Univerzális programozás

Írd meg a saját programozás tankönyvedet!

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

COLLABORATORS

	TITLE : Univerzális programozás	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Sántha, Máté Imre	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	2019-02-19	Aktualizálás, javítások.	nbatfai
0.0.5	2019-03-03	első fejezet kész	Sántha Máté Imre
0.0.6	2019-03-11	Második fejezet kész	Sántha Máté Imre
0.0.7	2019-03-18	Harmadik fejezet kész	Sántha Máté Imre
0.0.8	2019-03-26	Negyedik fejezet kész .	Sántha Máté Imre

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

Ajánlás

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

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

Tartalomjegyzék

Ábrák jegyzéke

Előszó

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

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

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

Hogyan forgasd

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

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

Milyen nyelven nyomjuk?

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

Hogyan nyomjuk?

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

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

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

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

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

I. rész

Bevezetés

1. fejezet

Vízió

1.1. Mi a programozás?

1.2. Milyen doksikat olvassak el?

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

1.3. Milyen filmeket nézzek meg?

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

II. rész

Tematikus feladatok

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

2. fejezet

Helló, Turing!

2.1. Végtelen ciklus

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

Megoldás videó:

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

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

```
int main() {
while(1){}

   return 0;

}
}
```

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

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



int main() {

        #pragma omp parallel
           while(1){}
```

```
        return 0;

}
}
```

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

```
include <unistd.h>

int main() {
  while(1) {
    sleep(1);

  }

}
```

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

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

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

Megoldás videó:

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

```
Program T100
{

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

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

```
}
```

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

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

akár önmagára

```
T100(T100)
false
```

ezt a kimenetet adja.

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

```
Program T1000
{

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

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

  main(Input Q)
  {
    Lefagy2(Q)
  }

}
```

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

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

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

Velemenyem szerint Nem tudunk olyan programot írni amely képes egy előrre megírt programról meghatorizni hogy az a program képes lesz e a lefutásra vagy nem lesz e kepes azaz le fog fagyni.

2.3. Változók értékének felcserélése

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

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

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

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

```
#include <stdio.h>

  int main() {

          int a = 6;
          int b = 2;

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

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

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

}
}
```

2.4. Labdapattogás

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

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

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

A feladatunk a labdapattogás feladat lesz . Velemenyem szerint ezzel a feladatban a bevprog során mindenki talalkozott hisz tobbszor is előkerült mind a sima mind az Ifes változata A kódot olvasva lathatjuk hogy több mindent is deklarálnunk kell. Meg kell adnunk a labda kezdőhelyzetét ezen felül az ablak méretét illetve a labda mozgásának pattogásának mértékét. Ahogy lathatjuk a kódban egész sok függvényt fogunk alkalmazni ilyen peldaul a refresh ami a nevéből adódóan a program frissítéséért felel ilyen a getmaxyx() amelyel a Dinamikusságot adjuk a programnak vagy eppen ilyen a mvprintw() aminek segítségével magát a labdát rajzoltatjuk ki vagy ilyen az uspeel() amelyel a labdánk gyorsaságát allíthatjuk be es tovabba ott van meg az initscr függvény is amelyel a terminal méretétével van kapcsolatban . A Maga program az IF függvények hasznalataval allapítja meg hogy eppen elerte e a labda az adott meretu terminal valamelyik szélét . Ha elérte akkor a labda irányt vált

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

{
    WINDOW *ablak;
    ablak = initscr();

    int x = 0;
    int y = 0;

    int xnov = 1;
    int ynov = 1;

    int mx;
    int my;

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

        x = x + xnov;

        y = y + ynov;

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

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

}
}
```

2.5. Szóhossz és a Linus Torvalds féle BogoMIPS

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

Megoldás videó:

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

Tehat ebben a megoldasban a Bitshift nevezetű operatorral fogunk dolgozni. Ahogy lathatjuk a megoldas során Shiftelésre leszünk szükségünk . Ez azt jelenti hogy folyamatosan haladunk egy adott iranyban. Ezt jelen esetben egy while ciklust használva fogunk megcsinalni de lehetne akar más ciklust is alkalmazni . Miutan a elshifteljük 0hoz es sikeres volt a muvelet illetve nem vétettünk hibat gyakorlatilag megkapjuk a szó méretét bitben

```
        #include <stdio.h>

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

    printf("Szóhossz: %d bit\n",a);

}
}
```

2.6. Helló, Google!

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

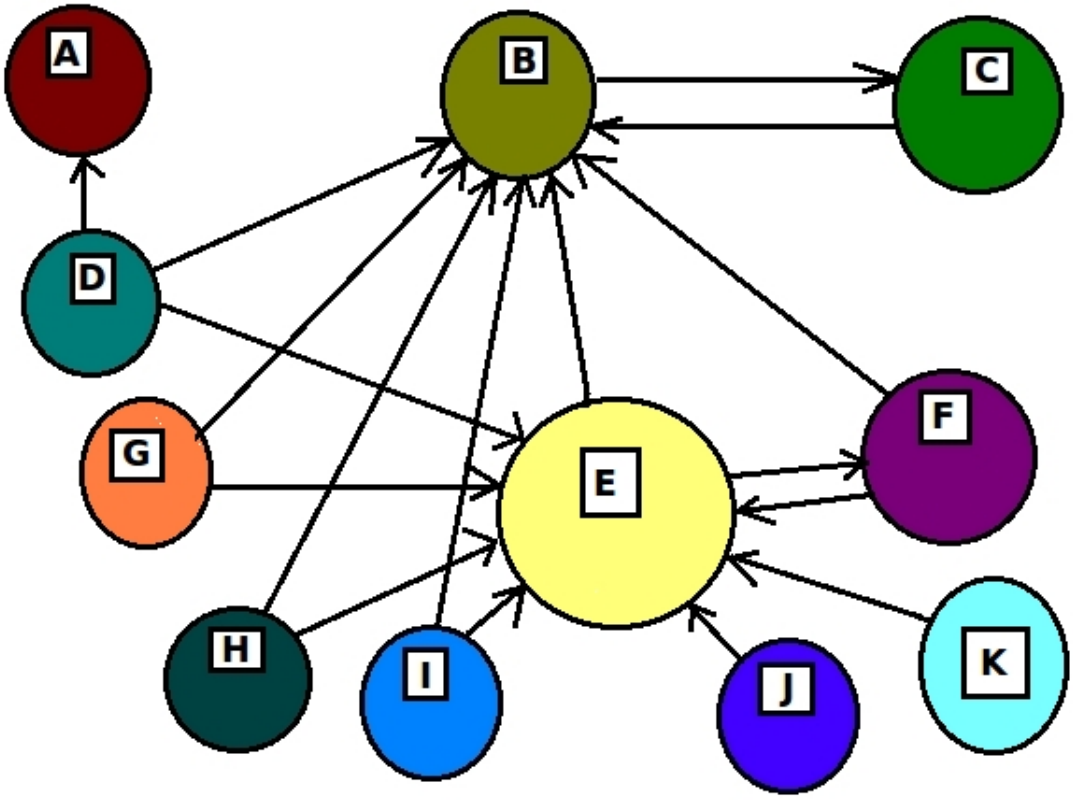
Megoldás videó:

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

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

```
          // nem sajat kód

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


void

kiir (double tomb[], int db)
{
  int i;

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

}

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

{

  double osszeg = 0.0;
  int i;
  for (i = 0; i < n; ++i)
    osszeg += (PRv[i] - PR[i]) * (PRv[i] - PR[i]);

  return sqrt(osszeg);
}

int main(void){
  double L[4][4] = {

    {0.0, 0.0, 1.0 / 3.0, 0.0},
    {1.0, 1.0 / 2.0, 1.0 / 3.0, 1.0},
    {0.0, 1.0 / 2.0, 0.0, 0.0},
```

