Univerzális programozás

Programozás kezdöknek

Ed. BHAX, DEBRECEN,
2019. Május 09, v. 1.0.0

COLLABORATORS			
	TITLE : Univerzális programozás		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Rácz, András István	2019. május 9.	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
0.0.1	2019-02-12	Az iniciális dokumentum szerkezetének kialakítása.	nbatfai
0.0.2	2019-02-14	Inciális feladatlisták összeállítása.	nbatfai
0.0.3	2019-02-16	Feladatlisták folytatása. Feltöltés a BHAX csatorna https://gitlab.com/nbatfai/bhax repójába.	nbatfai
0.0.4	2019-03-04	Második fejezet befejezése	randras
0.0.5	2019-03-11	Harmadik feladatsor befejezése.	randras
0.0.6	2019-03-18	Negyedik feladatsort megoldva.	randras
0.0.7	2019-03-25	Ötödik fejezet befejezve.	randras
0.0.8	2019-04-01	Hatodik fejezet megoldva.	randras

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
0.0.9	2019-04-08	Hetedik feladatsor megoldva.	randras
0.1.0	2019-04-22	Nyolcadik fejezet befejezve.	randras
0.1.1	2019-04-29	Kilencedik feladatsor megoldva.	randras
0.1.2	2019-05-06	Olvasónaplo befejezése.	randras
1.0.0	2019-04-09	Könyv befejezése.	randras

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

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?

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

1.2. Milyen doksikat olvassak el?

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

1.3. Milyen filmeket nézzek meg?

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

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

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

A végtelen ciklus egy olyan ciklus mely nem áll le mindaddig míg valamilyen külső behatás nem éri.Végtelen ciklus felléphet valamilyen hiba esetén, de sokszor használjuk szándékosn azokat például egy programablak nyitvatartásához.A mi feladatunk az volt, hogy írjunk egy végtelen ciklust amely egy szalat 100%-on pörget, egyet amely nulla százalékot használ illetve egy olyat amely minden szálat 100%-on használja.

Egy szál 100 százalékon:

Végtelen ciklust számtalan módon létrehozhatunk például while(1) vagy for(;;).Mi az utobbit fogjuk használni mivel abból egy másik programozó számára is egyértelmű, hogy az általuk létrehozot végtelen ciklus nem egy hiba vagy véletlen eredménye

```
int main ()
{
    for (;;);
    return 0;
}
```

Egy szál 0 százalékon:

```
#include <unistd.h>  //Az unistd.h a sleephez szükséges
int main ()
{
    for (;;);
    sleep(1); //A sleep parancs felel a program altatásáért
    return 0;
}
```

Minden szál 100 százalékon:

```
#include <stdlib.h>

int main()
```

```
    {
        #pragma omp parallel    //Ez osztja ki a programot az összes szálra ↩
           .Fordításkor gcc loopminden100.c -o fáljnév -fopenmp
        for (;;);
        return 0;
    }
```

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

A feladatunk az volt, hogy akkora haxorok legyünk, hogy megírjuk azt programot ami ellenőriz más programokat, hogy futásuk során előfordulhat-e végtelen ciklus amely során az adott program lefagy. Ez természetesen nem lehetséges de tegyük fel, hogy lehetséges és nézük végik az alábbi kódokat.Mint láthatjuk irtunk egy lefagy függvényt ami a megadot programban végtelen ciklust keres, ha megtalálja akkor kiírja, hogy a program lefagy.Eddig nincs baj de ha a kapot program futás során nem lép végtelen ciklusba akkor a mi programunk maga fog végtelen ciklusban ragadni és ennek következtében lefagyni.Ezen ok miatt nem tudunk függvényt írni a végtelen ciklus szűrésére, legalábbis jelenleg.

```
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)
  }

}
```

2.3. Változók értékének felcserélése

A feladatunk az volt, hogy cseréljünk fel két változót segédváltozó és logikai utasítások nélkül.Mint alább látható ezt egy egyszerű matematikai módszere megoldható.

```
#include <stdio.h>      //ez a printf és a scanf-hez szükséges

int main()
{
  int a = 0;
  int b = 0;
  printf("Adja meg az a szamot: ");
  scanf("%d" , &a );
  printf("Adja meg a b szamot: ");
  scanf("%d" , &b);
  b = b-a;
  a = a+b;
  b = a-b;
  printf("a=%d%s",a,"\n");
```

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

}
```

2.4. Labdapattogás

A feladatunk az volt, hogy standard outputon, labdához hasonlóan, patogtasunk egy karaktert.A könyebb érthetőség érdekében a magyarázat közvetlenül a kód melett lesz kommentek formájaban.

Labdapatogás if-el

```
#include <stdio.h>  //printf-et tartalmaza
#include <curses.h> //curses.h szükséges a getmaxyx-hez, a WINDOW *-hoz, az ↩
    initscr-hez, mvprintw-hez és a refresh-hez
#include <unistd.h> //Ez az usleep-hez szükséges

int
main ( void )
{
    WINDOW *ablak; //létrehozza az ablakot
    ablak = initscr (); //ez határoza meg a ablak méretét majd át lesz adva ↩
        a getmaxyx-nek

    int x = 0;
    int y = 0;

    int xnov = 1;
    int ynov = 1;

    int mx; //Az ablak magassága
    int my; //Az ablag szélessége

    for ( ;; ) {

        getmaxyx ( ablak, my , mx ); //az ablak méreteit atadja az my-nak  ↩
           és az mx-nek

        mvprintw ( y, x, "o" ); // kiírja a labdát

        refresh ();   //tényleges kimenetet ad,  meg kell hívni hogy  ↩
           kimenetet kapjunk
        usleep ( 50000 ); // a labda sebességét határoza meg a program  ↩
           altatásával (az esetunkben 50000 microssecundum)
        clear();

        x = x + xnov; //a labda vízszintes koordinatája
        y = y + ynov; //a labda függőleges koordinatája
```

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

    }
}
```

Labdapatogás if nélkül

```
#include <stdio.h>  //printf-et tartalmaza
#include <stdlib.h> //Ez tartalmaza az abs függvényt
#include <unistd.h>   //Ez az usleep-hez szükséges

//Az ablak méretnek megadása
#define SZEL 78 //fordításkor SZEL 78-al lesz egyenlő
#define MAG 22  //fordításkor MAG 22-vel lesz egyenlő

int putX(int x,int y)     //putX függvény
{
  int i;

  for(i=0;i<x;i++)  // a függőleges koordinátát határozza meg
    printf("\n");

  for(i=0;i<y;i++)  // a vízszintes koordinátát határozza meg
    printf(" ");

  printf("o\n");    // a Labda

  return 0;
}

int main()
{
  int x=0,y=0;

  while(1)
  {
    system("clear"); // Ez törli az elöző labdát az új kiírása előt
    putX(abs(MAG-(x++%(MAG*2))),abs(SZEL-(y++%(SZEL*2)))); // ez határoza ↩
       meg a labda pozicióját amihez a putX függvényt hivja meg
```

```
    usleep(50000); // a labda sebességét határoza meg a program altatásával
  }

  return 0;
}
```

2.5. Szóhossz és a Linus Torvalds féle BogoMIPS

Olyan programot kellet írnunk amely megadja hogy hány biten tárolunk egy adott integer számot.Ezt a bitek shiftelésével oldotuk meg egy while ciklusban ami addig fut mig el nem éri a nullát.A ciklus minden egyes periodusában növeljük az n-t ami a végén a bitek számat fogja megadni.Ennek megvalosítását lentebb láthatjuk.

```
#include <stdio.h>
int main()
{
  int a=1;
  int n=1;
  while((a<<=1))
  {
    n+=1;
  }
  printf("Megoldas:%d%s",n,"\n");
}
```

2.6. Helló, Google!

A feladatunk az volt, hogy újraalkosuk azt a page rank algoritmust amire a googel épult.Az algoritmus feladata az, hogy meghatározza az oldalak presztízsét és az alapján rangsorolja azokat.Egy megadott oldal presztízsét úgy határozza meg, hogy összeszámolja azt, hogy más oldalakról hány link mutat a mi altalunk megadott oldalra.De az oldal pontszámába az is beleszámit, hogy a mi oldalunkra mutató weboldalakra hány link mutat.Miután a programunk végzet az összes megadot oldalal a kapott presztízs pontokat rangsorolja majd kiírja a standard outputra.

PageRank:

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

void kiir (double tomb[], int db) //A kiir függvény felel a kiíratásért.Az ↩
   int db adja meg, hogy hányszor fut le a fügvényben lévő ciklus.
{
  int i;
```

```
  for (i=0; i<db; i++)
    printf("PageRank [%d]: %lf\n", i, tomb[i]);
}

double tavolsag(double pagerank[],double pagerank_temp[],int db)
{
  double tav = 0.0;
  int i;
  for(i=0;i<db;i++)
    if((pagerank[i] - pagerank_temp[i])<0)
    {
      tav +=(-1*(pagerank[i] - pagerank_temp[i]));
    }
    else
    {
      tav +=(pagerank[i] - pagerank_temp[i]);
    }
  return tav;
}

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

  double PR[4] = {0.0, 0.0, 0.0, 0.0}; //Ebben lesz benne a végeredmény
  double PRv[4] = {1.0 / 4.0, 1.0 / 4.0, 1.0 / 4.0, 1.0 / 4.0}; // Az ↩
     oldatak presztízs

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

  for (;;)
  {
    for(i=0;i<4;i++)
      PR[i] = PRv[i]; //Átadja PRv-t a PR-nek
    for (i=0;i<4;i++)
    {
      double temp=0;
      for (j=0;j<4;j++)
        temp+=L[i][j]*PR[j]; //Az L mátrixot összeszoroza a PR vektorral.
      PRv[i]=temp;
    }

    if ( tavolsag(PR,PRv, 4) < 0.00001) //A tavlság függvényt meghívva ↩
       határoza meg, hogy mikor áll le a ciklus.
```

```
    break;
  }
  kiir (PR,4);  //Meghívjuk a kiir függvényt a kiiratáshoz
  return 0;
}
```

2.7. 100 éves a Brun tétel

A feladatunk a Brun-tétel demonstrálása volt.A Brun tétel a prímszámokkal kapcsolatos és nélkülözhetetlen tudni, hogy mik is azok a prím számok és hogy melyeket nevezzük ikerprimeknek. Prímszámoknak nevezzük azokat a számokat amelyeknek pontosan két osztolyuk van, az egy és önmaga.Minden szám felírható prímszámok szorzataira és Euklidész Elemek című munkája óta azt is tudjuk, hogy végtelen prímszám létezik. Ami viszontv még nincs bebizonyítva, hogy az ikerprímek - azok a prím számpárok amelyek különbsége pontosan kettő - száma is végtelen-e. Ezekhez az ikerprimekhez kapcsolódik a Brun-tétel ami azt mondja ki, hogy az ikerprimek reciprokösszege egy Brun-konstans névvel ellátot értékhez konvergál.Ennek demonstrálására készült az alábbi R kód.

