/03/erdos_pal_mit_keresett_a_ ↩
   nagykonyvben_a_monty_hall-paradoxon_kapcsan
#

kiserletek_szama=10000000
kiserlet = sample(1:3, kiserletek_szama, replace=T)
jatekos = sample(1:3, kiserletek_szama, replace=T)
musorvezeto=vector(length = kiserletek_szama)

for (i in 1:kiserletek_szama) {

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

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

    }else{

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

    }

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

}

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

for (i in 1:kiserletek_szama) {
```

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

}

valtoztatesnyer = which(kiserlet==valtoztat)


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

Megoldás videó: https://bhaxor.blog.hu/2019/01/03/erdos_pal_mit_keresett_a_nagykonyvben_a_monty_hall-paradoxon_kapcsan

Megoldás később!

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfjával megadva írd meg ezt a gépet!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Az unáris, azaz egyes számrendszer a legegyszerűbb számrendszer. Nevéből eredően egyes a legnagyobb jegye. Amikor az kezünkön elszámolunk ötig akkor unáris számrendszerben dolgozunk. Ez le is fedi a lényeget, egy k szám átírása unáris számrendszerbe k darab egyes lesz. Például: 5 az annyi, mint 11111.

Egy c++ demonstráció:

```
            #include <iostream>

void tounar(int a){
    for(int i=0; i<a; i++)
        std::cout << "1";

    std::cout << std::endl;
}

int main(){

    int val;
    std::cout << "Type a number in decimal." << std::endl;
    while(std::cin >> val){
        tounar(val);
    }

    return 0;
}
```

A gépünk feladata pontosan ez lenne. Amikor az szalagról az író/olvasó fej beolvas egy decimális számot, akkor egy eljárás által átváltsa unárisba azt.

A gépünk első feladata az lenne, hogy a szalagon érkező számot megvizsgálja. El lépked az utolsó számjegy utániig, majd visszalép az utolsóra és elkezd kivonni egyeket. Először az utolsó tagból, majd az utolsó előttiből és így tovább. A folyamat zajlani fog, amíg nulla nem lesz az olvasott szám. A kivont egyeseket elhelyezi a tárban, az unáris számrendszerbe átváltott számunkat onnan fogjuk tudni kinyerni.

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

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

A generatív nyelvtan megalkotása Noam Chomsky nyelvész, illetve matematikus nevéhez fűződik. A lentebb látható generatív grammatikák lényege abban rejlik, hogy miképp tudjuk kezdőszimbólumunkból a megadott szabályok segítségével csupán nem terminális jelekből, azaz konstansokból felépíteni a mondtatot.

Az első:

```
         S, X, Y változók
a, b, c konstansok
S - abc, S - aXbc, Xb - bX, Xc - Ybcc, bY - Yb, aY - aaX, aY - aa

  S (S - aXbc)
  aXbc (Xb - bX)
  abXc (Xc - Ybcc)
  abYbcc (bY - Yb)
  aYbbcc (aY - aaX)
  aaXbbcc (Xb - bX)
  aabXbcc (Xb - bX)
  aabbXcc (Xc - Ybcc)
  aabbYbccc (bY - Yb)
  aabYbbccc (bY - Yb)
  aaYbbbccc (aY - aa)
  aaabbbccc
```

A második:

```
        A, B, C változók
a, b, c konstansok
A - aAB, A - aC, CB - bCc, cB - Bc, C - bc

  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
```

Miért is nem környezetfüggő? Azért, mert nem fordul olyan elő, hogy a szabályok alakjánál a bal oldalon csak nem terminális jel lenne.

3.3. Hivatkozási nyelv

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Először is bemutatnám a kódcsipetet, amely a c89-es szabvánnyal nem fordul le, viszont a c99-esnél nem okoz problémát:

```
            #include <stdio.h>

int main(){

    int tomb[3]={1,2,3};

    for(int i=0;i<3;i++)
       printf("%d\n", tomb[i]);
}
//ez itt hiba lesz
```

Fordítás után látható is a hibaüzenet:

```
mate@ubuntu:~/Asztal$ gcc c89.c -o c89 -std=c89
c89.c: In function ‘main’:
c89.c:7:5: error: ‘for’ loop initial declarations are only allowed in C99 ↩
   or C11 mode
     for(int i=0;i<3;i++)
     ^~~
c89.c:7:5: note: use option -std=c99, -std=gnu99, -std=c11 or -std=gnu11 to ↩
    compile your code
c89.c: At top level:
c89.c:10:1: error: C++ style comments are not allowed in ISO C90
 //ez itt hiba lesz
```

```
 ^
c89.c:10:1: error: (this will be reported only once per input file)
```

3.4. 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.

Különböző szabványok kisebb nagyobb eltéréseket mutathatnak, így jól kell tudnunk, hogy milyen szabványokkal is dolgozunk. Az alábbi kód a C99 szabvány alapján lefut viszont a C89-el nem. A kódunkat szokásos gcc-vel fordítjuk, de ahhoz, hogy megtudjuk nézni a különbségeket a `-std:c89` illetve a `-std:c99` kapcsolókat kell alkalmaznunk.

```
int main()
{
    for(int i = 0; i < 5; i++)
    {
        /* code */
    }


    return 0;
}
```

Ha lefuttattuk C89 szabvány szerint, akkor az alábbi hibaüzenetet kaptuk: `error: ‘for’ loop initial declarations are only allowed in C99 or C11 mode`

A régi szabvány szerint a for fejlécében nem megengedett a változó deklaráció. Viszont, ha a deklarációt a foron kívülre helyezzük, akkor minden ok. Íme:

```
#include <unistd.h>

int main()
{
    int i;

    for(i = 0; i < 5; i++)
    {
        /* code */
    }
```

```
        return 0;
    }
```

3.5. 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!

A megoldás tömör lényege: Használunk egy programot, ami a megadott utasítások alapján elkészíti nekünk a megoldást.

Az általunk írt kód csupán egy .l kiterjesztésű pár soros állomány, amiből a lexer program állítja majd elő a c kódunkat.

Ez a következő képpen történik:

- A kódunk elején %{ és %} között megadhatunk C kód részleteket, amiket konkrétan látni akarunk a generált programban.

```
%{
//#include <stdio.h>
int realnum = 0;
%}
```

  Az a stadnard input/output könyvátr hozzáadása azért van kikommentelve, mert a streames megoldásban ugyan szerepel, de a sajátomból kihagytam, mivel a lexer által előállított C forrást tanulmányozva fölöslegesnek tartottam.(A lexer magától include-ol pár gyakran használt header filet, köztük ezt is.)

- Ez után következnek még az első részben a definícók, ami ebben a programban a számjegyek definiálását jelenti. Ezt a nevének és az általa felvehető karaktereknek a megadásával tehető meg. Miután megadtunk minden szükséges definíciót lezárjuk az első részt %% segítségével.

```
digit [0-9]
%%
```

- A második részben a megadott definíciókat is felhasználva megadjuk a szabályt/szabályokat ami alapján felismerhetünk a bemenetből bizonyos részeket.Itt csak egy szabály van, ami a valós számok "kinézetét" írja le általánosan és hogy mit tegyen akkor ha ilyet talál.

```
{digit}*(\.{digit}+)? {++realnum_count;
printf("[realnum: %s - %f]",yytext,atof(yytext));}
%%
```

  A valós szám formális megadása: Elől lehet számjegy(digit) 0 vagy több, ezt opcionálisan követheti pont(nem kötelező), de ha van pont, akkor utána muszáj egy vagy több számjegynek utána állnia.

  Ha talál ilyet, akkor a számlálót növeljük egyel és kiírjuk a megtalált kifejezést(yytext), valamint az `atof()` függvény segítségével a stringet lebegőpontos értékké alakítva is kiíratjuk. Ennek a szekciónak a végét is %%-jel jelöljük.

- Ezek után már csak a főprogram maradt, amiben meghívjuk a lexerünket az `yylex()` függvénnyel, valamint miután a lexeléssel végeztünk, a megtalált valós számok darabszámát is kiíratjuk.

```
int main(){
    yylex();
    printf("The number of real numbers: %d",realnum_count);
    return 0;
}
```

Az elkészült kódunkat bele öntjük egy flex nevű programba a következő képpen:

```
lex valos.l -o valos.c
```

Eredményül egy C forráskódot kapunk, ami az általunk írt .l kódnál jelentősen hosszabb(1700+ sor). Már csak arányait tekintve is rengeteg munkát spóroltunk meg vele.

A kapott C kódból a gcc-vel lefordítva előállíthatjuk a futtatható állományunkat így:

```
gcc valos.c -o valos -lfl
```

A -lfl kapcsolóra a lexerünk miatt van szükség.

Hogy mit jelent az óriások vállán állni és nem kispályázni? Ez az általunk megírt lex kód és a valós generált programkód közötti viszont reprezentálja. Ötletes!

3.6. l33t.l

Lexelj össze egy l33t ciphert!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

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

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

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

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

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

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
    */
    %{
    #include <stdio.h>
    #include <stdlib.h>
    #include <time.h>
    #include <ctype.h>

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

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

    {'a', {"4", "4", "@", "/-\\"}},
    {'b', {"b", "8", "|3", "|}"}},
    {'c', {"c", "(", "<", "{"}},
    {'d', {"d", "|)", "|]", "|}"}},
    {'e', {"3", "3", "3", "3"}},
    {'f', {"f", "|=", "ph", "|#"}},
    {'g', {"g", "6", "[", "[+"}},
    {'h', {"h", "4", "|-|", "[-]"}},
    {'i', {"1", "1", "|", "!"}},
    {'j', {"j", "7", "_|", "_/"}},
    {'k', {"k", "|<", "1<", "|{"}},
    {'l', {"l", "1", "|", "|_"}},
    {'m', {"m", "44", "(V)", "|\\/|"}},
    {'n', {"n", "|\\|", "/\\/", "/V"}},
    {'o', {"0", "0", "()", "[]"}},
    {'p', {"p", "/o", "|D", "|o"}},
    {'q', {"q", "9", "O_", "(,)"}},
    {'r', {"r", "12", "12", "|2"}},
    {'s', {"s", "5", "$", "$"}},
    {'t', {"t", "7", "7", "'|'"}},
    {'u', {"u", "|_|", "(_)", "[_]"}},
    {'v', {"v", "\\/", "\\/", "\\/"}},
    {'w', {"w", "VV", "\\/\\/", "(/\\)"}},
    {'x', {"x", "%", ")(", ")("}},
    {'y', {"y", "", "", ""}},
    {'z', {"z", "2", "7_", ">_"}},

    {'0', {"D", "0", "D", "0"}},
    {'1', {"I", "I", "L", "L"}},
```

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

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

%}
%%
. {

    int found = 0;
    for(int i=0; i<L337SIZE; ++i)
    {

        if(l337d1c7[i].c == tolower(*yytext))
        {

        int r = 1+(int) (100.0*rand()/(RAND_MAX+1.0));

        if(r<91)
            printf("%s", l337d1c7[i].leet[0]);
        else if(r<95)
            printf("%s", l337d1c7[i].leet[1]);
        else if(r<98)
            printf("%s", l337d1c7[i].leet[2]);
        else
            printf("%s", l337d1c7[i].leet[3]);

        found = 1;
        break;
        }

    }

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

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

```
    return 0;
    }
```

3.7. A források olvasása

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

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

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

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

---

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

Ha a SIGINT nem volt figyelmen kívül hagyva akkor a jelkezelő kezelje. Ha figyelmen kívül volt hagyva tovább is maradjon úgy.

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

A forciklus fejlécébe hiányzik az i deklaráció, ha ez megvan akkor ok.

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

A forciklus fejlécébe hiányzik az i deklaráció, ha ez megvan akkor ok.

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

A kód hibamentes ha már létrehoztuk a látható változókat és mutatókat.

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

Ugyanaz, mint az előzőnél.

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

A printf függvény ki fog írni 2 decimális számot ha már megvan az f függvény, az a változó, és ha az a változó megfelelő típusú az f függvényhez. Arra kell figyelni hogy ha az f függvény visszatérési értéke nem int akkor a kiírt értékek nem biztos hogy pontosak lesznek.

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

A printf ki fogja írni az f függvény visszatérési értékét a-ra, és a értékét.

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

A kiiratás megtörténik viszont az f függvény most az a változó memória címével fog dolgozni nem az a értékével.

3.8. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

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

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

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

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

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

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

Tanulságok, tapasztalatok, magyarázat...

3.9. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

- egész
```
int a;
```
- egészre mutató mutató
```
int *b;
```
- egész referenciája
```
int &c;
```

- egészek tömbje

```
int T[3];
```

- egészek tömbjének referenciája (nem az első elemé)

```
int (&TR)[3] = T;
```

- egészre mutató mutatók tömbje

```
int *T[3];
```

- egészre mutató mutatót visszaadó függvény

```
int *func();
```

- egészre mutató mutatót visszaadó függvényre mutató mutató

```
int *(*func)();
```

- egészet visszaadó és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó függvény

```
int *(*func)();
```

- függvénymutató egy egészet visszaadó és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó függvényre

```
int *(*func)();
```

Mit vezetnek be a programba a következő nevek?

-
```
int c[5];
```
Öt elemű tömb deklaráció.
-
```
int a;
```
Egész declaráció.
-
```
int *b = &a;
```
Egy pointer, ami az ’a’ változóra mutat.
-
```
int &r = a;
```
Az ’a’ változó referenciája.
-
```
int c[5];
```
Öt elemű tömb deklaráció.

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

  A c tömbre referenciája.
- ```
  int *d[5];
  ```

  Egy int-re mutató 5 elemű pointer tömb.
- ```
  int *h ();
  ```

  Egy int-re mutató függvény pointer.
- ```
  int *(*l) ();
  ```

  Egy int-re mutató függvénypointer pointere.
- ```
  int (*v (int c)) (int a, int b)
  ```

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...
  ```

4. fejezet

Helló, Caesar!

4.1. int *** háromszögmátrix

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

int
main ()
{
    int nr = 5;
    double **tm;

    printf("%p\n", &tm);

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

    printf("%p\n", tm);

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

    }

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

    for (int i = 0; i < nr; ++i)
        for (int j = 0; j < i + 1; ++j)
            tm[i][j] = i * (i + 1) / 2 + j;
```

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

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

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

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

        free (tm);

        return 0;
    }
```

Csak a nem nulla értékeket tároljuk, mivel a háromszöghöz felesleges ezeket az értékeket mutatni, így nem négyzet alakot kapunk. Egy 'i' sornak 'i+1' oszlopot foglalunk le. Elsőnek deklaráljuk a változóinkat. 'nr'-ben a háromszögmátrix sorainak a kívánt számát állítjuk le. '**tm' egy mutató, ami a tömbünk első elemére mutat, ami a double* mutatokra, tehát a tömb soraira mutat.

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Az EXOR titkosítás egy egyszerű titkosítási eljárás. Lényege, hogy a titosítandó szöveg mellé rendelünk egy titkosító kulcsot (szöveg), majd ezzel végezzük el a titkosítást. Az alábbi C kód az EXOR titkosító algoritmus.

```
    #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);
    }
}
```

Működése:

Elsőként a kódban felveszünk 2 konstans változót. A `MAX_KULCS 100` lesz a kulcsunk maximális mérete, illetve `BUFFER_MERET 256` a maximálisan beolvasható stringek száma. Ezek után a `main()` függvénybe valósul meg a titkosító algoritmus. El van tárolva a kulcs illetve a bufferbe a szöveg. Ezután a kulcs méretét vizsgáljuk a `strncpy` függvényel. Ha túl nagy a kulcs mérete, akkor lecípi belőle a szükséges 100 karakternyi részt.

```
#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);
```

Továbbiakban egy while ciklus segítségével folyamatosan olvassuk be a bájtokat a titkosítandó szöveges állományból. A magban a lényegi dolgok történnek. A kiolvasott bájtot össze EXOR-ozza a kulcs adott bájtja segítségével. Az EXOR után a kulcsindexet növeljük. A titkosított bájtokat egy bufferbe olvassuk, majd kiirjuk egy output file-ba.