```
    {0.0, 0.0, 1.0 / 3.0, 0.0}

  };

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

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

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

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

}
}
```



2.1. ábra. PageRank : - Kép Wikipedia:

2.7. 100 éves a Brun tétel

Írj R szimulációt a Brun tétel demonstrálására!

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

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

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

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

```
// nem sajat kód
stp <- function(x){

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

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

}
}
```

2.8. A Monty Hall probléma

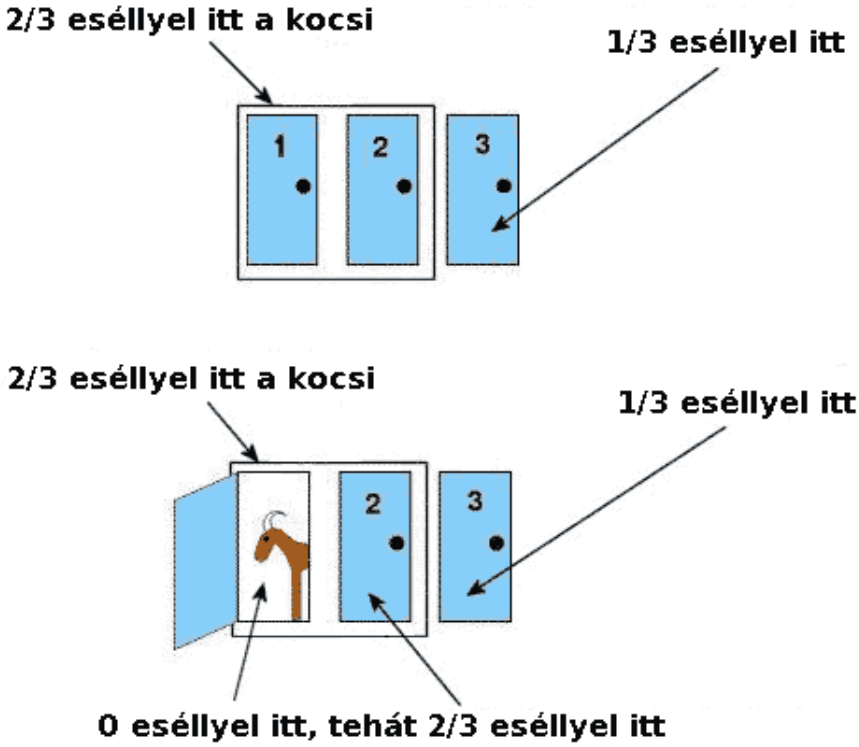
Írj R szimulációt a Monty Hall problémára!

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

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

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

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

for (i in 1:kiserletek_szama) {

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

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

    }else{

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

    }

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

}

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

for (i in 1:kiserletek_szama) {

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

}

valtoztatesnyer = which(kiserlet==valtoztat)


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

}
}
```



2.2. ábra. Monty Hall Forrás:Wikipedia:

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfjával megadva írd meg ezt a gépet!

Megoldás videó:

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

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

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

using namspace std;
int main()

{
```

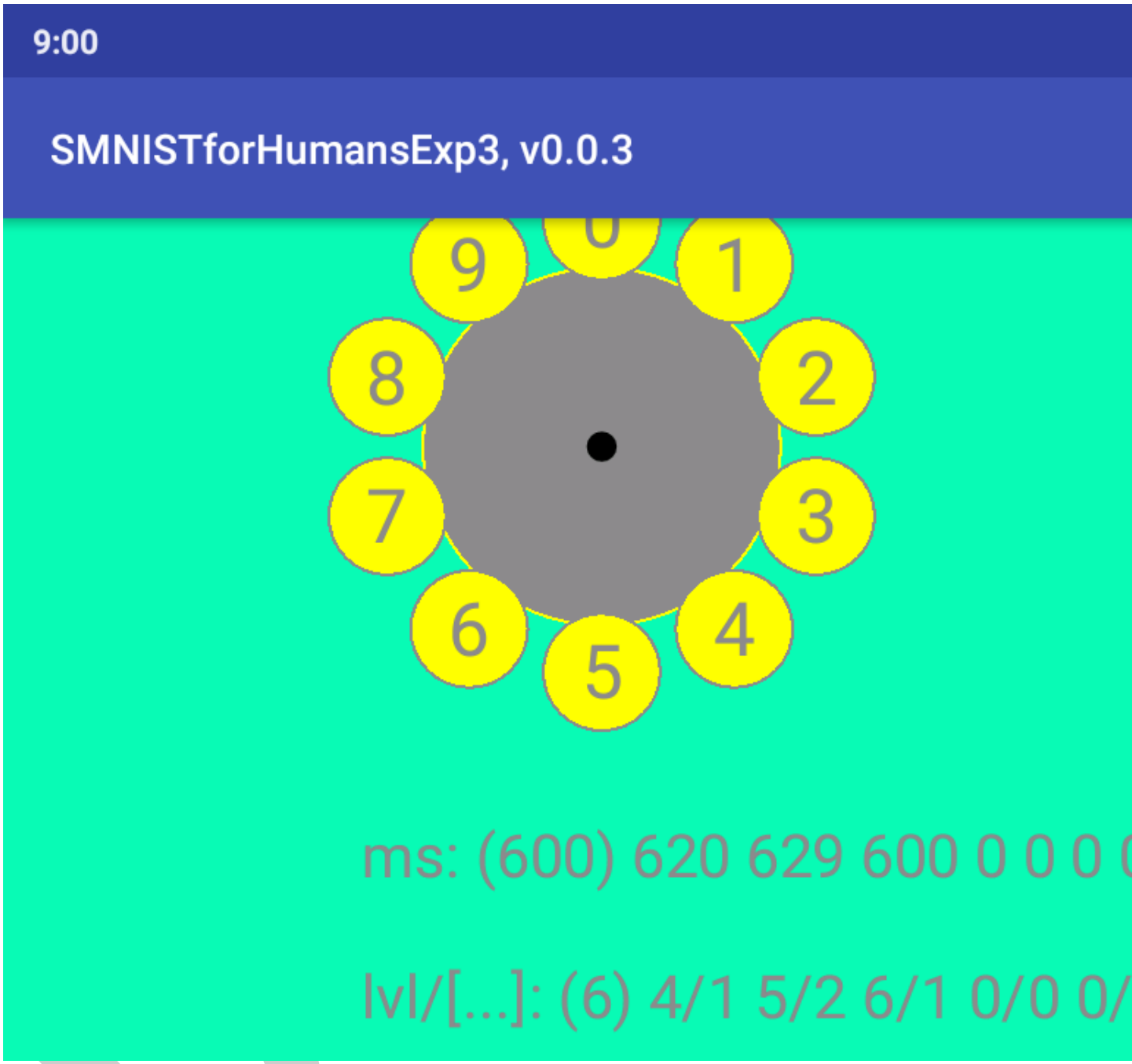
```
int a=0
cout << "írjon be egy számot : "<< '\n';
 cin >> a ;
  for (int i =0 ; i<a , i++ ) {
  cout << "I" ;

 }

 cout<< '\n';

}
}
```

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

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

Megoldás videó:

Megoldás forrása:



3.1. ábra. 1.

3.3. Hivatkozási nyelv

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

Megoldás videó:

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

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

3.4. Saját lexikális elemző

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

Megoldás videó:

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

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

```
// nem sajat kód
#include <stdio.h>
int realnumbers = 0;

%}

digit [0-9]
%%

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

```
%%
int

main (){
yylex ();

printf("The number of real numbers is %d\n", realnumbers);

return 0;
        }
```

3.5. l33t.l

Lexelj össze egy l33t ciphert!

