Univerzális programozás

Így neveld a programozód!

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

COLLABORATORS			
	TITLE : Univerzális programozás		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Bátfai, Norbert Ã©s Rátóczi, Martin Ferenc	2019. május 10.	

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

Ajánlás

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

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

Tartalomjegyzék

Előszó

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

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

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

Hogyan forgasd

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

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

Milyen nyelven nyomjuk?

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

Hogyan nyomjuk?

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

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

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

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

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

I. rész

Bevezetés

1. fejezet

Vízió

1.1. Mi a programozás?

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

1.2. Milyen doksikat olvassak el?

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

1.3. Milyen filmeket nézzek meg?

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

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

II. rész

Tematikus feladatok

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

2. fejezet

Helló, Turing!

2.1. Végtelen ciklus

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:

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

sleep();
};

}


int main(){
while(1) {


};

}



#include <omp.h>

int main(){

#pragma omp parallel

while(1) {
```

```
};

}
```

-vegtelen_0.c Egy szálat futtatunk 0%-on, ezt úgy érjük el, hogy a szálat a sleep paranccsal "elaltatjuk", azaz leállítjuk. -vegtelen_100.c Egy szálat futtatunk 100%-on, a while a zátójelben lévő egyes miatt mindig igaz lesz, ezért folyamatosan futni fog. -vegtelen_a.c Az open MP segítségével az összes szálra futtatni párhuzamosan, ugyan azt az elvet, amit az egy szál 100%-on történő futtatásakor használtunk.

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.

Tanulságok, tapasztalatok, magyarázat...

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:

```
            #include <stdio.h>



int a= 13;
int b= 9;
```

```
int main(){


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



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


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


}
```

A-nak és b-nek megadunk 2 egész értékű változót, majd kiiratjuk az a és b változókat eredeti alakjukban. Ezek után a két változót összeadjuk, és az összegből kivonjuk az eredeti változókat, majd megint kiiratjuk a változókat, de mostmár felcserélt állapotban.

2.4. Labdapattogás

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

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

Megoldás forrása:

```
            #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 ); //was 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 ) { // elerte-e a tetejet?
            ynov = ynov * -1;
        }
        if ( y>=my-1 ) { // elerte-e a aljat?
            ynov = ynov * -1;
        }

    }

    return 0;
}
```

A curses headerrel kezeljük az ablakot melyben pattog a labdánk. Az x és y alap éártékének 0-t állítunk be, majd ezeket az x- és ynov paranccsal nüveljük az értékét. A végtelen ciklusban a getmaxyx meghatározza a az ablak maximum x és y értékeit, míg az mvprint kiiratja a labdát. A refresh parancs frissíti az ablakot, a usleep késlelteti, a labda mozgását. Az ifek ellentétes irányba fordítják a labda útvonalát, ha eléri a labda a max értéket.

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:

Tanulságok, tapasztalatok, magyarázat...

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:

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

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

  int i;

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

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

  int i;
  double osszeg=0;

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

  return sqrt(osszeg);
}

void
pagerank(double T[4][4]){
  double PR[4] = { 0.0, 0.0, 0.0, 0.0 }; //ebbe megy az eredmény
  double PRv[4] = { 1.0/4.0, 1.0/4.0, 1.0/4.0, 1.0/4.0}; //ezzel szorzok

  int i, j;

  for(;;){

    // ide jön a mátrix művelet

    for (i=0; i<4; i++){
      PR[i]=0.0;
      for (j=0; j<4; j++){
        PR[i] = PR[i] + T[i][j]*PRv[j];
      }
    }
```

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

    // ide meg az átpakolás PR-ből PRv-be

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

  kiir (PR, 4);
}

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

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

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

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

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

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

  printf("\n");

  return 0;
}
```

Megadunk 2 függvényt a void típusban és 1 értéket a double változótípussal. Az adott vektor i-edik elemét írja ki a kiir függvény, a tavolsag nevű változó felfogható a matematikában használt 2 pont távolságának. A pagerank függyvényben megadunk 2 darab 1 dimenziós és 1 darab 2 dimenziós mátrixot(tömböt). A PR a végeredményn oszlopvektora, a PRv mátrix-szal pedig szorzok. A PR[i]-t egyenlővé tesszük a PRv és a T mátrix elemeinek szorzatával, majd hozzáadjuk a PR[i]-t. Minden PR[i] kiszámítás előtt az előző for ciklus ezt az értéket nullára állítja. Ezután átpakolom a PR-ből a PRv-be, majd a kiir függvény a PR-ből kiírja a benne lévő 4 PageRank értéket. a main (void) függvényben kiszámolja L-re L1-re és L2-re a pagerank értéket.

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

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

Tanulságok, tapasztalatok, magyarázat...

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfjával megadva írd meg ezt a gépet!

Megoldás videó:

Megoldás forrása:

A turinggép olvas egy számot, ami most 10. Kivon egyet a decimális számunk utolsó számjegyéből majd eltárolja az egyest. Mivel az utolsó helyen nem 0 áll ezért átváltja 0 helyett 9-re. Ezek után az első számjegyből is kivn egy egyest és azt is tárolja. Addig ismétli ezt a feladatsort, míg minden helyi értékben 0-át nem látunk. A tárolt egyesek sorozata lesz a jelen esetben unáris 10-esünk. Tutor: Rémiás Dávidot

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

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

Megoldás videó:

Megoldás forrása:

```
1. A, B, C „változók”a, b, c „konstansok”A → aAB, A → aC, CB ↩
   → bCc, cB → Bc, C → bcS-ből indulunk ki ↩
   ----------------------------A (A → aAB)aAB ( A → aC)aaCB ( ↩
   CB → bCc)aabCc (C → bc)aabbccRévész könyv, 13. o. (Bev. a ↩
   form. nyelvek elméletébe, Akadémiai Kiadó, 1979)A (A → aAB) ↩
   aAB ( A → aAB)aaABB ( A → aAB)aaaABBB ( A → aC)aaaaCBBB ( ↩
   CB → bCc)aaaabCcBB (cB → Bc)aaaabCBcB (cB → Bc)aaaabCBBc ( ↩
   CB → bCc)aaaabbCcBc (cB → Bc)aaaabbCBcc (CB → bCc) ↩
   aaaabbbCccc (C → bc)aaaabbbbcccc
2. S, X, Y „változók”a, b, c „konstansok”S → abc, S → aXbc, ↩
   Xb → bX, Xc → Ybcc, bY → Yb, aY -> aaX, aY → aaS-ből ↩
   indulunk ki----------------------------S (S → aXbc)aXbc (Xb ↩
   → bX)abXc (Xc → Ybcc)abYbcc (bY → Yb)aYbbcc (aY → aa) ↩
   aabbccRévész könyv, 12. o. (Bev. a form. nyelvek elméletébe, ↩
```

```
 Akadémiai Kiadó, 1979S (S → aXbc)aXbc (Xb → bX)abXc (Xc ↩
→ Ybcc)abYbcc (bY → Yb)aYbbcc (aY -> aaX)aaXbbcc (Xb → bX) ↩
aabXbcc (Xb → bX)aabbXcc (Xc → Ybcc)aabbYbccc (bY → Yb) ↩
aabYbbccc (bY → Yb)aaYbbbccc (aY → aa)aaabbbccc
```

```
P1XP2 → P1QP2, P1, P2 eleme (VN U VT)*, X VN beli, Q (VN U VT) ↩
   + beli, kivéve S → üres, de akkor S nem lehet jobb oldali ↩
   egyetlen szabályban sem
```

3.3. Hivatkozási nyelv

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

Megoldás videó:

Megoldás forrása:

```
printf("%d\n", sizeof(char)*8);
```

A kódrészlet áll konstansból, karakterláncból, azonosítóból és kulcsszavakból. A printf függvény az azonosítónk mely a kiírásért felel. A karakterlánc a %d. A kulcsszavak a : "sizeof" "char". "sizeof" tárolja az argumentumának a bitméretét. A "char" pedig karakterláncot fejez ki. Így nézhet ki: Van egy kifejezés ami most printf, mely argumentumakémt egy kifejezéslista szolgál. A printf-et tovább bontva, kapunk kifejezéslistát és még egy kifejezést. Az új kifejezésünk tovább bontható erre: sizeof(kifejezés). Tehát a fent leírt "printf" a következőképp áll elő: Elsődleges kifejezés: lista, kifejezés, konstans. kif. lista: adott kifejezés legkisebb kifejezés: sizeof(kifejezés) konstans: 8 Tutor: Rémiás Dávid

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:

```
%{
#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;
}
```

Deklarálunk egy real_numbers változót, ez megszabja, hogy a számjegyeink 0 és 9 között vehetnek fel értéket. Ezek után a lexerünk eldönti ({digit}*(\.{digit}+)?)-t alkalmazva, hogy valós-e az adott szám. Ha valós a szám a real_numbers értéke nő 1-gyel. A mainben meghívjuk a lexert és kiíratjuk a valós számok számát.

3.5. l33t.l

Lexelj össze egy l33t ciphert!

Megoldás videó:

Megoldás forrása: /* Forditas: $ lex -o l337d1c7.c l337d1c7.l Futtatas: $ gcc l337d1c7.c -o l337d1c7 -lfl (kilépés az input vége, azaz Ctrl+D)

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

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

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

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

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

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

```
  } l337d1c7 [] = {

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

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

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

%}
%%
. {

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

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

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

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

        found = 1;
        break;
      }

    }

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

  }
%%
int
main()
{
  srand(time(NULL)+getpid());
  yylex();
  return 0;
}
```

Egy leet táblázat alapjhán kicserélünk karaktereket leetekre. Minden karakterhez 4 leetet fűzünk, ezt tömbökbe osztjuk, azért hogy könnyebben tudjunk hivatkozni ezekre. Az r változó eldönti, hogy a karakter 4 lett értéke közül melyik kerüljön kiírásra. Generál egy számot 0 és 100 között véletlenszerűen és ezt az értéket veszi fel. Az értéktől függően kapjuk a leet[0], leet[1], leet[2] vagy leet[3] értéket. Tutor: Rémiás Dávid

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:

Tanulságok, tapasztalatok, magyarázat...

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. Minden x-re létezik nagyobb y amely prím. 2. Minden x-re létezik nagyobb y amely prím, s a rákövetkezőjének a rákövetkezője is prím. 3. Létezik y minden x-re ahol ha x prím, akkor kisebb, mint y. 4. Létezik y minden x-re ahol ha y kisebb, mint x, akkor x nem prím.

3.8. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

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

Mit vezetnek be a programba a következő nevek?

- ```
  int a;
  ```
- ```
  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:

```
int a;
int *b = &a;
int &r =a;
int c[5];
int (&tr) [5] = c;
int *d[5];
int *h ();
int *(*j ());
int (*i(*v (int e))) (int f, int g);

        </para>
        <para>
            int a; = egész
            int *b=&a; = egészre mutató mutató
            int &r=a; =egész referenciája
            int c[5]; = egészek tömbje
            int (&tr)[5] =c; =egészek tömbjének referenciája
            int *d[5]; = egészre mutató mutatók tömbje
            int *h(); = egészre mutató mutatót visszaadó függvény
            int *(*1()); = egészre mutató mutatót visszaadó függvényre ↩
               mutatót mutató (bár 1-es helyett egy betű használata jobb ↩
               választás volna)
            int (*v(int c)) (int a, int b); = egészet visszaadó és két ↩
               egészet kapó függvényre mutató mutatót visszaadó, egészet ↩
               kapó függvény
            int (*(*z) (int)) (int, int); = függvénymutató egy egészet ↩
               visszaadó és két egészet kapó függvényre mutató mutatót ↩
               visszaadó, egészet
                                                kapó függvényre ( az "int"-ek ↩
                                                   után hiányzik az elnevezés)
    Tutor: Rémiás Dávid
```
```

4. fejezet

Helló, Caesar!

4.1. int *** háromszögmátrix

Megoldás videó:

Megoldás forrása:

```
#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;
    *(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;
}
```

Létrehozunk egy háromszög mátrixot. Az int nr-ben megadjuk a mátrixunk sorainak számát. A tm-mel pedig egy double-t hoztunk létre. Az ifen belül memóriát foglal a programunkm mely a double lefoglalt 8 byte-ját jelszi összeszorozva az nr értékével. Ha a lefoglalandó memóriaterület 0 NULL akkor egy -1 return értékkel kilép a program. Miután megkaptuk a tm értékét kapunk egy for ciklust, ami tartalmazza azt az if kifejezést, ami foglalja a memóriát. A ciklusban látott ciklusváltozó int i=0 megtalálható az ifes sorban is. Az if sorában a tm i-edik eleme egyenlő a double értékre mutató pointerrel. Ezzel a formulával számítjuk ki az értéket: (i+1)*sizeof(double *): 1-szer vesszük a double pointer méretét (8 byte). Ha az értéke NULL (0) , a fent leírt return -1 -gyel bezárjuk a programot. Tutor: Rémiás Dávid

4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás videó:

Megoldás forrása:

```
#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);
    }
}
```

A program ezekkel a parancsosri argumentumokkal kezd el működni: (argc, **argv) Adjunk meg egy kulcsot és egy bufferta #include alatt találhatóak alapján. A deklarációk között megtaláljuk a kulcs_meret nevű függvényt, mely a második parancsosir argumentummal lesz egyenlő. Lemásoljuk a kulcsunkat, argv[1](2. argumentum)-ot, illetve a MAX_KULCS értékét a bufferünkbe. Beolvassuk a bufferben lévő byte-okat és ez össze XOR-ozza a bufferben található byte-okat. XOR = Logikai kizáró vagy Ezt minden kulcs byte-ján elvégzi. A write funkcióval kiírjuk a buffer tartalmát és a byte-okat. Tutor: Rémiás DÁvid

4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

Megoldás forrása:

```
public class ExorTitkosító {

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

        byte [] kulcs = kulcsSzöveg.getBytes();
        byte [] buffer = new byte[256];
        int kulcsIndex = 0;
        int olvasottBájtok = 0;
        while((olvasottBájtok =
                bejövőCsatorna.read(buffer)) != -1) {

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

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

```
            }

            kimenőCsatorna.write(buffer, 0, olvasottBájtok);

        }

    }

    public static void main(String[] args) {

        try {

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

        } catch(java.io.IOException e) {

            e.printStackTrace();

        }

    }

}
```

4.4. C EXOR törő

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

Megoldás videó:

Megoldás forrása:

```
#define MAX_TITKOS 4096
#define OLVASAS_BUFFER 256
#define KULCS_MERET 8
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <string.h>
double
atlagos_szohossz (const char *titkos, int titkos_meret)
{
  int sz = 0;
  for (int i = 0; i < titkos_meret; ++i)
    if (titkos[i] == ' ')
      ++sz;
  return (double) titkos_meret / sz;
}
int
```

```
tiszta_lehet (const char *titkos, int titkos_meret)
{
  // a tiszta szoveg valszeg tartalmazza a gyakori magyar szavakat
  // illetve az átlagos szóhossz vizsgálatával csökkentjük a
  // potenciális töréseket
  double szohossz = atlagos_szohossz (titkos, titkos_meret);
  return szohossz > 6.0 && szohossz < 9.0
    && strcasestr (titkos, "hogy") && strcasestr (titkos, "nem")
    && strcasestr (titkos, "az") && strcasestr (titkos, "ha");
}
void
exor (const char kulcs[], int kulcs_meret, char titkos[], int titkos_meret)
{
  int kulcs_index = 0;
  for (int i = 0; i < titkos_meret; ++i)
    {
      titkos[i] = titkos[i] ^ kulcs[kulcs_index];
      kulcs_index = (kulcs_index + 1) % kulcs_meret;
    }
}
int
exor_tores (const char kulcs[], int kulcs_meret, char titkos[],
      int titkos_meret)
{
  exor (kulcs, kulcs_meret, titkos, titkos_meret);
  return tiszta_lehet (titkos, titkos_meret);
}
int
main (void)
{
  char kulcs[KULCS_MERET];
  char titkos[MAX_TITKOS];
  char *p = titkos;
  int olvasott_bajtok;
  // titkos fajt berantasa
  while ((olvasott_bajtok =
    read (0, (void *) p,
    (p - titkos + OLVASAS_BUFFER <
     MAX_TITKOS) ? OLVASAS_BUFFER : titkos + MAX_TITKOS - p)))
    p += olvasott_bajtok;
  // maradek hely nullazasa a titkos bufferben
  for (int i = 0; i < MAX_TITKOS - (p - titkos); ++i)
    titkos[p - titkos + i] = '\0';
  // osszes kulcs eloallitasa
  for (int ii = '0'; ii <= '9'; ++ii)
    for (int ji = '0'; ji <= '9'; ++ji)
      for (int ki = '0'; ki <= '9'; ++ki)
  for (int li = '0'; li <= '9'; ++li)
    for (int mi = '0'; mi <= '9'; ++mi)
      for (int ni = '0'; ni <= '9'; ++ni)
```

```
        for (int oi = '0'; oi <= '9'; ++oi)
    for (int pi = '0'; pi <= '9'; ++pi)
      {
        kulcs[0] = ii;
        kulcs[1] = ji;
        kulcs[2] = ki;
        kulcs[3] = li;
        kulcs[4] = mi;
        kulcs[5] = ni;
        kulcs[6] = oi;
        kulcs[7] = pi;
        if (exor_tores (kulcs, KULCS_MERET, titkos, p - titkos))
          printf
      ("Kulcs: [%c%c%c%c%c%c%c%c]\nTiszta szoveg: [%s]\n",
       ii, ji, ki, li, mi, ni, oi, pi, titkos);
        // ujra EXOR-ozunk, igy nem kell egy masodik buffer
        exor (kulcs, KULCS_MERET, titkos, p - titkos);
      }
  return 0;
}
```