```
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")
```

2.8. A Monty Hall probléma

A Monty Hall-probléma egy valószínüségi paradoxon.Ez a paradoxon egy amerikai vetélkedőn alapul, amiben a játékosnak három ajtó közül kellett választania.Két ajtó mögött egy-egy kecske volt a harmadik mögött pedig egy autó.A játékosnak rá kellett mutatnia az egyik ajtóra.Ezután a műsorvezető a másik két ajtó közül kinyitota az egyiket ami mögött a kecske volt (a műsorvezető tudja, hogy melyik ajtó mögöt mi található) és megkérdezte a játékost, hogy akar-e változtatni a döntésén.A Monty Hall-proléma ezen kérdésből jött létre, hogy érdemes-e a játékosnak változtatni vagy sem, illetve számít-e.Egyszerű valószínűségszámítással megmutattható, hogy igen, számít mivel elsőnek 3 ajtó közül választhatunk így annak az esélye,

hogy az autót választjuk 1/3, annak pedig, hogy kecskét 2/3.Miután a műsorvezető kinyitotta az egyik ajtót annak a valószínűsége, hogy az általunk választott ajtó mögött van a kocsi továbbra is 1/3.Ezen a ponton viszont a másik két ajtó közül az egyik kinyílt ezért a másik csukott ajtó mögöt 2/3 valószínüséggel a kocsi van.Ez azonban annyira ellentmond a józan észnek, hogy a problémát paradoxonnak tekintjük és ennek a szemléltetésére készült az alábbi kód R-ben

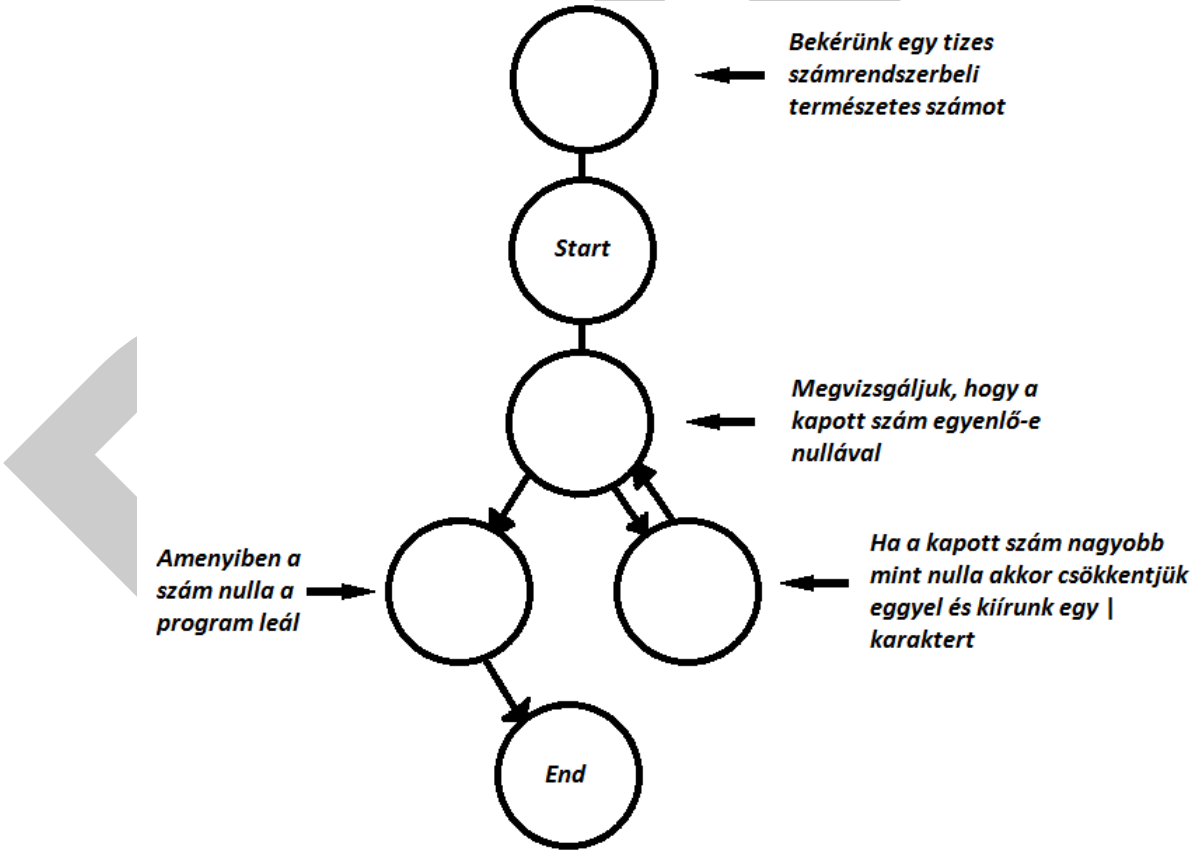
```
kiserletek_szama=10000000
kiserlet = sample(1:3, kiserletek_szama, replace=T)
jatekos = sample(1:3, kiserletek_szama, replace=T)
musorvezeto=vector(length = kiserletek_szama)

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)
```

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Az unáris számrendszer a legegyszerübb számrendszer.Csak természetes számokat lehet velük ábrázolni.Ha például az 5 számot akarjuk unárisban ábrázólni akkor elöször is be kell vezetnünk egy szimbólumont amivel az 1 számot foja jelenteni.Nekünk most ez a | karakter lesz.Szóval az 5 unáris számrendszerbe így fog kinézni: |||||.A mi feladatunk az, hogy irjunk egy olyan turing gépet ami egy decimális számot unárisá alakit.Ennek a programnak az álapotmenet gráfja lejjebb látható.



3.2. Az $a^nb^nc^n$ nyelv nem környezetfüggetlen

Ebben a feladatban két környezetfüggő generatív grammatikát kellett bemutatnunk ami az $a^nb^nc^n$ nyelvet generálja.A generatív nyelvtan elméletét Noam Chomsky amerikai nyelvész alkota meg.Chomsky a generativ nyelveket négy csoportra osztota, amiből mi a környezetfüggő nyelvtanokkal fogunk foglalkozni.A környezetfüggetlen nyelvtanokkal ellentétben itt a képzési szabályok mindkét oldalán állhatnak terminális szimbólumok. A szabályokat a -> jellel lehet megadni de ez a lentebbi megoldásunkból is látható.

```
S, X, Y "változók"
a, b, c "konstansok"
S -> abc, S -> aXbc, Xb -> bX, Xc -> Ybcc, bY -> Yb, aY -> aaX, aY -> aa
S-ből indulunk ki
-----------------
S (S -> aXbc)
aXbc (Xb -> bX)
abXc (Xc -> Ybcc)
abYbcc (bY -> Yb)
aYbbcc (aY - aa)
aabbcc
-----------------
S (S -> aXbc)
aXbc (Xb -> bX)
abXc (Xc -> Ybcc)
abYbcc (bY -> Yb)
aYbbcc (aY -> aaX)
aaXbbcc (Xb -> bX)
aabXbcc (Xb -> bX)
aabbXcc (Xc -> Ybcc)
aabbYbccc (bY -> Yb)
aabYbbccc (bY -> Yb)
aaYbbbccc (aY -> aa)
aaabbbccc
```

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

```
aaaabCBcB (cB -> Bc)
aaaabCBBc (CB -> bCc)
aaaabbCcBc (cB -> Bc)
aaaabbCBcc (CB -> bCc)
aaaabbbCccc (C -> bc)
aaaabbbbcccc
```

3.3. Hivatkozási nyelv

A C egy általános célú programozási nyelv és mint minden nyelv a C is változik, bővűl bizonyos dolgokkal.Így ennek következtében előfordulhat az, hogy egy régebbi szabványal nem fog lefordulni a programunk.Erre példa az alábbi for ciklus amit ha c89 szabvánnyal probálunk lefordítani (ezt a -std=c89 kapcsolóval tudjuk megtenni) akkor egy hibaüzeneten belül a fordító közli velünk, hogy mely szabványokkal tudjuk lefordítani a programunkat.

```
//gcc fordule.c -o teszt -std=c89 -el nem fordul le
//gcc fordule.c -o teszt le fog fordulni
int main()
{
    for(int i; i>0; ++i)
    {}
}
```

3.4. Saját lexikális elemző

A feladat az volt, hogy írjunk egy olyan lexikális ellemzőt ami felismeri a bemeneten megjelenő valós számokat és megszámolja azokat majd kiírja, hogy pontosan hány darab számot tartalmazot a szöveg.A feladat érdekesége az volt, hogy a lexer lexikális elemzőjét használjuk.Ezzel is szokva, hogy munkánk során gyakran fogjuk más programjait használni, azaz "óriások válán állni".A kód maga azt csinálja, hogy bekér egy szöveget, majd ha ütünk egy entert akkor láthatjuk, hogy a szövegben lévő számokat kiírja először stringként majd float formátumban.Amikor ki akarunk lépni a programból, ez ctrl+D lenyomásaval tehetjük meg, akkor ki fogja írni, hogy a szövegben pontosan hány valós számot találtunk.

```
%{/*
Forditas:
$ lex -o realnumber.c realnumber.l

Futtatas:
$ gcc realnumber.c -o realnumber -lfl
(kilépés az input vége, azaz Ctrl+D)
*/
#include <stdio.h>
int realnumbers = 0;
%}
digit   [0-9]
```

```
%%
{digit}*(\.{digit}+)?   {++realnumbers;
    printf("[realnum=%s %f]", yytext, atof(yytext));}
%%
int
main ()
{
 yylex ();
 printf("The number of real numbers is %d\n", realnumbers);
 return 0;
}
```

3.5. l33t.l

A leet egy olyan írásmód amely szavakban egyes betüket átír más karakterekre (pl.haxor - h4x0r).Amellet, hogy az így leírt szavak sokkal rajosabbak, még hasznuk is van mivel általába az online világban egyre gyakraban megjelenő AI szűrők nincsenek felkészitve az e módon leírt szavakra.A program hossza ellenére egyáltalán nem bonyolult.Elsőnek szokásosan includeoljuk az includeolnivalokat majd definiáljuk a L337SIZE-ot.Ezután bevezetünk egy c változót egy leet nevü négy elemű tömböt és megadjuk l337d1c7 tömböt amely megadja hogy mely karaktereket mire cserélhetunk ki.A program végső része pedig egy for ciklus ami végig megy a kapott szón és if-ek segítségével eldönti, hogy mit mire cseréljen.

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

Futtatas:
$ gcc l337d1c7.c -o l337d1c7 -lfl
(kilépés az input vége, azaz Ctrl+D)
*/
%{
  #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;
}
```

3.6. A források olvasása

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

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

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

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

i.

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

A legelső kóddal ellentétben a SIGINT jel kezelése nem let figyelmen kivül hagyva.Ezt leszámitva a két kód megegyezik de ez az apró külömbség is elegendő ahhoz, hogy a két kód egymás ellentetjei legyenek.

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

Ez egy for ciklus ami nullától indul és egyesével léptetve 5-ig tart.A ++i miat az i az elött fog eggyel nőni, hogy bármilyen müvelet végrehajtódna.

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

Ez egy for ciklus ami nullától indul és egyesével léptetve 5-ig tart.A i++ miat az i az után fog eggyel nőni, hogy bármilyen művelet végrehajtódna.

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

Itt szintén egy for ciklust láthatunk ami egy tömbben akarjuk 0-tól 4-ig tárolni a számokat.De ez a megvalósitás ilyen formán kerülendő mivel a tomb[i] = i++ első alkalommal nem kerül végrehajtásra a tömb 0 indexű eleme valamilyen memória szemét lesz.

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

Ez egy feltételes for ciklus ahol i kisebb mint n és *d++ megegyezik *s++ -al.Visszont ez a feltétel rosszul van megírva mivel a (*d++=*s++) nem egy logikai értéket fog visszaadni így az and logikai művelet nem lesz értelmezhrtő.

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

A printf az f() függvény értékét adja vissza amelynek paraméterei sorrendje nincs megadva.

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

Két számot iratunk ki az egyik f() által visszadot érték és az a értéke.

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

A printf, két számot, az a változót és f() függvény által, a memoriacímből meghatározott értéket írja ki.