```
        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!

```
        public class ExorTitkosító
        {
            public ExorTitkosító(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 ExorTitkosító(args[0], System.in, System.out);
      }
            catch(java.io.IOException e)
            {
        e.printStackTrace();
      }
    }
```

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

```
        public class ExorTitkosító
        {
            public ExorTitkosító(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 ExorTitkosító(args[0], System.in, System.out);
      }
            catch(java.io.IOException e)
            {
        e.printStackTrace();
      }
    }
```

4.4. C EXOR törő

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

```
#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
tiszta_lehet (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_szohossz (titkos, titkos_meret);

return szohossz > 6.0 && szohossz < 9.0
    && strcasestr (titkos, "hogy") && strcasestr (titkos, "nem")
    && strcasestr (titkos, "az") && strcasestr (titkos, "ha");
```

```
}

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

int kulcs_index = 0;

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

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

    }

}

int
exor_tores (const char kulcs[], int kulcs_meret, char titkos[],
        int titkos_meret)
{

exor (kulcs, kulcs_meret, titkos, titkos_meret);

return tiszta_lehet (titkos, titkos_meret);

}

int
main (void)
{

char kulcs[KULCS_MERET];
char titkos[MAX_TITKOS];
char *p = titkos;
int olvasott_bajtok;

// titkos fajt berantasa
while ((olvasott_bajtok =
    read (0, (void *) p,
        (p - titkos + OLVASAS_BUFFER <
        MAX_TITKOS) ? OLVASAS_BUFFER : titkos + MAX_TITKOS - p)))
    p += olvasott_bajtok;

// maradek hely nullazasa a titkos bufferben
for (int i = 0; i < MAX_TITKOS - (p - titkos); ++i)
    titkos[p - titkos + i] = '\0';
```

```
        // osszes kulcs eloallitasa
        for (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, igy nem kell egy masodik buffer
                    exor (kulcs, KULCS_MERET, titkos, p - titkos);
                }

        return 0;
        }
```

Az exor törő minden esetben egy karakter sorozatból megadott kulcs alapján próbálja visszafejteni a szöveget. Bruteforce módszert használva minden lehetséges kombinációval kipróbálva töri fel. A tiszta szöveg előállítását is egy algoritmus végzi, ami egyes szavak alapján megpróbál értelmes szöveget visszafejteni.

További magyarázat, kidolgozás (később): Lorem, ipsum dolor sit amet consectetur adipisicing elit. Omnis delectus temporibus obcaecati eveniet saepe expedita dolorem ratione natus veniam asperiores, commodi pariatur iusto accusamus ut sed unde. Magni, reprehenderit voluptates. Lorem ipsum dolor sit amet consectetur adipisicing elit. Ex illo officia asperiores est itaque vel totam, quia voluptatibus dolores ducimus inventore alias id, nesciunt ab possimus harum quis error quod!

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...

4.6. Neurális OR, AND és EXOR kapu

R

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

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

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

Ez a kód igazából négy kis eltéréssel ismétlődő részből áll. Minden részben lényegében ugyanaz történik, egyedül a logikai művelet változik, amelyre feltanítjuk a neurális hálót. Ezalól kivétel az utolsó, amiről lentebb szó lesz.

Első rész:

```
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])
```

Az elején megadjuk, hogy milyen bementi adatokból milyen eredményt kell megközelítenie a threshold-dal jelölt hibahatáron belül. Ezután ezt megadjuk a neurális hálónak is, majd a neurális hálót feltanítjuk a feladatra. Itt meghívjuk a `neuralnet` függvényt, amely megkapja a bementi adatokat és az elvárt kimeneteket, 0 rejtett réteggel, 0.000001-es hibahatárral. Ezután a `plot` függvénnyel kirajzoljuk a neurális háló sematikus képét egy gráf segítségével.

Majd a `compute` függvénnyel meghívjuk a már feltanított neurális hálót az elején megadott adatokkal, hogy kiszámolja a logikai műveletek eredményét.

```
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])
```

Itt annyi különbség van, hogy míg a harmadik részben feltanított neurális háló kb. 50%-os pontossággal dolgozott, ami annyit jelent, mintha véletlenszeűen találgatott volna, itt már több neuron van a hálóban,

növelve a pontosságot. Ebben az esetben három rejtett neuronréteg van, amelyek rendre 6, 4, 6 neuronból állnak. Ezeket a "hidden=c(6,4,6)" argumentum jelöli. Ezzel már a hibahatáron belülre kerül többnyire az EXOR logikai művelet értékének kiszámítása.

4.7. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5. fejezet

Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

A matematikában a Mandelbrot-halmaz azon c komplex számokból áll (a „komplex számsík” azon pontjainak mértani helye, halmaza), melyekre az alábbi (komplex szám értékű) x_n rekurzív sorozat:

$x_1 := c$

$x_{n+1} := (x_n)^2 + c$

nem tart végtelenbe, azaz abszolút értékben (hosszára nézve) korlátos. Ez a komplex számokon egy nevezetes fraktálalakzatot formál. A továbbiakban ezt fogjuk leimplementálni C++ nyelven.

Elősször szükség van egy `Makefile`-ra, ami a .cpp file-ből előállítja a megfelelő kimenetet. Jelen esetben létrehozza az output-ot és a képet. Íme így néz ki a Makefile:

```
all: mandelbrot clean

mandelbrot.o: mandelbrot.cpp
    @g++ -c mandelbrot.cpp `libpng-config --cflags`

mandelbrot: mandelbrot.o
    @g++ -o mandelbrot mandelbrot.o `libpng-config --ldflags`

clean:
    @rm -rf *.o
    @./mandelbrot
    @rm -rf mandelbrot
```

Ahhoz, hogy előállíthassuk a képünket szükség van a png++/png.hpp header file-ra. Ha ez megvan, akkor a `GeneratePNG(int tomb[N][M])` eljárás fogja legenerálni a `kimenet.png` állományt. Ezt ugy teszi meg, hogy az előállítanó kép mérete fix 500x500-as képpontú. Egy forciklus végigmegy az eljárás paraméterében átadott mátrixon, majd pixelről pixelre színez az értékek alapján. A paraméterként átadott mátrix a Mandelbrot-halmaz a komplex számsíkon vett pontjait tartalmazza.

```
void GeneratePNG( int tomb[N][M])
{
    png::image< png::rgb_pixel > image(N, M);
    for (int x = 0; x < N; x++)
    {
        for (int y = 0; y < M; y++)
        {
            image[x][y] = png::rgb_pixel(tomb[x][y], tomb[x][y ↩
               ], tomb[x][y]);
        }
    }
    image.write("kimenet.png");
}
```

Létrehozunk egy Komplex nevű struktúrát majd a `re` és `in` mezőkben el fogjuk tárolni a komplex számunk halmazán vett valós és képzetes egységeket.

```
struct Komplex
{
    double re, im;
};
```

A `main()` függvénybe van megírva a a Mandelbrot-halmaz algoritmusa.

```
/*
* Program: Mandelbrot halmaz
* Dátum: 2014. február. 26.
* Tutor: Szabó Attila
* Tutoriált: Tuza József
*/

#include <png++/png.hpp>

#define N 500
#define M 500
#define MAXX 0.7
#define MINX -2.0
#define MAXY 1.35
#define MINY -1.35

void GeneratePNG( int tomb[N][M])
{
    png::image< png::rgb_pixel > image(N, M);
    for (int x = 0; x < N; x++)
    {
```

```
            for (int y = 0; y < M; y++)
            {
                image[x][y] = png::rgb_pixel(tomb[x][y], tomb[x][y],  ↩
                    tomb[x][y]);
            }
        }
        image.write("kimenet.png");
    }

    struct Komplex
    {
        double re, im;
    };

    int main()
    {
        int tomb[N][M];

        int i, j, k;

        double dx = (MAXX - MINX) / N;
        double dy = (MAXY - MINY) / M;

        struct Komplex C, Z, Zuj;

        int iteracio;

        for (i = 0; i < M; i++)
        {
            for (j = 0; j < N; j++)
            {
                C.re = MINX + j * dx;
                C.im = MAXY - i * dy;

                Z.re = 0;
                Z.im = 0;
                iteracio = 0;

                while(Z.re * Z.re + Z.im * Z.im < 4 && iteracio++ <  ↩
                    255)
                {
                    Zuj.re = Z.re * Z.re - Z.im * Z.im + C.re;
                    Zuj.im = 2 * Z.re * Z.im + C.im;
                    Z.re = Zuj.re;
                    Z.im = Zuj.im;
                }

                tomb[i][j] = 256 - iteracio;
            }
        }
```

```
        GeneratePNG(tomb);

        return 0;
    }
```

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

Lényegében ugyanaz, mint a felső implementáció, viszont míg ott mi irtunk egy külön struktúrát a Komplex számok kezelésére, itt a már meglévő std::complex osztállyal fogunk dolgozni illetve az ő metódusaival. Cpp kód kicsit átírva `std:complex` osztályra.

```
        /*
        * Program: Mandelbrot halmaz komplex osztállyal
        * Dátum: 2014. március. 5.
        * A feladatot Szabó Attila és Tuza József által készített ↩
           alapfeladat alapján
        * Dalmadi Zoltán módosította
        */

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

        const int N = 500;
        const int M = 500;
        const double MAXX = 0.7;
        const double MINX = -2.0;
        const double MAXY = 1.35;
        const double MINY = -1.35;

        void GeneratePNG(const int tomb[N][M])
        {
            png::image< png::rgb_pixel > image(N, M);
            for (int x = 0; x < N; x++)
            {
                for (int y = 0; y < M; y++)
                {
                    image[x][y] = png::rgb_pixel(tomb[x][y], tomb[x][y ↩
                       ], tomb[x][y]);
                }
            }
            image.write("kimenet.png");
        }

        int main()
```

```
{
    int tomb[N][M];

    double dx = (MAXX - MINX) / N;
    double dy = (MAXY - MINY) / M;

    std::complex<double> C, Z, Zuj;

    int iteracio;

    for (int i = 0; i < M; i++)
    {
        for (int j = 0; j < N; j++)
        {
            C.real(MINX + j * dx);
            C.imag(MAXY - i * dy);

            Z = 0;
            iteracio = 0;

            while(abs(Z) < 2 && iteracio++ < 255)
            {
                Zuj = Z*Z+C;
                Z = Zuj;
            }

            tomb[i][j] = 256 - iteracio;
        }
    }

    GeneratePNG(tomb);

    return 0;
}
```

5.3. Biomorfok

```
// Verzio: 3.1.3.cpp
// Forditas:
// g++ 3.1.3.cpp -lpng -O3 -o 3.1.3
// Futtatas:
// ./3.1.3 bmorf.png 800 800 10 -2 2 -2 2 .285 0 10
// Nyomtatas:
// a2ps 3.1.3.cpp -o 3.1.3.cpp.pdf -1 --line-numbers=1  --left- ↩
   footer="BATF41 HAXOR STR34M" --right-footer="https://bhaxor. ↩
```

```
   blog.hu/" --pro=color
// ps2pdf 3.1.3.cpp.pdf 3.1.3.cpp.pdf.pdf
//
// BHAX Biomorphs
// Copyright (C) 2019
// Norbert Batfai, batfai.norbert@inf.unideb.hu
//
//  This program is free software: you can redistribute it and/ ↩
   or modify
//  it under the terms of the GNU General Public License as  ↩
   published by
//  the Free Software Foundation, either version 3 of the  ↩
   License, or
//  (at your option) any later version.
//
//  This program is distributed in the hope that it will be  ↩
   useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty ↩
    of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See  ↩
   the
//  GNU General Public License for more details.
//
//  You should have received a copy of the GNU General Public  ↩
   License
//  along with this program.  If not, see <https://www.gnu.org/ ↩
   licenses/>.
//
// Version history
//
// https://youtu.be/IJMbgRzY76E
// See also https://www.emis.de/journals/TJNSA/includes/files/ ↩
   articles/Vol9_Iss5_2305--2315 ↩
   _Biomorphs_via_modified_iterations.pdf
//

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

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

    int szelesseg = 1920;
    int magassag = 1080;
    int iteraciosHatar = 255;
    double xmin = -1.9;
    double xmax = 0.7;
    double ymin = -1.3;
```

```
    double ymax = 1.3;
    double reC = .285, imC = 0;
    double R = 10.0;

    if ( argc == 12 )
    {
        szelesseg = atoi ( argv[2] );
        magassag =  atoi ( argv[3] );
        iteraciosHatar =  atoi ( argv[4] );
        xmin = atof ( argv[5] );
        xmax = atof ( argv[6] );
        ymin = atof ( argv[7] );
        ymax = atof ( argv[8] );
        reC = atof ( argv[9] );
        imC = atof ( argv[10] );
        R = atof ( argv[11] );

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

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

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

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

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

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

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

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

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

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

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

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

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

        }
```

A biomorf példákon végégmenve úgy tapasztaljuk, hogy a logikája hasonló a Mandelbrot halmaz kódjához így egy kis átalakítással létrehozható a biomorf. Mi az a biomorf? Egy elő szervezetre (pl. baktériumra, egysejtűre) hasonlító forma vagy modell. Nem feltétlen jelent elő organizmust.

Tanulságok, tapasztalatok, magyarázat...

5.4. A Mandelbrot halmaz CUDA megvalósítása

Megoldás videó:

Megoldás forrása:

5.5. Mandelbrot nagyító és utazó C++ nyelven

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

Ebben a megoldásban a GUI létrehozására és kezelésére SFML grafikus library-t használok. Ezt a csomagot telepítjük a `sudo apt-get install libsfml-dev` parancssal. Logika ugyanaz, mint a korábbi

Mandelbrot os példáknál. `void generate_mandelbrot_set(sf::VertexArray vertexarray, int pixel_shift_x, int pixel_shift_y, int precision, float zoom)` függvény fogja legenerálni a MBH_t egy Vertex array segítségével illetve mindig az aktuális paraméterekként átadott értékek alapján. Pl. Ha zoom történik, akkor ujra meghívódik a függvény az éppen aktuális MBH részeként.

```
// Forrás: https://github.com/SullyChen/Mandelbrot-Set-Plotter


#include "SFML/Graphics.hpp"

//resolution of the window
const int width = 1280;
const int height = 720;

//used for complex numbers
struct complex_number
{
    long double real;
    long double imaginary;
};

//mandelbrot komplex alapján legenerál egy mandelbrot halmazt
void generate_mandelbrot_set(sf::VertexArray& vertexarray, int pixel_shift_x, int pixel_shift_y, int precision, float zoom)
{
    #pragma omp parallel for
    for(int i = 0; i < height; i++)
    {
        for (int j = 0; j < width; j++)
        {
            //scale the pixel location to the complex plane for calculations
            long double x = ((long double)j - pixel_shift_x) / zoom;
            long double y = ((long double)i - pixel_shift_y) / zoom;
            complex_number c;
            c.real = x;
            c.imaginary = y;
            complex_number z = c;
            int iterations = 0; //keep track of the number of iterations
            for (int k = 0; k < precision; k++)
            {
                complex_number z2;
                z2.real = z.real * z.real - z.imaginary * z.imaginary;
                z2.imaginary = 2 * z.real * z.imaginary;
                z2.real += c.real;
```

```
                        z2.imaginary += c.imaginary;
                        z = z2;
                        iterations++;
                        if (z.real * z.real + z.imaginary * z.imaginary ↩
                            > 4)
                            break;
                    }
                    //color pixel based on the number of iterations
                    if (iterations < precision / 4.0f)
                    {
                        vertexarray[i*width + j].position = sf:: ↩
                           Vector2f(j, i);
                        sf::Color color(iterations * 255.0f / ( ↩
                           precision / 4.0f), 0, 0);
                        vertexarray[i*width + j].color = color;
                    }
                    else if (iterations < precision / 2.0f)
                    {
                        vertexarray[i*width + j].position = sf:: ↩
                           Vector2f(j, i);
                        sf::Color color(0, iterations * 255.0f / ( ↩
                           precision / 2.0f), 0);
                        vertexarray[i*width + j].color = color;
                    }
                    else if (iterations < precision)
                    {
                        vertexarray[i*width + j].position = sf:: ↩
                           Vector2f(j, i);
                        sf::Color color(0, 0, iterations * 255.0f /  ↩
                           precision);
                        vertexarray[i*width + j].color = color;
                    }
                }
            }
        }
```