Megoldás videó:

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

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

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

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


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

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

  {'x', {"x", "%", ")(", ")("}},
  {'y', {"y", "", "", ""}},
  {'z', {"z", "2", "7_", ">_"}},

  {'0', {"D", "0", "D", "0"}},
  {'1', {"I", "I", "L", "L"}},
  {'2', {"Z", "Z", "Z", "e"}},
  {'3', {"E", "E", "E", "E"}},
  {'4', {"h", "h", "A", "A"}},
  {'5', {"S", "S", "S", "S"}},
  {'6', {"b", "b", "G", "G"}},
  {'7', {"T", "T", "j", "j"}},
  {'8', {"X", "X", "X", "X"}},
  {'9', {"g", "g", "j", "j"}}

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

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

      }
```

3.6. A források olvasása

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

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

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

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

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

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

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

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

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

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

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

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

Megoldás forrása:

Megoldás videó:

1.Hat a Sigint jel figyelve volt tehat nem volt figyelmen kivul hagyva akkor az adott jelkezelő függvény kezelje ,de ha figyelmen kivul volt hagyva akkor ő is hagyja figyelmen kivul.

2. Hat a masodik kód nem tul nehez egy ciklusrol beszelhetunk ami 4ig megy . A ciklus addig fog lefutni még az i értéke kisebb mint 5. Ha igaz a feltetel noveli az I értékét .

3. Szerintem ez a 2 kód ugyanaz azzal a kivetellel hogy az I vegső értéke nem 5 lesz mint az előző esetben hanem 4.

4. A ciklus során elemeket kepezünk , Minden lefutás utan egyre nagyobb és nagyobb elemeket kapunk es ezeket beletoltjuk a Tomb megfelelő elemébe

5.A kovetkező kódban is egy ciklussal talalkozhatunk itt addig megy a folyamat még az I kisebb mint n illetve a d eleme megegyezik az s elemével

6. Ebben az esetben kiiratunk 2 szamot amelyet aez F függvény határoz meg

7. Itt is kiiratunk 2 erteket az egyik az F(a) értéke lesz a masik pedig az 'a' értéke

8. Ebben a helyzetben is kiiratunk 2 erteket ebben az esetben az F nagy mertekben meghatarozza az a értékét

.

3.7. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

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

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

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

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

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

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

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

3.8. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

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

Mit vezetnek be a programba a következő nevek?

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

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Egy változót A néven

Egy változóra mutatót

A valtozó azaz a refernciajat

A kovetkező peldaban egy öt elemből álló tömbel talakozhatunk.

Az előző feladatban talalható tömb referenciajat lathatjuk.

Az öt elemből álló tömbre mutató mutatót lathatunk.

Egy függvényre mutatót lathatunk .

Egy függvényre mutató mutató függvényt.

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

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

4. fejezet

Helló, Caesar!

4.1. int *** háromszögmátrix

Megoldás videó:

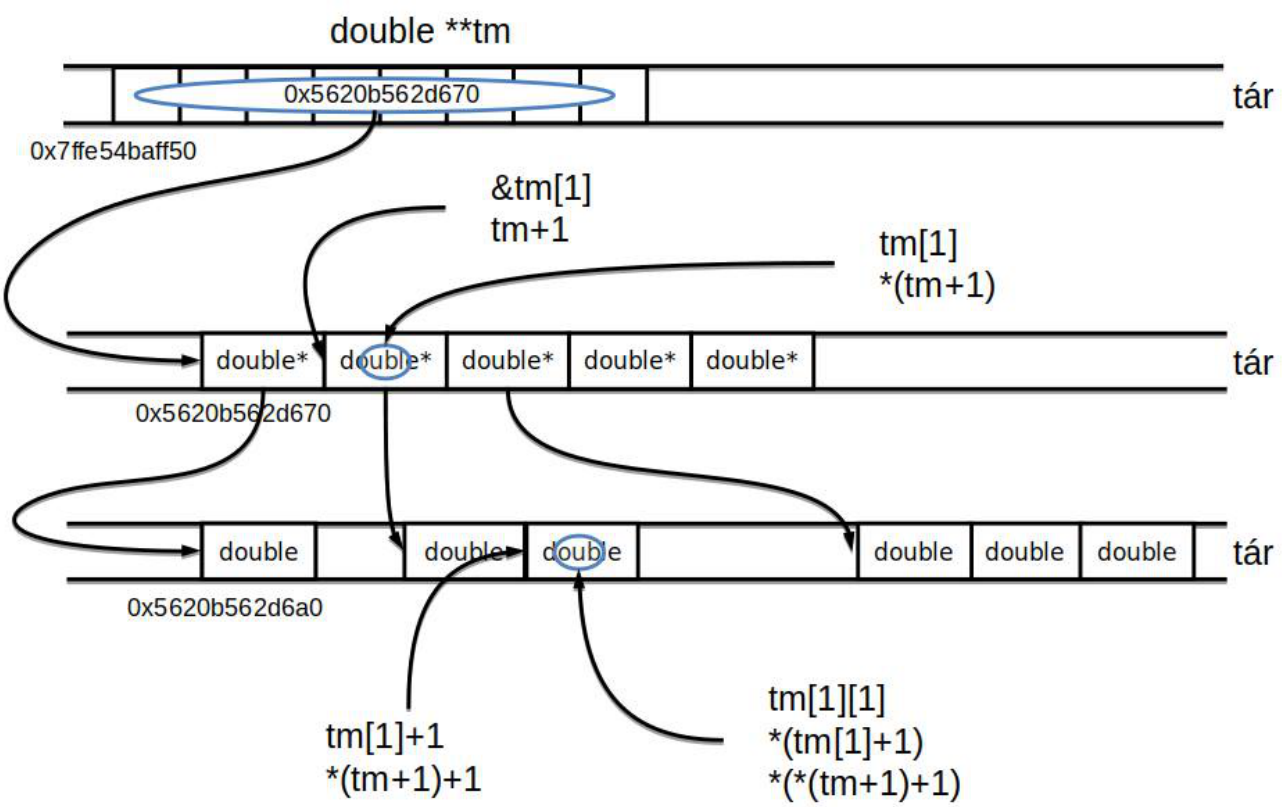
Megoldás forrása: https://github.com/Samate99/Hello-world/blob/master/Prog1/Ceaser/matrix.c

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

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



int main ()

{

    int nr = 5;
```

```
double **tm;

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

{
    return -1;

}

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

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

printf("%p\n", tm[0]);
for (int i = 0; i < nr; ++i)
    for (int j = 0; j < i + 1; ++j)
        tm[i][j] = i * (i + 1) / 2 + j;



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

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

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

{

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

}
```

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

    return 0;
}
}
```



4.1. ábra. Forrás:Batfai Norbert

4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás videó:

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

Tehat a feladatunk egy C exor titkosító létrehozása . A feladatban meg kell adnunk 2 nagyon fontos dolgot egy szöveget amit a program letitkosít és egy kulcsot amiből a karaktereket nyeri. A program az elorre megadott kulcsban szereplő karakterrekkel exorozzza , keveri össze a szöveget . Ennek a segítésével hozzuk letre az exorozott szöveget. Nagyon érdekese hogy egyes kulcsok megadasaval milyen érdekes kódolt szövegek jönnek létre . A programhoz szükség van 2 TXT allomanyra az egyikbe a tiszta szövegett kell tennünk, a masikba pedig kesobb fog kerülni a titkos szöveg. A programhoz includeolnunk kell a megfelelő

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

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

#define MAX_KULCS 100
#define BUFFER_MERET 256


int main (int argc, char **argv)

{

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

  int kulcs_index = 0;
  int olvasott_bajtok = 0;

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


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

    {

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

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

  }

      write (1, buffer, olvasott_bajtok);

    }
}
```