A #include rész után aazonnal függvényeket deklarálunk. Az "atlagos_szohossz" egyszerű. A tiktos szó hosszát elosztja az átlag szóshosszal. tiszta_lehet (pontos megértés szükséges) Az exor a titkosítás miatt ismerőssé válhat. A main részban a p* pointert egy maximális értékig léptetjük , ami azt jelenti, hogy egyenlővé tesszük a saját plusz az olvasott byte-ok számával. Majd lefuttatun kegy úgynevezett bruteforce mechanizmust, ez előállítja a lehetséges kulcskombinációk összességét, és ezután kiírja a tiszta szöveget. Tutor: Rémiás Dávid

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

```
#    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])
```

Ez az R program egyszerű logikai műveletekre tanítja a gépünket egy neutrális hálót használva. Mindig adunk egy mintát, ez alapján a számítógép egy adott súlyt ad értékként, majd elvégzi a műveleteket elvégzi és ezt adja meg értéknek. Csak közelítő értékeket kaphatunk. (pl.: igaz érték esetén 0.9998 érték, míg hamis érték esetén 0,01) Tutor: Rémiás Dávid

4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5. fejezet

Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

Megoldás videó:

Megoldás forrása:

```
// 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)
    struct tms tmsbuf1, tmsbuf2;
    times (&tmsbuf1);
    // számítás adatai
    float a = -2.0, b = .7, c = -1.35, d = 1.35;
    int szelesseg = MERET, magassag = MERET, iteraciosHatar = ITER_HAT;
    // a számítás
    float dx = (b - a) / szelesseg;
    float dy = (d - c) / magassag;
    float reC, imC, reZ, imZ, ujreZ, ujimZ;
    // Hány iterációt csináltunk?
    int iteracio = 0;
    // Végigzongorázzuk a szélesség x magasság rácsot:
    for (int j = 0; j < magassag; ++j)
    {
        //sor = j;
        for (int k = 0; k < szelesseg; ++k)
        {
            // c = (reC, imC) a rács csomópontjainak
            // megfelelő komplex szám
            reC = a + k * dx;
            imC = d - j * dy;
            // z_0 = 0 = (reZ, imZ)
            reZ = 0;
            imZ = 0;
            iteracio = 0;
            // z_{n+1} = z_n * z_n + c iterációk
            // számítása, amíg |z_n| < 2 vagy még
            // nem értük el a 255 iterációt, ha
            // viszont elértük, akkor úgy vesszük,
            // hogy a kiinduláci c komplex számra
            // az iteráció konvergens, azaz a c a
            // Mandelbrot halmaz eleme
            while (reZ * reZ + imZ * imZ < 4 && iteracio < iteraciosHatar)
            {
                // z_{n+1} = z_n * z_n + c
                ujreZ = reZ * reZ - imZ * imZ + reC;
                ujimZ = 2 * reZ * imZ + imC;
                reZ = ujreZ;
                imZ = ujimZ;
                ++iteracio;
            }
            kepadat[j][k] = iteracio;
        }
```

```
    }
    times (&tmsbuf2);
    std::cout << tmsbuf2.tms_utime - tmsbuf1.tms_utime
              + tmsbuf2.tms_stime - tmsbuf1.tms_stime << std::endl;
    delta = clock () - delta;
    std::cout << (float) delta / CLOCKS_PER_SEC << " sec" << std::endl;
}
int
main (int argc, char *argv[])
{
    if (argc != 2)
    {
        std::cout << "Hasznalat: ./mandelpng fajlnev";
        return -1;
    }
    int kepadat[MERET][MERET];
    mandel(kepadat);
    png::image < png::rgb_pixel > kep (MERET, MERET);
    for (int j = 0; j < MERET; ++j)
    {
        //sor = j;
        for (int k = 0; k < MERET; ++k)
        {
            kep.set_pixel (k, j,
                           png::rgb_pixel (255 -
                                           (255 * kepadat[j][k]) / ITER_HAT ↩
                                              ,
                                           255 -
                                           (255 * kepadat[j][k]) / ITER_HAT ↩
                                              ,
                                           255 -
                                           (255 * kepadat[j][k]) / ITER_HAT ↩
                                              ));
        }
    }
    kep.write (argv[1]);
    std::cout << argv[1] << " mentve" << std::endl;
}
```

A kód a mandelbrot halmazt vizualizálja. a Void mandle halmazban definiáljuk a halmazt. (Matematikailag a halmaz egyik z n+1-edik elemét (xn^2)+c -képpen írhatjuk fel.) Komplex számokkal végezzük el a fenti műveletet Emiatt van egy koordináta rendszerünk, melyet griddel jelenítünk meg, és csomópontjait komplex számok alkotják. A halmaz elemeihez, a fekete színt rendeljük a kiírandó pontokhoz, a megadott határokon belül. (Az iterációs határ fontos szerepet tölt be , hiszen ha a számok nem esnek bele ebbe az iterációs határba, akkor nem elemai a halmaznak, azaz a végtelenbe is 'ugorhatnak/ugranak'.) Tutor: Rémiás Dávid

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

Megoldás videó:

Megoldás forrása: // 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  ↩
   0.213538705176874649138696327097512154281  ↩
   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

Ahhoz, hogy vizualizáljuk és prezentáljuk a fent elkészített mandelbrot halmazt elkészített programokat veszünk alapul. A "complex" headerrel ellátott programba egy képlet implementálásával biomorf képet kapunk. (jelen esetben z_n = z_n * z_n + c; ----default mandelbrot halmaz----)

5.4. A Mandelbrot halmaz CUDA megvalósítása

Megoldás videó:

Megoldás forrása:

```
// 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
//  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 <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;
}
```

A CUDA-s megoldás gyorsabb mint a fent prezentált programunk. Az első program a processzorunkat veszi igénybe. míg ez a videókártyánkat ezért a CUDA (GPU) esetében 60x60x100 szál dolgozik. Ezáltal párhuzamosan számolunk ki pontokat. A gyorsaságunk a 'delta clock()' funkció miatt még látványosabb.(időt mérünk vele) Tutor: Rémiás Dávid

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:

Megoldás videó:

Megoldás forrása:

```
// frakablak.cpp
//
// Mandelbrot halmaz nagyító
#include "frakablak.h"
FrakAblak::FrakAblak(double a, double b, double c, double d,
                     int szelesseg, int iteraciosHatar, QWidget *parent)
                         : QMainWindow(parent)
{
    setWindowTitle("Mandelbrot halmaz");
    szamitasFut = true;
    x = y = mx = my = 0;
    this->a = a;
    this->b = b;
    this->c = c;
    this->d = d;
    this->szelesseg = szelesseg;
    this->iteraciosHatar = iteraciosHatar;
```

```
    magassag = (int)(szelesseg * ((d-c)/(b-a)));
    setFixedSize(QSize(szelesseg, magassag));
    fraktal= new QImage(szelesseg, magassag, QImage::Format_RGB32);
    mandelbrot = new FrakSzal(a, b, c, d, szelesseg, magassag,  ↩
       iteraciosHatar, this);
    mandelbrot->start();
}
FrakAblak::~FrakAblak()
{
    delete fraktal;
    delete mandelbrot;
}
void FrakAblak::paintEvent(QPaintEvent*) {
    QPainter qpainter(this);
    qpainter.drawImage(0, 0, *fraktal);
    if(!szamitasFut) {
        qpainter.setPen(QPen(Qt::white, 1));
        qpainter.drawRect(x, y, mx, my);
    }
    qpainter.end();
}
void FrakAblak::mousePressEvent(QMouseEvent* event) {
    // A nagyítandó kijelölt területet bal felsõ sarka:
    x = event->x();
    y = event->y();
    mx = 0;
    my = 0;
    update();
}
void FrakAblak::mouseMoveEvent(QMouseEvent* event) {
    // A nagyítandó kijelölt terület szélessége és magassága:
    mx = event->x() - x;
    my = mx; // négyzet alakú
    update();
}
void FrakAblak::mouseReleaseEvent(QMouseEvent* event) {
    if(szamitasFut)
        return;
    szamitasFut = true;
    double dx = (b-a)/szelesseg;
    double dy = (d-c)/magassag;
    double a = this->a+x*dx;
    double b = this->a+x*dx+mx*dx;
    double c = this->d-y*dy-my*dy;
    double d = this->d-y*dy;
    this->a = a;
    this->b = b;
    this->c = c;
    this->d = d;
    delete mandelbrot;
```

```
    mandelbrot = new FrakSzal(a, b, c, d, szelesseg, magassag,  ↩
       iteraciosHatar, this);
    mandelbrot->start();
    update();
}
void FrakAblak::keyPressEvent(QKeyEvent *event)
{
    if(szamitasFut)
        return;
    if (event->key() == Qt::Key_N)
        iteraciosHatar *= 2;
    szamitasFut = true;
    delete mandelbrot;
    mandelbrot = new FrakSzal(a, b, c, d, szelesseg, magassag,  ↩
       iteraciosHatar, this);
    mandelbrot->start();
}
void FrakAblak::vissza(int magassag, int *sor, int meret)
{
    for(int i=0; i<meret; ++i) {
        QRgb szin = qRgb(0, 255-sor[i], 0);
        fraktal->setPixel(i, magassag, szin);
    }
    update();
}
void FrakAblak::vissza(void)
{
    szamitasFut = false;
    x = y = mx = my = 0;
}
```