A `main()` metódusba kezeljük le a GUI generálását. Először is `sf::` ablaknak beállítunk címet, ablakméretet stb. Ezután a default értékekkel (zoom, precision, x_shift, y_shift) legeneráltatjuk a MBH-t a `generate_mandelbrot_set(...)` függvényel. Ameddig az ablak nyitott állapotba van addig két eseményt figyel. 1. Ha rákattintunk az X gombra, akkor zárja be az ablakot. 2. Ha bal gombbal belekattintunk a MBH-ba, akkor az átadott értékek segítségével újra legeneráltatja a MBH-t 2X nagyítással.

```
int main()
{
    sf::String title_string = "Mandelbrot Set Plotter"; //ablak ↩
        címe
    sf::RenderWindow window(sf::VideoMode(width, height),  ↩
```

```
    title_string); //ablak objektum(létrehozza az ablakot a ↩
    megadott méretekkel és címmel)
window.setFramerateLimit(30); //frissített ablak/s vagy ↩
    ilyesmi
sf::VertexArray pointmap(sf::Points, width * height);


//értékek inicializálása
float zoom = 300.0f;
int precision = 100;
int x_shift = width / 2;
int y_shift = height / 2;

//legenerálja a mbh-t
generate_mandelbrot_set(pointmap, x_shift, y_shift, ↩
    precision, zoom);


/**
*
*
*
*
* */
while (window.isOpen())
{

    //ciklikusan figyeli az előforduló különböző event-eket ↩
        , ha egy olyan esemény következik be, hogy ↩
        rákattolunk az X gombra, akkor bezárja az ablakot
    sf::Event event;
    while (window.pollEvent(event))
    {
        if (event.type == sf::Event::Closed)
            window.close();
    }

    //ha a bal egérgommbal kattintunk, akkor az egér ↩
        helyére nagyít az alábbi algoritmus segítségével. ↩
        Minden nagyítás után újra legenerálja a mbh-t.
    //zoom into area that is left clicked
    if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
    {
        sf::Vector2i position = sf::Mouse::getPosition( ↩
            window);
        x_shift -= position.x - x_shift;
        y_shift -= position.y - y_shift;
        zoom *= 2;
        precision += 200;
```

```
                #pragma omp parallel for
                for (int i = 0; i < width*height; i++)
                {
                    pointmap[i].color = sf::Color::Black;
                }
                generate_mandelbrot_set(pointmap, x_shift, y_shift, ↩
                    precision, zoom);
            }
            window.clear();
            window.draw(pointmap);
            window.display();
        }

        return 0;
    }
```

5.6. Mandelbrot nagyító és utazó Java nyelven

```
import java.awt.*;
import java.awt.image.BufferedImage;
import javax.swing.*;
import java.awt.event.*;

public class Mandelbrot extends JFrame implements  ↩
   ActionListener
{

    private JPanel ctrlPanel;
    private JPanel btnPanel;
    private int numIter = 50;
    private double zoom = 130;
    private double zoomIncrease = 100;
    private int colorIter = 20;
    private BufferedImage I;
    private double zx, zy, cx, cy, temp;
    private int xMove, yMove = 0;
    private JButton[] ctrlBtns = new JButton[9];
    private Color themeColor = new Color(150,180,200);

    public Mandelbrot() {
        super("Mandelbrot Set");
        setBounds(100, 100, 800, 600);
        setResizable(false);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        plotPoints();
```

```
        Container contentPane = getContentPane();

        contentPane.setLayout(null);




        ctrlPanel = new JPanel();
        ctrlPanel.setBounds(600,0,200,600);
        ctrlPanel.setBackground(themeColor);
        ctrlPanel.setLayout(null);

        btnPanel = new JPanel();
        btnPanel.setBounds(0,200,200,200);
        btnPanel.setLayout(new GridLayout(3,3));
        btnPanel.setBackground(themeColor);

        ctrlBtns[1] = new JButton("up");
        ctrlBtns[7] = new JButton("down");
        ctrlBtns[3] = new JButton ("left");
        ctrlBtns[5] = new JButton("right");
        ctrlBtns[2] = new JButton("+");
        ctrlBtns[0] = new JButton("-");
        ctrlBtns[8] = new JButton(">");
        ctrlBtns[6] = new JButton("<");
        ctrlBtns[4] = new JButton();

        contentPane.add(ctrlPanel);
        contentPane.add(new imgPanel());
        ctrlPanel.add(btnPanel);

        for (int x = 0; x<ctrlBtns.length;x++){
            btnPanel.add(ctrlBtns[x]);
            ctrlBtns[x].addActionListener(this);
        }

        validate();

    }

    public class imgPanel extends JPanel{
        public imgPanel(){
            setBounds(0,0,600,600);

        }

        @Override
        public void paint (Graphics g){
            super.paint(g);
```

```
                g.drawImage(I, 0, 0, this);
            }
        }

        public void plotPoints(){
            I = new BufferedImage(getWidth(), getHeight(),  ↩
               BufferedImage.TYPE_INT_RGB);
            for (int y = 0; y < getHeight(); y++) {
                for (int x = 0; x < getWidth(); x++) {
                    zx = zy = 0;
                    cx = (x - 320+xMove) / zoom;
                    cy = (y - 290+yMove) / zoom;
                    int iter = numIter;
                    while (zx * zx + zy * zy < 4 && iter > 0) {
                        temp = zx * zx - zy * zy + cx;
                        zy = 2 * zx * zy + cy;
                        zx = temp;
                        iter--;
                    }
                    I.setRGB(x, y, iter | (iter << colorIter));
                }
            }
        }

        public void actionPerformed(ActionEvent ae){
            String event = ae.getActionCommand();

            switch (event){
            case "up":
                yMove-=100;
                break;
            case "down":
                yMove+=100;
                break;
            case "left":
                xMove-=100;
                break;
            case "right":
                xMove+=100;
                break;
            case "+":
                zoom+=zoomIncrease;
                zoomIncrease+=100;
                break;
            case "-":
                zoom-=zoomIncrease;
                zoomIncrease-=100;
                break;
            case ">":
                colorIter++;
```

```
                    break;
                case "<":
                    colorIter--;
                    break;
                }



                plotPoints();
                validate();
                repaint();
            }
            public static void main(String[] args)
            {
                new Mandelbrot().setVisible(true);
            }
        }
```

A program az Eclipse gui library-t (swing, awt) használok, ami nagyban segíti az ablakozó rendszer létrejöttét. A Mandelbrot osztályt származtatjuk a JFrame osztályból így elérjük a JFram tulajdonságait. A konstrutorba létrehozzuk az ablakot a megfelelő méretekkel illetve felpakoluk a gombokat és a Mandelbrot halmazt megjelenítő ablakocskát. A `plotPoints()` függvény rajzoltatja ki a Mandelbrot halmazt a mb algoritmus alapján. Továbbiakban minden egyes gombok által elérhető eseményekre feliratkozunk és megmondjuk hogy azon eseményre mi történjen.Ilyen eseményekre többnyire a MB halmaz értékeit változtatjuk specifikusan. Például ha nagyítunk vagy kicsinyítünk stb. Minden eseménykor újrageneráltatjuk a MB-halmazt.

6. fejezet

Helló, Welch!

6.1. Első osztályom

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

Megoldás videó:

Megoldás forrása:

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

6.2. LZW

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

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
typedef struct node{
    char c;
    struct node* left;
    struct node* right;
} Node;

Node* fa;
Node gyoker;

#define null NULL

Node* create_empty()
```

```
{
    Node* tmp = &gyoker;
    tmp->c= '/';
    tmp->left = null;
    tmp->right = null;
    return tmp;
}

Node* create_node(char val)
{
    Node* tmp = (Node*)malloc(sizeof(Node));
    tmp->c=val;
    tmp->left = null;
    tmp->right = null;
    return tmp;
}

void insert_tree(char val)
{
    if(val=='0')
    {
        if(fa->left == null)
        {
            fa->left = create_node(val);
            fa = &gyoker;
            //printf("Inserted into left.");
        }
        else
        {
            fa = fa->left;
        }
    }
    else
    {
        if(fa->right == null)
        {
            fa->right = create_node(val);
            fa = &gyoker;
            //printf("Inserted into left.");
        }
        else
        {
            fa = fa->right;
        }
    }
}

void inorder(Node* elem,int depth)
{
```

```
        if(elem==null)
        {
            return;
        }
        inorder(elem->left,depth+1);
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth]='\0';

            printf("%s%c\n",spaces,elem->c);
        }
        else
        {
            printf("%c\n",elem->c);
        }
        inorder(elem->right,depth+1);
    }

    void preorder(Node* elem,int depth)
    {
        if(elem==null)
        {
            return;
        }
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth*2]='\0';

            printf("%s%c\n",spaces,elem->c);
        }
        else
        {
            printf("%c\n",elem->c);
        }
        preorder(elem->left,depth+1);
        preorder(elem->right,depth+1);
```

```
}
void postorder(Node* elem,int depth)
{
    if(elem==null)
    {
        return;
    }
    postorder(elem->left,depth+1);
    postorder(elem->right,depth+1);
    if(depth)
    {
        char *spaces;
        spaces =(char*) malloc(sizeof(char)*depth*2+1);
        for(int i=0;i<depth;i+=2)
        {
            spaces[i]='-';
            spaces[i+1]='-';
        }
        spaces[depth*2]='\0';

        printf("%s%c\n",spaces,elem->c);
        free(spaces);
    }
    else
    {
        printf("%c\n",elem->c);
    }
}

void destroy_tree(Node* elem)
{
    if(elem==null)
    {
        return;
    }
    destroy_tree(elem->left);
    destroy_tree(elem->right);
    if(elem->c == gyoker.c)
    {

    }
    else
    {
        free(elem);
    }
}

void usage()
{
    printf("Használat: ./binfa KAPCSOLÓ\n");
```

```
        printf("Az KAPCSOLÓ lehet:\n");
        printf("--preorder\tA bináris fa preorder bejárása\n");
        printf("--inorder\tA bináris fa inorder bejárása\n");
        printf("--postorder\tA bináris fa postorder bejárása\n");
    }

    int main(int argc, char** argv)
    {
        srand(time(null));
        fa = create_empty();
        //gyoker = *fa;
        for(int i=0;i<10000;i++)
        {
            int x=rand()%2;
            if(x)
            {
                insert_tree('1');
            }
            else
            {
                insert_tree('0');
            }
        }
        if(argc == 2)
        {
            if(strcmp(argv[1],"--preorder")==0)
            {
                preorder(&gyoker,0);
            }
            else if(strcmp(argv[1],"--inorder")==0)
            {
                inorder(&gyoker,0);
            }
            else if(strcmp(argv[1],"--postorder")==0)
            {
                postorder(&gyoker,0);
            }
            else
            {
                usage();
            }
        }
        else
        {
            usage();
        }
        destroy_tree(&gyoker);
        return 0;
    }
```

A fenti programban az LZW algoritmussal kódolt binfa változata van megírva. Ez a feladat több részből áll, amit a továbbiakban részletekre bontva taglalunk.

```
typedef struct node
{
    char c;
    struct node* left;
    struct node* right;
} Node;

Node* fa;
Node gyoker;

#define null NULL

Node* create_empty()
{
    Node* tmp = &gyoker;
    tmp->c= '/';
    tmp->left = null;
    tmp->right = null;
    return tmp;
}

Node* create_node(char val)
{
    Node* tmp = (Node*)malloc(sizeof(Node));
    tmp->c=val;
    tmp->left = null;
    tmp->right = null;
    return tmp;
}
```

Létrehozunk egy adatstruktúrát, aminek a neve és típusa `Node` lesz. Ebbe három property lesz. `char c` property-be tároljuk el az input karaktert. A `left` property egy saját node-ra mutató pointer. Ugyanez a `left` property-re. Ezután definiálunk egy Node* fa pointer objektumot és egy Node típusu objektumot.

A `create_empty()` függvény lényege, hogy létrehoz egy új Node* típusú pointer objektumot, aminek beállítjuk a bal, jobb gyermekét nullára majd a függvény visszatér ezzel a pointer objektummal.

A `create_node(char val)` függvény lényege, hogy paraméterként kapott char érték alapján először helyet foglal a memóriában, majd a `val` értékét eltárolja az legfoglalt memóriacímen. Beállítja a jobb és bal fiát nullára, majd visszatér egy Node* pointerrel.

```
void insert_tree(char val)
{
    if(val=='0')
    {
        if(fa->left == null)
        {
            fa->left = create_node(val);
            fa = &gyoker;
```

```
                    //printf("Inserted into left.");
                }
                else
                {
                    fa = fa->left;
                }
            }
            else
            {
                if(fa->right == null)
                {
                    fa->right = create_node(val);
                    fa = &gyoker;
                    //printf("Inserted into right.");
                }
                else
                {
                    fa = fa->right;
                }
            }
        }
```

A `insert_tree(char val)` eljárás a paraméterében megkapott érték alapján felépít egy ÚJ csomópontot a csomópont éppen aktuális jobb vagy bal fiával. Ha pl. a val értéke 1 akkor megnézi, hogy az fa ponter objektumban a aktuális csomópontnak van-e 1-es gyermeke. Ha null az érték a jobb gyermeknél akkor beállítja az 1-es értéket a jobb gyermekhez. Ez után a fát ráállítjuk a binfa gyökerére. Ha viszont már van 1-es gyermeke az aktuális node-nak akkor továbbhalad a jobb gyermekre és beállítja erre a fa mutatóját. Ugynezen logika mentén játszódik le a 0-ás érték esetén.

```
        void inorder(Node* elem,int depth)
        {

            if(elem==null)
            {
                return;
            }
            inorder(elem->left,depth+1);
            if(depth)
            {
                char *spaces;
                spaces =(char*) malloc(sizeof(char)*depth*2+1);
                for(int i=0;i<depth;i+=2)
                {
                    spaces[i]='-';
                    spaces[i+1]='-';
                }
                spaces[depth]='\0';

                printf("%s%c\n",spaces,elem->c);
            }
```

```
        else
        {
            printf("%c\n",elem->c);
        }
        inorder(elem->right,depth+1);
    }

    void preorder(Node* elem,int depth)
    {
        if(elem==null)
        {
            return;
        }
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth*2]='\0';

            printf("%s%c\n",spaces,elem->c);
        }
        else
        {
            printf("%c\n",elem->c);
        }
        preorder(elem->left,depth+1);
        preorder(elem->right,depth+1);
    }
    void postorder(Node* elem,int depth)
    {
        if(elem==null)
        {
            return;
        }
        postorder(elem->left,depth+1);
        postorder(elem->right,depth+1);
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
```

```
                spaces[depth*2]='\0';

                printf("%s%c\n",spaces,elem->c);
                free(spaces);
            }
            else
            {
                printf("%c\n",elem->c);
            }
        }
```

Ezekbe a metódusokba van megírta az három fabejárás(inorder, postorder, preorder). Rekurzív rendezések. Rendezéstől függ, hogy melyik algoritmus alapján járjuk be a fát. Pl. Inorder: Mindig a bal oldalt vizsgáljuk majd ha megvan a legutolsó bal elem akkor visszatér az ő szülejéhez majd pedig a jobb gyerekéhez. Ezt az agoritmust minden node-on lejátsza rekurzívan.

```
        void destroy_tree(Node* elem)
        {
            if(elem==null)
            {
                return;
            }
            destroy_tree(elem->left);
            destroy_tree(elem->right);
            if(elem->c == gyoker.c)
            {

            }
            else
            {
                free(elem);
            }
        }
```

A `destroy_tree(Node* elem)` függvény a felesleges memóriacímeket szabadítja fel. Ha elem null értékű akkor üres a visszatérési értéke. Ha viszont a paraméterként átadott csomópont nem null akkor önmagát meghívja a jobb illetva majd a bal fiára. Utánna a free(elem) metódussal felszabadítja a jobb vagy éppen a bal fia memóriacímét.