4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

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

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

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

    public ExorTitkosito(String kulcsSzoveg,
            java.io.InputStream bejovoCsatorna,
            java.io.OutputStream kimenoCsatorna)
            throws java.io.IOException {


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

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

                buffer[i] = (byte)(buffer[i] ^ kulcs[kulcsIndex]);
                kulcsIndex = (kulcsIndex+1) % kulcs.length;
            }
            kimenoCsatorna.write(buffer, 0, olvasottBajtok);
        }
    }
    public static void main(String[] args) {
        try {
          new ExorTitkosito(args[0], System.in, System.out);

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

4.4. C EXOR törő

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

Megoldás videó:

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

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

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

#define OLVASAS_BUFFER 256

#define KULCS_MERET 5

#define _GNU_SOURCE



#include <stdio.h>

#include <unistd.h>

#include <string.h>

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

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

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

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

  double szohossz = atlagos_szohossz (titkos, titkos_meret);

  return szohossz > 6.0 && szohossz < 9.0

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

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

{

  int kulcs_index = 0;

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

    {

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

    }

}

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

}

int
main (void)
{

  char kulcs[KULCS_MERET];
  char titkos[MAX_TITKOS];
```

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

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

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



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

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

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

4.5. Neurális OR, AND és EXOR kapu

R

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

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

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

```
Program T100
#   Copyright (C) 2019  Dr. Norbert Bátfai, nbatfai@gmail.com
#
#   This program is free software: you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation, either version 3 of the License, or
#   (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program.  If not, see <http://www.gnu.org/licenses/>
#
# https://youtu.be/Koyw6IH5ScQ

library(neuralnet)

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

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

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

plot(nn.or)

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


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

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

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

```
plot(nn.orand)

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



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

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

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

plot(nn.exor)

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



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

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

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

plot(nn.exor)

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


                }
```

4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó:

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

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

létrejött rengeteg embernek koszonhetjuk de az első valtozatanak elkeszítője Perceptron Rosenbalt volt . Aki ki nem talalnak a nevadó is volt . A program eleg bonyolult de szerintem mindenki szamara latható hogy korulbelul 3 reszre oszthatjuk a kódot . Lathatunk egy részt amelyben ahogy latjuk fogadjuk az adatokat lathatunk egy masodik reszt amely kepes lesz osszegezni az adatokat illetve eltarolni illetve van egy harmadik resz amely kepes donteseket hozni es lathatjuk hogy kepes kezeli az adatokat.

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

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

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

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

  delete p;
  delete [] image;

}
}
```

5. fejezet

Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

Megoldás videó:

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

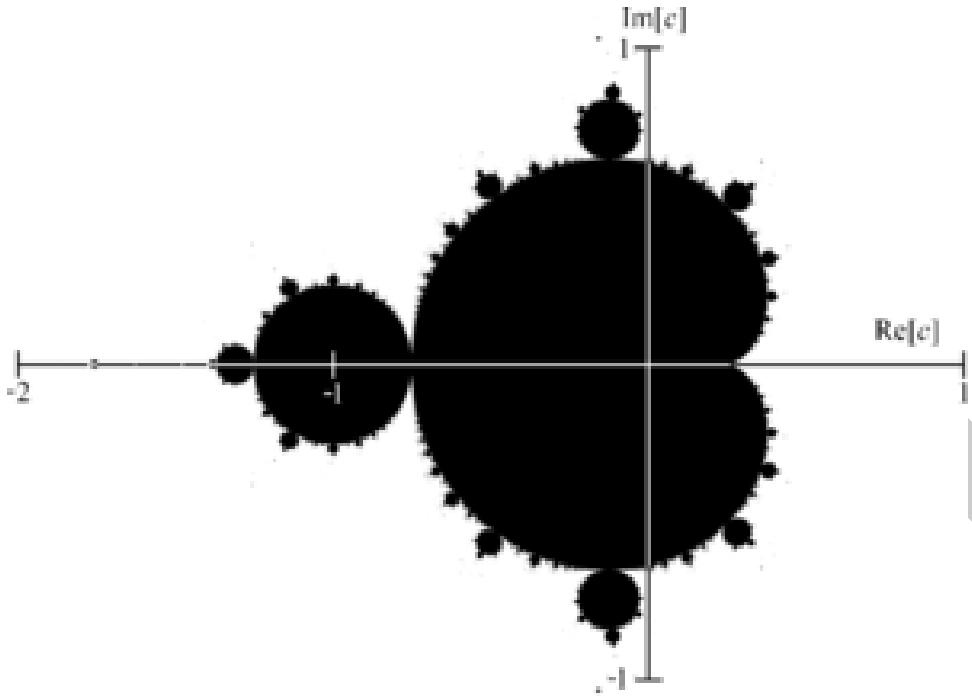
Első lépésként itt is fontos tisztázni hogy mi is az ugynevezett Mandelbrot halmaz . Ez gyakorlatilag egy a komplex szamsíkon talalt halamaz . Felfedezését Benoit Mandelbrotnak köszönhetjük aki 1980ban találta meg . Tehetjük fel a kérdést hogy miért is fontosak szamunkra a komplex szamsíkon talalható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

```
Program T100

/ mandelpngt.c++
// Copyright (C) 2019
// Norbert Bátfai, batfai.norbert@inf.unideb.hu
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
```

```
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <https://www.gnu.org/licenses/>.
//
// Version history
//
//  Mandelbrot png
//  Programozó Páternoszter/PARP
//   https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0063 ↩
   _01_parhuzamos_prog_linux
//
//  https://youtu.be/gvaqijHlRUs
//

#include <iostream>
#include "png++/png.hpp"
#include <sys/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)
  sruct tms tmsbuf1, tmsbuf2;
    times (&tmsbuf1);



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







    // a számítás
```

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



    int iteracio = 0;

    // Végigzongorázzuk a szélesség x magasság rácsot:

    for (int j = 0; j < magassag; ++j)
    {
       //sor = j;
        for (int k = 0; k < szelesseg; ++k)
        {
            // c = (reC, imC) a rács csomópontjainak
            // megfelelő komplex szám
            reC = a + k * dx;
            imC = d - j * dy;
            // z_0 = 0 = (reZ, imZ)
            reZ = 0;
            imZ = 0;

            iteracio = 0;
            // z_{n+1} = z_n * z_n + c iterációk
            // számítása, amíg |z_n| < 2 vagy még
            // nem értük el a 255 iterációt, ha
            // viszont elértük, akkor úgy vesszük,
            // hogy a kiinduláci c komplex számra
            // az iteráció konvergens, azaz a c a
            // Mandelbrot halmaz eleme

           while (reZ * reZ + imZ * imZ < 4 && iteracio < iteraciosHatar)
            {
               // z_{n+1} = z_n * z_n + c
               ujreZ = reZ * reZ - imZ * imZ + reC;
                ujimZ = 2 * reZ * imZ + imC;
                reZ = ujreZ;
                imZ = ujimZ;
                ++iteracio;

            }

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

    times (&tmsbuf2);
```

```
    std::cout << tmsbuf2.tms_utime - tmsbuf1.tms_utime
              + tmsbuf2.tms_stime - tmsbuf1.tms_stime << std::endl;
    delta = clock () - delta;
    std::cout << (float) delta / CLOCKS_PER_SEC << " sec" << std::endl;

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

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

    }

    int kepadat[MERET][MERET];
    mandel(kepadat);

    png::image < png::rgb_pixel > kep (MERET, MERET);
    for (int j = 0; j < MERET; ++j)
    {
        //sor = j;
        for (int k = 0; k < MERET; ++k)
        {
            kep.set_pixel (k, j,
                           png::rgb_pixel (255 -

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

        }
    }

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



}


}
```