Felveszi a programunk a mandelbrot halmaz nagyságát, ezután a kattintással létrehozott objektum kiinduló, azaz a bal felső sarkát számolja ki, és a kijelölt rész többi tagját ehhez igazítja. Az egérgomb felengedését követően felnagyítja ezt. Tutor: Rémiás Dávid A header file linkje : https://sourceforge.net/p/udprog/code/ci/mast

5.6. Mandelbrot nagyító és utazó Java nyelven

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

```
 * @version 0.0.1
 */
public class MandelbrotHalmazNagyító extends MandelbrotHalmaz {
    /** A nagyítandó kijelölt területet bal felső sarka. */
    private int x, y;
    /** A nagyítandó kijelölt terület szélessége és magassága. */
    private int mx, my;
    /**
     * Létrehoz egy a Mandelbrot halmazt a komplex sík
     * [a,b]x[c,d] tartománya felett kiszámoló és nygítani tudó
     * <code>MandelbrotHalmazNagyító</code> objektumot.
     *
     * @param      a              a [a,b]x[c,d] tartomány a koordinátája.
     * @param      b              a [a,b]x[c,d] tartomány b koordinátája.
     * @param      c              a [a,b]x[c,d] tartomány c koordinátája.
     * @param      d              a [a,b]x[c,d] tartomány d koordinátája.
     * @param      szélesség      a halmazt tartalmazó tömb szélessége.
     * @param      iterációsHatár a számítás pontossága.
     */
    public MandelbrotHalmazNagyító(double a, double b, double c, double d,
            int szélesség, int iterációsHatár) {
        // Az ős osztály konstruktorának hívása
        super(a, b, c, d, szélesség, iterációsHatár);
        setTitle("A Mandelbrot halmaz nagyításai");
        // Egér kattintó események feldolgozása:
        addMouseListener(new java.awt.event.MouseAdapter() {
            // Egér kattintással jelöljük ki a nagyítandó területet
            // bal felső sarkát:
            public void mousePressed(java.awt.event.MouseEvent m) {
                // A nagyítandó kijelölt területet bal felső sarka:
                x = m.getX();
                y = m.getY();
                mx = 0;
                my = 0;
                repaint();
            }
            // Vonszolva kijelölünk egy területet...
            // Ha felengedjük, akkor a kijelölt terület
            // újraszámítása indul:
            public void mouseReleased(java.awt.event.MouseEvent m) {
                double dx = (MandelbrotHalmazNagyító.this.b
                        - MandelbrotHalmazNagyító.this.a)
                        /MandelbrotHalmazNagyító.this.szélesség;
                double dy = (MandelbrotHalmazNagyító.this.d
                        - MandelbrotHalmazNagyító.this.c)
                        /MandelbrotHalmazNagyító.this.magasság;
                // Az új Mandelbrot nagyító objektum elkészítése:
                new MandelbrotHalmazNagyító(MandelbrotHalmazNagyító.this.a+ ↩
                   x*dx,
                        MandelbrotHalmazNagyító.this.a+x*dx+mx*dx,
```

```
                        MandelbrotHalmazNagyító.this.d-y*dy-my*dy,
                        MandelbrotHalmazNagyító.this.d-y*dy,
                        600,
                        MandelbrotHalmazNagyító.this.iterációsHatár);
            }
        });
        // Egér mozgás események feldolgozása:
        addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
            // Vonszolással jelöljük ki a négyzetet:
            public void mouseDragged(java.awt.event.MouseEvent m) {
                // A nagyítandó kijelölt terület szélessége és magassága:
                mx = m.getX() - x;
                my = m.getY() - y;
                repaint();
            }
        });
    }
    /**
     * Pillanatfelvételek készítése.
     */
    public void pillanatfelvétel() {
        // Az elmentendő kép elkészítése:
        java.awt.image.BufferedImage mentKép =
                new java.awt.image.BufferedImage(szélesség, magasság,
                java.awt.image.BufferedImage.TYPE_INT_RGB);
        java.awt.Graphics g = mentKép.getGraphics();
        g.drawImage(kép, 0, 0, this);
        g.setColor(java.awt.Color.BLUE);
        g.drawString("a=" + a, 10, 15);
        g.drawString("b=" + b, 10, 30);
        g.drawString("c=" + c, 10, 45);
        g.drawString("d=" + d, 10, 60);
        g.drawString("n=" + iterációsHatár, 10, 75);
        if(számításFut) {
            g.setColor(java.awt.Color.RED);
            g.drawLine(0, sor, getWidth(), sor);
        }
        g.setColor(java.awt.Color.GREEN);
        g.drawRect(x, y, mx, my);
        g.dispose();
        // A pillanatfelvétel képfájl nevének képzése:
        StringBuffer sb = new StringBuffer();
        sb = sb.delete(0, sb.length());
        sb.append("MandelbrotHalmazNagyitas_");
        sb.append(++pillanatfelvételSzámláló);
        sb.append("_");
        // A fájl nevébe belevesszük, hogy melyik tartományban
        // találtuk a halmazt:
        sb.append(a);
        sb.append("_");
```

```
        sb.append(b);
        sb.append("_");
        sb.append(c);
        sb.append("_");
        sb.append(d);
        sb.append(".png");
        // png formátumú képet mentünk
        try {
            javax.imageio.ImageIO.write(mentKép, "png",
                    new java.io.File(sb.toString()));
        } catch(java.io.IOException e) {
            e.printStackTrace();
        }
    }
    /**
     * A nagyítandó kijelölt területet jelző négyzet kirajzolása.
     */
    public void paint(java.awt.Graphics g) {
        // A Mandelbrot halmaz kirajzolása
         g.drawImage(kép, 0, 0, this);
        // Ha éppen fut a számítás, akkor egy vörös
        // vonallal jelöljük, hogy melyik sorban tart:
        if(számításFut) {
            g.setColor(java.awt.Color.RED);
            g.drawLine(0, sor, getWidth(), sor);
        }
        // A jelző négyzet kirajzolása:
        g.setColor(java.awt.Color.GREEN);
        g.drawRect(x, y, mx, my);
    }
    /**
     * Példányosít egy Mandelbrot halmazt nagyító obektumot.
     */
    public static void main(String[] args) {
        // A kiinduló halmazt a komplex sík [-2.0, .7]x[-1.35, 1.35]
        // tartományában keressük egy 600x600-as hálóval és az
        // aktuális nagyítási pontossággal:
        new MandelbrotHalmazNagyító(-2.0, .7, -1.35, 1.35, 600, 255);
    }
}
```

6. fejezet

Helló, Welch!