3.7. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

\text = szöveg kiíratása

\forall = univerzális kvantor

\exists = egzisztenciális kvantor

\supset = implikáció

\wedge = konjunkció

\neg = negáció

\vee = diszjunkció

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

    Végtelen sok prímszám létezik

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

    Végtelen sok ikerprím létezik

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

    Véges sok prímszám létezik

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

    Véges sok prímszám létezik
```

3.8. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

- ```
  int a;         // integer típusú változó bevezetése
  ```
- ```
  int *b = &a;        // egy pointer ami a memoriacímére mutatt
  ```
- ```
  int &r = a;         //referencia változó ami megkapja az a értékét
  ```
- ```
  int c[5];       // 5 elemű tömb ami integer értékeket tárol
  ```
- ```
  int (&tr)[5] = c;       // egészek töbjének referenciája
  ```
- ```
  int *d[5];          //egészre mutató mutatók tömbje
  ```
- ```
  int *h ();          // egészre mutató mutatót visszaadó függvény
  ```
- ```
  int *(*l) ();       //egészre mutató mutatót visszaadó függvényre mutató  ↩
     mutató
  ```

- ```
  int (*v (int c)) (int a, int b)        // egészet visszaadó és két  ↩
     egészet kapó függvényre mutató mutatót visszaadó, egészet kapó  ↩
     függvény
  ```

- ```
  int (*(*z) (int)) (int, int);      //függvénymutató egy egészet visszaadó ↩
      és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó ↩
      függvényre
  ```

4. fejezet

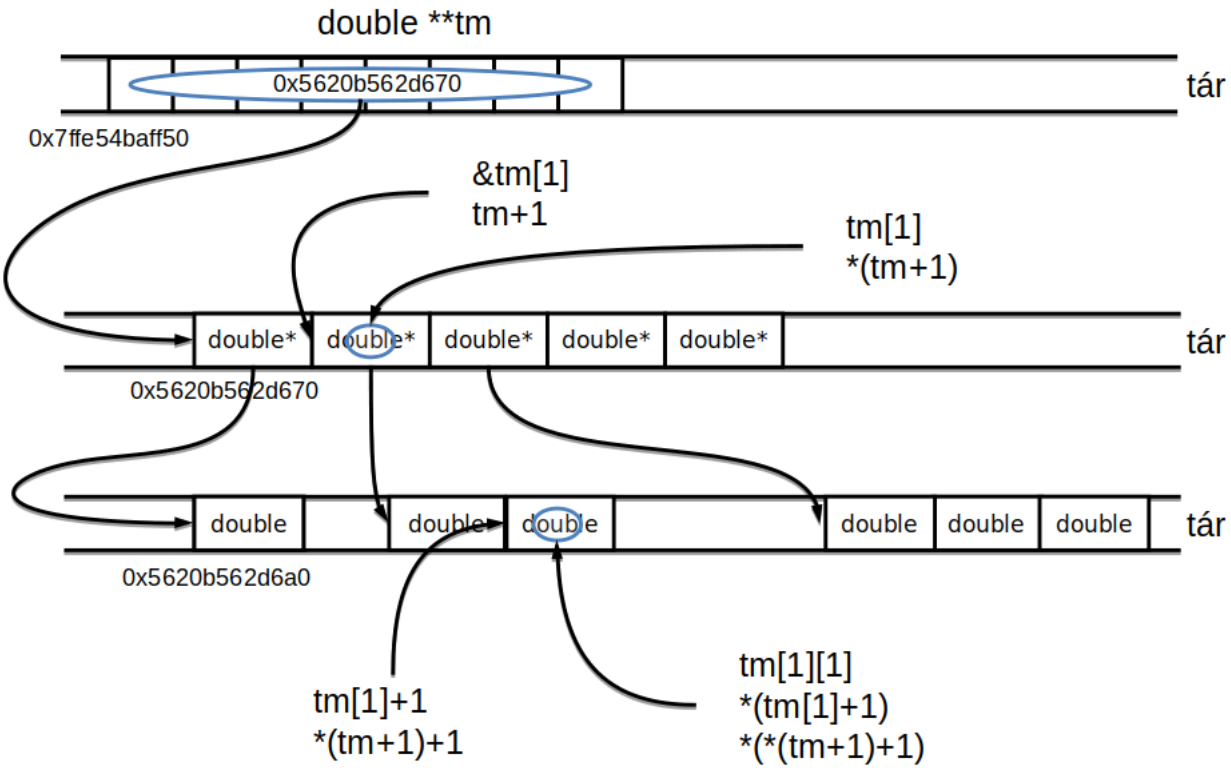
Helló, Caesar!

4.1. Double ** háromszögmátrix

Ebben a feladatban háromszögmátrixokkal fogunk foglalkozni ezért fontos letisztázni, hogy mi is az a mátrix és mikor beszélünk háromszögmátrixról.A mátrixok m darab sorral és n darab oszloppal rendelkező táblázatok.Ha egy mátrix négyzetes, azaz a sorok és oszlopok száma megegyezik és főátlója alatt vagy felett csupa 0 elem található, akkor alsó vagy felső háromszög mátrixról beszélünk. Az alsó háromszögmátrix elemeit sorfolytonosan bejárva el tudjuk helyezni egy tömbben vagy vektorban. A `malloc` függvénnyel képesek vagunk lefoglalni a dinamikus területen egy, addot méretű területet,amit paraméterként kapunk meg.Végül a lefoglalt területet a `free` függvény felszabadítjuk.

```
#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");
    }

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

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

    free (tm);

    return 0;
}
```



4.2. C EXOR titkosító

A feladatunk az volt, hogy készitsönk egy EXOR-os titkositót C-ben.A program maga egy fáljból kapott szöveget fog titkositani amit aztán kiír egy cél fáljba.A kódot forditás után(gcc titkosito.c -o titkosito) úgy lehet futatni Linux terminálban, hogy a következő sort begépeljük:

```
./titkosito kód < titkositandó.szöveg > titkositot.szöveg
```

A futtáshoz szükséges parancssorban észrevehetjük, hogy szükségünk lesz egy bemeneti fájlra ami tartalmaza a titkositani kívánt szöveget, egy kimeneti fájlra ahova majd a titkositot szöveg fog kerülni illetve egy kódra ami alapján titkosítani fogunk.Az általunk megadott kóddal úgy fog titkosjtani, hogy a kód bitjeit kizáró vaggyal (EXOR-al) összeküti. Az így létrejőt szöveget pedig kiírja a megadott output fájlunkba.

A kódban láthatjuk, hogy a szokásos includeok után láthatunk két definet az egyik a maximális kulcsméretett a másik a buffer méretét rogzíti.A main függvényben láthatunk pár változó deklarálást illetve egy while ciklust. Utóbbinak a feladata, hogy a kapott szöveg bájtjait a kulcsal elshiftelje.Az így kapott szöveget végül kiírjuk egy fájlba.

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

#define MAX_KULCS 100
#define BUFFER_MERET 256


int
```

```
main (int argc, char **argv)
{

  char kulcs[MAX_KULCS];
  char buffer[BUFFER_MERET];

  int kulcs_index = 0;
  int olvasott_bajtok = 0;

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

  while ((olvasott_bajtok = read (0, (void *) buffer, BUFFER_MERET)))
    {

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

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

  }

      write (1, buffer, olvasott_bajtok);

    }
}
```

4.3. Java EXOR titkosító

Az elöző feladatban megírt C kizáro vagyos titkositonkat át fogjuk írni java-ba.A program működési elve úgyan az, azt leszámítva, hogy itt a bemeneti fájl helyett a standard inputon keresztül visszük be a titkositandó szöveget.

A kód attekintése után láthatjuk, hogy itt is egy kulccsal fogjuk bájtonként elshiftelni a szöveget és így fog létrejönni a titkosítot szövegünk. Ez ugy fogjuk megtenni, hogy inditunk egy while ciklust ami adig fog tartani amedig az éppen bejövő szövegen végig nem ment bájtonként.

```
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

Az elözö két feladatban titkosítotunk és most a titkositot fájlunkat fogjuk feltörni.A mi EXOR törőnk egy öt, az angol ábc betüiből álló kódot fogja feltőrni illetve dekodolja a szöveget. A program végigmegy a kódolt szövegen és egymásba ágyazot ciklusokkal legyártja az összes lehetséges kulcsot és megprobálja dekódolni a szöveget.A tiszta lehet függvény magyar nyelvben gyakran előforduló szavakat megkeresi és azok alapján próbálja megtippelni, hogy a szöveg tiszta-e.A program működésének részetesebb elemzése a kódban kommentek formájában található meg.

```
#define MAX_TITKOS 4096
#define OLVASAS_BUFFER 256
#define KULCS_MERET 5
#define _GNU_SOURCE
```

```c
#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;
  char kod[28]= {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o ↩
     ','p','q','r','s','t','u','v','w','x','y','z'};

  // 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 <= 28; ++ii)
    for (int ji = 0; ji <= 28; ++ji)
      for (int ki = 0; ki <= 28; ++ki)
        for (int li = 0; li <= 28; ++li)
          for (int mi = 0; mi <= 28; ++mi)
      {
        kulcs[0] = kod[ii];
        kulcs[1] = kod[ji];
        kulcs[2] = kod[ki];
        kulcs[3] = kod[li];
        kulcs[4] = kod[mi];

        if (exor_tores (kulcs, KULCS_MERET, titkos, p - titkos))
      {
          printf
      ("Kulcs: [%c%c%c%c%c]\nTiszta szoveg: [%s]\n",
       kod[ii], kod[ji], kod[ki], kod[li], kod[mi], titkos);
        return 0;
      }
```

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

  return 0;
}
```

4.5. Neurális OR, AND és EXOR kapu

Az emberi agyban található neuronok hálozata felel az információk feldolgozásáért.A neuron felelős az elektromos jelek fogadásáért, feldolgozásáért és továbbadásáért.Neurális hálonak nevezzük azt az információfeldolgozó eszközt, amely nagyszámú, hasonló típussal rendelkező adat feldolgozására képes.Illetve tanulási algoritmussal is rendelkezik így képes előhívni a korábban megtanult információkat.Ezt fogjuk tapasztalni a lentebb található R kódban is.A program azt fogja csinálni, hogy az általunk megadot szabályok alapján elkezdi megtanulni az alap logikai műveleteket.

```
library(neuralnet)

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

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

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

plot(nn.or)

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


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

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

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

plot(nn.orand)

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

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

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

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

plot(nn.exor)

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



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

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

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

plot(nn.exor)

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

4.6. Hiba-visszaterjesztéses perceptron

Ebben a feladatban továbbrai is a neurális hálokkal, pontosaban a percetronokkal fogunk foglalkozni.Ez az algoritmus a számítógépnek megtanitja a bináris osztályzást. A bináris osztályzást legegyszerübben úgy érthetjük meg ha elképzelünk egy vonalat ami fölött fehér, alatta pedig fekete pontok vannak.Ha perceptronnak megadok egy fehéret és egy feketét onnantól kezdve el fogja tudni dönteni,hogy az adottpont fehér vagy fekete-e.Ezt az osztályzást azért nevezzük bináris osztályzásnak mivel vannak a vonal feletti és a vonal alatti pontok, tehát két választási lehetőségünk van.A mi általunk használ program egy általunk megadot képen fogja végrehajtani a bináris osztályozást.

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

5. fejezet

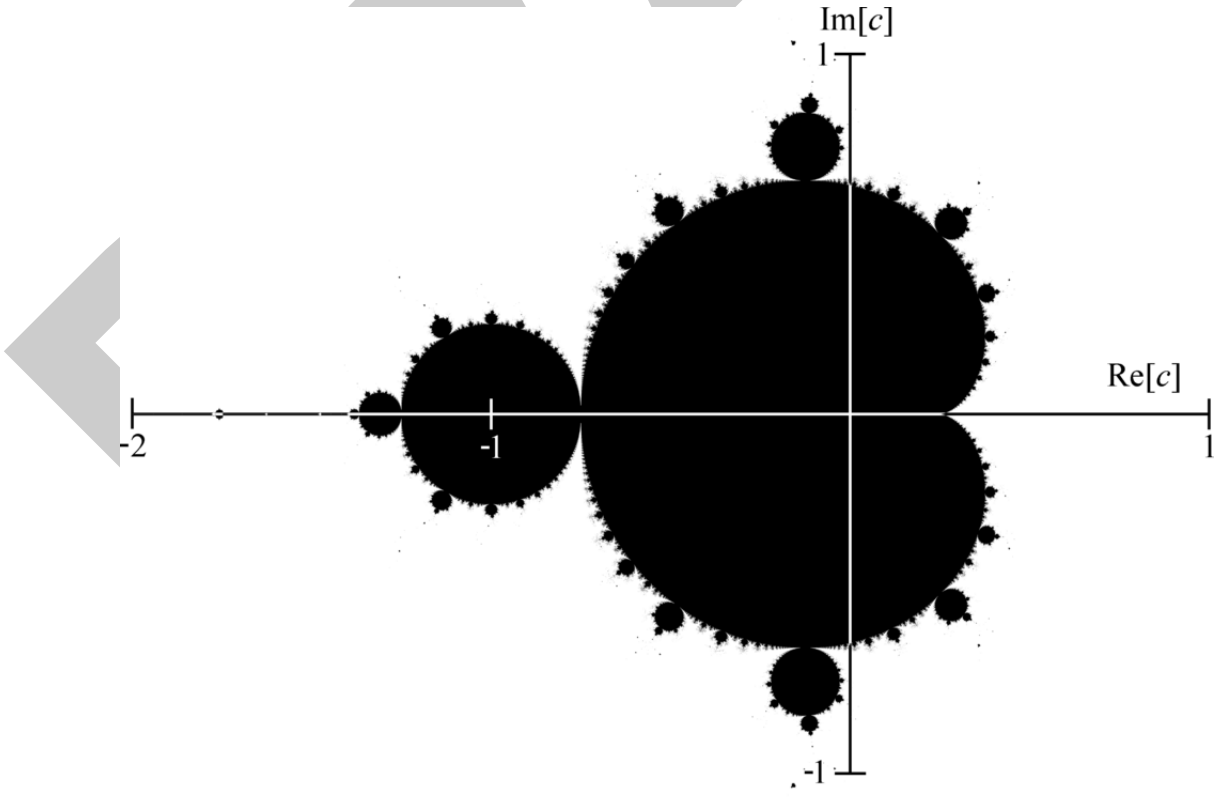
Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

A Mandelbrot halmazt Benoit Mandelbrot fedezte fel komplex számsíkon.A komplex számok azok a számok, amelyek megoldást jelenetenek az olyan gyökös kifejezések meghatározásában amikor a gyök alatt negatív szám található.Ezeket a számokat az imaginárius egység segítségével fejezük ki.Imaginárius egységnek nevezzük azt a komplex számot amelynek négyzete -1.A Mandelbrot-halmaz azokból a komplex számokból áll melyekre az alábbi rekurzív sorozat:

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

nem tart végtelenbe, azaz abszolút értékében korlátos.A Mandelbrot-halmaz a komplex számsíkon ábrázolva, egy nevezetes fraktálalakzat adodik.



A Mandelbrot-halmazt kirajzóló program:

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

#define MERET 600
#define ITER_HAT 32000

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

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

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

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