```
        int main(int argc, char** argv)
        {
            srand(time(null));
            fa = create_empty();
            //gyoker = *fa;
            for(int i=0;i<10000;i++)
            {
                int x=rand()%2;
                if(x)
                {
                    insert_tree('1');
                }
```

```
            else
            {
                insert_tree('0');
            }
        }
        if(argc == 2)
        {
            if(strcmp(argv[1],"--preorder")==0)
            {
                preorder(&gyoker,0);
            }
            else if(strcmp(argv[1],"--inorder")==0)
            {
                inorder(&gyoker,0);
            }
            else if(strcmp(argv[1],"--postorder")==0)
            {
                postorder(&gyoker,0);
            }
            else
            {
                usage();
            }
        }
        else
        {
            usage();
        }
        destroy_tree(&gyoker);
        return 0;
    }
```

Az összes eddig szétbontogatott részek felhasználása a `main()` metóduson belül történik.

6.3. Fabejárás

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

Az előző feladatba részletezve van mindhárom fabejárás. Azt felhasználva illetve kiegészítve rakom be ehhez a feladathoz.

```
    void inorder(Node* elem,int depth)
    {

        if(elem==null)
        {
            return;
        }
        inorder(elem->left,depth+1);
```

```
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth]='\0';

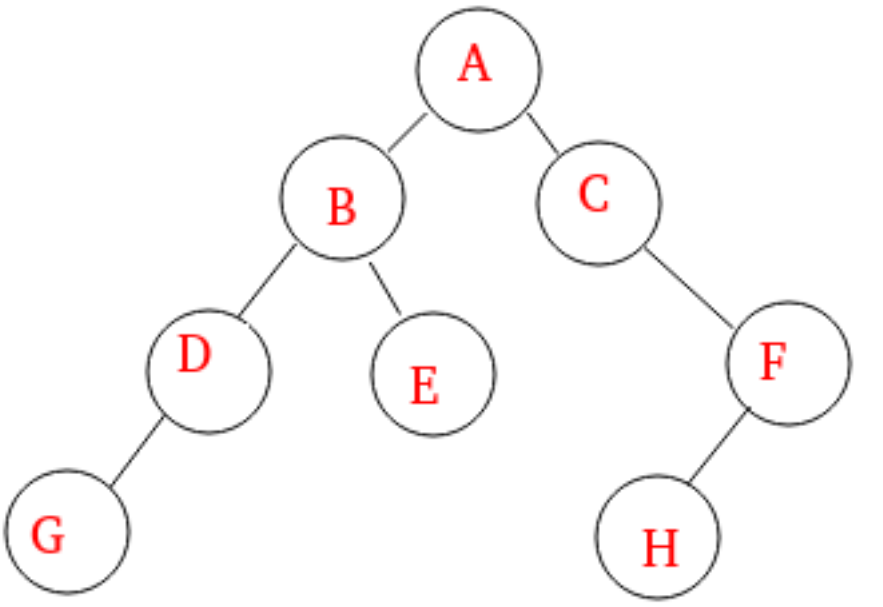
            printf("%s%c\n",spaces,elem->c);
        }
        else
        {
            printf("%c\n",elem->c);
        }
        inorder(elem->right,depth+1);
    }

    void preorder(Node* elem,int depth)
    {
        if(elem==null)
        {
            return;
        }
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth*2]='\0';

            printf("%s%c\n",spaces,elem->c);
        }
        else
        {
            printf("%c\n",elem->c);
        }
        preorder(elem->left,depth+1);
        preorder(elem->right,depth+1);
    }
    void postorder(Node* elem,int depth)
    {
        if(elem==null)
        {
```

```
            return;
        }
        postorder(elem->left,depth+1);
        postorder(elem->right,depth+1);
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth*2]='\0';

            printf("%s%c\n",spaces,elem->c);
            free(spaces);
        }
        else
        {
            printf("%c\n",elem->c);
        }
    }
```



INORDER (LPR): G,D,B,E,A,C,H,F
PREORDER (PLR): A,B,D,G,E,C,F,H
POSTORDER (LRP): G,D,E,B,H,F,C,A

6.1. ábra. Bejaras

6.4. Tag a gyökér

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

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string.h>

#define null NULL

class Binfa
{
```

```
    private:
        class Node
        {
        public:
            Node(char c='/')
            {
                this->c=c;
                this->left = null;
                this->right = null;
            }
            char c;
            Node* left;
            Node* right;
        };
        Node* fa;


    public:
        Binfa(): fa(&gyoker)
        {

        }

        void operator<<(char c)
        {
            if(c=='0')
            {
                if(fa->left == null)
                {
                    fa->left = new Node('0');
                    fa = &gyoker;
                }
                else
                {
                    fa = fa->left;
                }
            }
            else
            {
                if(fa->right == null)
                {
                    fa->right = new Node('1');
                    fa = &gyoker;
                }
                else
                {
                    fa = fa->right;
                }
            }
        }
```

```
void preorder(Node* elem,int depth=0)
{
    if(elem==null)
    {
        return;
    }
    if(depth)
    {
        char *spaces;
        spaces =(char*) malloc(sizeof(char)*depth*2+1);
        for(int i=0;i<depth;i+=2)
        {
            spaces[i]='-';
            spaces[i+1]='-';
        }
        spaces[depth*2]='\0';

        printf("%s%c\n",spaces,elem->c);
    }
    else
    {
        printf("%c\n",elem->c);
    }
    preorder(elem->left,depth+1);
    preorder(elem->right,depth+1);
}

void inorder(Node* elem,int depth=0)
{
    if(elem==null)
    {
        return;
    }
    inorder(elem->left,depth+1);
    if(depth)
    {
        char *spaces;
        spaces =(char*) malloc(sizeof(char)*depth*2+1);
        for(int i=0;i<depth;i+=2)
        {
            spaces[i]='-';
            spaces[i+1]='-';
        }
        spaces[depth*2]='\0';

        printf("%s%c\n",spaces,elem->c);
    }
    else
    {
```

```
                printf("%c\n",elem->c);
            }
            inorder(elem->right,depth+1);
        }

        void postorder(Node* elem,int depth=0)
        {
            if(elem==null)
            {
                return;
            }
            postorder(elem->left,depth+1);
            postorder(elem->right,depth+1);
            if(depth)
            {
                char *spaces;
                spaces =(char*) malloc(sizeof(char)*depth*2+1);
                for(int i=0;i<depth;i+=2)
                {
                    spaces[i]='-';
                    spaces[i+1]='-';
                }
                spaces[depth*2]='\0';

                printf("%s%c\n",spaces,elem->c);
            }
            else
            {
                printf("%c\n",elem->c);
            }
        }

        void destroy_tree(Node* elem)
        {
            if(elem==null)
            {
                return;
            }
            destroy_tree(elem->left);
            destroy_tree(elem->right);
            if(elem->c!='/') delete elem;
        }

        Node gyoker;

    };

    void usage()
    {
        printf("Használat: ./binfa KAPCSOLÓ\n");
```

```
        printf("Az KAPCSOLÓ lehet:\n");
        printf("--preorder\tA bináris fa preorder bejárása\n");
        printf("--inorder\tA bináris fa inorder bejárása\n");
        printf("--postorder\tA bináris fa postorder bejárása\n");
    }

    int main(int argc, char** argv)
    {
        srand(time(0));
        Binfa bfa;
        for(int i=0;i<100;i++)
        {
            int x=rand()%2;
            if(x)
            {
                bfa<<'1';
            }
            else
            {
                bfa<<'0';
            }
        }
        if(argc == 2)
        {
            if(strcmp(argv[1],"--preorder")==0)
            {
                bfa.preorder(&bfa.gyoker);
            }
            else if(strcmp(argv[1],"--inorder")==0)
            {
                bfa.inorder(&bfa.gyoker);
            }
            else if(strcmp(argv[1],"--postorder")==0)
            {
                bfa.postorder(&bfa.gyoker);
            }
            else
            {
                usage();
            }
        }
        else
        {
            usage();
        }
        bfa.destroy_tree(&bfa.gyoker);
        return 0;
    }
```

Ebben a változatban tagként van definiálva a csomópont gyökér. Mivel tag így, hogy elérjük szükségünk

van a referenciájára. Mindenhol, ahol szükség van a gyökér tagra ott alkalmazni kell a referencia operátort.

6.5. Mutató a gyökér

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

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string.h>

#define null NULL

class Binfa
{
private:
    class Node
    {
    public:
        Node(char c='/')
        {
            this->c=c;
            this->left = null;
            this->right = null;
        }
        char c;
        Node* left;
        Node* right;
    };
    Node* fa;


public:
    Binfa()
    {
        gyoker=fa=new Node();
    }


    void operator<<(char c)
    {
        if(c=='0')
        {
            if(fa->left == null)
            {
                fa->left = new Node('0');
                fa = gyoker;
            }
            else
```

```
                {
                    fa = fa->left;
                }
            }
            else
            {
                if(fa->right == null)
                {
                    fa->right = new Node('1');
                    fa = gyoker;
                }
                else
                {
                    fa = fa->right;
                }
            }
        }

        void preorder(Node* elem,int depth=0)
        {
            if(elem==null)
            {
                return;
            }
            if(depth)
            {
                char *spaces;
                spaces =(char*) malloc(sizeof(char)*depth*2+1);
                for(int i=0;i<depth;i+=2)
                {
                    spaces[i]='-';
                    spaces[i+1]='-';
                }
                spaces[depth*2]='\0';

                printf("%s%c\n",spaces,elem->c);
            }
            else
            {
                printf("%c\n",elem->c);
            }
            preorder(elem->left,depth+1);
            preorder(elem->right,depth+1);
        }

        void inorder(Node* elem,int depth=0)
        {
            if(elem==null)
            {
                return;
```

```
        }
        inorder(elem->left,depth+1);
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth*2]='\0';

            printf("%s%c\n",spaces,elem->c);
        }
        else
        {
            printf("%c\n",elem->c);
        }
        inorder(elem->right,depth+1);
    }

    void postorder(Node* elem,int depth=0)
    {
        if(elem==null)
        {
            return;
        }
        postorder(elem->left,depth+1);
        postorder(elem->right,depth+1);
        if(depth)
        {
            char *spaces;
            spaces =(char*) malloc(sizeof(char)*depth*2+1);
            for(int i=0;i<depth;i+=2)
            {
                spaces[i]='-';
                spaces[i+1]='-';
            }
            spaces[depth*2]='\0';

            printf("%s%c\n",spaces,elem->c);
        }
        else
        {
            printf("%c\n",elem->c);
        }
    }

    void destroy_tree(Node* elem)
```

```
        {
            if(elem==null)
            {
                return;
            }
            destroy_tree(elem->left);
            destroy_tree(elem->right);
            if(elem->c!='/') delete elem;
        }

        Node* gyoker;

};

void usage()
{
    printf("Használat: ./binfa KAPCSOLÓ\n");
    printf("Az KAPCSOLÓ lehet:\n");
    printf("--preorder\tA bináris fa preorder bejárása\n");
    printf("--inorder\tA bináris fa inorder bejárása\n");
    printf("--postorder\tA bináris fa postorder bejárása\n" ↩
       );
}

int main(int argc, char** argv)
{
    srand(time(0));
    Binfa bfa;
    for(int i=0;i<100;i++)
    {
        int x=rand()%2;
        if(x)
        {
            bfa<<'1';
        }
        else
        {
            bfa<<'0';
        }
    }
    if(argc == 2)
    {
        if(strcmp(argv[1],"--preorder")==0)
        {
            bfa.preorder(bfa.gyoker);
        }
        else if(strcmp(argv[1],"--inorder")==0)
        {
            bfa.inorder(bfa.gyoker);
        }
```