6.1. Első osztályom

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:

```
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <ctime>
class PolarGen
    {
public:
 PolarGen ()
 {
   nincsTarolt = true;
   std::srand (std::time (NULL));
  }
   ~PolarGen ()
  {
  }
  double kovetkezo ()
 {
   if (nincsTarolt)
      {
       double u1, u2, v1, v2, w;
       do
  {
      u1 = std::rand () / (RAND_MAX + 1.0);
      u2 = std::rand () / (RAND_MAX + 1.0);
     v1 = 2 * u1 - 1;
```

```
        v2 = 2 * u2 - 1;
        w = v1 * v1 + v2 * v2;
       }
           while (w > 1);
         double r = std::sqrt ((-2 * std::log (w)) / w);
         tarolt = r * v2;
          nincsTarolt = !nincsTarolt;
         return r * v1;
       }
      else
       {
          nincsTarolt = !nincsTarolt;
          return tarolt;
        }
    }
  private:
   bool nincsTarolt;
    double tarolt;
  };
  int
  main (int argc, char **argv)
  {
   PolarGen pg;
   for (int i = 0; i < 10; ++i)
     std::cout << pg.kovetkezo () << std::endl;
    return 0;
  }
  "
        </para>
        <para>
           "
                           import java.util.Random;
  public class PolarGen
  {
    private double tarolt;
  private boolean nincsTarolt;
  private Random r;
  private int RAND_MAX;

  public PolarGen()
  {
  nincsTarolt = true;
  r = new Random();
  r.setSeed(20);
  this.RAND_MAX=100;
  }
  public PolarGen(Integer RAND_MAX)
  {
  nincsTarolt = true;
  r = new Random();
```

```
    r.setSeed(20);
    this.RAND_MAX=RAND_MAX;
    }

    public double kovetkezo()
    {
      if (nincsTarolt)
        {

      double u1, u2, v1, v2, w;
      int i=0;
          do
        {
            u1 = r.nextInt() / (RAND_MAX + 1.0);
            u2 = r.nextInt() / (RAND_MAX + 1.0);
            v1 = 2 * u1 - 1;
            v2 = 2 * u2 - 1;
            w = v1 * v1 + v2 * v2;
        }
        while (w > 1 && i++ < 40000000);
          double r = Math.sqrt ((2 * Math.log10(w)) / w);
          tarolt = r * v2;
          nincsTarolt = !nincsTarolt;
          return r * v1;
       }
        else
        {
          nincsTarolt = !nincsTarolt;
          return tarolt;
        }
      }
    }
```

Egy olyan class-t hozunk letre, melyben meg tudjuk adni, hogy nincs szám tárolva, és a Random Number Generatornak véletlenszerű értéket adunk. A program megvizsgálja, hogy van-e szám, ha nincs, kér 2 számot, melyekből az egyiket tárolja, a változóját hamisra változtaja, ezután a második számot adja vissza. A megoldás JAVA nyelvben is hasonló. Tutoriált: Rémiás Dávid

6.2. LZW

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

Megoldás videó:

Megoldás forrása:

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

```
typedef struct node{
char c;
struct node* left;
struct node* right;
} Node;
Node* fa;
Node gyoker;
#define null NULL
Node* create_empty()
{
Node* tmp = &gyoker;
tmp->c= '/';
tmp->left = null;
tmp->right = null;
return tmp;
}
Node* create_node(char val)
{
Node* tmp = (Node*)malloc(sizeof(Node));
tmp->c=val;
tmp->left = null;
tmp->right = null;
return tmp;
}
void insert_tree(char val)
{
if(val=='0')
{
if(fa->left == null)
{
  fa->left = create_node(val);
  fa = &gyoker;
  //printf("Inserted into left.");
}
else
{
  fa = fa->left;
}
}
else
{
if(fa->right == null)
{
  fa->right = create_node(val);
  fa = &gyoker;
  //printf("Inserted into left.");
}
else
{
  fa = fa->right;
```

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

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

    }
    else
    {
      free(elem);
    }
    }
    void usage()
    {
    printf("Használat: ./binfa KAPCSOLÓ\n");
    printf("Az KAPCSOLÓ lehet:\n");
```

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

Csinálunk egy struktúrát melynek a Node nevet adjuk,ez jelzi a binfában lévő csomópontokat. EZ rendelkezik egy változóval, és két mutatóval. A mutatók az egyesre és nullásra mutatnak.

Egy függvény inicializál egy kitüntetett gyökérelemmel, ami '/'-el van megjelenítve.

A függvény csomópontot hoz létre.

A insert_tree csinálja a binfát. Megnézi, hogy 0 van-e, ha igen, megvizsgálja, hogy a pontoknak melyekre a mutató mutat van-e 0-s gyermeke. Ha van, akkor lép oda a mutató. Ha nincs, generál egyet, majda gyökérre állítja a mutatót. Ha az érték nem 0, akkor az utasítások végrehajtódnak, az ópont jobb gyermekére.

A fét inroder módon járja be. Így először a bal oldalon megyünk végig, majd feldolgozzuk a részfának a gyökér elemét, ezután a fa jobb oldalát.

A postorder módon bejárás a destroy_tree-nek köszönhető. Ez a rekurzió végén szabaddá teszi a gyökérelemet. Tutoriált: Rémiás Dávid

6.3. Fabejárás

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

Megoldás videó:

Megoldás forrása:

```
            "#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
typedef struct node{
  char c;
  struct node* left;
  struct node* right;
} Node;
Node* fa;
Node gyoker;
#define null NULL
Node* create_empty()
{
  Node* tmp = &gyoker;
  tmp->c= '/';
  tmp->left = null;
  tmp->right = null;
  return tmp;
}
Node* create_node(char val)
{
  Node* tmp = (Node*)malloc(sizeof(Node));
  tmp->c=val;
  tmp->left = null;
  tmp->right = null;
  return tmp;
}
void insert_tree(char val)
{
```

```
  if(val=='0')
  {
    if(fa->left == null)
    {
      fa->left = create_node(val);
      fa = &gyoker;
      //printf("Inserted into left.");
    }
    else
    {
      fa = fa->left;
    }
  }
  else
  {
    if(fa->right == null)
    {
      fa->right = create_node(val);
      fa = &gyoker;
      //printf("Inserted into left.");
    }
    else
    {
      fa = fa->right;
    }
  }
}
void inorder(Node* elem,int depth)
{
  if(elem==null)
  {
    return;
  }
  inorder(elem->left,depth+1);
  if(depth)
  {
    char *spaces;
    spaces =(char*) malloc(sizeof(char)*depth*2+1);
    for(int i=0;i<depth;i+=2)
    {
      spaces[i]='-';
      spaces[i+1]='-';
    }
    spaces[depth]='\0';
    printf("%s%c\n",spaces,elem->c);
  }
  else
  {
    printf("%c\n",elem->c);
  }
```

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

```
    printf("%c\n",elem->c);
  }
}
void destroy_tree(Node* elem)
{
  if(elem==null)
  {
    return;
  }
  destroy_tree(elem->left);
  destroy_tree(elem->right);
  if(elem->c == gyoker.c)
  {
  }
  else
  {
    free(elem);
  }
}
void usage()
{
  printf("Használat: ./binfa KAPCSOLÓ\n");
  printf("Az KAPCSOLÓ lehet:\n");
  printf("--preorder\tA bináris fa preorder bejárása\n");
  printf("--inorder\tA bináris fa inorder bejárása\n");
  printf("--postorder\tA bináris fa postorder bejárása\n");
}
int main(int argc, char** argv)
{
  srand(time(null));
  fa = create_empty();
  //gyoker = *fa;
  for(int i=0;i<10000;i++)
  {
    int x=rand()%2;
    if(x)
    {
      insert_tree('1');
    }
    else
    {
      insert_tree('0');
    }
  }
  if(argc == 2)
  {
    if(strcmp(argv[1],"--preorder")==0)
    {
      preorder(&gyoker,0);
    }
```

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

Preorder bejárásnál a fa gyökérelemit dolgozzuk fel , ezután bejárjuk a részfa bal oldalát majd utána a jobb oldalát.

A postorder bejárás során a fa bal, majd jobb oldalán és végül a gyökerén megyünk végig.

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! Tutoriált: Rémiás Dávid

Megoldás videó:

Megoldás forrása:

```
            "#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string.h>
#define null NULL
class Binfa
{
private:
  class Node
  {
  public:
    Node(char c='/')
    {
      this->c=c;
```

```
      this->left = null;
      this->right = null;
    }
    char c;
    Node* left;
    Node* right;
  };
  Node* fa;

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

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

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

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

```
    if(depth)
    {
      char *spaces;
      spaces =(char*) malloc(sizeof(char)*depth*2+1);
      for(int i=0;i<depth;i+=2)
      {
        spaces[i]='-';
        spaces[i+1]='-';
      }
      spaces[depth*2]='\0';
      printf("%s%c\n",spaces,elem->c);
    }
    else
    {
      printf("%c\n",elem->c);
    }
  }
  void destroy_tree(Node* elem)
  {
    if(elem==null)
    {
      return;
    }
    destroy_tree(elem->left);
    destroy_tree(elem->right);
    if(elem->c!='/') delete elem;
  }
  Node gyoker;
};
void usage()
{
  printf("Használat: ./binfa KAPCSOLÓ\n");
  printf("Az KAPCSOLÓ lehet:\n");
  printf("--preorder\tA bináris fa preorder bejárása\n");
  printf("--inorder\tA bináris fa inorder bejárása\n");
  printf("--postorder\tA bináris fa postorder bejárása\n");
}
int main(int argc, char** argv)
{
  srand(time(0));
  Binfa bfa;
  for(int i=0;i<100;i++)
  {
    int x=rand()%2;
    if(x)
    {
      bfa<<'1';
    }
    else
    {
```

```
      bfa<<'0';
    }
  }
  if(argc == 2)
  {
    if(strcmp(argv[1],"--preorder")==0)
    {
      bfa.preorder(&bfa.gyoker);
    }
    else if(strcmp(argv[1],"--inorder")==0)
    {
      bfa.inorder(&bfa.gyoker);
    }
    else if(strcmp(argv[1],"--postorder")==0)
    {
      bfa.postorder(&bfa.gyoker);
    }
    else
    {
      usage();
    }
  }
  else
  {
    usage();
  }
  bfa.destroy_tree(&bfa.gyoker);
  return 0;
}
">Binfa c++
```

A függvények a binfa osztályában vannak elhelyezve. A Noda privát a fán belül. A fa építését a bitshioft operator veszi át.