5.1. ábra. Mandelbrot Forrás :Wikipedia:

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

Megoldás videó:

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

Tehat az előző feladatban mar beszéltünk a Mandelbrot halmazról illetve a komplex szamsíkról . Itt külön nem emelném ki a történetét illetve a komplex szamsík és a halmaz jelentőségét. A programunk egyszeruen kifogja számolni az előzőhőz hasonlóan a mandelbrot halmazt viszont ebben az esetben az std:complex osztályt fogja segítéségük hívni a feladathoz . Tehat itt is kiemelném hogy ehez a programhoz is kellenek a mar fentebb említett könyvtárak illetve egyéb eszközöt telepítése hisz nélkülük ez a program se fog megfelelően lefutni. Szóval mivel itt külön az std:complex osztalyt kell hasznalnunk erdemes első lépésként ezt importalnunnk a programba. Ahogy az előző feladatban ebben is előrre meg kell adnunk azaz definialnunk kell a kép adatik A program hasonló képpen mukodik mint az előző program tehat ebben az esetben is for ciklusok segítségével megy végig a pontokon , egy while ciklussal ellenorzi magat illetve tolti fel a jónak talalt pontokat. Fontos kiemelni hogy a programnak van egy része amely folyamatosan kiirja szamunkra hogy hany %nál jár a programunk. Végül de nem utolsó sorban szeretnem kiemelni hogy az előzővel ellentétben az itt hasznalunk szinezest emiatt ez a csodas kép egy szines kép lesz. Ha mindent jól csinaltunk mindent megfelelően leírtunk , deklaraltunk akkor a program lefutása utan itt is egy képpel lehetünk gazdagabbak valamelyik meghajtón .

```
Program T100
// Verzio: 3.1.2.cpp
// Forditas:
// g++ 3.1.2.cpp -lpng -O3 -o 3.1.2
// Futtatas:
```

```
// ./3.1.2 mandel.png 1920 1080 2040  ↩
   -0.019473810573093663922605855987058021128 18  ↩
   -0.019473810572541341845642648422654019668 7  ↩
   0.798505756933826860155534177465597167611 1  ↩
   0.798505756934379196110285192844457924366
// ./3.1.2 mandel.png 1920 1080 1020  ↩
   0.412765541820958925534057470940751954913 1  ↩
   0.412765541824581805308014281763462349772 5  ↩
   0.213538705176874649138696327099751215428 1  ↩
   0.213538705180497528912653137922461610287 4
// Nyomtatas:
// a2ps 3.1.2.cpp -o 3.1.2.cpp.pdf -1 --line-numbers=1  --left-footer=" ↩
   BATF41 HAXOR STR34M" --right-footer="https://bhaxor.blog.hu/" --pro= ↩
   color
// ps2pdf 3.1.2.cpp.pdf 3.1.2.cpp.pdf.pdf
//
//
// Copyright (C) 2019
// Norbert Bátfai, batfai.norbert@inf.unideb.hu
//
//  This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <https://www.gnu.org/licenses/>.


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

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

{

  int szelesseg = 1920;
  int magassag = 1080;

  int iteraciosHatar = 255;

  double a = -1.9;
  double b = 0.7;
```

```
double c = -1.3;
double d = 1.3;


if ( argc == 9 )
  {

    szelesseg = atoi ( argv[2] );
    magassag =  atoi ( argv[3] );
    iteraciosHatar =  atoi ( argv[4] );

    a = atof ( argv[5] );
    b = atof ( argv[6] );
    c = atof ( argv[7] );
    d = atof ( argv[8] );
  }
else

  {

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

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

double dx = ( b - a ) / szelesseg;
double dy = ( d - c ) / magassag;
double reC, imC, reZ, imZ;
int iteracio = 0;
std::cout << "Szamitas\n";
// j megy a sorokon



for ( int j = 0; j < magassag; ++j )
  {

    // k megy az oszlopokon
    for ( int k = 0; k < szelesseg; ++k )
      {

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

        reC = a + k * dx;
        imC = d - j * dy;

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

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

          while ( std::abs ( z_n ) < 4 && iteracio < iteraciosHatar )
            {

             z_n = z_n * z_n + c
              ++iteracio;
            }
          kep.set_pixel ( k, j,

                          png::rgb_pixel ( iteracio%255, (iteracio*iteracio) ↩
                             %255, 0 ) );


        }
      int szazalek = ( double ) j / ( double ) magassag * 100.0;
      std::cout << "\r" << szazalek << "%" << std::flush;
    }
  kep.write ( argv[1] );
  std::cout << "\r" << argv[1] << " mentve." << std::endl;

}

}
```

5.3. Biomorfok

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

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

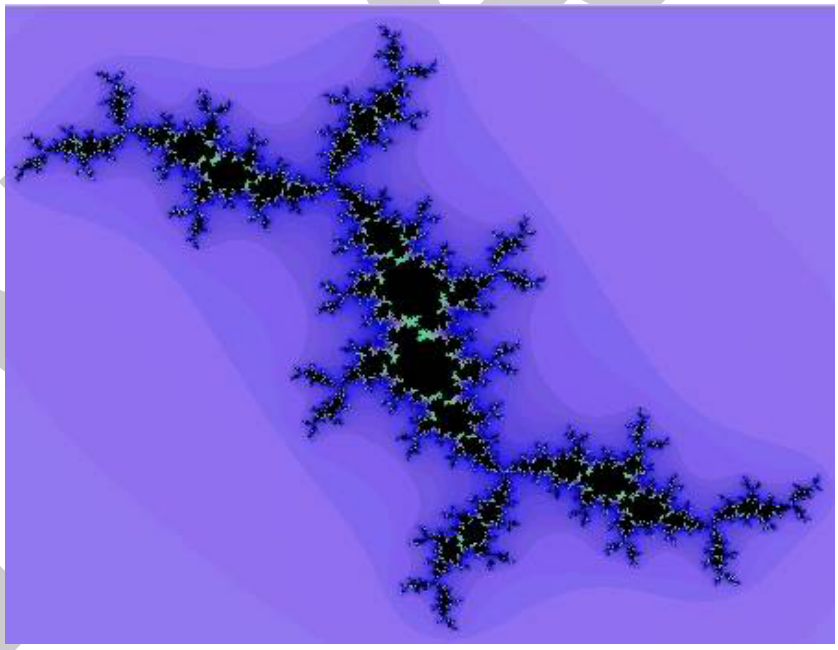
Tehetjük fel hogy honnan is erednek a biomorfok . Hát gyakorlatilag egy Hiba , tévedés miatt talaltak ra ezekre mint egyéb más dolgokra. Clifford Pickover volt az személy aki megpróbálta a Julia halmazokat kirajzoltatni sajnos az alltala megírt program több helyen is hibákkal rendelkezik de szerencséjére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

```
Program T100
// Verzio: 3.1.3.cpp
```

```
// Forditas:
// g++ 3.1.3.cpp -lpng -O3 -o 3.1.3
// Futtatas:
// ./3.1.3 bmorf.png 800 800 10 -2 2 -2 2 .285 0 10
// Nyomtatas:
// a2ps 3.1.3.cpp -o 3.1.3.cpp.pdf -1 --line-numbers=1  --left-footer=" ↩
   BATF41 HAXOR STR34M" --right-footer="https://bhaxor.blog.hu/" --pro= ↩
   color
// ps2pdf 3.1.3.cpp.pdf 3.1.3.cpp.pdf.pdf
//
// BHAX Biomorphs
// Copyright (C) 2019
// Norbert Batfai, batfai.norbert@inf.unideb.hu
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <https://www.gnu.org/licenses/>.
//
// Version history
//
// https://youtu.be/IJMbgRzY76E
// See also https://www.emis.de/journals/TJNSA/includes/files/articles/ ↩
   Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf
//

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

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

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