```
                    else if(strcmp(argv[1],"--postorder")==0)
                    {
                        bfa.postorder(bfa.gyoker);
                    }
                    else
                    {
                        usage();
                    }
                }
                else
                {
                    usage();
                }
                bfa.destroy_tree(bfa.gyoker);
                return 0;
            }
```

Lényegében a Node gyoker tagok változtattuk át Node* gyoker-re. A konstruktort átalakítjuk úgy hogy nem referenciát adunk át, hanem a gyökérnek egy új helyet osztunk fel a memóriában a new operátor segítségével. Továbbá a gyökér minden előfordulásánál ki kell venni a referencia szerinti hivatkozásokat.

6.6. Mozgató szemantika

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

Megoldás videó:

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <vector>


class LZWBinFa {
public:


    LZWBinFa () :fa ( &gyoker ) {

    }
    ~LZWBinFa () {
        std::cout << "LZWBinFa dtor" << std::endl;
        szabadit ( gyoker.egyesGyermek () );
        szabadit ( gyoker.nullasGyermek () );
    }

    LZWBinFa ( const LZWBinFa & regi ) {
```

```
        std::cout << "LZWBinFa copy ctor" << std::endl;

        gyoker.ujEgyesGyermek ( masol ( regi.gyoker.egyesGyermek () ↩
           , regi.fa ) );
        gyoker.ujNullasGyermek ( masol ( regi.gyoker.nullasGyermek  ↩
           (), regi.fa ) );

        if ( regi.fa == & ( regi.gyoker ) )
            fa = &gyoker;

    }

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

        gyoker.ujEgyesGyermek ( regi.gyoker.egyesGyermek() );
        gyoker.ujNullasGyermek ( regi.gyoker.nullasGyermek() );

        regi.gyoker.ujEgyesGyermek ( nullptr );
        regi.gyoker.ujNullasGyermek ( nullptr );

    }


    LZWBinFa& operator<< ( char b ) {

        if ( b == '0' ) {

            if ( !fa->nullasGyermek () ) {
                    Csomopont *uj = new Csomopont ( '0' );

                    fa->ujNullasGyermek ( uj );

                    fa = &gyoker;
            } else {

                    fa = fa->nullasGyermek ();
            }
        }

        else {
            if ( !fa->egyesGyermek () ) {
                    Csomopont *uj = new Csomopont ( '1' );
                    fa->ujEgyesGyermek ( uj );
                    fa = &gyoker;
            } else {
                    fa = fa->egyesGyermek ();
            }
        }
```

```
            return *this;
        }

        void kiir ( void ) {
            /
            melyseg = 0;

            kiir ( &gyoker, std::cout );
        }

        {
        szabadit (gyoker.egyesGyermek ());
        szabadit (gyoker.nullasGyermek ());

        }

        int getMelyseg ( void );
        double getAtlag ( void );
        double getSzoras ( void );


        friend std::ostream & operator<< ( std::ostream & os, LZWBinFa  ↩
           & bf ) {
            bf.kiir ( os );
            return os;
        }
        void kiir ( std::ostream & os ) {
            melyseg = 0;
            kiir ( &gyoker, os );
        }

    private:
        class Csomopont {
        public:

            Csomopont ( char b = '/' ) :betu ( b ), balNulla ( 0 ),  ↩
               jobbEgy ( 0 ) {
            };
            ~Csomopont () {
            };

            Csomopont *nullasGyermek () const {
                return balNulla;
            }

            Csomopont *egyesGyermek () const {
                return jobbEgy;
            }

            void ujNullasGyermek ( Csomopont * gy ) {
```

```
            balNulla = gy;
        }

        void ujEgyesGyermek ( Csomopont * gy ) {
            jobbEgy = gy;
        }

        char getBetu () const {
            return betu;
        }

    private:


        char betu;

        Csomopont *balNulla;
        Csomopont *jobbEgy;

        Csomopont ( const Csomopont & );
        Csomopont & operator= ( const Csomopont & );

    };


    Csomopont *fa;

    int melyseg, atlagosszeg, atlagdb;
    double szorasosszeg;


    void kiir ( Csomopont * elem, std::ostream & os ) {

        if ( elem != NULL ) {
            ++melyseg;
            kiir ( elem->egyesGyermek (), os );

            for ( int i = 0; i < melyseg; ++i )
                    os << "---";
            os << elem->getBetu () << "(" << melyseg - 1 << ")" << ↩
               std::endl;
            kiir ( elem->nullasGyermek (), os );
            --melyseg;
        }
    }

    void szabadit ( Csomopont * elem ) {

        if ( elem != NULL ) {
            szabadit ( elem->egyesGyermek () );
```

```
                szabadit ( elem->nullasGyermek () );

                delete elem;
            }
        }


        Csomopont *  masol ( Csomopont * elem, Csomopont * regifa ) {

            Csomopont * ujelem = NULL;

            if ( elem != NULL ) {
                ujelem = new Csomopont ( elem->getBetu() );

                ujelem->ujEgyesGyermek ( masol ( elem->egyesGyermek (), ↩
                    regifa ) );
                ujelem->ujNullasGyermek ( masol ( elem->nullasGyermek  ↩
                   (), regifa ) );

                if ( regifa == elem )
                        fa = ujelem;

            }

            return ujelem;
        }

    protected:      /
        Csomopont gyoker;
        int maxMelyseg;
        double atlag, szoras;

        void rmelyseg ( Csomopont * elem );
        void ratlag ( Csomopont * elem );
        void rszoras ( Csomopont * elem );

    };


    int
    LZWBinFa::getMelyseg ( void )
    {
        melyseg = maxMelyseg = 0;
        rmelyseg ( &gyoker );
        return maxMelyseg - 1;
    }

    double
    LZWBinFa::getAtlag ( void )
    {
```

```
        melyseg = atlagosszeg = atlagdb = 0;
        ratlag ( &gyoker );
        atlag = ( ( double ) atlagosszeg ) / atlagdb;
        return atlag;
    }

    double
    LZWBinFa::getSzoras ( void )
    {
        atlag = getAtlag ();
        szorasosszeg = 0.0;
        melyseg = atlagdb = 0;

        rszoras ( &gyoker );

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

        return szoras;
    }

    void
    LZWBinFa::rmelyseg ( Csomopont * elem )
    {
        if ( elem != NULL ) {
            ++melyseg;
            if ( melyseg > maxMelyseg )
                maxMelyseg = melyseg;
            rmelyseg ( elem->egyesGyermek () );

            rmelyseg ( elem->nullasGyermek () );
            --melyseg;
        }
    }

    void
    LZWBinFa::ratlag ( Csomopont * elem )
    {
        if ( elem != NULL ) {
            ++melyseg;
            ratlag ( elem->egyesGyermek () );
            ratlag ( elem->nullasGyermek () );
            --melyseg;
            if ( elem->egyesGyermek () == NULL && elem->nullasGyermek ↩
                () == NULL ) {
                ++atlagdb;
                atlagosszeg += melyseg;
            }
```

```
        }
    }

    void
    LZWBinFa::rszoras ( Csomopont * elem )
    {
        if ( elem != NULL ) {
            ++melyseg;
            rszoras ( elem->egyesGyermek () );
            rszoras ( elem->nullasGyermek () );
            --melyseg;
            if ( elem->egyesGyermek () == NULL && elem->nullasGyermek  ↩
                () == NULL ) {
                ++atlagdb;
                szorasosszeg += ( ( melyseg - atlag ) * ( melyseg -  ↩
                    atlag ) );
            }
        }
    }



    void
    usage ( void )
    {
        std::cout << "Usage: lzwtree in_file -o out_file" << std::endl;
    }


    void
    fgv ( LZWBinFa binFa )
    {
        binFa << '1';

        std::cout << binFa;

        std::cout << "depth = " << binFa.getMelyseg () << std::endl;
        std::cout << "mean = " << binFa.getAtlag () << std::endl;
        std::cout << "var = " << binFa.getSzoras () << std::endl;
    }

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



        if ( argc != 4 ) {

            usage ();
```

```
        return -1;
    }


    char *inFile = *++argv;


    if ( * ( ( *++argv ) + 1 ) != 'o' ) {
        usage ();
        return -2;
    }


    std::fstream beFile ( inFile, std::ios_base::in );


    if ( !beFile ) {
        std::cout << inFile << " nem letezik..." << std::endl;
        usage ();
        return -3;
    }

    std::fstream kiFile ( *++argv, std::ios_base::out );

    unsigned char b;
    LZWBinFa binFa;


    binFa << '0' << '1' << '0' << '1' << '1' << '1' << '1' << '1'  ↩
       << '1' << '1';

    fgv ( binFa );

    binFa << '0';

    kiFile << binFa;

    kiFile << "depth = " << binFa.getMelyseg () << std::endl;
    kiFile << "mean = " << binFa.getAtlag () << std::endl;
    kiFile << "var = " << binFa.getSzoras () << std::endl;

    LZWBinFa binFa3 = std::move ( binFa );

    kiFile << "depth = " << binFa3.getMelyseg () << std::endl;
    kiFile << "mean = " << binFa3.getAtlag () << std::endl;
    kiFile << "var = " << binFa3.getSzoras () << std::endl;

    kiFile.close ();
    beFile.close ();
```

```
        return 0;
    }
```

A másoló konstruktor és az értékadó operátor ugyanazt a feladatot látja el (objektum adattagjainak átmásolása), azonos esetekben működnek jól, vagy okoznak problémát. Ezért, ha az egyiket letiltjuk, vagy definiáljuk, akkor a másikat is célszerű. Mivel lényegében ugyanazt csinálják, ezért érdemes egy külön private metódusban megírni a lényeget, és annak segítségével definiálni a másoló konstruktort és az értékadó operátort.

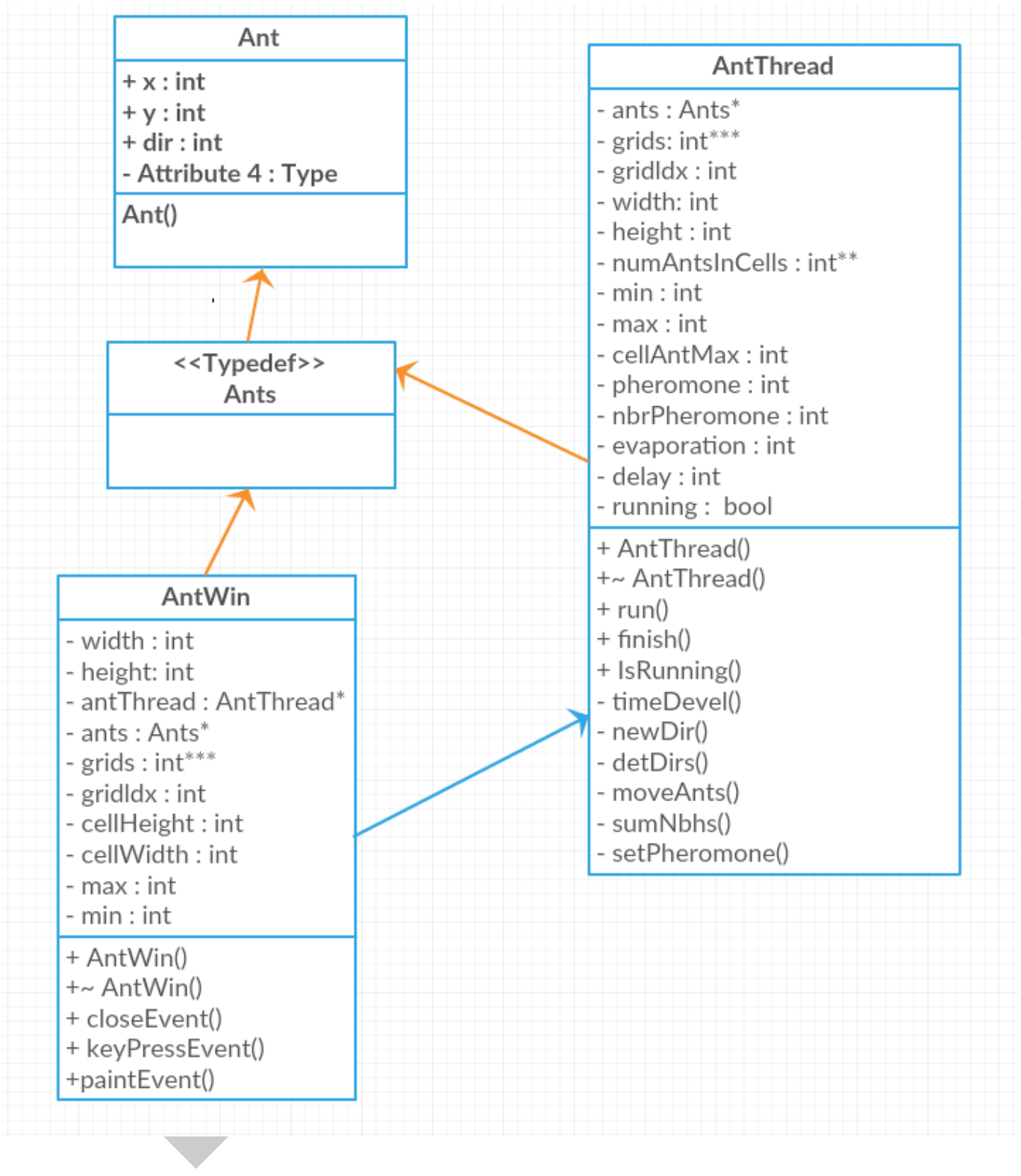
7. fejezet

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



7.1. ábra. Hangyaszimuláció UML

Az UML ábrában minden blokk egy egy osztályt jelent. Blokkon belül 3 tagrészt különítünk el. Fentről lefele haladva: osztálynév, tulajdonságok, viselkedés. A + vagy minusz jelek a láthatóságot jelentik. + ha más osztályok láthatják illetve - ha nem (private).

`Ant`

Az `Ant` osztály fogja létrehozni a hangya objektumokat. Vannak különböző mezői, ami a hangyára jellemző illetve egy Ant() függvénye.

`AntThread`

Ez az osztály írja le a hangya egyed tulajdonságait. A tulajdonságok meghatározzák pl. az egyes egyedek elhelyezkedését, viselkedését, mozgását stb. Valamit ezekhez a tulajdonságokhoz párosulnak függvények. Ilyen pl. a run(), newDir(), detDirs() függvények, amik a hangya mozgását írják le.

`AntWin`

Ebben az osztályban történik meg a rácsvonalak valamint a hangyák és feromon útvonalak kirajzoltatása. Tartalmaz továbbá egy AntThread pointert is, ami az egyes hangya egyedet tulajdonságait és viselkedéseit írják le.

7.2. Java életjáték

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

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Rectangle;
import java.awt.Shape;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.image.ImageObserver;
import java.text.AttributedCharacterIterator;
import java.util.ArrayList;
import java.awt.Event;
public class game_of_life extends JFrame {
    RenderArea ra;
    private int i;

    public game_of_life() {
        super("Game of Life");
        this.setSize(1005, 1030);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.setVisible(true);
        this.setResizable(false);
        ra = new RenderArea();
        ra.setFocusable(true);
        ra.grabFocus();
```

```
            add(ra);
            ra.edit_mode = true;
            ra.running = true;
        }
        public void update() {
            ArrayList<ArrayList<Boolean>> entities = new ArrayList< ↩
               ArrayList<Boolean>>();// = ra.entities;
            int size1 = ra.entities.size();
            int size2 = ra.entities.get(0).size();
            for(int i=0;i<size1;i++)
            {
                entities.add( new ArrayList<Boolean>());
                for(int j=0;j<size2;j++)
                {
                    int alive = 0;

                    if(ra.entities.get((size1+i-1)%size1).get((size2+j ↩
                       -1)%size2)) alive++;
                    if(ra.entities.get((size1+i-1)%size1).get((size2+j) ↩
                       %size2)) alive++;
                    if(ra.entities.get((size1+i-1)%size1).get((size2+j ↩
                       +1)%size2)) alive++;
                    if(ra.entities.get((size1+i)%size1).get((size2+j-1) ↩
                       %size2)) alive++;
                    if(ra.entities.get((size1+i)%size1).get((size2+j+1) ↩
                       %size2)) alive++;
                    if(ra.entities.get((size1+i+1)%size1).get((size2+j ↩
                       -1)%size2)) alive++;
                    if(ra.entities.get((size1+i+1)%size1).get((size2+j) ↩
                       %size2)) alive++;
                    if(ra.entities.get((size1+i+1)%size1).get((size2+j ↩
                       +1)%size2)) alive++;

                    /*for(int k=-1;i<2;k++)
                    {
                        for(int l = -1; l < 2 ;l++)
                        {
                            if(!(k==0 && l == 0))
                            {
                                if(ra.entities.get((size1+i+k)%size1). ↩
                                   get((size2+j+l)%size2)) alive++;
                            }
                        }
                    }*/
                    if(ra.entities.get(i).get(j))
                    {
                        if(alive < 2 || alive > 3)
                        {
                            //ra.entities.get(i).set(j,false);
                            entities.get(i).add(false);
```

```
                        }
                        else
                        {
                            entities.get(i).add(true);
                        }
                    }
                    else
                    {
                        if(alive == 3)
                        {
                            //ra.entities.get(i).set(j,true);
                            entities.get(i).add(true);
                        }
                        else
                        {
                            entities.get(i).add(false);
                        }
                    }
                }
            }
            ra.entities = entities;
        }
        class RenderArea extends JPanel implements KeyListener {
            public ArrayList<ArrayList<Boolean>> entities;
            public int diff;
            public boolean edit_mode;
            public boolean running;
            public RenderArea() {
                super();
                setSize(1000, 1000);
                setVisible(true);
                setBackground(Color.WHITE);
                setForeground(Color.BLACK);
                setLocation(0, 0);
                diff = 20;

                this.addMouseListener((MouseListener) new MouseListener ↩
                    (){

                    @Override
                    public void mouseReleased(MouseEvent arg0) {

                    }

                    @Override
                    public void mousePressed(MouseEvent arg0) {
                        clicked(arg0);
                    }

                    @Override
```

```
                public void mouseExited(MouseEvent arg0) {

                }

                @Override
                public void mouseEntered(MouseEvent arg0) {

                }

                @Override
                public void mouseClicked(MouseEvent arg0) {

                }
            });
            this.addKeyListener(this);
            entities = new ArrayList<ArrayList<Boolean>>();
            for(int i=0;i<1000/diff;i++)
            {
                entities.add(new ArrayList<Boolean>());
                for(int j=0;j<1000/diff;j++)
                {
                    entities.get(i).add(false);
                }
            }

        }
        void clicked(MouseEvent arg0)
        {
            System.out.println("Button "+(arg0.getButton()== 1 ? " ↩
               Left" : "Right"));
            System.out.println("X:"+arg0.getX()/diff);
            System.out.println("Y:"+arg0.getY()/diff);
            if(edit_mode)
            {
                entities.get(arg0.getX()/diff).set(arg0.getY()/diff ↩
                   ,!entities.get(arg0.getX()/diff).get((arg0.getY ↩
                   ()/diff)));
                this.update(this.getGraphics());
            }

        }
        @Override
        public void keyTyped(KeyEvent e) {
            //System.out.println(e.getKeyChar());
        }

        @Override
        public void keyReleased(KeyEvent e) {
            System.out.println("Key pressed:"+e.getKeyChar());
            if(e.getKeyChar()=='e')
```

```
                {
                    edit_mode = !edit_mode;
                }
                else if(e.getKeyChar()=='q')
                {
                    this.running = false;
                }
                else if(e.getKeyChar()=='c')
                {
                    if(edit_mode)
                    {
                        for(int i=0;i<this.entities.size();i++)
                        {
                            for(int j=0;j<this.entities.get(1).size();j ↩
                               ++)
                            {
                                this.entities.get(i).set(j,false);
                            }
                        }
                        this.update(this.getGraphics());
                    }

                }
            }

            @Override
            public void keyPressed(KeyEvent e) {
                //System.out.println(e.getKeyChar());
            }
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.clearRect(0, 0, 1000, 1000);
                for(int i=0;i<1000;i+=diff)
                {
                    g.drawLine(i, 0, i, 1000);
                }
                for(int j=0;j<1000;j+=diff)
                {
                    g.drawLine(0, j, 1000, j);
                }
                for(int i=0;i<1000;i+=diff)
                {
                    for(int j=0;j<1000;j+=diff)
                    {
                        if(entities.get(i/diff).get(j/diff))
                        {
                            g.setColor(Color.BLACK);

                        }
```

```
                            else
                            {
                                g.setColor(Color.WHITE);
                            }

                            g.fillRect(i+2, j+2, diff-3, diff-3);
                        }
                    }
                }

                private static final long serialVersionUID = 1L;

            }
            private static final long serialVersionUID = 1L;
            public static void main(String args[])
            {
                game_of_life gol = new game_of_life();
                while(gol.ra.running)
                {
                    if(!gol.ra.edit_mode)gol.update();
                    try{Thread.sleep(200);}
                    catch(Exception ex)
                    {
                    }
                    gol.ra.update(gol.ra.getGraphics());
                }
                gol.dispose();
            }
}
```

Lényegében majdnem ugyanaz van megírva, mint a C++ verzióban. Annyi, hogy a Java-s verzióban a beépített gui library-ket (swing, awt) használom. A program lényege az ez alatt lévő C++ életjátékba van kifejtve. Továbbiakban hozzá szeretnék írni

7.3. Qt C++ életjáték

Most Qt C++-ban!

Az életjátékot John Conway Cambridge Egyetem matematikusa találta ki. Ez egy nullszemélyes játék. Lényege, hogy a játékos megad kezdő alakzatot vagy alakzatokat és ha elinditjuk egy számítás eredményeként bizonyos feltételek mellett új alakzatot kapunk. Sejtautomaták közé tartozik ez a fajta játék. Szabályok:

1. Túléli a sejt(kocka), ha a közvetlen közelébe 2 vagy 3 szomszédja van.

2. A sejt elpusztul, ha 2-nél kevesebb vagy 3-nél több szomszédja van. Az előbbit túlnépesedésnek a utobbit elszigetelődésnek nevezzük.

3. Új sejt születik minden olyan cellában, amelynek környezetében párom sejt található.

Jellegzetes alakzat a Bill Gosper féle "siklóágyú", amely időközönként siklókat lő ki.

Tanulságok, tapasztalatok, magyarázat...