6.5. Mutató a gyökér

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

Megoldás videó:

Megoldás forrása:

```
          "#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string.h>
#define null NULL
class Binfa
{
```

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

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

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

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

```
  void postorder(Node* elem,int depth=0)
  {
    if(elem==null)
    {
      return;
    }
    postorder(elem->left,depth+1);
    postorder(elem->right,depth+1);
    if(depth)
    {
      char *spaces;
      spaces =(char*) malloc(sizeof(char)*depth*2+1);
      for(int i=0;i<depth;i+=2)
      {
        spaces[i]='-';
        spaces[i+1]='-';
      }
      spaces[depth*2]='\0';
      printf("%s%c\n",spaces,elem->c);
    }
    else
    {
      printf("%c\n",elem->c);
    }
  }
  void destroy_tree(Node* elem)
  {
    if(elem==null)
    {
      return;
    }
    destroy_tree(elem->left);
    destroy_tree(elem->right);
    if(elem->c!='/') delete elem;
  }
  Node* gyoker;
};
void usage()
{
  printf("Használat: ./binfa KAPCSOLÓ\n");
  printf("Az KAPCSOLÓ lehet:\n");
  printf("--preorder\tA bináris fa preorder bejárása\n");
  printf("--inorder\tA bináris fa inorder bejárása\n");
  printf("--postorder\tA bináris fa postorder bejárása\n");
}
int main(int argc, char** argv)
{
  srand(time(0));
  Binfa bfa;
  for(int i=0;i<100;i++)
```

```
  {
    int x=rand()%2;
    if(x)
    {
      bfa<<'1';
    }
    else
    {
      bfa<<'0';
    }
  }
  if(argc == 2)
  {
    if(strcmp(argv[1],"--preorder")==0)
    {
      bfa.preorder(bfa.gyoker);
    }
    else if(strcmp(argv[1],"--inorder")==0)
    {
      bfa.inorder(bfa.gyoker);
    }
    else if(strcmp(argv[1],"--postorder")==0)
    {
      bfa.postorder(bfa.gyoker);
    }
    else
    {
      usage();
    }
  }
  else
  {
    usage();
  }
  bfa.destroy_tree(bfa.gyoker);
  return 0;
}
"
```

Módosítottunk az előző feladaton annyit, hogy a bejárás miatt létrehoztun kegy gyökérobjektumot. Ezelőtt, ahol a program azt kérte hogy a gyökér referenciáját adjuk meg, most a gyökeret magát kell megadnunk. Tutoriált: Rémiás Dávid

6.6. Mozgató szemantika

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

Megoldás videó:

Megoldás forrása:

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:

Tanulságok, tapasztalatok, magyarázat...

7.2. Java életjáték

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7.3. Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó: // sejtablak.cpp

```
//
// Életjáték rajzoló
// Programozó Páternoszter
//
// Copyright (C) 2011, Bátfai Norbert, nbatfai@inf.unideb.hu, nbatfai@gmail ↩
   .com
//
```

```
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License
// along with this program.  If not, see <http://www.gnu.org/licenses/>.
//
// Ez a program szabad szoftver; terjeszthető illetve módosítható a
// Free Software Foundation által kiadott GNU General Public License
// dokumentumában leírtak; akár a licenc 3-as, akár (tetszőleges) későbbi
// változata szerint.
//
// Ez a program abban a reményben kerül közreadásra, hogy hasznos lesz,
// de minden egyéb GARANCIA NÉLKÜL, az ELADHATÓSÁGRA vagy VALAMELY CÉLRA
// VALÓ ALKALMAZHATÓSÁGRA való származtatott garanciát is beleértve.
// További részleteket a GNU General Public License tartalmaz.
//
// A felhasználónak a programmal együtt meg kell kapnia a GNU General
// Public License egy példányát; ha mégsem kapta meg, akkor
// tekintse meg a <http://www.gnu.org/licenses/> oldalon.
//
//
// Version history:
//
// 0.0.1    A két osztály tervezésének fő szempontja az volt, hogy
// ne vagy alig különbözzön az első C++-os példától, a Mandelostól:
// http://progpater.blog.hu/2011/02/26/ ↩
   tan_csodallak_amde_nem_ertelek_de_kepzetem_hegyvolgyedet_bejarja
// ezért az olyan kényesebb dolgokkal, hogy kezeljük a racsIndex-et a
// két osztályra bontott C++ megoldásban, amikor írjuk át a Javásból, nem  ↩
   foglalkoztunk
// a kiinduló Javás: http://www.tankonyvtar.hu/informatika/javat-tanitok ↩
   -1-2-080904-1
// (a bazár eszme: Release Early, Release Often" írjuk ki a posztra)
//
#include "sejtablak.h"
SejtAblak::SejtAblak(int szelesseg, int magassag, QWidget *parent)
: QMainWindow(parent)
{
  setWindowTitle("A John Horton Conway-féle életjáték");

  this->magassag = magassag;
  this->szelesseg = szelesseg;
```

```
  cellaSzelesseg = 6;
  cellaMagassag = 6;
  setFixedSize(QSize(szelesseg*cellaSzelesseg, magassag*cellaMagassag));

  racsok = new bool**[2];
  racsok[0] = new bool*[magassag];
  for(int i=0; i<magassag; ++i)
    racsok[0][i] = new bool [szelesseg];
  racsok[1] = new bool*[magassag];
  for(int i=0; i<magassag; ++i)
    racsok[1][i] = new bool [szelesseg];
  racsIndex = 0;
  racs = racsok[racsIndex];
  // A kiinduló racs minden cellája HALOTT
  for(int i=0; i<magassag; ++i)
    for(int j=0; j<szelesseg; ++j)
      racs[i][j] = HALOTT;
    // A kiinduló racsra "ELOlényeket" helyezünk
    //siklo(racs, 2, 2);
    sikloKilovo(racs, 5, 60);
  eletjatek = new SejtSzal(racsok, szelesseg, magassag, 120, this);
  eletjatek->start();

}
void SejtAblak::paintEvent(QPaintEvent*) {
  QPainter qpainter(this);

  // Az aktuális
  bool **racs = racsok[racsIndex];
  // racsot rajzoljuk ki:
  for(int i=0; i<magassag; ++i) { // végig lépked a sorokon
    for(int j=0; j<szelesseg; ++j) { // s az oszlopok
      // Sejt cella kirajzolása
      if(racs[i][j] == ELO)
  qpainter.fillRect(j*cellaSzelesseg, i*cellaMagassag,
        cellaSzelesseg, cellaMagassag, Qt::black);
  else
    qpainter.fillRect(j*cellaSzelesseg, i*cellaMagassag,
          cellaSzelesseg, cellaMagassag, Qt::white);
    qpainter.setPen(QPen(Qt::gray, 1));

  qpainter.drawRect(j*cellaSzelesseg, i*cellaMagassag,
        cellaSzelesseg, cellaMagassag);
    }
  }

  qpainter.end();
}
SejtAblak::~SejtAblak()
{
```

```
  delete eletjatek;

  for(int i=0; i<magassag; ++i) {
    delete[] racsok[0][i];
    delete[] racsok[1][i];
  }

  delete[] racsok[0];
  delete[] racsok[1];
  delete[] racsok;


}
void SejtAblak::vissza(int racsIndex)
{
  this->racsIndex = racsIndex;
  update();
}
/**
 * A sejttérbe "ELOlényeket" helyezünk, ez a "sikló".
 * Adott irányban halad, másolja magát a sejttérben.
 * Az ELOlény ismertetését lásd például a
 * [MATEK JÁTÉK] hivatkozásban (Csákány Béla: Diszkrét
 * matematikai játékok. Polygon, Szeged 1998. 172. oldal.)
 *
 * @param   racs    a sejttér ahová ezt az állatkát helyezzük
 * @param   x       a befoglaló tégla bal felső sarkának oszlopa
 * @param   y       a befoglaló tégla bal felső sarkának sora
 */
void SejtAblak::siklo(bool **racs, int x, int y) {

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

}
/**
 * A sejttérbe "ELOlényeket" helyezünk, ez a "sikló ágyú".
 * Adott irányban siklókat lő ki.
 * Az ELOlény ismertetését lásd például a
 * [MATEK JÁTÉK] hivatkozásban /Csákány Béla: Diszkrét
 * matematikai játékok. Polygon, Szeged 1998. 173. oldal./,
 * de itt az ábra hibás, egy oszloppal told még balra a
 * bal oldali 4 sejtes négyzetet. A helyes ágyú rajzát
 * lásd pl. az [ÉLET CIKK] hivatkozásban /Robert T.
 * Wainwright: Life is Universal./ (Megemlíthetjük, hogy
 * mindkettő tartalmaz két felesleges sejtet is.)
 *
```

```
 * @param   racs    a sejttér ahová ezt az állatkát helyezzük
 * @param   x       a befoglaló tégla bal felső sarkának oszlopa
 * @param   y       a befoglaló tégla bal felső sarkának sora
 */
void SejtAblak::sikloKilovo(bool **racs, int x, int y) {

  racs[y+ 6][x+ 0] = ELO;
  racs[y+ 6][x+ 1] = ELO;
  racs[y+ 7][x+ 0] = ELO;
  racs[y+ 7][x+ 1] = ELO;

  racs[y+ 3][x+ 13] = ELO;

  racs[y+ 4][x+ 12] = ELO;
  racs[y+ 4][x+ 14] = ELO;

  racs[y+ 5][x+ 11] = ELO;
  racs[y+ 5][x+ 15] = ELO;
  racs[y+ 5][x+ 16] = ELO;
  racs[y+ 5][x+ 25] = ELO;

  racs[y+ 6][x+ 11] = ELO;
  racs[y+ 6][x+ 15] = ELO;
  racs[y+ 6][x+ 16] = ELO;
  racs[y+ 6][x+ 22] = ELO;
  racs[y+ 6][x+ 23] = ELO;
  racs[y+ 6][x+ 24] = ELO;
  racs[y+ 6][x+ 25] = ELO;

  racs[y+ 7][x+ 11] = ELO;
  racs[y+ 7][x+ 15] = ELO;
  racs[y+ 7][x+ 16] = ELO;
  racs[y+ 7][x+ 21] = ELO;
  racs[y+ 7][x+ 22] = ELO;
  racs[y+ 7][x+ 23] = ELO;
  racs[y+ 7][x+ 24] = ELO;

  racs[y+ 8][x+ 12] = ELO;
  racs[y+ 8][x+ 14] = ELO;
  racs[y+ 8][x+ 21] = ELO;
  racs[y+ 8][x+ 24] = ELO;
  racs[y+ 8][x+ 34] = ELO;
  racs[y+ 8][x+ 35] = ELO;

  racs[y+ 9][x+ 13] = ELO;
  racs[y+ 9][x+ 21] = ELO;
  racs[y+ 9][x+ 22] = ELO;
  racs[y+ 9][x+ 23] = ELO;
  racs[y+ 9][x+ 24] = ELO;
  racs[y+ 9][x+ 34] = ELO;
```

```
  racs[y+ 9][x+ 35] = ELO;

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

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

}
```