```
    double R = 10.0;

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

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

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

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

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

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

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

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

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

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

                z_n = std::pow(z_n, 3) + cc;
                //z_n = std::pow(z_n, 2) + std::sin(z_n) + cc;
```

```
                if(std::real ( z_n ) > R || std::imag ( z_n ) > R)
                {
                    iteracio = i;
                    break;
                }
            }

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

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

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

}


}
```



5.2. ábra. Julia halmaz Forrás :www.t-es-t.h

5.4. A Mandelbrot halmaz CUDA megvalósítása

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

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

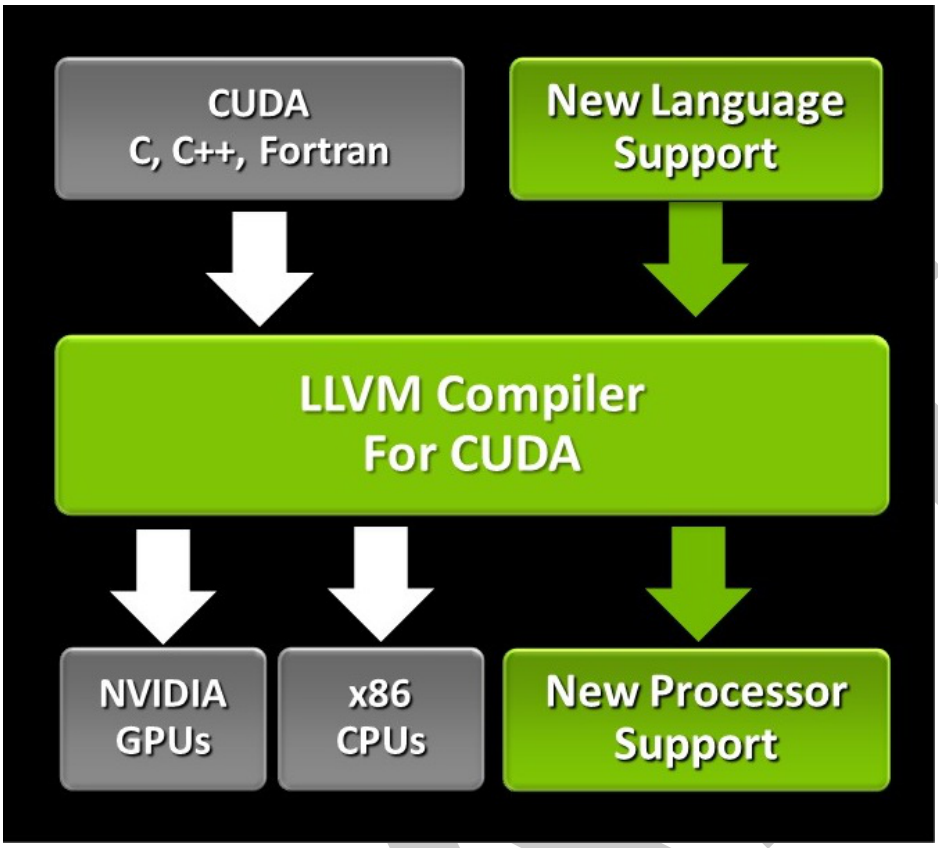
```
Program T100
// mandelpngc_60x60_100.cu
// Copyright (C) 2019
// Norbert Bátfai, batfai.norbert@inf.unideb.hu
//
//  This program is free software: you can redistribute it and/or modify
//  at under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 3 of the License, or
//  (at your option) any later version.
//
//  This program is distributed in the hope that it will be useful,
//  but WITHOUT ANY WARRANTY; without even the implied warranty of
//  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
//  GNU General Public License for more details.
//
//  You should have received a copy of the GNU General Public License
//  along with this program.  If not, see <https://www.gnu.org/licenses/>.
//
// Version history
//
//  Mandelbrot png
//  Programozó Páternoszter/PARP
//   https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0063 ↩
   _01_parhuzamos_prog_linux
//
// https://youtu.be/gvaqijHlRUs
//



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


}

}
```



5.3. ábra. CUDA Forrás: NVIDIA Developer

5.5. Mandelbrot nagyító és utazó C++ nyelven

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

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

Megoldás videó:

A feladat megfelelő megoldásához szükségünk lesz tobb csomag telepítésére többek között a libqt4-dev csomagra is . Ahogy már ismerjek ezt folyamatot a qmake parancsal letre kell hoznuink a makefilet es mint ahogy eddig is tettük a Makefileos feladatok esetbéne le kell futtatnunk Ha a telepítések és a futtatások sikeresek voltak 4 abalkot fogunk kapni . Mind a 4 képen ugyanaz tekinthető meg . Az egyik kepen az eredeti kepet lathatjuk a maradek 3 kepen pedig erről a képről bezoomolt kepeket tekinthetünk meg. Kiemelném hogy a program nem egyetlen egy fajból hanem összesen 6 fajlból áll. Mindegyik fajlnak fontos szerepe van van amelyik a mandelbrot halmaz kirajzolásában van szerepe van emlyik van amelyik csak osztalyokat tartalmaz viszont a megfelelő mukodéshez az osszesre van szükségünk

5.6. Mandelbrot nagyító és utazó Java nyelven

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

Az előző feladathoz nagyon hasonló feladatot kaptunk. Az előzőhez hasonlóan ez is ki fog szamunkra rajzolni egy kepet . Illetve ez a kep is egy Mandelbrot halmaz lesz. Ebben az esetben is fontos a megfelelő fajlok , csomagok telepítése mert a program megfelelő futásához elengedhetetlenek. Magát a halmazt szinte ugyanugy hozza létre mint a c++ esetében az eltérő változás a Zoomban tekinthető hisz ebben az esetben ez interaktív lesz . Az eger segítéségvel tudunk kijelolni részeket majd ezekbol a reszekbol kapunk uj kepet. Az előzőhoz hasonlóan itt is egy pillanatkeppel lehetunk gazdagabbak ,de az előzővel ellentétben nem arról van szó hogy előrre megadott kepeket kapunk hanem mi valaszthatjuk ki hogy a mandelbrot halmaz eppen melyik szegletére vagyunk kivancsiak.