```
#include <SFML/System.hpp>
#include <SFML/Graphics.hpp>
#include <vector>
#include <iostream>
using namespace sf;
using std::vector;
using std::cout;
using std::endl;
class Grid
{
public:
    Grid(unsigned int x = 1000, unsigned int y = 1000, unsigned int diffs = ↩
        50) : w(x),h(y),diff(diffs)
    {

    }
    void draw(RenderWindow & window)
    {
        for(int i=0;i<w;i+=diff)
        {
            Vertex line[] =
            {
                sf::Vertex(sf::Vector2f(i,0)),
                sf::Vertex(sf::Vector2f(i, h))
            };
            line[0].color = Color(0,0,0);
            line[1].color = Color(0,0,0);
            window.draw(line, 2, sf::Lines);
        }
        for(int i=0;i<h;i+=diff)
        {
            Vertex line[] =
            {
                sf::Vertex(sf::Vector2f(0,i)),
                sf::Vertex(sf::Vector2f(w,i))
            };
            line[0].color = Color(0,0,0);
            line[1].color = Color(0,0,0);
            window.draw(line, 2, sf::Lines);
        }
    }
    unsigned int w;
    unsigned int h;
    unsigned int diff;
};
class Square
{
public:
    Square()
```

```
    {
    }
    Square(int x_pos, int y_pos, float w, bool alive = false)
    {
        square = new RectangleShape(Vector2f(w,w));
        square->setPosition(Vector2f(x_pos,y_pos));
        aliveState = alive;
    }
    /*Square (const Square& other )
    {
        if(this != &other)
        {
            delete this->square;
            this->square = other.square;
        }
    }
    Square& operator=(const Square& other)
    {
        if(this != &other)
        {
            delete this->square;
            this->square = other.square;
        }
        return *this;
    }*/
    ~Square()
    {
        delete square;
    }
    void update()
    {
        if(aliveState)
        {
            square->setFillColor(Color::Black);
        }
        else
        {
            square->setFillColor(Color::White);
        }

    }
    void setFill(Color c = Color::White)
    {
        square->setFillColor(c);
    }
    void draw(RenderWindow &window)
    {
        window.draw(*square);
    }
    RectangleShape* square;
```

```
    bool aliveState;
private:

};
vector<vector<Square*>> update(vector<vector<Square*>> v)
{
    vector<vector<Square*>> tmp ;//= v;
    for(int i=0;i<v.size();i++)
    {
        tmp.push_back(vector<Square*>());
        for(int j=0;j<v[0].size();j++)
        {
            tmp[i].push_back(new Square(v[i][j]->square->getPosition().x,v[ ↩
               i][j]->square->getPosition().y,v[i][j]->square->getSize().x, ↩
               v[i][j]->aliveState));
        }
    }
    for(int i=0;i<v.size();i++)
    {
        for(int j=0;j<v[0].size();j++)
        {
            int live_neighbours = 0;
            live_neighbours +=  v[(i-1)%v.size()][(j-1)%v[0].size()]-> ↩
               aliveState;
            live_neighbours +=  v[(i-1)%v.size()][(j)%v[0].size()]-> ↩
               aliveState;
            live_neighbours +=  v[(i-1)%v.size()][(j+1)%v[0].size()]-> ↩
               aliveState;
            live_neighbours +=  v[(i)%v.size()][(j-1)%v[0].size()]-> ↩
               aliveState;
            live_neighbours +=  v[(i)%v.size()][(j+1)%v[0].size()]-> ↩
               aliveState;
            live_neighbours +=  v[(i+1)%v.size()][(j-1)%v[0].size()]-> ↩
               aliveState;
            live_neighbours +=  v[(i+1)%v.size()][(j)%v[0].size()]-> ↩
               aliveState;
            live_neighbours +=  v[(i+1)%v.size()][(j+1)%v[0].size()]-> ↩
               aliveState;
            //cout <<" X:"<<i << " y:"<< j << " Live neighbours:"<<  ↩
               live_neighbours<<endl;
            if(v[i][j]->aliveState)
            {
                if(live_neighbours < 2)
                {
                    tmp[i][j]->aliveState = false;
                }
                else if(live_neighbours > 3)
                {
                    tmp[i][j]->aliveState = false;
                }
```

```
            }
            else
            {
                if(live_neighbours == 3)
                {
                    tmp[i][j]->aliveState = true;
                }
            }
        }
    }
    return tmp;
}
void killall(vector<vector<Square*>> &v)
{
    for(int i=0;i<v.size();i++)
    {
        for(int j=0;j<v[0].size();j++)
        {
            v[i][j]->aliveState=false;
        }
    }
}
int main()
{
    RenderWindow window(VideoMode(1000,1000),"Game of Life");
    window.setFramerateLimit(10);
    window.setActive();

    Vector2u size = window.getSize();
    Grid g(size.x,size.y,1000/40);


    int h = g.h/g.diff+1;
    int w = g.w/g.diff+1;
    //Square squares[h][w];
    std::vector<std::vector<Square*>> squares;
    bool edit_mode = true;
    for(int i=0;i<h;i++)
    {
        squares.push_back(vector<Square*>());
        for(int j=0;j<w;j++)
        {
            squares[i].push_back(new Square(i*g.diff+1,j*g.diff+2,g.diff-3) ↩
               );
        }
    }
    //squares[4][5]->aliveState=true;
    while (window.isOpen())
    {
        window.clear(sf::Color::White);
```

```
        // check all the window's events that were triggered since the last ↩
            iteration of the loop
        sf::Event event;
        while (window.pollEvent(event))
        {
            // "close requested" event: we close the window
            if (event.type == sf::Event::Closed)
            {
                window.close();
            }
            else if(event.type == Event::MouseButtonPressed)
            {
                if(edit_mode && event.mouseButton.button == Mouse::Button:: ↩
                   Left)
                {
                    /*cout<<event.mouseButton.x<<" "<<event.mouseButton.y<< ↩
                       endl;
                    cout<<event.mouseButton.x/g.diff<< " "<< event. ↩
                       mouseButton.y/g.diff<<endl;*/
                    squares[event.mouseButton.x/g.diff][event.mouseButton.y ↩
                       /g.diff]->aliveState= !squares[event.mouseButton.x/g ↩
                       .diff][event.mouseButton.y/g.diff]->aliveState;
                    cout<< "Changed state on entity at X:"<< event. ↩
                       mouseButton.x/g.diff << " Y:"<<event.mouseButton.y/g ↩
                       .diff << " to "<< (squares[event.mouseButton.x/g. ↩
                       diff][event.mouseButton.y/g.diff]->aliveState? " ↩
                       Alive" : "Dead")<<endl;
                }
            }
            else if(event.type == Event::KeyPressed)
            {
                if(event.key.code == Keyboard::Q)
                {
                    cout<<"Close request recieved. Application will exit." ↩
                       <<endl;
                    window.close();
                }
                if(edit_mode && event.key.code == Keyboard::C)
                {
                    cout<< "Killed all entities." <<endl;
                    killall(squares);
                }
                if(event.key.code == Keyboard::E)
                {
                    edit_mode = !edit_mode;
                    if(edit_mode)
                    {
                        cout<< "Changed to edit mode."<<endl;
                    }
                    else
```

```
                    {
                        cout<< "Changed to simulation mode."<<endl;
                    }

                }
            }


        }
        /*s.draw(window);
        s.square->setPosition(Vector2f(s.square->getPosition().x+1,s.square ↩
           ->getPosition().y));*/
        g.draw(window);
        for(int i=0;i<h;i++)
        {
            for( int j=0; j<w;j++)
            {
                squares[i][j]->draw(window);
            }
        }
        window.display();
        if(!edit_mode) squares = update(squares);
        for(int i=0;i<h;i++)
        {
            for( int j=0; j<w;j++)
            {
                squares[i][j]->update();
            }
        }
    }

    return EXIT_SUCCESS;
}
```

Fordítás: "g++ *.cpp -o sfml-app -lsfml-graphics lsfml-window -lsfml.system". Amikor már a programunk fut , az esetben kézzel kell rajzolnunk egy alakzatot , majd nyomni egy "e" betűt , aminek következtében a program elkezdi az "életjátékot".

A program SFML ablakozó rendszerre épül. Elősször létrehozzuk az ablakunkat, majd megrajzoltatjuk a rács-pont rendszert. A `update()` függvénybe van megírva az egyes szabályok mentén történő matematikai műveletek végrehajtása. Ez ha a szabály teljesül visszatér egy Alive értékkel, ami jellemzi a rácspontot. A többi az SFML-hez tartozó kódokat mutatja. Úgy, mint az egéreségyek, kirajzoltatás update stb.

Idő híján a csak a lényegi részt írtam ki. Később bővebben is kifejtem a program működését, mint Java, mint C++ verziókban.

7.4. BrainB Benchmark

Ez a program vagy játék egy készségmérő program, ami azt méri, hogy egy bizonyos objektumot mennyire tudunk lekövetni. Kezdetben egy négyzetben lévő karikára kell ráfocuszálni. Cél, hogy az egérgomb folyamatos nyomvatartása mellett kövessük a Samu Entropy nevű objektumot a kurzorral. Bizonyos időközönként új négyzetek (objektumok) jelennek meg a képernyőn, ami nagyban nehezíti a Samu objektum követését. Illetve az egyes objektumok mozgása, rezgése is megnőhet.

A játék során erős koncentrációs és reagálási képesség kell.

Ehhez hasonló képességfelmérés létezik a League of Legends játékon belül is, amit elsőként Veres Dávid Msc hallgató munkájában láttam. A játékon belül különböző behatások érnek minket játék közben. (pl. teamfight, roam, active items stb.) Ezekre mind együttesen kell odafigyelni, ha hasznos tagjai akarunk lenni a csapatunknak. Ez nyilván nem könnyű így sok gyakorlást és odafigyelést igényel az egyes objektumok követése, figyelése.

8. fejezet

Helló, Schwarzenegger!

8.1. Szoftmax Py MNIST

Először is, hogy használni tudjuk le kell töltenünk a TensorFlow-t. A TF telepítéséhez nem kell külön ügyködni, elég ha felmegyünk a hivatalos oldalára és onnan az útmutatók segítségével megcsináljuk. A program lényege, hogy 1-9 ig számokat mutató kis 28x28 as képekből fel kell ismernie az éppen aktuális számjegyet. Ez ugye 784 pontot, azaz 784 db számot jelent. Ezt a 784 számot felfoghatjuk úgy is, mint egy pont koordinátáit a 784 dimenziós térben (ne, még csak ne is próbáljuk meg elképzelni), és puff, már meg is van a bemeneti tenzor. Az eredmény meg ugye 0-9-ig egy szám, pontosabban 10 db érték, ami azt mondja meg, hogy rendszerünk az adott bemenetre milyen számot tippel. Ugye ha jó a rendszer, akkor arra a számra fogja mondani a legnagyobb esélyt, ami oda van írva. Ezt úgy kell elképzelni, hogy mondjuk egy írott 6-osra azt mondja, hogy 10%, hogy 8, 20% hogy 9, és 70%, hogy 6-os számot lát. Szóval a kimenet 10 érték. Ezt is el lehet képzelni egy pontnak a 10 dimenziós térben, így ez is leírható egy tenzorként. Szóval kb. megvan amit kerestünk. A problémát oda redukáltuk, hogy bemegy egy tenzor, és kijön egy másik.

A Szoftmax python kódja:

```
# 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.
#  ↩
   ==============================================================================


# Norbert Batfai, 27 Nov 2016
# Some modifications and additions to the original code:
# https://github.com/tensorflow/tensorflow/blob/r0.11/ ↩
   tensorflow/examples/tutorials/mnist/mnist_softmax.py
# See also http://progpater.blog.hu/2016/11/13/ ↩
   hello_samu_a_tensorflow-bol
#  ↩
   ==============================================================================


"""A very simple MNIST classifier.