Megoldás forrása:

Létrehottuk a COnway-féle sejtautomatát a program segítségével. AZ automatát egy griden ábrázoljuk, mely eldönti, hogy élő-e a sejt vagy nem. Egy sejtautomata akkor élő ha 2 szomszédja nem élő és 3 élő. Ha egy üres cellának 3 élő szomszédja van a cella helyére élő sejt kerül. TUTOR: Rémiás Dávid

7.4. BrainB Benchmark

Megoldás videó:

Megoldás forrása:

```
/**
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBThread.cpp
 * @author  Norbert Bátfai <nbatfai@gmail.com>
 * @version 6.0.1
 *
 * @section LICENSE
 *
 * Copyright (C) 2017, 2018 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/>.
 *
 * @section DESCRIPTION
 *
 */
```

```
#include "BrainBThread.h"
BrainBThread::BrainBThread ( int w, int h )
{
        dispShift = heroRectSize+heroRectSize/2;
        this->w = w - 3 * heroRectSize;
        this->h = h - 3 * heroRectSize;
        std::srand ( std::time ( 0 ) );
        Hero me ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ) -  ↩
           100,
                this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0 ) -  ↩
                   100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), 9 );
        Hero other1 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
           ) - 100,
                    this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
                       ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ↩
                        5, "Norbi Entropy" );
        Hero other2 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
           ) - 100,
                    this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
                       ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ↩
                        3, "Greta Entropy" );
        Hero other4 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
           ) - 100,
                    this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
                       ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ↩
                        5, "Nandi Entropy" );
        Hero other5 ( this->w / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
           ) - 100,
                    this->h / 2 + 200.0 * std::rand() / ( RAND_MAX + 1.0  ↩
                       ) - 100, 255.0 * std::rand() / ( RAND_MAX + 1.0 ), ↩
                        7, "Matyi Entropy" );
        heroes.push_back ( me );
        heroes.push_back ( other1 );
        heroes.push_back ( other2 );
        heroes.push_back ( other4 );
        heroes.push_back ( other5 );
}
BrainBThread::~BrainBThread()
{
}
void BrainBThread::run()
{
        while ( time < endTime ) {
                QThread::msleep ( delay );
                if ( !paused ) {
                        ++time;
                        devel();
                }
                draw();
        }
```

```
        emit endAndStats ( endTime );
}
void BrainBThread::pause()
{
        paused = !paused;
        if ( paused ) {
                ++nofPaused;
        }
}
void BrainBThread::set_paused ( bool p )
{
        if ( !paused && p ) {
                ++nofPaused;
        }
        paused = p;
}
```

A program jelentős részét a header file képezi. A headerben a hősök tulajdonságait találjuk, tartózkodási hely x és y koordinátákkal, valamint egy "agility" érték, amely a hősünk mozgásáért felelős (hány egységet léphet egyszerre). //a header pontos megértése szükséges A cpp fájlban 4 játékost láthatunk, valamint egy pause funkciót. Tutor: Rémiás Dávid

8. fejezet

Helló, Schwarzenegger!

8.1. Szoftmax Py MNIST

Python

Megoldás videó: https://youtu.be/j7f9SkJR3oc

Megoldás forrása: https://github.com/tensorflow/tensorflow/releases/tag/v0.9.0 (/tensorflow-0.9.0/tensorflow/exa
https://progpater.blog.hu/2016/11/13/hello_samu_a_tensorflow-bol

Tanulságok, tapasztalatok, magyarázat...

FIGYELEM
A futtáshoz telepíteni kell a Tensorflow megfelelő csomagjait és a Python3 fejlesztői csomagokat is. Kövesd az általad használt Linux disztribúció telepítési útmutatóját hozzá!

Jelenleg(2019.04.21.) így telepíthetjük egy Ubuntu disztrón:

```
sudo apt install python3-dev python3-pip
sudo pip3 install -U virtualenv  # system-wide install
virtualenv --system-site-packages -p python3 ./venv
source ./venv/bin/activate  # sh, bash, ksh, or zsh
pip install --upgrade pip
pip install --upgrade tensorflow
#ellenőrizzük, hogy helyesen települt-e
python -c "import tensorflow as tf; tf.enable_eager_execution(); print(tf. ↩
   reduce_sum(tf.random_normal([1000, 1000])))"
```

Az MNIST egy viszonylag nagy adatbázis ami tanitóanyagokat biztosít , például, olyan hálózatok(szoftveres) tanítására amiknek feladata, hogy képelemzési technikákkal és háttértudások(tanult minták alapján) segítségével elemezni tudjanak bemeneti képeket. Az adatbázis kézírott arab számokat tartalmaz.

A Tensorflow egy nyílt forrású matematikai könyvtár amit a gépi tanulásban, például neurális hálózatoknál, használják. Mivel ezeknek a témaköröknek rengeteg erőforrásra van szükséges, hogy érdemi munkát hajtsanak végre a Tensorflow képes egyszerre több CPU-n és GPU-n futni(például CUDA segítségével).

Tensorflow vagy hasonló könyvtár nélkül is implementálhatunk különféle gépi tanuláshoz kapcsolódó algoritmust akármilyen programozási nyelvben. Azonban Tensorflow már eleve optimalizált és tesztelt algoritmusokat biztosít a számításainkhoz, saját kóddal *from scratch* nem 100%, hogy elérjük a teljesítményüket. Egyedül a neurális hálózat megírása marad ránk.

Az adatbázist felhasználva 60.000 képpel fogjuk kioktatni a hálózatunkat és 10.000-rel megpróbáljuk megközelíteni, hogy mennyire pontos volt.