```
            {
                // z_{n+1} = z_n * z_n + c
                ujreZ = reZ * reZ - imZ * imZ + reC;
                ujimZ = 2 * reZ * imZ + imC;
                reZ = ujreZ;
                imZ = ujimZ;

                ++iteracio;

            }

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

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

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

}

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

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

    int kepadat[MERET][MERET];

    mandel(kepadat);

    png::image < png::rgb_pixel > kep (MERET, MERET);

    for (int j = 0; j < MERET; ++j)
    {
        //sor = j;
        for (int k = 0; k < MERET; ++k)
        {
            kep.set_pixel (k, j,
                           png::rgb_pixel (255 -
                                           (255 * kepadat[j][k]) / ITER_HAT ↩
                                              ,
                                           255 -
```

```
                                                (255 * kepadat[j][k]) / ITER_HAT ↩
                                                   ,
                                                255 -
                                                (255 * kepadat[j][k]) / ITER_HAT ↩
                                                   ));
        }
    }

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

}
```

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

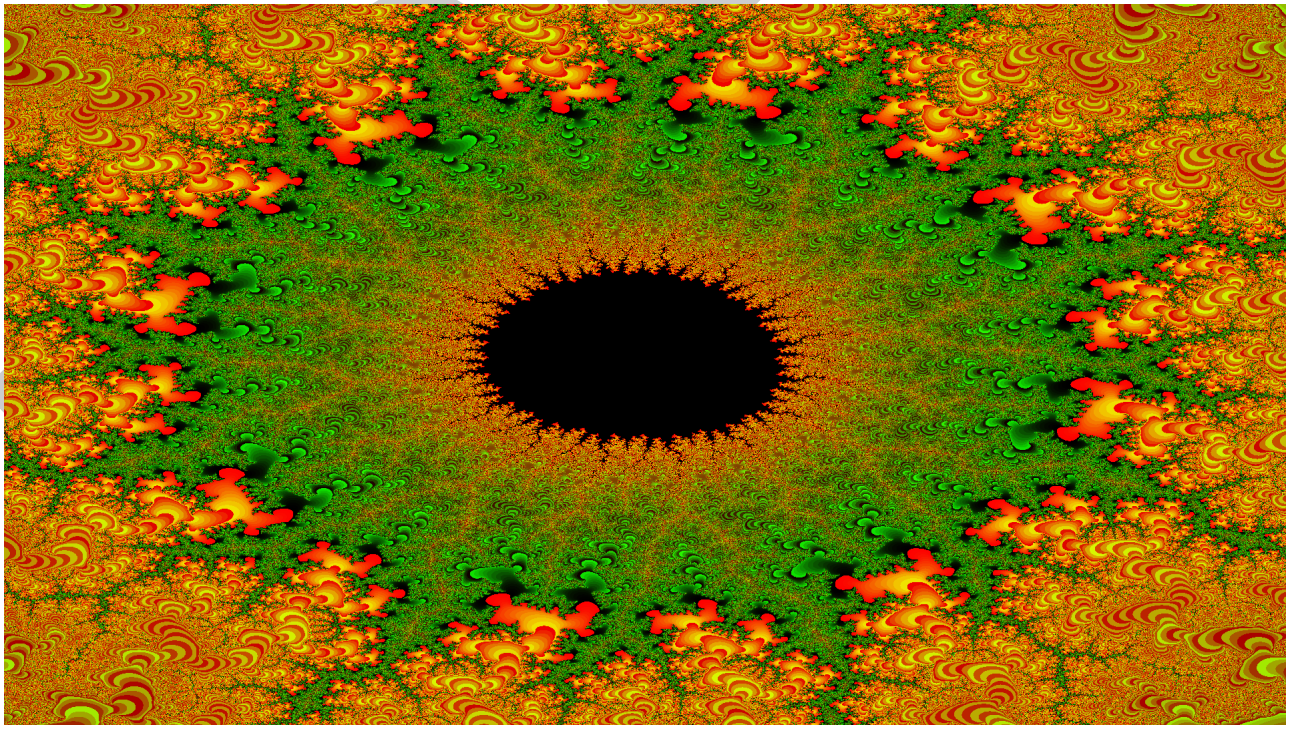
Ebben a feladatban egy C++ programot írtunk amely meghatározza a Mandelbrot-halmazt az std::complex osztály segítségével.A komplex számmolkkal való számításhoz szükséges a #include <complex> beinclu-deolása.A program gyakorlati megvalósítása lejebb található.

```
// Forditas:
// g++ 3.1.2.cpp -lpng -O3 -o 3.1.2
// Futtatas:
// ./3.1.2 mandel.png 1920 1080 2040  ↩
   -0.01947381057309366392260585598705802112818  ↩
   -0.01947381057254134184564264842265401966 87  ↩
   0.79850575693382686015553417746559716761 11  ↩
   0.79850575693437919611028519284445792436 6
// ./3.1.2 mandel.png 1920 1080 1020  ↩
   0.41276554182095892553405747094075195491 31  ↩
   0.41276554182458180530801428176346234977 25  ↩
   0.21353870517687464913869632709975121542 81  ↩
   0.21353870518049752891265313792246161028 74
// Nyomtatas:
// a2ps 3.1.2.cpp -o 3.1.2.cpp.pdf -1 --line-numbers=1  --left-footer=" ↩
   BATF41 HAXOR STR34M" --right-footer="https://bhaxor.blog.hu/" --pro= ↩
   color
// ps2pdf 3.1.2.cpp.pdf 3.1.2.cpp.pdf.pdf

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

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

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

if ( argc == 9 )
  {
    szelesseg = atoi ( argv[2] );
    magassag =  atoi ( argv[3] );
    iteraciosHatar =  atoi ( argv[4] );
    a = atof ( argv[5] );
    b = atof ( argv[6] );
    c = atof ( argv[7] );
    d = atof ( argv[8] );
  }
else
  {
    std::cout << "Hasznalat: ./3.1.2 fajlnev szelesseg magassag n a b c d ↩
       " << std::endl;
    return -1;
  }

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

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

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

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

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

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

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

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

```
          iteracio = 0;

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

              ++iteracio;
            }

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

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

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

}
```

Futatás után, a megadott számoktól függően, ezt vagy egy ehhez hasonló képet kapunk:



5.3. Biomorfok

A biomorfokat Clifford Pickover fedezte fel, amikor a Julia halmazokat rajzoló programjában hiba lépet fel.A különbség a Julia és a Mandelbrot halmazok között, hogy utóbiban a c változó.Ennek egy programban való megvalósításához az elöző feladatban elkészített programot átírni mivel eleje teljesen megegyezik igy elegendő csak a végét változtatnunk.Ebben az esetben a felhasználótól már a cc konstanst és a küszöbszámot kérjük be.

```
// 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

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

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

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

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

    }
```

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

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

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

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

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

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

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

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

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

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

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

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

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

}
```

5.4. A Mandelbrot halmaz CUDA megvalósítása

Ebben a feladatban az Nvidia CUDA technológiáját fogjuk igénybe venni, amelyel jelentősen fel tudjuk gyorsítani a Mandelbrotos programunk kép genérálását.A technológia lényege, hogy a program futását párhuzamositjuk a CUDA magok között.Ehhez termlészetesen Nvidia grafikus kártyára van szükséges.

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

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


#define MERET 600
#define ITER_HAT 32000

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

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

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

  // c = (reC, imC) a rács csomópontjainak
  // megfelelő komplex szám
  reC = a + k * dx;
  imC = d - j * dy;
  // z_0 = 0 = (reZ, imZ)
  reZ = 0.0;
  imZ = 0.0;
  iteracio = 0;
  // z_{n+1} = z_n * z_n + c iterációk
```

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

      ++iteracio;

    }
  return iteracio;
}


/*
__global__ void
mandelkernel (int *kepadat)
{

  int j = blockIdx.x;
  int k = blockIdx.y;

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

}
*/

__global__ void
mandelkernel (int *kepadat)
{

  int tj = threadIdx.x;
  int tk = threadIdx.y;

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

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

}

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

```
{

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

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

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

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

}

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

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

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

  int kepadat[MERET][MERET];

  cudamandel (kepadat);

  png::image < png::rgb_pixel > kep (MERET, MERET);

  for (int j = 0; j < MERET; ++j)
    {
      //sor = j;
      for (int k = 0; k < MERET; ++k)
  {
    kep.set_pixel (k, j,
       png::rgb_pixel (255 -
           (255 * kepadat[j][k]) / ITER_HAT,
           255 -
           (255 * kepadat[j][k]) / ITER_HAT,
```

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

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

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

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

}
```

5.5. Mandelbrot nagyító és utazó C++ nyelven

A feladatunk az volt, hogy készitsünk egy 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! A mi programunk a QT GUI-t használja, mivel ez az egyik legnépszerübb grafikus interfésze a C++ nyelvben. Ennek a megoldását itt találhatjuk: https://github.com/AndrasIstvanRacz/Prog1/tree/-master/P_K%C3%B6nyv/source/Mandelbrot/MandelMozgatoC%2B%2B

Fordításhoz elöször fel kell telepítenünk a libqt4-dev-et (sudo apt-get install libqt4-dev).Ez követöen gépeljük be sorba a következő utasításokat:

qmake -project

qmake -mandelmozgatoc++.pro

make

Futatás:

./mandelmozgatoc++

5.6. Mandelbrot nagyító és utazó Java nyelven

Ebben a feladatban az elözö C++ programot kellet átirni Javaba.Ennek a megoldását itt találhatjuk: https://github.c AndrasIstvanRacz/Prog1/tree/master/P_K%C3%B6nyv/source/Mandelbrot/NagyitoJava

Fordításhoz elöször fel kell telepítenünk egy java fordítót amit úgy tehetünk meg, hogy linux terminálba beirjuk a következő parancsot: sudo apt-get install openjdk-8-jdk.Ez követöen gépeljük be sorba a következő utasításokat:

javac MandelbrotHalmazNagyító.java

Futatás:

java MandelbrotHalmazNagyító

```
/*
 * MandelbrotHalmazNagyító.java
 *
 * DIGIT 2005, Javat tanítok
 * Bátfai Norbert, nbatfai@inf.unideb.hu
 *
 */
/**
 * A Mandelbrot halmazt nagyító és kirajzoló osztály.
 *
 * @author Bátfai Norbert, nbatfai@inf.unideb.hu
 * @version 0.0.1
 */
public class MandelbrotHalmazNagyító extends MandelbrotHalmaz {
    /** A nagyítandó kijelölt területet bal felső sarka. */
    private int x, y;
    /** A nagyítandó kijelölt terület szélessége és magassága. */
    private int mx, my;
    /**
     * Létrehoz egy a Mandelbrot halmazt a komplex sík
     * [a,b]x[c,d] tartománya felett kiszámoló és nygítani tudó
     * <code>MandelbrotHalmazNagyító</code> objektumot.
     *
     * @param      a              a [a,b]x[c,d] tartomány a koordinátája.
     * @param      b              a [a,b]x[c,d] tartomány b koordinátája.
     * @param      c              a [a,b]x[c,d] tartomány c koordinátája.
     * @param      d              a [a,b]x[c,d] tartomány d koordinátája.
     * @param      szélesség      a halmazt tartalmazó tömb szélessége.
     * @param      iterációsHatár a számítás pontossága.
     */
    public MandelbrotHalmazNagyító(double a, double b, double c, double d,
            int szélesség, int iterációsHatár) {
        // Az ős osztály konstruktorának hívása
        super(a, b, c, d, szélesség, iterációsHatár);
        setTitle("A Mandelbrot halmaz nagyításai");
        // Egér kattintó események feldolgozása:
        addMouseListener(new java.awt.event.MouseAdapter() {
            // Egér kattintással jelöljük ki a nagyítandó területet
            // bal felső sarkát:
            public void mousePressed(java.awt.event.MouseEvent m) {
                // A nagyítandó kijelölt területet bal felső sarka:
                x = m.getX();
                y = m.getY();
                mx = 0;
                my = 0;
                repaint();
            }
```