See extensive documentation at
http://tensorflow.org/tutorials/mnist/beginners/index.md
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse

# Import data
from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf

import matplotlib.pyplot


FLAGS = None


def readimg():
    file = tf.read_file("sajat8a.png")
    img = tf.image.decode_png(file)
    return img

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

# Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.matmul(x, W) + b
```

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

# The raw formulation of cross-entropy,
#
#   tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(tf.nn.softmax(y)) ↩
   ,
#                                 reduction_indices=[1]))
#
# can be numerically unstable.
#
# So here we use tf.nn.softmax_cross_entropy_with_logits on the ↩
    raw
# outputs of 'y', and then average across the batch.
cross_entropy = tf.reduce_mean(tf.nn. ↩
   softmax_cross_entropy_with_logits(y, y_))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize( ↩
   cross_entropy)

sess = tf.InteractiveSession()
# Train
tf.initialize_all_variables().run()
print("-- A halozat tanitasa")
for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
    if i % 100 == 0:
    print(i/10, "%")
print(" ↩
   ------------------------------------------------------------")

# Test trained model
print("-- A halozat tesztelese")
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1) ↩
   )
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf. ↩
   float32))
print("-- Pontossag: ", sess.run(accuracy, feed_dict={x: mnist. ↩
   test.images,
                                    y_: mnist.test.labels}))
print(" ↩
   ------------------------------------------------------------")

print("-- A MNIST 42. tesztkepenek felismerese, mutatom a  ↩
   szamot, a tovabblepeshez csukd be az ablakat")

img = mnist.test.images[42]
image = img
```

```
            matplotlib.pyplot.imshow(image.reshape(28, 28), cmap=matplotlib ↩
               .pyplot.cm.binary)
            matplotlib.pyplot.savefig("4.png")
            matplotlib.pyplot.show()

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

            print("-- Ezt a halozat ennek ismeri fel: ", classification[0])
            print(" ↩
               ----------------------------------------------------------")

            print("-- A sajat kezi 8-asom felismerese, mutatom a szamot, a  ↩
               tovabblepeshez csukd be az ablakat")

            img = readimg()
            image = img.eval()
            image = image.reshape(28*28)

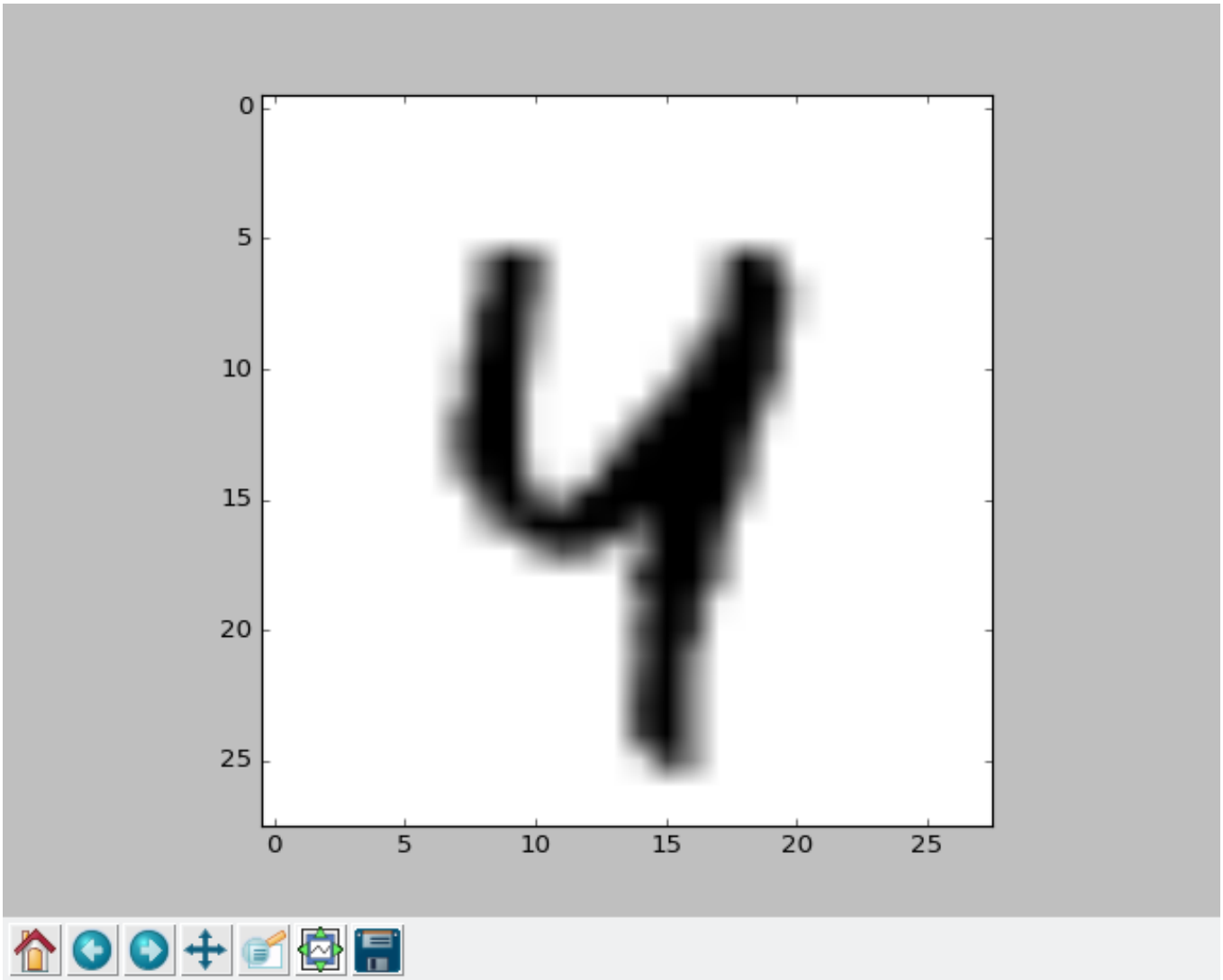
            matplotlib.pyplot.imshow(image.reshape(28, 28), cmap=matplotlib ↩
               .pyplot.cm.binary)
            matplotlib.pyplot.savefig("8.png")
            matplotlib.pyplot.show()

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

            print("-- Ezt a halozat ennek ismeri fel: ", classification[0])
            print(" ↩
               ----------------------------------------------------------")

            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 = parser.parse_args()
            tf.app.run()
```

Az fenti kód két részre bontható. Van az első rész, ahol a feltanítjuk a hálózatunkat a felismerni kívánt "objektumokkal". Feltölti a képet, majd ezek alapján egy bizonyos pontosságot belőve meghatározza, hogy az épp milyen objektum. A második rész a tesztelése a hálózatnak, aholis felhasználói inputokat vizsgál a hálózat és eldönti, hogy az inputon melyik számjegy található. Futtatni a `python` értelmezővel tudjuk. Futtatás után a felismert számot kiírja a kimenetre.



8.1. ábra. Bátfai tanár úr ábrája a megjelenített számokról a MNIST-ben.

8.2. Mély MNIST

```
# 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)
```

A deep learning (mély tanulás) paradigmán alapuló mély neurális hálózat. A fentiekhez képest több különbség is adódik..

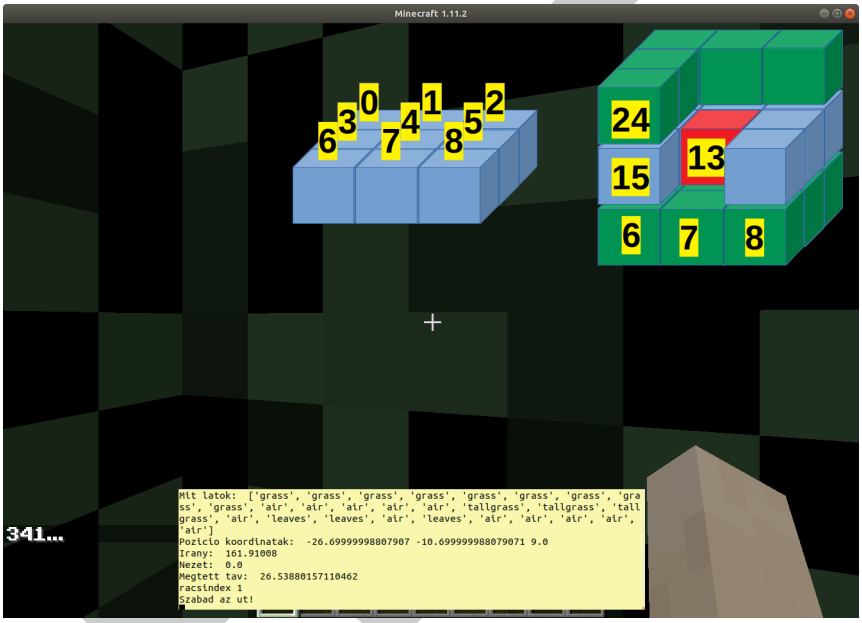
Itt még lesz valami

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Voluptates suscipit culpa adipisci inventore in eaque, et omnis aperiam dignissimos incidunt magnam, similique dolores blanditiis atque nihil autem ea, voluptatibus libero!

8.3. Minecraft-MALMÖ

Megoldás videó: https://youtu.be/bAPSu3Rndi8

Az 1. ábrán azt látjuk, hogy az 1.5-ös proginkban Stivi csapdába esik: elakad (2,3 blokk) között A jobb felső sarokban láthatjátok, hogy (a 9x3 kiskockás nagykocka) Steve midig a piroson áll, a középső kék kockasort,az a ugyanaz mint a mellette lévő korábbi kockasor. Mostmár 27 kockát (blokkot) lát Steve. A régebi progamunkban Steve mindig a 4-en ált most már a 13-as on áll.



8.2. ábra. Bátfai Tanár úr ábrája a blokkok számozásáról.

Minecraft Steve szemüvege 2018. október 28. - nb Az 1. ábrán azt látjuk, hogy az 1.5-ös proginkban Stivi csapdába esik: elakad (2,3 blokk) között A jobb felső sarokban láthatjátok, hogy (a 9x3 kiskockás nagykocka) Steve midig a piroson áll, a középső kék kockasort,az a ugyanaz mint a mellette lévő korábbi kockasor. Mostmár 27 kockát (blokkot) lát Steve. A régebi progamunkban Steve mindig a 4-en ált most már a 13-as on áll. 3d1.png 1. ábra Stivnek most nem 9 kockáját hanem 27 hetet nézünk.Igy a lába aiatti és a feje fölöti kockákat is látja. Igy amikor elöre néz a vizalatti kockákat is látja. Most az első képröl írunk. A 0-ásón grass, az 1-en grass, a 3 is grass, a 4 is grass, amúgy a grass, angolul a fű,föld az 5 is grass, a 6 is grass, a 7 is grass, a 8 is grass, ez a talajszint, az alsó 9 kocka (0-8). A 9. air, az air azt jejenti angolul,hogy levegő, a 10 is air, a 11 is air, a 12 is air, a 13 is air .A 14 tallgrass a tallgrass azt jelenti, hogy magas fű, a 15 is tallgrass, a 16 is tallgrass, azomban 17 megint air. Ez a 2. szinten lévő 9 kocka. Ez látszik a 2. ábrán. A 18. kockával kezdődik a 3. emelet, leaves-el a leaves azt jelenti angolul, hogy levél, 19 is leaves,a 20 megint air,a 21 megint leave, a 22 air,a 23 air, a 24 is air,a 25 is air, a 26, az utolsó is air. Ez látszik a 3. ábrán.

8.3. ábra. Bátfai Tanár úr ábrája a különböző blokkok jelentéséről.

9. fejezet

Helló, Chaitin!

9.1. Iteratív és rekurzív faktoriális Lisp-ben

Ahhoz, hogy tudjuk hackelni a GIMP-et tisztába kell lenni néhány kifejezéssel.

Kifejezések:

A lisp változók és minden kifejezést kerek zárójelek () közé kell tenni. A zárójelen belüli kifejezés(ek) meghatározott sorrendet követnek. pl.: (- 5 5) A kifejezések mindig egy függvénnyel kezdődnek majd utána a megfelelő paraméterek. A Sheme nem veszi figyelembe a szóközöket így szóval külön is tudjuk írni az egyes kifejezéseket. További kifejezés lehet: (* (+ 5 5) (- 10 5))

Függvények:

Függvényeket a `define` kulcsszó segítségével definiálunk. Zárójelek közt megadjuk a define kulcsszót ezzel jelezve hogy függvénydefinicíió következik, majd név és utánaírjuk a kifejezést. pl.: (define (square x) (* x x))

Ezek alapján már egyszerűen megtudjuk írni a faktoriális függvényt Lisp-ben.

Ahogy írtuk a függvénydefiníció a `define` kulcsszóval kezdődik majd megadjuk a fg nevét `(fakt n)`. Ezután írjuk a függvény "törzsét", ami kiszámolja az n faktoriálisát. Egybepakolva így néz ki:

```
(define (fakt n) (if(< n 0) 1 (* n (fakt(- n 1)))))
```

9.2. Gimp Scheme Script-fu: króm effekt

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

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

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

A program egésze így néz ki:

```
; bhax_chrome3.scm
;
; BHAX-Chrome creates a chrome effect on a given text.
; Copyright (C) 2019
; Norbert Bátfai, batfai.norbert@inf.unideb.hu
; Nándor Bátfai, batfai.nandi@gmail.com
;
;    This program is free software: you can redistribute it and ↩
   /or modify
;    it under the terms of the GNU General Public License as  ↩
   published by
;    the Free Software Foundation, either version 3 of the  ↩
   License, or
;    (at your option) any later version.
;
;    This program is distributed in the hope that it will be  ↩
   useful,
;    but WITHOUT ANY WARRANTY; without even the implied  ↩
   warranty of
;    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See  ↩
   the
;    GNU General Public License for more details.
;
;    You should have received a copy of the GNU General Public  ↩
   License
;    along with this program.  If not, see <https://www.gnu.org ↩
   /licenses/>.
;
; Version history
;
;  This Scheme code is partially based on the Gimp tutorial
;  http://penguinpetes.com/b2evo/index.php?p=351
;  (the interactive steps of this tutorial are written in  ↩
   Scheme)
;
;  https://bhaxor.blog.hu/2019/01/10/ ↩
   a_gimp_lisp_hackelese_a_scheme_programozasi_nyelv
;

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

```
                (aset tomb 6 200)
                (aset tomb 7 190)
            tomb)
        )

        ;(color-curve)

        (define (elem x lista)

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

        )

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

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

            (list text-width text-height)
            )
        )

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(script-fu-register "script-fu-bhax-chrome"
    "Chrome3"
    "Creates a chrome effect on a given text."
    "Norbert Bátfai"
    "Copyright 2019, Norbert Bátfai"
    "January 19, 2019"
    ""
    SF-STRING       "Text"      "Bátf41 Haxor"
    SF-FONT         "Font"      "Sans"
    SF-ADJUSTMENT   "Font size" '(100 1 1000 1 10 0 1)0"

    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"
)
```

A program úgy kezdődik, hogy definiálunk egy `color-curve` függvényt. A let kulcsszóval megadunk egy lokális változót, ami egy 8 elemű tömb lesz. Ezután feltöltjük az értékeit különböző értékekkel. Ez lesz a színátmenetért felelős függvény.

```
(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)
```

A függvény 3 paramétert vár. Magát a szöveget, amit formázni szeretnénk. A szöveg betűstílusát illetve a szöveg méretét. `set!` kulcsszóval beállítunk értékeket a változóknak és a változók globális értékekké válnak. Létrehozunk két változót text-width illetve text-height-t és beállítjuk az értékeiket 1-re. Majd a set! segítségével beállítjuk a további értékeket a paraméterként megkapott 3 érték alapján.

```
(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)
```

Az alábbi programban fog megtörténni a chrome effect leimplementálása. A sript-fu-bhax-chrome függvény 7 paramétert vár. Ezek a következők: (script-fu-bhax-chrome "formázandó szöveg" "betűstílus" betűméret szélesség magasság színskála "színezési stílus") A továbbiakban írni fogok még róla. Időhiány stb.

```
(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")
```

9.3. Gimp Scheme Script-fu: név mandala

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

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

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

Ebben a programban egy mandalát készítünk Scheme segítségével Lisp nyelven. A mandala egy körkörös stílusú vallási alagzat. A programban egy szöveget fogunk használni a mandala elkészítéséhez. Úgy működik, hogy a `fu-bhax-mandala` függvény a megadott paraméterek segítségével előállítja a mandalát. A mandalához forgatást használezen kód segítségével. `(gimp-item-transform-rotate text-layer (/ *pi* 4) TRUE 0 0)` Az elforgatott formázott szöveg után megkapjuk a mintát. A kódot illetve lefuttatva az Új/ Létrehozás menüpont alatt GUI-s interface-n tudjuk megadni a paramétereinket.