```
Program T100

 * MandelbrotHalmazNagyító.java
 *
 * DIGIT 2005, Javat tanítok
 * Bátfai Norbert, nbatfai@inf.unideb.hu

 */
/**
 * A Mandelbrot halmazt nagyító és kirajzoló osztály.
 *
 * @author Bátfai Norbert, nbatfai@inf.unideb.hu
 * @version 0.0.1
 */


public class MandelbrotHalmazNagyító extends MandelbrotHalmaz {


    /** A nagyítandó kijelölt területet bal felső sarka. */
    private int x, y;
    /** A nagyítandó kijelölt terület szélessége és magassága. */
    private int mx, my;
    /**
     * Létrehoz egy a Mandelbrot halmazt a komplex síkon
     * [a,b]x[c,d] tartománya felett kiszámoló és 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();

            }

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

            // újraszámítása indul:

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

                double dy = (MandelbrotHalmazNagyító.this.d

                        - MandelbrotHalmazNagyító.this.c)
                        /MandelbrotHalmazNagyító.this.magasság;
                // Az új Mandelbrot nagyító objektum elkészítése:

                new MandelbrotHalmazNagyító(MandelbrotHalmazNagyító.this.a+ ↩
                   x*dx,
                        MandelbrotHalmazNagyító.this.a+x*dx+mx*dx,
                        MandelbrotHalmazNagyító.this.d-y*dy-my*dy,
                        MandelbrotHalmazNagyító.this.d-y*dy,

                        600,
```

```
                    MandelbrotHalmazNagyító.this.iterációsHatár);

            }

        });

        // Egér mozgás események feldolgozása:

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

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

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

    }

    /**

     * Pillanatfelvételek készítése.
     */

    public void pillanatfelvétel() {
        // Az elmentendő kép elkészítése:
        java.awt.image.BufferedImage mentKép =

                new java.awt.image.BufferedImage(szélesség, magasság,
                java.awt.image.BufferedImage.TYPE_INT_RGB);

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



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

        if(számításFut) {
            g.setColor(java.awt.Color.RED);
            g.drawLine(0, sor, getWidth(), sor);
```

```
        }
        g.setColor(java.awt.Color.GREEN);
        g.drawRect(x, y, mx, my);
        g.dispose();
        // A pillanatfelvétel képfájl nevének képzése:



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

        // A fájl nevébe 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

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

Megoldás videó:

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

Tehat első ránézésre a polartranszformációs algoritmus talan ilyesztőnek tünhet viszont ha jobban beleolvasunk a programkódban akkor azért megnyugodhatunk. A polártranszformációval fogunk random számokat generalni amelyeket fel fogunk hasznalni a program kulonboző részein ! A program több nyelven is megtalalható jelen esetben en egy java nyelven írt kódot mutatok be . Szóval kezdjuk is el nezni a programunkat szerintem ami egyből feltünik a kódot olvasók szamara hogy rendelkezünk egy PolarGen Classal aminek talan az a specifikus tulajdonsaga hogy 2 reszből all Beszélhetünk egy Privat illetve egy publikus részről is . De akkor lessük is meg hogy hogyan dolgozik a program . A program megnezi hogy van e tarolt ertek ezutan 2 felé válik a történet ha nem rendelkezünk akkor szamulonk megfelelő mennyiségű értéket . Az egyiket visszaadjuk a programunknak a masikra még később szükségünk lesz emiatt azt eltaroljuk. Tehat utana vegzunk par müveletet es egy ciklus segítéségével kiiratjuk az adott szamu erteket , eredményt .

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

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

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

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


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

    dValue = r * v2;
    bExists = !bExists;

    return r * v1; //idáig tart az algoritmus

  }
  else

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

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

  }
}
```

6.2. LZW

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

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

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

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

6.3. Fabejárás

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

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

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