```
            // Vonszolva kijelölünk egy területet...
            // Ha felengedjük, akkor a kijelölt terület
            // újraszámítása indul:
            public void mouseReleased(java.awt.event.MouseEvent m) {
                double dx = (MandelbrotHalmazNagyító.this.b
                        - MandelbrotHalmazNagyító.this.a)
                        /MandelbrotHalmazNagyító.this.szélesség;
                double dy = (MandelbrotHalmazNagyító.this.d
                        - MandelbrotHalmazNagyító.this.c)
                        /MandelbrotHalmazNagyító.this.magasság;
                // Az új Mandelbrot nagyító objektum elkészítése:
                new MandelbrotHalmazNagyító(MandelbrotHalmazNagyító.this.a+ ↩
                   x*dx,
                        MandelbrotHalmazNagyító.this.a+x*dx+mx*dx,
                        MandelbrotHalmazNagyító.this.d-y*dy-my*dy,
                        MandelbrotHalmazNagyító.this.d-y*dy,
                        600,
                        MandelbrotHalmazNagyító.this.iterációsHatár);
            }
        });
        // Egér mozgás események feldolgozása:
        addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
            // Vonszolással jelöljük ki a négyzetet:
            public void mouseDragged(java.awt.event.MouseEvent m) {
                // A nagyítandó kijelölt terület szélessége és magassága:
                mx = m.getX() - x;
                my = m.getY() - y;
                repaint();
            }
        });
    }
    /**
     * Pillanatfelvételek készítése.
     */
    public void pillanatfelvétel() {
        // Az elmentendő kép elkészítése:
        java.awt.image.BufferedImage mentKép =
                new java.awt.image.BufferedImage(szélesség, magasság,
                java.awt.image.BufferedImage.TYPE_INT_RGB);
        java.awt.Graphics g = mentKép.getGraphics();
        g.drawImage(kép, 0, 0, this);
        g.setColor(java.awt.Color.BLUE);
        g.drawString("a=" + a, 10, 15);
        g.drawString("b=" + b, 10, 30);
        g.drawString("c=" + c, 10, 45);
        g.drawString("d=" + d, 10, 60);
        g.drawString("n=" + iterációsHatár, 10, 75);
        if(számításFut) {
            g.setColor(java.awt.Color.RED);
            g.drawLine(0, sor, getWidth(), sor);
```

```
        }
        g.setColor(java.awt.Color.GREEN);
        g.drawRect(x, y, mx, my);
        g.dispose();
        // A pillanatfelvétel képfájl nevének képzése:
        StringBuffer sb = new StringBuffer();
        sb = sb.delete(0, sb.length());
        sb.append("MandelbrotHalmazNagyitas_");
        sb.append(++pillanatfelvételSzámláló);
        sb.append("_");
        // A fájl nevébe belevesszük, hogy melyik tartományban
        // találtuk a halmazt:
        sb.append(a);
        sb.append("_");
        sb.append(b);
        sb.append("_");
        sb.append(c);
        sb.append("_");
        sb.append(d);
        sb.append(".png");
        // png formátumú képet mentünk
        try {
            javax.imageio.ImageIO.write(mentKép, "png",
                    new java.io.File(sb.toString()));
        } catch(java.io.IOException e) {
            e.printStackTrace();
        }
    }
    /**
     * A nagyítandó kijelölt területet jelző négyzet kirajzolása.
     */
    public void paint(java.awt.Graphics g) {
        // A Mandelbrot halmaz kirajzolása
         g.drawImage(kép, 0, 0, this);
        // Ha éppen fut a számítás, akkor egy vörös
        // vonallal jelöljük, hogy melyik sorban tart:
        if(számításFut) {
            g.setColor(java.awt.Color.RED);
            g.drawLine(0, sor, getWidth(), sor);
        }
        // A jelző négyzet kirajzolása:
        g.setColor(java.awt.Color.GREEN);
        g.drawRect(x, y, mx, my);
    }
    /**
     * Példányosít egy Mandelbrot halmazt nagyító obektumot.
     */
    public static void main(String[] args) {
        // A kiinduló halmazt a komplex sík [-2.0, .7]x[-1.35, 1.35]
        // tartományában keressük egy 600x600-as hálóval és az
```

```
        // aktuális nagyítási pontossággal:
        new MandelbrotHalmazNagyító(-2.0, .7, -1.35, 1.35, 600, 255);
    }
}
```

6. fejezet

Helló, Welch!

6.1. Első osztályom

Ebben a feladatban az volt a dolgunk, hogy kódoljuk le a polártranszformációs algoritmust C++-ban illetve javában. A C++-os verzió három fáljból fog állni egy main.cpp-ből,egy polargen.cpp-ből és apolargen.h-ból.

A main.cpp-ben elöször beincludeoljuk az iostreamet illetve az általunk létrehozot polargen.h-t utobbira később még viszatérunk.A main függvényben declaráljuk a PolarGen osztályú pg változót és egy 1-től 10-ig indít egy for ciklust amiben pg-re meghívja a kovetkezo függvényt és kiírja a pg-ből kapot számot.

```
#include <iostream>
#include "polargen.h"

int
main ()
{
  PolarGen pg;

  for (int i = 0; i < 10; ++i)
    std::cout << pg.kovetkezo () << std::endl;

  return 0;
}
```

A polargen.cpp-be szintén be kell includeoljuk a polargn.h-t.Később majd láthatjuk,hogy erre azért van szükség mivel polargen.h-ban lévő, PolarGen osztályban declaráljuk a kovetkezo függvényt amit itt a polargen.cpp-ben fejtünk ki.Ha a nincsTarolt igaz, azaz nincs tárolt adat akkor a viszatérítési érték r*v1 lesz.Elenkező esetben a tarolt változót fogjuk visszkapni.

```
#include "polargen.h"

double
PolarGen::kovetkezo ()
{
```

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

      double r = std::sqrt ((-2 * std::log (w)) / w);

      tarolt = r * v2;
      nincsTarolt = !nincsTarolt;

      return r * v1;
    }
  else
    {
      nincsTarolt = !nincsTarolt;
      return tarolt;
    }
}
```

Elöször beincludeoljuk a cstdlib-et, a cmath-ot és a ctime-ot majd declaráljuk a PolarGen osztályt.Deffiniáljuk a nincsTarolt logikai változót igazra, declaráljuk a kovetkezo függvényt és a tarolt double típusu változót.

```
#ifndef POLARGEN__H
#define POLARGEN__H

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

class PolarGen
{
public:
  PolarGen ()
  {
    nincsTarolt = true;
    std::srand (std::time (NULL));
  }
   ~PolarGen ()
  {
  }
  double kovetkezo ();
```

```
private:
  bool nincsTarolt;
  double tarolt;

};

#endif
```

Illetve végül itt láthatjuk a polárgen programunk javás átírását.A kód mükődési elve úgyan az mint a c++-os verziójé csak a java letisztult egyszerűsége miat sokkal tisztább és átláthatobb. Például a random szám generálása itt jelentősen egyszerubben történik meg.

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

    public PolárGenerátor() {
        nincsTárolt = true;
    }

    public double következő() {
        if (nincsTárolt) {
            double u1, u2, v1, v2, w;
            do {
                u1 = Math.random();
                u2 = Math.random();
                v1 = 2*u1-1;
                v2 = 2*u2-1;
                w = v1*v1+v2*v2;
            } while (w > 1);
            double r = Math.sqrt((-2*Math.log(w)) / w);
            tárolt = r*v2;
            nincsTárolt = !nincsTárolt;
            return r*v1;
        } else {
            nincsTárolt = !nincsTárolt;
            return tárolt;
        }
    }

    public static void main(String[] arps) {
        PolárGenerátor g = new PolárGenerátor();
        for (int i = 0; i < 10; ++i) {
            System.out.println(g.következő());
            }
        }
    }
```

6.2. LZW

A feladatunk az volt, hogy Lempel-Ziv-Welch tömöritési algoritmust - amit Terry Welch amerikai informatikus publikált 1984-ben - írjuk meg C-ben.Ez az algoritmus az adatokat egy fa strukturában ábrázolja. A binfában megadjuk a fa bal és jobb oldali mutatóját.Ez a gyökér jobb és bal gyerekére fog mutatni,majd ez után létrehozzuk a binfa típusra mutató mutatót.Ha az általunk beolvasott érték 0, akkor ez azt jelenti,hogy a fa mutatónak nincs bal oldali gyereke, ezért lefoglalunk neki helyet és nullára állítjuk az értékét.Ezután nullára állítjuk az új gyermeknek a jobb és a bal mutatóját és a fa mutatóját ráállítjuk az új gyökérre. A bal gyermek lesz a gyökér ha a fa bal gyermeke nem 0-ra mutat.A jobb oldali gyermeket 1 esetén vizsgáljuk.

```
//>gcc z.c -lm -o z        fordítása

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

typedef struct binfa
{
  int ertek;
  struct binfa *bal_nulla;
  struct binfa *jobb_egy;
} BINFA, *BINFA_PTR;      // definiáljuk a binfa típust

BINFA_PTR
uj_elem ()
{
  BINFA_PTR p;

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

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

int
main ()
{
  int argc;
  char **argv;
  char b;
  int egy_e;
  int i;
```

```
  unsigned char c;

  BINFA_PTR gyoker = uj_elem ();
  gyoker->ertek = '/';
  gyoker->bal_nulla = gyoker->jobb_egy = NULL;
  BINFA_PTR fa = gyoker;
  long max=0;
while (read (0, (void *) &b, sizeof(unsigned char)))
    {
        for(i=0;i<8; ++i)
        {
            egy_e= b& 0x80;
            if ((egy_e >>7)==0)
                c='1';
            else
                c='0';
        }
      if (c == '0')
    {
      if (fa->bal_nulla == NULL)
        {
          fa->bal_nulla = uj_elem ();
          fa->bal_nulla->ertek = 0;
          fa->bal_nulla->bal_nulla = fa->bal_nulla->jobb_egy = NULL;
          fa = gyoker;
        }
      else
        {
          fa = fa->bal_nulla;
        }
    }
      else
    {
      if (fa->jobb_egy == NULL)
        {
          fa->jobb_egy = uj_elem ();
          fa->jobb_egy->ertek = 1;
          fa->jobb_egy->bal_nulla = fa->jobb_egy->jobb_egy = NULL;
          fa = gyoker;
        }
      else
        {
          fa = fa->jobb_egy;
        }
    }
    }

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

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

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

  /* Átlagos águhossz kiszámítása */

  atlagosszeg = 0;
  melyseg = 0;
  atlagdb = 0;
  ratlag (gyoker);
  atlag = ((double) atlagosszeg) / atlagdb;

  /* Águhosszak szórásának kiszámítása */

  atlagosszeg = 0;
  melyseg = 0;
  atlagdb = 0;
  szorasosszeg = 0.0;

  rszoras (gyoker);

  double szoras = 0.0;

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

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

  szabadit (gyoker);
}


int atlagosszeg = 0, melyseg = 0, atlagdb = 0;

void
ratlag (BINFA_PTR fa)
{

  if (fa != NULL)
    {
      ++melyseg;
      ratlag (fa->jobb_egy);
      ratlag (fa->bal_nulla);
      --melyseg;