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import argparse
# Import data
from tensorflow.examples.tutorials.mnist import input_data
import tensorflow as tf
old_v = tf.logging.get_verbosity()
tf.logging.set_verbosity(tf.logging.ERROR)
import matplotlib.pyplot
FLAGS = None
def main(_):
  mnist = input_data.read_data_sets(FLAGS.data_dir, one_hot=True)
  # Create the model
  x = tf.placeholder(tf.float32, [None, 784])
  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))
  y = tf.matmul(x, W) + b
```

Az adatok (MNIST adatbázis) behívása és eltárolása után inicializáljuk lépésekben a Tensorflow könyvtárat.

Egy neurális hálózat rétegekből épül fel, lásd Perceptron feladat, ezeket a rétegeket be kell állítanunk annak megfelelően ami el akarunk érni a neurális hálózat működése során. A réteg a bemeneti adatokból különféle információkat olvasnak ki.

Létrehozunk egy tenzort `x`, ebbe fogjuk küldeni az értékeket, beállítjuk, hogy 784 pixelt tartalmazó képeket fog kapni. Ezeken felül létrehozzuk az `y` értékeit adó függvényhez még szükséges `W` (súly) és `b` (bias) változókat. Az y értékei 0-1. Ezzel a softmax modelt hoztuk létre.

```
  # Define loss and optimizer
  y_ = tf.placeholder(tf.float32, [None, 10])
  # The raw formulation of cross-entropy,
  #
  #   tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(tf.nn.softmax(y)),
  #                                 reduction_indices=[1]))
  #
  # can be numerically unstable.
  #
  # So here we use tf.nn.softmax_cross_entropy_with_logits on the raw
```

```
# outputs of 'y', and then average across the batch.
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits( ↩
    labels = y_, logits = y))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize( ↩
    cross_entropy)
```

A `cross_entropy` sorban a *cost* függvényt állítjuk be, ez arra szolgál, hogy megtudjuk mennyi a különbség az eredeti érték és a becsült érték között, ezt az értékét akarjuk minimálizálni.

Lépésekben történik a tanítás és egy beépített `GradientDescentOptimizer` függvényt használunk fel tanításra. Ez a függvény úgy működik, hogy van egy kezdeti érték amit frissítget egészen addig amíg a *cost* függvény el nem ér egy minimumot, például a legmagasabb pontossági értéket. Ebből látjuk, hogy iteráció függő lesz ez a kalkuláció.

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

Inicializáljuk a sessionünket, feldaraboljuk az adatot `batch`-ekre tehát részekre és futtatjuk a modellünket. Közben a jelenlegi pontosságot is kiiratjuk.

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

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

img = mnist.test.images[42]
image = img
matplotlib.pyplot.imshow(image.reshape(28, 28), cmap=matplotlib.pyplot.cm ↩
    .binary)
matplotlib.pyplot.savefig("4.png")
matplotlib.pyplot.show()
classification = sess.run(tf.argmax(y, 1), feed_dict={x: [image]})
print("-- Ezt a halozat ennek ismeri fel: ", classification[0])
print("------------------------------------------------------------")
```

```
  print("-- A MNIST 11. tesztkepenek felismerese, mutatom a szamot, a ↩
     tovabblepeshez csukd be az ablakat")
  img = mnist.test.images[11]
  image = img
  matplotlib.pyplot.imshow(image.reshape(28,28), cmap=matplotlib.pyplot.cm. ↩
     binary)
  matplotlib.pyplot.savefig("8.png")
  matplotlib.pyplot.show()
  classification = sess.run(tf.argmax(y, 1), feed_dict={x: [image]})
  print("-- Ezt a halozat ennek ismeri fel: ", classification[0])
  print("----------------------------------------------------------")
if __name__ == '__main__':
  parser = argparse.ArgumentParser()
  parser.add_argument('--data_dir', type=str, default='/tmp/tensorflow/ ↩
     mnist/input_data',
                      help='Directory for storing input data')
  FLAGS = parser.parse_args()
  tf.app.run()
```

Az elkészült hálózatunkat teszteljük az MNIST adatbázis néhány képével, az output alapján sikeresen működik és működtettük a Tensorflow projektünket.

A kiválaszott képet megjelenítjük külön ablakban is, hogy összetudjuk mi is hasonlítani a képen látható objektumot az eredmény szerinti objektummal.

8.2. Mély MNIST

PASSZOLÁS 1 - UDPROG SMINST FOR HUMANS feladat alapján
Az UDPROG SMINST for humans bejegyzés feladatát teljesítettem, elértem a level 10-s szintet a felmérésben, így felhasználom egyik passzolási lehetőségemet ennél a feladatnál. Összesen 2 passzolás járt lvl 9 és lvl 10-ért lásd lenti link.

BEJEGYZÉS POSTJÁNAK LINKJE

Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.3. Minecraft-MALMÖ

PASSZOLÁS 2 - UDPROG SMINST FOR HUMANS feladat alapján
Az UDPROG SMINST for humans bejegyzés feladatát teljesítettem, elértem a level 10-s szintet a felmérésben, így felhasználom az utolsó passzolási lehetőségemet ennél a feladatnál. Összesen 2 passzolás járt lvl 9 és lvl 10-ért lásd lenti link.

BEJEGYZÉS POSTJÁNAK LINKJE

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

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9. fejezet

Helló, Chaitin!

9.1. Iteratív és rekurzív faktoriális Lisp-ben

Megoldás videó: https://youtu.be/z6NJE2a1zIA

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

Tanulságok, tapasztalatok, magyarázat...

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

Tanulságok, tapasztalatok, magyarázat...

10. fejezet

Helló, Gutenberg!

10.1. Programozási alapfogalmak

[?] z emberek folyton modellekben gondolkodnak, főleg problémák megoldása közben. Ez teljes mértékben igaz a programozásra is, mivel ebben problémákat kell megoldanunk, melyek megoldását nagyban elősegíti egy előre felállított modell, működési elv. A számítógépek processzorai egy előre meghatározott nyelvet felismernek, viszont több magas szintű programnyelv létezik, ezért az ezen nyelveken írt programokat le kell fordítanunk, erre szolgálnak segítségül a fordító porgramok (gcc, g++ ...stb.) Ezek a fordítóporgramok lexerrel avagy lexikális elemlzővel felszereltek, melyek a lefordítandó programkódot hivatottak ellenőrizni. Illetve a fellépő hibákat (legyen az szintaktikai, vagy más egyéb) a felhasználó/programozó felé jelezni. A programnyelveket osztályokba sorolhatjuk , miszerint léteznek : - Imperatív nyelvek : algoritmikus nyelv, utasítások sorozata - Deklaratív nyelvek: pl.: logikai nyelvek - Más nyelvek : ide azokat soroljuk, melyek egy imperatív tényezőt/tulajdonságot tagadnak

10.2. Programozás bevezetés

[KERNIGHANRITCHIE]

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

10.3. Programozás

[BMECPP] Minden programkód, karakterekből áll elő, ezek egy adott karakterkészletből érhetőek el, minden nyelv definiálja a sajátját. (betűk , számjegyek, stb..) Ily módon épülnek fel a lexikális -, a szintaktikai -, a program -, a fordítási egységek, az utasítások, illetve maga a program. A programkódon belüli lexikális egységek, melyeket a fordítóprogram tokenizál a következőek : Többkarakteres szimbólum, szimbolikus név, címke, megjegyzés, literál. A szimbolikus neveken belül találkozhatunk az azonosítóval, melynek kötelezően betűvel kell kezdődnie, majd betűvel vagy számmal folytatódnia. Programozói eszközök megnevezésére szolgál, majd a későbbiekben az ezekre való hivatkozást is megkönnyíti. Találkozhatunk még a programnyelvekben kulcsszavakkal, melyeknek az adott nyelv tulajdonít jelentést, tehát azonosítói szempontból kötöttek. Általában utasítások elnevezései. Léteznek még standard azonosítók, melyeknek szintén

az adott programnyelv tualjdonít jelentést a kulcsszavakhoz hasonló módon, ezt a programozó átnevezheti, átértelmezheti (C-ben ilyen például a NULL). A programkódok egy hasznos egységét képezik a megjegyzések, kommentek. Ezek segítik a programozónak egy adott szegmens működési elvének leírását, ha esetleg hosszabb a kód és a későbbiekben vissza kell térni rá, de már a programozó nem teljesen emlékszik rá. Problémák feljegyzése adott szegmensekkel kapcsolatban. A kommentek szinte bármilyen nagyságúak lehetnek, nem befolyásolják a program működését.

III. rész

Második felvonás

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

11. fejezet

Helló, Arroway!

11.1. A BPP algoritmus Java megvalósítása

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

11.2. Java osztályok a Pi-ben

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

IV. rész

Irodalomjegyzék

11.3. Általános

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

11.4. C

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

11.5. C++

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

11.6. Lisp

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

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

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