```
Program T100

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

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

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


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

}
```

6.4. Tag a gyökér

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

Megoldás forrása: //github.com/Samate99/Hello-world/blob/master/Prog1/lwzbinfa.cpp

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

6.5. Mutató a gyökér

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

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

Tehat a feladatunkban tovabbra is a binfat kell jobban tanulmanyoznunk illetve a binfaban kelll különböző modosításokat létrehoznunk . A feladatunk hogy a gyökér ne kompozícióban legyen hanem aggregacióban

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

6.6. Mozgató szemantika

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

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

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

7. fejezet

Helló, Conway!

7.1. Hangyaszimulációk

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

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

Megoldás forrása: hhttps://github.com/Samate99/Hello-world/tree/master/Prog1/Myrmecologist

Ez a szimulació egy nagyon erdekes muvelet . Vegrehatjasahoz viszont tobb egyéb csomag , program telepítésére van szükség ilyen peladaul a libqt4 csomag amelyet telepítenünk kell . Miutan a telepítéseék lezajlottak kiemelném a qmake hasznalatat , ennek segítségével fogjuk megkapni a makefilet ami a program futtatásához elengedhetetlen. Nézzük is meg a kódot ahogy lathatjuk a main függényen belül több helyen adatokat kell megadnunk illetve deklarálnunk kell . Lathatjuk hogy a program legalbb 3 elkulonithető osztallyal /resszel rendelekezik . Ant , Antwim és Anthread lesz ez a 3 osztály. Nézzük is meg őket egyesével nézzük meg hogy mi miért felelős . Tehat az Antwin lesz az osztaly amelynek feladata a megjelenítés ebben tudjuk megadni az ablak tulajdonságait , méreteit , szélességét pixeleinek szamat stb.... Az Ant maga a virtualis "hangyak" tulajdonsagait fogja tartalmazni . Tehat ebben a reszben szerepelnek a hangyaink mozgasat , koordinalasat rögzítő beállítások . Itt lathatjuk a Hangyak mereteit , illetve mozgasuk elorre meghatarozott iranyat . Vegul nezzuk meg az Anthreadet hogy mit is tartalmaz . Az Antthread eleg sok dolgot tartalmaz . Igy első korben kiemelném hogy tartalmazza a szimulacióban résztvevő hangyak számát, illetve tartalmazza a hangyak utvonalait , tovabba ebben a reszben adhatjuk meg a feromont a feromon párolgásának mértékét és egyéb feromon adatokat amelynek segítségével képesek leszünk a hangyak "iranyítására" Kiemelném hogy a programban tobb elorre definialt bind is talalható ilyen peldauil a P amely feltételezéseim szerint az angol PAUSE szóból eredhet . Lenyomásával termeszetesn meg tudjuk allítani a folyamatot szüneteltetni tudjuk. A kovetkező ilyen a Q lesz ez az angol QUIT szóbol ered gondolom . Ennek lenyomásával kitudunk lepni a folyamatból.

7.2. Java életjáték

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

Megoldás videó:

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

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

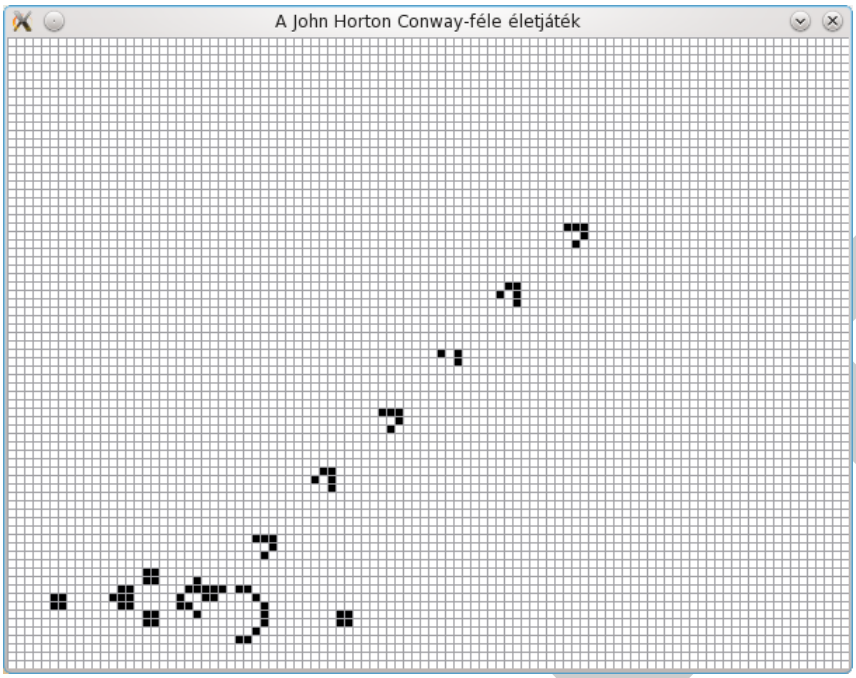
7.3. Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó:

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

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



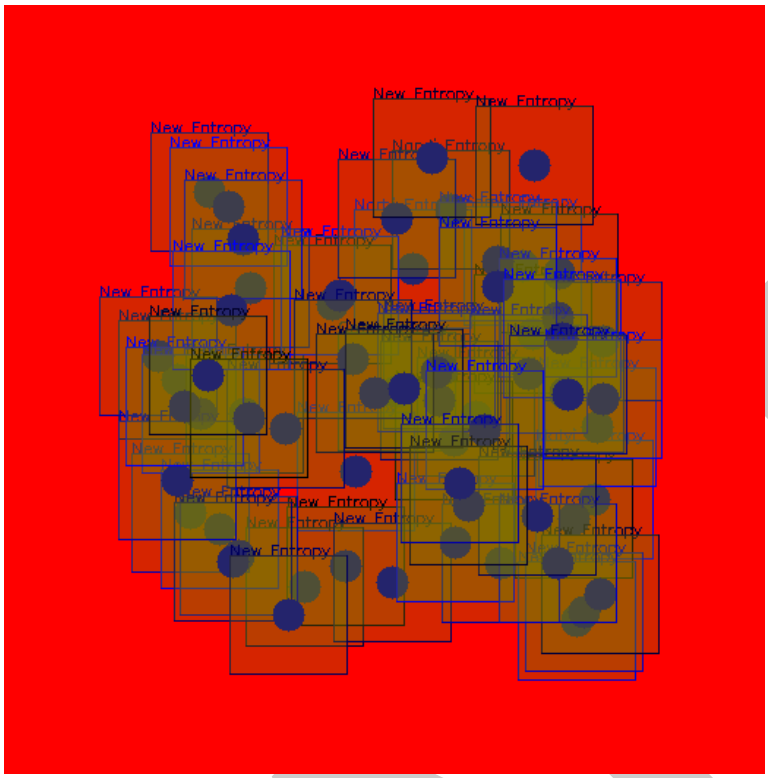
7.1. ábra. John Horton Conway-féle életjáték Forrás : Batfai Norbert

7.4. BrainB Benchmark

Megoldás videó:

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

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



7.2. ábra. BrainB Benchmark : Batfai Norbert

8. fejezet

Helló, Schwarzenegger!

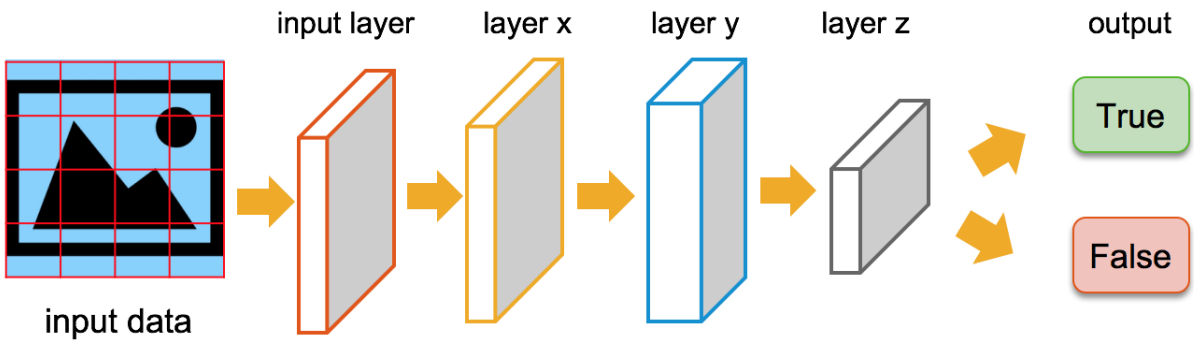
8.1. Szoftmax Py MNIST

aa Python

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

Megoldás forrása:

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



8.1. ábra. TensorFlow Forrás:SAP Blogs

8.2. Szoftmax R MNIST (Passz)

R

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.3. Minecraft-MALMÖT(Passz)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9. fejezet

Helló, Chaitin!

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

Megoldás videó:

Megoldás forrása:

9.2. Gimp Scheme Script-fu: króm effekt

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

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

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

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

```
; bhax_chrome3.scm
;
; BHAX-Chrome creates a chrome effect on a given text.
; Copyright (C) 2019
; Norbert Bátfai, batfai.norbert@inf.unideb.hu
; Nándor Bátfai, batfai.nandi@gmail.com
;
```

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

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

;(color-curve)

(define (elem x lista)

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

)

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

```
        (text-height 1)
    )

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

    (list text-width text-height)
    )
)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
}
```

9.3. Gimp Scheme Script-fu: név mandala

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

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

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

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

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

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

(define (elem x lista)

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

)

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

    text-width
    )
)

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

    (list text-width text-height)
    )
)


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

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

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

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

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

    (gimp-context-set-foreground color)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

}
```

10. fejezet

Helló, Gutenberg!

10.1. Programozási alapfogalmak

[?]

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

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

10.2. Programozás bevezetés

[KERNIGHANRITCHIE]

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

A kovetkező szamunkra fontos konyv A Brian W.Kernighan és Dennis M. Ritchie . The C programming lanugage nevet viseli.Ebben a konyvben a C nyelv sajátosságaival fogunk megismerkedni.Így kezdésként fontos kiemelni hogy a C nyelv maga rendkivul sok specialis tulajdonsággal rendelkezik nezzuk meg akart a vesszőt ha egy sort vagy tobb sort egy vessző követ akkor alltalaban annak elvalasztasi funkciót adunk velemenyem szerint. A C nyelvben való kódoláskor nem érdemes ilyet hasznalni mivel könnyen lefordulási problémmakkal talalkozhatunk . Mivel ha egy sort vesszővel zarunk az itt utasításnak minősül . Ebben a nyelvben kifejezzen fontos a kapcsos zarójelek megfelelő hasznalata mert ez is konnyen hibahoz vezethet. Tovabba fontos kiemelni hogy tobb dologban megegyezik a tobbi nyelvek ugyanugy hasznalhatunk megfelelő ciklusokat ugyanugy Ifel tudunk választ kérni / adni ugyanugy tudunk Switchet alkallmazni hogy elágazásokat vezessünk be. A konyvben kulon szóba kerülnek a ciklusok illetve a ciklusok alkalmazása . Hisz tudjuk hogy a különböző ciklusok között (while, do-while , for) akkora kulonbség nincs tehat ugyanugy feltételt kell megadni. Talan ami fontosabbnak tekinthető hogy van olyan fajta ciklus amely 1x mindenfelepkeppen lefut illetve van olyan amelynek ha nincs megfelelő feltétel egyszer sem fut le. A C++ ban mar megtanult cout helyett Cben a printF függvényt kell hasznalnunk hogy a megfelelő dolgokat kiirassuk az outputra. Így összességében a C nyelv egy nagyon érdekes nyelv amely talan az egyik nehezebb de velemenyem szerint ez ízlések és pofonok hogy eppen ki miben szeret és akar vagy eppen miben köteles programozni.

10.3. Programozás

[BMECPP]

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

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(Passz)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

11.2. Java osztályok a Pi-ben(Passz)

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.