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

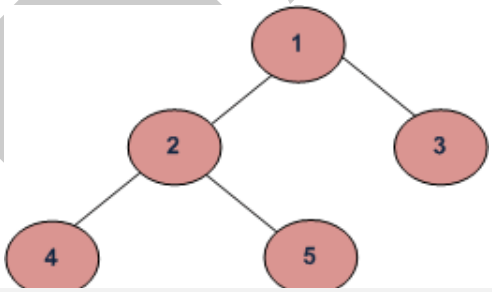
```
      ++atlagdb;
      atlagosszeg += melyseg;

    }

    }

}

double szorasosszeg = 0.0, atlag = 0.0;

void
rszoras (BINFA_PTR fa)
{

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

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

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

    }

    }

}

int max_melyseg = 0;

void
kiir (BINFA_PTR elem)
{
  if (elem != NULL)
    {
      ++melyseg;
      if (melyseg > max_melyseg);
    max_melyseg = melyseg;
      kiir (elem->jobb_egy);

      // ez a postorder bejáráshoz képest
      // 1-el nagyobb mélység, ezért -1
```

```
      for (int i = 0; i < melyseg; ++i)
    printf ("---");
      printf ("%c(%d)\n", elem->ertek < 2 ? '0' + elem->ertek : elem->ertek ↩
          ,
           melyseg - 1);
      kiir (elem->bal_nulla);
      --melyseg;
    }
}

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

6.3. Fabejárás

Egy bináris fának három bejáraási módja van: inorder, postorder, preorder.Az eredeti LZW binfa építőnk a fát inorder módon építi fel.

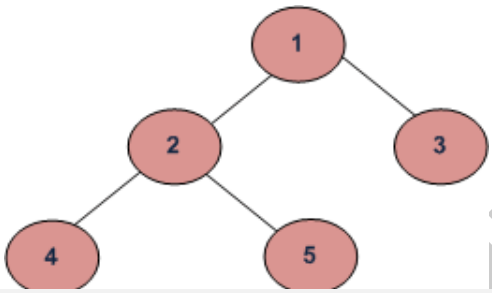
Inorder bejárás: 4, 2, 5, 1, 3




```
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;
```

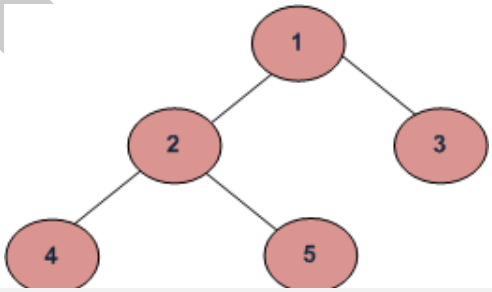
```
        }
    }
```

Postorder bejárás: 1, 2, 4, 5, 3




```
void kiir (Csomopont* elem, std::ostream& os)
    {
        if (elem != NULL)
        {
            ++melyseg;
            kiir (elem->egyesGyermek(), os);

            kiir (elem->nullasGyermek(), os);
            --melyseg;

            for (int i = 0; i < melyseg; ++i)
                os << "---";
            os << elem->getBetu() << "(" << melyseg - 2 << ")" << std::endl ↩
               ;
        }
    }
```

Preorder bejárás: 4, 5, 2, 3, 1




```
void kiir (Csomopont* elem, std::ostream& os)
    {
        // Nem létező csomóponttal nem foglalkozunk... azaz ez a rekurzió  ↩
           leállítása
        if (elem != NULL)
        {

            for (int i = 0; i < melyseg; ++i)
                os << "---";
```

```
            os << elem->getBetu() << "(" << melyseg - 2 << ")" << std::endl ↩
               ;
            ++melyseg;
            kiir (elem->egyesGyermek(), os);
        kiir (elem->nullasGyermek(), os);
            --melyseg;
        }
    }
```

6.4. Tag a gyökér

A feladatunk az volt, hogy az LZW algoritmust ültesük át egy C++ osztályba. Az osztály definíciójába beágyazzuk a fa egy csomópontjának az absztrakt jellemzését, ez lesz a beágyazott Csomopont osztály.Csak a fa részeként fogunk számolni vele, egyéb szerepet nem szánunk neki.A a << operátort tagfüggvényként túlterheljük, ezzel fogjuk a fába nyomni az elemeket.Az alapértelmezet paraméter nélküli konstruktor a '/'-el hozza létre a csomópontot.A programot futatni ezzel a sorral lehet ./lzwtree be_file -o ki_file, amenyiben valamelyik tag hiányzik a program kiírja, hogy hogyan tudjuk futatni a programot.

```
            // z3a7.cpp

#include <iostream>     // mert olvassuk a std::cin, írjuk a std::cout  ↩
   csatornákat
#include <cmath>        // mert vonunk gyököt a szóráshoz: std::sqrt
#include <fstream>      // fájlból olvasunk, írunk majd

class LZWBinFa          //itt hozzuk létre LZWBinFa osztályt
{
public:

    LZWBinFa ():fa (&gyoker)
    {
    }
    ~LZWBinFa ()
    {
        szabadit (gyoker.egyesGyermek ());
        szabadit (gyoker.nullasGyermek ());
    }

    /* Tagfüggvényként túlterheljük a << operátort amely után így  ↩
       nyomhatjuk a fába az inputot: binFa << b; ahol a b
    egy '0' vagy '1'-es betű lesz.*/

    void 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 ();
            }
        }
    }

    void kiir (void)        // Ez a függvény felel a kiíratásért.
    {

        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;
    LZWBinFa (const LZWBinFa &);
    LZWBinFa & operator= (const LZWBinFa &);

    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;
        }
    }

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;
}

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;

    while (beFile.read ((char *) &b, sizeof (unsigned char)))
        if (b == 0x0a)
            break;

    bool kommentben = false;

    while (beFile.read ((char *) &b, sizeof (unsigned char)))
    {

        if (b == 0x3e)
        {
            kommentben = true;
            continue;
        }

        if (b == 0x0a)
        {
            kommentben = false;
            continue;
        }

        if (kommentben)
            continue;

        if (b == 0x4e)
            continue;
```

```
        for (int i = 0; i < 8; ++i)
        {

            if (b & 0x80)
                binFa << '1';
            else
                binFa << '0';
            b <<= 1;
        }

    }



    kiFile << binFa;

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

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

    return 0;
}
```

6.5. Mutató a gyökér

Az elöző feladatal elentétben itt a gyökeret nem egy objektumként fogjuk kezelni.Ezt úgy fogjuk megvalósítani, hogy a famutót ráálitjuk a gyökérre a konstruktorban, amit lent hozzunk létre.Ebben az esetben a gyökér is mutató lesz.Ez azt vonja maga után, hogy ahol referencia volt a gyökér ott referencia nélkülivé kell tennünk.Ezen kívül helyet kell foglalnunk a memóriában.Szabaditáskor mivel a mutató mutatóit kell elérnünk nyilat fogunk használni.

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

class LZWBinFa
{
public:

    LZWBinFa () {
        gyoker = new Csomopont ('0');
        fa = gyoker;
```

```
    };

void 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 ();
        }
    }
}
void kiir (void)
{
    melyseg = 0;
    kiir (gyoker, std::cout);
}
void szabadit (void)
{
    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;
    LZWBinFa (const LZWBinFa &);
    LZWBinFa & operator=(const LZWBinFa &);

    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;
        }
    }

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;
}

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

    unsigned char b;
    LZWBinFa binFa;

    while (beFile.read ((char *) &b, sizeof (unsigned char))) {
        for (int i = 0; i < 8; ++i)
        {
            int egy_e = b & 0x80;
            if ((egy_e >> 7) == 1)
                binFa << '1';
            else
                binFa << '0';
            b <<= 1;
        }

    }

    kiFile << binFa;

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

    binFa.szabadit ();

    kiFile.close();
```

```
    beFile.close();

    return 0;
}
```

6.6. Mozgató szemantika

Ebben a feladatban a Tag a gyökér fejezetben használt kódhoz fogunk mozgató konstruktort és értékadást.Ennek az a lényege, hogy felesleges másolatok létrehozását nélkül képesek leszünk a binfánkat átmozgatni egyik fáljból a másikba.Ezt a programunk úgy fogja megoldani, hogy a létrehozot binfát átrakja egy másik fáljba majd az eredetit törli.Ennek a gyakorlati megvalósítása a lentebb található kód.

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

class LZWBinFa
{
public:
    LZWBinFa ()
    {
        gyoker = new Csomopont ('/');
        fa = gyoker;
    }
    ~LZWBinFa ()
    {
        if (gyoker != nullptr)
        {
            szabadit (gyoker->egyesGyermek ());
            szabadit (gyoker->nullasGyermek ());
            delete(gyoker);
        }
    }
    LZWBinFa (LZWBinFa&& eredeti)
    {
        std::cout<<"Move ctor\n";
        gyoker = nullptr;
        *this = std::move(eredeti);

    }
    LZWBinFa& operator= (LZWBinFa&& eredeti)
    {
        std::cout<<"Move assignment ctor\n";
        std::swap(gyoker, eredeti.gyoker);
```

```
        return *this;
    }

    void 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 ();
            }
        }
    }

    void kiir (void)
    {
        melyseg = 0;
        kiir (gyoker, std::cout);
    }

    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 &); //másoló konstruktor
        Csomopont & operator= (const Csomopont &);
    };

    Csomopont *fa;
    int melyseg, atlagosszeg, atlagdb;
    double szorasosszeg;
    LZWBinFa (const LZWBinFa &);
    LZWBinFa & operator= (const LZWBinFa &);

    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;
        }
    }

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));
        }
    }
}

LZWBinFa move(LZWBinFa&& eredeti)
{
    LZWBinFa b(std::move(eredeti));
    return b;
}
void
usage (void)
{
    std::cout << "Usage: lzwtree in_file -o out_file" << 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;

    while (beFile.read ((char *) &b, sizeof (unsigned char)))
        if (b == 0x0a)
            break;

    bool kommentben = false;

    while (beFile.read ((char *) &b, sizeof (unsigned char)))
    {

        if (b == 0x3e)
        {
            kommentben = true;
            continue;
        }

        if (b == 0x0a)
        {
            kommentben = false;
            continue;
        }

        if (kommentben)
            continue;

        if (b == 0x4e)
            continue;

        for (int i = 0; i < 8; ++i)
        {
            if (b & 0x80)
                binFa << '1';
            else
                binFa << '0';
            b <<= 1;
        }

    }

    kiFile << binFa;
```

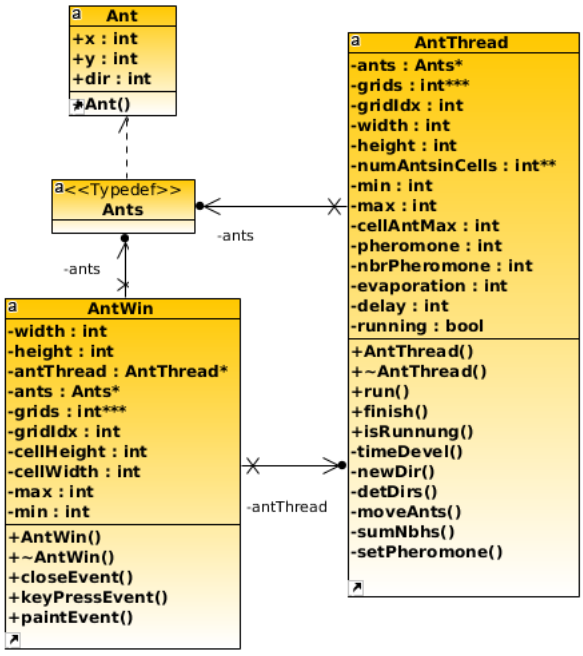
```
    kiFile << "depth = " << binFa.getMelyseg () << std::endl;
    kiFile << "mean = " << binFa.getAtlag () << std::endl;
    kiFile << "var = " << binFa.getSzoras () << std::endl;

    LZWBinFa binFa2 = std::move(binFa);

    kiFile << binFa2;

    kiFile << "depth = " << binFa2.getMelyseg () << std::endl;
    kiFile << "mean = " << binFa2.getAtlag () << std::endl;
    kiFile << "var = " << binFa2.getSzoras () << std::endl;