```
; bhax_mandala9.scm
;
; BHAX-Mandala creates a mandala from a text box.
; Copyright (C) 2019  Norbert Bátfai, batfai.norbert@inf.unideb ↩
   .hu
;
;    This program is free software: you can redistribute it and ↩
   /or modify
;    it under the terms of the GNU General Public License as  ↩
   published by
;    the Free Software Foundation, either version 3 of the  ↩
   License, or
;    (at your option) any later version.
;
;    This program is distributed in the hope that it will be  ↩
   useful,
;    but WITHOUT ANY WARRANTY; without even the implied  ↩
   warranty of
;    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See  ↩
   the
;    GNU General Public License for more details.
;
;    You should have received a copy of the GNU General Public  ↩
   License
;    along with this program.  If not, see <https://www.gnu.org ↩
   /licenses/>.
;
; Version history
;
;  This Scheme code is partially based on the Python code
;  Pat625_Mandala_With_Your_Name.py by Tin Tran, which is  ↩
   released under the GNU GPL v3, see
;  https://gimplearn.net/viewtopic.php/ ↩
   Pat625-Mandala-With-Your-Name-Script-for-GIMP?t=269&p=976
;
;  https://bhaxor.blog.hu/2019/01/10/ ↩
   a_gimp_lisp_hackelese_a_scheme_programozasi_nyelv
;
```

```
(define (elem x lista)

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

)

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

    text-width
    )
)

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

    (list text-width text-height)
    )
)


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

(define (script-fu-bhax-mandala text text2 font fontsize width ↩
   height color gradient)
(let*
    (
        (image (car (gimp-image-new width height 0)))
        (layer (car (gimp-layer-new image width height ↩
           RGB-IMAGE "bg" 100 LAYER-MODE-NORMAL-LEGACY)))
        (textfs)
        (text-layer)
        (text-width (text-width text font fontsize))
        ;;;
```

```
                    (text2-width (car (text-wh text2 font fontsize)))
                    (text2-height (elem 2 (text-wh text2 font fontsize)))
                    ;;;
                    (textfs-width)
                    (textfs-height)
                    (gradient-layer)
                )

                (gimp-image-insert-layer image layer 0 0)

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

                (gimp-context-set-foreground color)

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

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

                (set! text-layer (car (gimp-layer-new-from-drawable
tfs image)))
                (gimp-image-insert-layer image text-layer 0 -1)
                (gimp-item-transform-rotate text-layer (/ *pi* 2) TR
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"
        )
```

10. fejezet

Helló, Gutenberg!

10.1. PICI Juhász István

II. heti előadás - 1.2 Alapfogalmak

A programozásban használatos nyelvek közül több szintet különböztetünk meg. gépi nyelv assembly szintű nyelv magas szintű nyelv Az előbbi 2 szintről csak említés szintén tanultunk, hiszen a mai időkben a legelterjedtebbek a magas szintű nyelvek. Minden magas szintű nyelvet a szintaktika és a szemantika határoz meg. A szintaktika a nyelv által lefektetett és jól meghatározott szabályok gyűjteménye. Ezek formai, nyelv specifikus követelmények. Pl.: Szintaktikai hiba lehet egy ; záró tag elhagyása. A szemantikai szabályok pedig a tartalmi, jelentésbeli formális meghatározások. Pl.: Szemantikai hiba esetén a programunk futhat viszont nem várt működést eredményezhet. Minden program az őt futtató processzor utasításkészlete alapján fordul. (gépi kód) Alapesetben a programunk kódja nem gépi kódon íródott így át kell alakítani a forráskódokat a gép számára értelmezhető nyelvezetűre. Erre két megoldás adott a magas szintű nyelvek esetében. fordítóprogramos nyelv interpreteres nyelv A fordítóprogram a forráskódból gép kódú úgynevezett tárgy kódot állít elő. Ez m ég nem futtatható így egy kapcsolatszerkesztő előállítja a tárgykód ból a megfelelő futtatható kód állományt. Interpreteres nyelv esetében pedig a forráskódot sorról sorra értelmezi és hajtja végre. Tipikus interpreteres nyelv a Java, ahol a forráskódból köztes .class kód majd gépi kód állítódik elő. Minden programozási nyelvhez létezik olyan IDE (Integrated Developer Environment), ami nagyban segíti az adott nyelvben történő hatékony programozást. IDE funkciók a kódszínezés, kódkiegészítés, debuggolás, tesztelés stb. Eddigi tanulmányaim során többnyire objektumorientált nyelvekben programoztam, ami az imperatív nyelvi programozás csoportjába tartozik. Valamint ide tartozik még az eljárásorientált nyelvek is. Viszont sose hallottam még deklaratív nyelvekről. Ezek többnyire a programozó által meghatározott problémára keresik a megoldást a nyelvi implementációk segítségével. Ezek nem algoritmikus nyelvek.

III. heti előadás (28. oldal, a "2.4. Adattípusok" című rész):

Az adatabsztrakció első formája az adattípus. Az adattípus rendelkezik névvel, amely azonosítja a típust, például int, double. Léteznek típusos és nem típusos programozási nyelvek. A típusososok engedik, hogy a programozó adja meg a változók típusát. Ilyenek például a C++ és a Java. A nem típusosok automatikusan állapítják meg a változó típusát. Ilyenek például a R és a Python. Adattípusoknak két csoportja van, az egyszerű és az összetett. Az egyszerű adattípusok azok, amelyeket nem lehet tovább bontani, például int. Az összetett típusok például a struktúrák vagy a felhasználó által definiált típusok.

III. heti előadás (34. oldal, a "2.5. A nevesített konstans" című rész):

A nevesített konstansok azt a célt szolgálják a programokban, hogy a konstansoknak olyan nevet adjunk, amely jelképezi annak típusát és értékét. Illetve másik célja, hogy sokszori használat esetén csak a definiálásnál kelljen változtatni az értékét, ha szükséges. Ezeket a konstansokat mindig definiálni kell.

III. heti előadás (35. oldal a "2.6. A változó" című rész):

A változónak négy komponense van: a név, az attribútomok, a cím és az érték. A név az egy azonosító, a másik három komponenst egy névhez rendeljük hozzá. A legfőbb atribútom, a típus, amely a változó által felvett értéket határolja be. A változóhoz az attribútumok deklarációk segítségével rendelődnek. A deklarációnak különböző fajtáit simerjük: Explicit deklaráció, Implicit deklaráció, Automatikus deklaráció. A változó címe meghatározza a változó értékének a helyét. A címrendelésnek három fajtáját ismerjük: a Statikus tárkiosztás, a Dinamikus tárkiosztás, és a programozó által vezérelt kiosztás. A változó értékének a meghatározására több opció is van: értékadó utasítás, kezdőérték adás.

III. heti előadás (39. oldal, az "2.7. Alapelemek az egyes nyelvekben" című rész):

C-ben az aritmetikai típusok az egyszerű típusok, a származtatottak az összetett típusok. A karakter típus elemeit belső kódok alkotják. Logikai típus nincs, a hamis az int 0 az igaz pedig az int 1. A struktúra egy fix szerkeztű rekord. A void tartománya üres. A felsorolásos típusok nem fedhetik egymást. Különböző elemekhez ugyanazt az értéket hozzárendelhetjük.

IV. heti előadás - 3. Kifejezések

Kifejezések olyan “szintaktikai” eszközök, amelyekkel új értékeket adhatunk különböző a programon belül található kód részletekből. Két részből épül fel, az érték és típusból. Formális összetevők az operandusok, operátorok és a kerek zárójelek. Az operandusok reprezentálják az értékeket. Ezek lehetnek különböző változók, konstansok, függvényhívások stb. Az operátorok a műveleti jelek, amik meghatározzák az egyes értékekkel való műveletet. Pl. egyenlőségjel (=), mutatók (->), pontok (.) vagy akár az összeadásjel is. Operandusok számát tekintve 3 típust különböztetünk meg. Az egyoperandusút (pl. ++a). a kétoperandusút (pl. a + a), és a háromoperandusút, ami jellegzetesen lehet egy “kicsi if”. (pl. a ? xyz : asd) Két operandusú kifejezéseknél három sorrend van. Prefix, itt a operátor az operandusok előtt áll. Postfix, ahol az operátor az operandusok után áll. Illetve az infix, ahol az operátor az operandusok között helyezkedik el. Kiértékelésnek nevezzük azt a folyamatot, amikor a kifejezésünk értéke kiértékelődik. A műveletek végrehajtása balról jobbra az egyes műveletek által meghatározott erősség (precedencia) szerint történik.

V. heti előadás (56. oldal, az "4. Utasítások" című rész):

Az utasítások megalkotják a programok egységeit: az algoritmusok egyes lépései, a fordítóprogram ezzel generálja a tárgyporgramot. Két csoportjuk van: a deklarációs utasítások, és a végrehajtó utasítások. A deklarációs utasítások mögött nem áll tárgykód, a fordítóprogramnak szólnak. A végrehajtó utasításokból pedig a fordító generálja a kódot. A végrehajtó utasításokat csoportosíthatjuk: értékadó utasítás, üres utasítás, ugró utasítás, elágaztató utasítás, ciklusszervező utasítás, hívó utasítás, vezérlésátadó utasítás, I/O utasítás, egyéb utassítás. A vezérlési szerkezetet megvalósító utasítások: ugró utasítás, elágaztató utasítás, ciklusszervező utasítás, hívó utasítás, vezérlésátadó utasítás.

VII. heti előadás (78-84. oldal):

A paraméterátadásnak többféle módja is lehet, ezek nyelvfüggőek, hogy melyik nyelv melyiket alkalmazza.

Történhet érték szerint, mint a C-ben például. Ekkor a formális paraméter értékül kapja az aktuális paraméter értékét. Ennél a módszernél a függvényben nem lehet megváltoztatni a aktuális paraméter értékét. Lehet címszerinti a paraméterátadás. Ekkor a formális paraméter címe értékül kapja az aktuális paraméter címét. Ilyenkor a függvényben meg lehet változtatni az aktuális paraméter értékét. Lehet eredmény szerinti átadás is, ekkor a formális paraméter szintén megkapja az aktuális paraméter címét, de nem használja,

csak a végén beletölti az adatokat. Létezik még érték-eredmény szerinti, ekkor másolódik a cím szintén, és használja is az adatokat, majd a függvény végén belemásolja a formális paraméterbe az adatokat.

10.2. Programozás bevezetés

[KERNIGHANRITCHIE]

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

V.heti előadás - Vezérlési szerkezetek

Egy nyelv vezérlésátadó utasításai az egyes műveletek végrehajtási sorrendjét határozzák meg. Ha egy kifejezés után pontosvesszőt (;) rakunk akkor az utasítás lesz. Ezeket egy idő után automatikusan alkalmazzuk viszont ha elhagyjuk nagy eséllyel hibát okoz. A ; jel elhagyása tipikus kezdő programozók hibája. :) Vannak utasítások, amelyekhez több utasítás párosulhat, ezeket blokkokba rendezzük a kapcsos zárójelek ({}) segítségével. IF_ELSE utasítás egy elágazás, amely egy feltételt vizsgálva vagy az igaz ágba vagy a hamis ágba fordul. (Szintaxis: if(feltétel) {feltétel teljesülése esetén} else {különben}). Van több ágat vizsgáló IF_ELSE IF_ELSE szerkezet is. Ez többféle feltételt vizsgál és több ágba fordul. (Szintaxis: if(feltétel) {feltétel teljesülése esetén} else if(feltétel2) {feltétel2 teljesülése esetén} else {különben}). Létezik többágú elágazás ez a SWITCH. A switch egy feltételt vizsgálva több ágba is fordulhat a feltétel teljesülése esetén. Szintaxisa a switch(feltétel) majd ezt követi a törzs. Itt különböző case ágakba vannak definiálva a lehetséges feltételnek megfelelő értékek és az azokhoz tartozó utasítások. Minden case ágat egy (;break;) zár. Ha nem illeszkedik egyik megadott értékre sem, akkor a default ágba fordul. Elágazások után tipikus vezérlési szerkezetek a ciklusok. Ezek közül hármat különböztetünk meg. A for ciklust, az elöl tesztelő ciklust(while) illetve a hátul tesztelő ciklust (do while). FOR ciklus áll egy ciklusfejből. Itt három lezáró taggal szeparált tagok kell megadni. A ciklus kezdeti értékét a végpontot illetve a egy ciklus alatti lépés számát vagy “mértékét” (pl. i++). Ha a fej megvan ugyanúgy, mint az if-nél meg kell adnunk a ciklus törzsét kapcsos zárójelek között. WHILE ciklus először egy feltételt vagy feltételeket vizsgál és ezek alapján, ha teljesül megismétli önmagát különben meg kilép a ciklusból. A feltétel után megadjuk a ciklus törzsét kapcsos zárójelekben. DO WHILE ciklus a while-hoz hasonlóan egy feltételt vizsgálva ismételgeti önmagát, viszont itt a feltételvizsgálat a ciklusmag után helyezkedik el. Ez azt jelenti, hogy amit megadtunk a magban az egyszer mindenképpen lefordul, majd utána vizsgálja a feltételt.

V. heti előadás (Függelékből az Utasítások című fejezet):

Az utasítások a leírásuk sorrendjében hajtódnak végre, általános a szintaktikai leírásuk, és számos csoportba sorolhatók: Címkézett utasítások, mint például a "case" és "default" címkéi a "switch" utasítással használhatók. A címke egy azonosító nélküli deklarált azonosítóból áll. Kifejezésutasítsok, az utasítások(kifejezésutasítás, értékadás, függvényhívás) többsége ilyen. Összetett utasítás, több utasítást egyetlen utasításként kezeli, ez a fordításhoz szükséges, mivel sok fordítóprogram csak egyetlen utasítást fogad el. Kiválasztott utasítások, minden esetben a ehetséges végrehajtási sorrendek egyikét választják ki(if, if-else, switch). Iterációs utasítások, egy ciklust határoznak meg(while, do-while, for). Vezérlésátadó utasítások, vezérlés feltétel nélküli átadására alkalmasak(goto, continue, break, return).

10.3. Programozás

[BMECPP]

V. heti előadás (1.-16.):

Míg C-ben egy függvény deklaráció üres paraméterlistája tetszőleges számú paramétert eredményezhet ugyanez C++-ba a paraméterként megadott void kulcsszó segítségével történik. További ilyen C++ tetszőleges paraméter lehet a (...) paraméter definiálás. A program fő lefutási és indulási pontja a main metódus. Ezt kétféleképpen is definiálhatjuk. Üres paraméterrel (pl. int main()) vagy a paraméterben megadott parancssori argumentumokkal illetve azon számával. (pl. int main(int argc, char* argv[])). Hasonlóképpen, mint a többi magasszintű nyelvben változó deklarációt célszerű ott használi, ahol utasítás áll és használja azt. Ha nem így teszünk akkor warning-ot kaphatunk, ami arra figyelmeztet, hogy nem használt változót deklaráltunk. Vannak olyan előre megírt függvények, amelyeknek alapértelmezetten léteznek paraméteri argumentumai, ezeket meg kell adnunk, ha fel szeretnénk használni az adott függvényt. C-ben kizárólag csak érték szerinti paraméter átadás történhet ezzel szemben a C++-ban lehetőség van referencia szerinti paraméterátadásra is. Érték szerint

VI. heti előadás (17-58.):

Ez a fejezet a C++ osztályairól szól. Az objektum orientált programozás alapelve, hogy a probléma megoldását segítse azzal, hogy az emberi gondolkodáshoz közelebb hozza a programozást az osztályok és objektumok bevezetésével. Az egységbe zárás jelenti azt, hogy az összetartozó változók és függvények egy egységben legyen, ezek lesznek az adattagok és a tagfüggvények. Adatrejtés a private és protected adattagok és tagfüggvények bevezetésével jött létre. Az adatrejtés célja, hogy az osztály egyes tagjait ne lehessen kívülről elérni. A konstruktor szerepe, hogy lefusson, amikor létrejön az objektum, ezáltal akár inicializálva az adattagokat. A destruktor célja, hogy lefusson, amikor az objektum megsemmisül, ezáltal akár felszabadítva a diamikus adattagokat. A dinamikus adattagok osztályon belüli pointerek, amelyeket futásidőben hozzuk létre dinamikus memóriafoglalással, ezért ezeket, ha már nincs rájuk szükség, de legkésőbb a destruktorban fel kell szabadítani. A friend osztályok, illetve függvények olyan osztályok, illetve függvénynek, amelyek ugyan nem tagjai az osztálynak, viszont hozzáférnek azok private tagjaihoz. A tagváltozók inicializálása történhet a konstruktoron belül, illetve tagfüggvénnyel, vagy külső függvénnyel is. A statikus tagok azzal a tulajdonsággal rendelkeznek, hogy nem kell az osztályt példányosítani, hogy használni tudjuk. Az osztályok tartalmazhatnak beágyazott definíciókat, amelyek lehetnek enumerációk, struktúrák vagy akár osztályok is.

III. rész

Második felvonás

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.

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. és Ritchie Dennis M., *A C programozási nyelv*, Bp., Műszaki, 1993.

11.5. C++

[BMECPP] Benedek Zoltán és 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.