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

    return 0;
}
```

7. fejezet

Helló, Conway!

7.1. Hangyaszimulációk

Ebben a feladatban a hangya szimulácival fogunk foglalkozni.A szimulció, ahogy a nevéből is kitalálhatjuk a hangyákkal kapcsolatos, méghozzá a hangyák azon tulajdonságával kapcsolatos, hogy két pont között mindig megkeresik a legoptimálisabb utvonalat és miután megtalálták csak azt fogják használni. A mi programunk is ezt fogja csinálni. A mi esetunkben a kis pontok lesznek a hangyák amelyek először a különböző pontokba különböző utakon fognak eljutni.Egy kis idő után azonban észrevehetjük, hogy egy-egy pontba már csak egy vonalon fognak haladni a hangyáink.Ez azt jelenti, hogy megtalálták a legrövidebb utat az adott pontba.



Megoldás forrása: https://github.com/AndrasIstvanRacz/Prog1/tree/master/P_K%C3%B6nyv/source/Conway/Myrmecologist

7.2. Java életjáték

Ebben a feladatban a John Conway-féle életjátékot kellet megvalósitsuk egy java programban azon belül is a sikló-kilövőt.

John Conway, a Cambridge Egyetem egyik matematikusa találta ki az életjátékot.Ez matematikai szempontból az úgynevezet sejtautómaták közé tartozik (Az olyan diszkrét modelleket, amiket a számelméletben mikrostruktúrák modellezésére használnak sejtautomatáknak nevezzük).A program egy négyzetrácsot fog megjeleníteni.A négyzetrács fekete cellákat sejteknek nevezzük.Egy cellához legközelebb lévő 8 cellát a cella környezetének nevezzük.Az életjáték körökre van osztva.Az első körben elhelyezünk tetszőleges számú sejtet.Ez után a játékosnak nincs beleszólása a játékmenetbe.Egy sejttel ezután három dolog történhet:

1) A sejt túléli a kört, ha két vagy három szomszédja van.

2) A sejt elpusztul, ha kettőnél kevesebb (elszigetelődés), vagy háromnál több (túlnépesedés) szomszédja van.

3) Új sejt születik minden olyan cellában, melynek környezetében pontosan három sejt található.

Fontos, hogy ezek a változások csak kör végén következnek be.Ez azt jelenti, hogy az elhalálozók nem befolyásolják a túlélést és a születéseket illetve a születök sem befolyásolják az elhalálozókat.

Az életjáték eredményezhet speciális alakztokat (alkzatoknak nevezzük a sejtek egy halmazát), ezek közül mi a sikló-kilövővel fogunk foglalkozni.Ennek az az érdekesége, hogy négy fázis után önmagába alakul vissza miközben egy kockányit mindig elmozdul,tehát átlos mozgásal végez.Ez az alakzat a hackerek hivatalos emblémája is.

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

public Sejtautomata(int szélesség, int magasság) {
        this.szélesség = szélesség;
        this.magasság = magasság;
        rácsok[0] = new boolean[magasság][szélesség];
        rácsok[1] = new boolean[magasság][szélesség];
        rácsIndex = 0;
        rács = rácsok[rácsIndex];
        for(int i=0; i<rács.length; ++i)
            for(int j=0; j<rács[0].length; ++j)
                rács[i][j] = HALOTT;
```

```
        siklóKilövő(rács, 5, 60);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent e) {
                setVisible(false);
                System.exit(0);
        }
});

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

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

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

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

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

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

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

public int szomszédokSzáma(boolean [][] rács,
            int sor, int oszlop, boolean állapot) {
        int állapotúSzomszéd = 0;
        for(int i=-1; i<2; ++i)
            for(int j=-1; j<2; ++j)
```

```
                if(!((i==0) && (j==0))) {
            int o = oszlop + j;
            if(o < 0)
                o = szélesség-1;
            else if(o >= szélesség)
                o = 0;

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

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

        return állapotúSzomszéd;
}

public void időFejlődés() {

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

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

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

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

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

                    if(élők==3)
                        rácsUtána[i][j] = ÉLŐ;
                    else
                        rácsUtána[i][j] = HALOTT;
            }
        }
    }
        rácsIndex = (rácsIndex+1)%2;
}

public void run() {
```

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

            időFejlődés();
            repaint();
        }
}

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

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

}

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

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

        rács[y+ 3][x+ 13] = ÉLŐ;

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

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

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

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

```
        rács[y+ 7][x+ 23] = ÉLŐ;
        rács[y+ 7][x+ 24] = ÉLŐ;

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

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

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

        rács[y+ 11][x+ 25] = ÉLŐ;

}

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

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

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

7.3. Qt C++ életjáték

Ebben a feladatban is John Conway életjátékát fogjuk megvalósítani, csak ezutal C++-ban Qt gui segitségével.A progam müködési elve úgyan az, illetve a szabályok is megegyeznek az elöző feladatban lévökkel.

Megoldás forrása: https://github.com/AndrasIstvanRacz/Prog1/tree/master/P_K%C3%B6nyv/source/Conway/Gol_Qt_C%2B%2B

7.4. BrainB Benchmark

Ebben a feladatban a Brain Benchmarkkal kellett foglalkoznunk, ami az agy koncetrációs készégét méri.A Brain Benchmark meghatároza, hogy az agyunk milyen gyorsan képes reagálni, menyire képes összpontosítanni egy adott feladatra illetve teszteli a memóriát.Az általunk használt program ezt úgy teszi meg, hogy a képernyőn megjelenit kis négyzeteket bennük egy körrel.A felhasználó feladata az, hogy válaszon egyet a négyzetek közül és a benne lévö körre kattintson, és nyomva tartvva az egeret kövese azt. Eközben újabb és újabb négyzetek fognak megjelenni.Abban az esetben ha az egeret elengedjük vagy az általunk kiválasztott kört elveszitjük a négyzetek elkezdenek eltünni.A program ezen a datok alapján ki fogja számolni a reakcióidőnket.Ez az esprtolok körében eléggé elterjedt mivel azon a területen fontos, hogy az agy a lehető leggyorsabban reagáljon.

Megoldás forrása: https://github.com/AndrasIstvanRacz/Prog1/tree/master/P_K%C3%B6nyv/source/Conway/BrainB

8. fejezet

Helló, Schwarzenegger!

8.1. Szoftmax Py MNIST

Ezen feladat helyet az SMNIST-et csináltam.

8.2. Mély MNIST

Ezen feladat helyet az SMNIST-et csináltam.

8.3. Minecraft-MALMÖ

A MALMÖ project a microsoft álltal létrehozott olyan platform amely segitségével a mesterséges inteligenciával végezhetünk kutatásokat, méghozá az igen nagy népszeruséget élvező Minecraft játékon keresztül.A megoldáshoz nagy segítséget nyujtot a Malmo Platform Tutorial amit a következő címen találhatunk meg https://microsoft.github.io/malmo/0.17.0/Python_Examples/Tutorial.pdf.

A feladat elkezdése elött meg kell látogatnunka a következő linket https://github.com/Microsoft/malmo. Itt láthatjuk, hogy a Malmö telepítése elvégezhető Python Pip-el is illetve használhatjuk a pre-built verziókat is.Én a Linux pre-built verziót használtam a feladat megoldásához.Bármelyik verzió használásához szukségünk lesz az open JDK 8-ra. Ezután megnyitotuk a letöltött Malmo mappát és az abban található Minecraft file-t megnyitjuk terminálban.Majd ./launchClient.sh parancssal elinditjuk a clienst.

Mostmár, hogy fut a cliensünk elkezdhetünk foglalkozni a feladatunkkal ami az volt, hogy Steve képes legyen mozogni a neki megadot világban anélkül, hogy megakadna.Ehhez elösször ismernünk kell az alap mozgási parancsokat.

```
agent_host.sendCommand("turn 1") // Jobbra fordul, teljes sebességgel. -1 ↩
   esetén elenkező irányba fog fordulni. agent_host.sendCommand("move 1") ↩
   // 1-el előre -1-el hátrafelé indul teljes sebeséggel

agent_host.sendCommand("pitch 1") // 1-el lefelé tekintünk -1-el pedig az ↩
   elenkező irányba
```

```
agent_host.sendCommand("strafe -1") //bal oldalra mozdul, teljes  ↩
   sebességgel

agent_host.sendCommand("jump 1") //ugrálás

agent_host.sendCommand("crouch 1") //guggolás

agent_host.sendCommand("attack 1") //támadás
```

MalmoPython.MissionSpec() függvény a missionXML-ből meghatároza, hogy milyen biomot akarunk létrehozni, milyen játékmodba legyünk állítva illetve meddig fusson egy adott mission.Ezt majd átadja a my_mission változonak.

Példa a missionXML-re

```
'''<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
            <Mission xmlns="http://ProjectMalmo.microsoft.com" xmlns:xsi=" ↩
               http://www.w3.org/2001/XMLSchema-instance">

              <About>
                <Summary>gitlab.com/whoisZORZ</Summary>
              </About>

              <ServerSection>
                <ServerHandlers>
                  <FlatWorldGenerator generatorString="3;7,220*1,5*3,2;3;, ↩
                     biome_1"/>
                  <DrawingDecorator>
                        <DrawSphere x="-27" y="70" z="0" radius="30" type=" ↩
                           air"/>
                  </DrawingDecorator>
                  <ServerQuitFromTimeUp timeLimitMs="360000"/>
                  <ServerQuitWhenAnyAgentFinishes/>
                </ServerHandlers>
              </ServerSection>

              <AgentSection mode="Survival">
                <Name>ZORZBot</Name>
                <AgentStart/>
                <AgentHandlers>
                  <ObservationFromFullStats/>
                  <ContinuousMovementCommands turnSpeedDegs="180"/>
                </AgentHandlers>
              </AgentSection>
            </Mission>'''
```

Mielöt bármit is csinálnánk az agentnek meg kel adnunk, hogy folyamatosan elöre haladjon és declarálnunk kel néhány változót: a pozíció koordinátákat, irány változót, a nézetet pedig a stevepitch változó fogja

tárolni.A yaw értéke a Minecraft koordinátarendszeréhez igazodik a 180 északot,a 0 délt,a 90 nyugatot és a -90 pedig keletet.

```
# Loop until mission starts:
print("Waiting for the mission to start ", end=' ')
world_state = agent_host.getWorldState()
while not world_state.has_mission_begun:
    print(".", end="")
    time.sleep(0.1)
    world_state = agent_host.getWorldState()
    for error in world_state.errors:
        print("Error:",error.text)

print()
print("Mission running ", end=' ')

    agent_host.sendCommand( "move 1" )

Sx = 0
Sz = 0
sy = 0
Syaw = 0
Spitch = 0
eidx = 0
eidxj = 0
eidxb = 0
Barrier = 0
```

Steveet mozgás közben blokkok fogják körül venni ezért elöfordulhat,hogy megakad valamilyen akadályba például nekimegy egy fának.Ennek elkerülése érdekében a json függvénnyel információt fogunk gyüjteni a Stevet körülvevő blokkokról.Ezeket az információkat terminálra is kiírathatjuk.

```
if world state.number of observations since last state > 0:
    msg = world_state.observations[-1].text
    observations = json.loads(msg)
    nbr = observations.get("nbr3x3", 0)
    print("Mit latok: ", nbr)

    if "Yaw" in observations:
        Syaw = observations["Yaw"]
    if "Pitch" in observations:
        Spitch = observations["Pitch"]
    if "XPos" in observations:
        Sx = observations["XPos"]
    if "ZPos" in observations:
        Sz = observations["ZPos"]
    if "YPos" in observations:
        Sy = observations["YPos"]

    print ("Pozicio koordinatak: ", Sx, Sz, Sy)
    print ("Irany: ", Syaw)
```

```
    print ("Nezet: ", Spitch)

    if Syaw >= 180-22.5 and Syaw <= 180+22.5 :
        eidx = 1
        eidxj = 2
        eidxb = 0

    if Syaw >= 180+22.5 and Syaw <= 270-22.5 :
        eidx = 2
        eidxj = 5
        eidxb = 1

    if Syaw >= 270-22.5 and Syaw <= 270+22.5 :
        eidx = 5
        eidxj = 8
        eidxb = 2

    if Syaw >= 270+22.5 and Syaw <= 360-22.5 :
        eidx = 8
        eidxj = 7
        eidxb = 5

    if Syaw >= 360-22.5 or Syaw  <= 0+22.5 :
        eidx = 7
        eidxj = 6
        eidxb = 8

    if Syaw >= 0+22.5 and Syaw <= 90-22.5 :
        eidx = 6
        eidxj = 3
        eidxb = 7

    if Syaw >= 90-22.5 and Syaw <= 90+22.5 :
        eidx = 3
        eidxj = 0
        eidxb = 6

    if Syaw >= 90+22.5 and Syaw <= 180-22.5 :
        eidx = 0
        eidxj = 1
        eidxb = 3
```

Amikor Stevenek nem lesz szabad utja,akkor arra fogjuk utasítani, hogy kezdjen el fordulni és növeljük a Barrier változót eggyel.Amenyiben szabad az út Stevnek nem kell csinálnia semmit csak előre haladnia.Egyes akadályokat Steve ugrással is kivédhet.

```
if nbr[eidx+9]!="air" or nbr[eidxj+9]!="air" or nbr[eidxb+9]!="air":
        print ("Nincs szabad utam, elottem: ", nbr[eidx+9])
        agent_host.sendCommand ("turn" + str(turn))
```

```
        Barrier = Barrier + 1
    else:
        print ("Szabad az ut!")
        agent_host.sendCommand ("turn 0")
        agent_host.sendCommand ("jump 0")
        agent_host.sendCommand ("attack 0")
        Barrier = 0

    if Barrier > 8:
        agent_host.sendCommand ("jump 1")

    lepes = lepes + 1
    if lepes > 100:
    lepes = 0
    if tav < 20:
        prevSx = Sx
        prevSz = Sz
        prevSy = Sy
        turn = turn * -1
        agent_host.sendCommand ("attack 1")

    time.sleep(1)

print()
print("Mission ended")
# Mission has ended.
```

9. fejezet

Helló, Chaitin!

9.1. Iteratív és rekurzív faktoriális Lisp-ben

Ebben a feladatban egy olyan programot kellett készítenünk, amely meghatározza egy szám faktoriálisát. Faktoriálisnak nevezzük a matematikában egy n nemnegatív egész szám, n-nél kisebb vagy egyenlő pozitív egész számok szorzatát (Jelölése: n!).Ezt a programot iteratív és rekurzív módon is meg kellett valósítani Lisp-ben.

Factoriális iteratívan

Elsönek itt láthatjuk az iteratív megoldásunkat amit egy for loop-al oldottunk meg.Láthatjuk, hogy z-nek bekérünk egy számot aminek a faktoriálisát meg akarjuk tudni.Itt még definiáltunk egy y-t is amiben a megoldásunkat fogjuk tárolni. Ezután inditunk egy for ciklust ami 1-től z-ig össze fogja szorozni a számokat amit eltárolunk y-ban.Végül princ paranccsal kiíratjuk y-t ami a bekért számuk faktoriálisa lesz.

```
(princ "Szam: ")
(setq z (read))
(setq y 1)
(loop for x from 1 to z
    do (setq y (* y x))
       )
(princ y)
```

Factoriális rekurzívan

A rekurzív megvalósításban elsőnek írtunk egy factorial nevű függvényt.Ez maga nem olyan bonyolult, mindössze egy if-es eldöntésből áll. ha a szam kisebb mint 2 akkor a faktoriálisunk 1 lesz. Ellenkező esetben visszont a szam változótt megszoroza azzal a számal amit a factorial függvény fog vissza adni a (szam - 1)-re. Ezen a ponton láthatjuk, hogy ez egy rekurzív függvény mivel a végrehajtás során önmagát hívja meg.A függvény megírása után nincs más dolgunk csak bekérünk egy számot z néven, majd egy kiíratáson belül meghívjuk rá a factorial függvényt. Ez vissza is fogja adni a bekért szám faktoriálisát.

```
(defun factorial(szam)
    (if (< szam 2)
    1
    (* szam (factorial ( - szam 1)))))
```

```
    )
(princ "Szam: ")
(setq z (read))

(princ (factorial z))
```

9.2. Gimp Scheme Script-fu: króm effekt

A feladatunk az volt, hogy írjunk olyan script-fukiterjesztést a GIMP programhoz, amely megvalósítja a króm effektet egy bemenő szövegre!

Elöször létrehozunk egy 8 elemből álló tömböt egy függvény segitségével amelynek beállítjuk az értékeit.

```
            (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)
)
```

Ennek a definenak a segitségével tudjuk majd elérni a lista x-edik elemét.

```
(define (elem x lista)

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

)
```

A következő kodrész felel a kép megformázásért.

```
(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)
    )
)



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

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

    (gimp-image-select-rectangle image CHANNEL-OP-ADD 0 (/ text-height 2)  ↩
       width height)
    (gimp-context-set-foreground '(255 255 255))
    (gimp-drawable-edit-fill layer FILL-FOREGROUND )

    (gimp-image-select-rectangle image CHANNEL-OP-REPLACE border-size (+ (/ ↩
        text-height 2) border-size) (- width (* border-size 2)) (- height  ↩
       (* border-size 2)))
    (gimp-context-set-foreground '(0 0 0))
    (gimp-drawable-edit-fill layer FILL-FOREGROUND )

    (gimp-image-select-rectangle image CHANNEL-OP-REPLACE (* border-size 3) ↩
        0 text-width text-height)
    (gimp-drawable-edit-fill layer FILL-FOREGROUND )

    (gimp-selection-none image)

    ;step 1
    (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 (* border-size 3) 0)

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

```
       LAYER)))

    ;step 2
    (plug-in-gauss-iir RUN-INTERACTIVE image layer 25 TRUE TRUE)

    ;step 3
    (gimp-drawable-levels layer HISTOGRAM-VALUE .18 .38 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 (/ text-height  ↩
       2)) 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 0 width (+ height (/ text-height 2)))

    ;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-image-scale image new-width (/ (* new-width (+ height (/ text- ↩
       height 2))) width))

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

A következő kodrészletben a felhasználó használhatja az általunk alapértelmezetté tett beálitásokat illetve lehetöséget biztosítunk neki, hogy más értékeket használjon.

```
(script-fu-register "script-fu-bhax-chrome-border"
    "Chrome3-Border2"
    "Creates a chrome effect on a given text."
    "Norbert Bátfai"
    "Copyright 2019, Norbert Bátfai"
    "January 19, 2019"
    ""
```

```
    SF-STRING       "Text"      "Rácz András István"
    SF-FONT         "Font"      "Sans"
    SF-ADJUSTMENT   "Font size" '(160 1 1000 1 10 0 1)
    SF-VALUE        "Width"     "1920"
    SF-VALUE        "Height"    "1080"
    SF-VALUE        "New width" "400"
    SF-COLOR        "Color"     '(255 0 0)
    SF-GRADIENT     "Gradient"  "Crown molding"
    SF-VALUE        "Border size"   "7"
)
(script-fu-menu-register "script-fu-bhax-chrome-border"
    "<Image>/File/Create/BHAX"
)
```

9.3. Gimp Scheme Script-fu: név mandala

Ebben a feladatban egy olyan scriptet kellet irjunk a GIMP-hez, amely egy névből mandalát készit.Ezt a kapott szöveg töbszöros elforgatásával tehetjük meg.Ennek megvalósítását szemlélteti a következö kód.

Tanulságok, tapasztalatok, magyarázat...

```
        (define (elem x lista)

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

)

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

    text-width
    )
)

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

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

    (list text-width text-height)
    )
)


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

(define (script-fu-bhax-mandala text text2 font fontsize width height color ↩
    gradient)
(let*
    (
        (image (car (gimp-image-new width height 0)))
        (layer (car (gimp-layer-new image width height RGB-IMAGE "bg" 100  ↩
           LAYER-MODE-NORMAL-LEGACY)))
        (textfs)
        (text-layer)
        (text-width (text-width text font fontsize))
        ;;;
        (text2-width (car (text-wh text2 font fontsize)))
        (text2-height (elem 2 (text-wh text2 font fontsize)))
        ;;;
        (textfs-width)
        (textfs-height)
        (gradient-layer)
    )

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

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

    (gimp-context-set-foreground color)

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

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

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

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

(set! text-layer (car (gimp-layer-new-from-drawable textfs image)))
(gimp-image-insert-layer image text-layer 0 -1)
(gimp-item-transform-rotate text-layer (/ *pi* 6) TRUE 0 0)
(set! textfs (car(gimp-image-merge-down image text-layer CLIP-TO-BOTTOM ↩
   -LAYER)))

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

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

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

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

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

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

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

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

(gimp-image-insert-layer image gradient-layer 0 -1)
```

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

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

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

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

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



(script-fu-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"      "Rácz Andras István"
    SF-STRING       "Text2"     "BHAX"
    SF-FONT         "Font"      "Sans"
    SF-ADJUSTMENT   "Font size" '(100 1 1000 1 10 0 1)
    SF-VALUE        "Width"     "1000"
    SF-VALUE        "Height"    "1000"
    SF-COLOR        "Color"     '(255 0 0)
    SF-GRADIENT     "Gradient"  "Deep Sea"
)
(script-fu-menu-register "script-fu-bhax-mandala"
    "<Image>/File/Create/BHAX"
)
```

10. fejezet

Helló, Gutenberg!

10.1. Programozási alapfogalmak

1. Fejezet: Bevezetés

Ebben a fejezetben megismerkedhetünk az alapvető fogalmakkal.Továbbá olvashatunk arról, hogy mi alapján osztályozzuk a programnyelveket.Megismerjük a programnyelvek két csoportjait: imperatív, deklaratív illetve, tudatja velünk hogy mik sorolhatók az egyéb nyelvek közé.

2. Fejezet: Alapelemek

A második fejezetben egy programozási nyelv alapeszközeit, alapfogalmait ismerjük meg.Részletesen tárgyalunk a karakterkészletről illetve a többkarakteres szimbólumokról.A második fejezet negyedik részétől kezdve a különböző adattípusokról, konstansokról, mutatókról és változókról olvashatunk.

3. Fejezet: Kifejezések

Ebben a fejezetben a kifejezésekröl beszélünk.Megnézük, hogy egy kifejezés milyen konponensekből tevődnek össze (operandusok, operátorok, zárojelek).Megtudjuk, hogy egy kifejezésnek három alakja lehet inflix, prefix és postfix.Ezután rátérunk a C nyelvben használt operátorok megismerésére.

10.2. Programozás bevezetés

1. Fejezet: Alapismeretek

Az első fejezetben arról olvshatunk, hogy, hogyan kell elkezdeni egy számunkra új programozási nyelv tanulását.Továbbá ebben a fejezetben megismerkedhetünk a C alapjaival mint például a kiíratás, változók és konstansok megadása, ciklusok.Ezeken kivül bevezet minket a tönbök és függvények világába is, illetve röviden összefoglalta, hogy mivel is fog foglalkozni a könyv a továbbiakban.

2. Fejezet: Típusok, operátorok és kifejezések

Ebben a részben a különböző declarációkról részletesen olvashatunk.Arról, hogy milyen módon érdemes a változó neveket létrehozni illetve, hogy vannak a C nyelv által lefoglalt kifejezések, illetve, hogy meg kell adnunk a változók típusát is.Továbbá ebben a részben olvashatunk a különböző (aritmetikai,relációs, logikai, inkrementáló, dekrementáló és értékadó) operátorokról is.

3. Fejezet: Vezérlési szerkezetek

Itt beszélünk az if-else utsítással való döntésekröl, hogy hol adhatjuk meg a feltételt illetve, hogy hova kell írjuk a végrehajtani kívánt utasítást.Beszélünk továbbá a switch utasításról ahol megemlítődi a break parancs amit majd részletezni fog.Ez után rátérunk a while, a for és a do while ciklusok részletezésére.Ezt a fejezetet a ritkán használt continue-val és goto-val zárjuk.

10.3. Programozás

1. Fejezet: Bevezetés

2. Fejezet: A C++ nem objektumorientált újdonságai

A második fejezetben a C és a C++ nyelv összehasonlítását láthatjuk.Elöször a függvényparaméterek megadását hasonlitjuk össze.Ezzutá olvashatunk, hogy milyne módon declarálhatunk változókat, illetve itt még megemlítjük a változótipusokat is.Ezek után visszatérunk a függvényekre, hogy, hogyan terhelhetünk túl őket.A fejezetet a praméterátadással és referenciatípusokkal zárjuk.

3. Fejezet: Objektumok és osztályok

Ebben a fejezetben az osztályokról olvashatunk, arról, hogy mit nevezzünk osztálynak, mire használjuk azokat, illetve megemlítődik még az objektumok fogalma, illetve az ezeken belüli adatvédelem és adatrejtés is.Itt továbbá részletes betekintést kapunk a konstruktorok és destruktorok világába és beszélünk a dinamikus adatkezelésről.Ez után visszatérünk a konstruktorkra nevezetesen a másoló konstruktort elemzük ki benne és olvashatunk a tagfüggvényekről is.Továbbá a könyv ezen részében olvasunk a frien függvényekről, a statikus tagokról és a beágyazot definiciókról.

4. Fejezet: Konstansok és inline függvények

A könyv negyedik fejezete elöször bemutatja nekünk, hogy a konstansokat hol és hogyan érdemes használni.Ezután részletesen olvashatunk a konstansok fontosabb fajtáiról például a konstans változókról, pointerekről, függvényparaméterekről, tagváltozókról stb.Információt kapunk arról, hogy abban az esetben ha egy konstans tagfüggvény egy adot változóját meg akarjuk változtatni akkor ezt úgy tehetjük meg, hogy a változót mutable-ként kel definiálnunk.Ezen fejezet második és egyben utólsó részében az inline függvényekről beszünk.

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. Ã©s Ritchie, Dennis M., *A C programozási nyelv*, Bp., Műszaki, 1993.

11.5. C++

[BMECPP] Benedek, Zoltán Ã©s 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.