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, Bátfai, Mátyás, Bátfai, Nándor, Ã©s Bátfai, Margaréta	2020. november 23.	

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

Ábrák jegyzéke

Előszó

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

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

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

Hogyan forgasd

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

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

Milyen nyelven nyomjuk?

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

Hogyan nyomjuk?

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

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

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

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

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

I. rész

Bevezetés

1. fejezet

Vízió

1.1. Mi a programozás?

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

Azt nem tudom, hogy mit jelent a program írása, de arról már kezd lenni egy kis tudásom hogy hogyan kell programot írni és az mindig egy irányba mutat egy bonyolultnak tűnő problémát kell olyan szintű kis elemi lépésekre leírni, hogy egy olyan buta készülék ami csak tranzisztorokból épül fel végre tudja hajtani. De ez az eljárás minden esetben egy jobb megértést biztosít mintha csak meg tanultam volna az adott problémáról szóló dolgokat.

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.

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: https://youtu.be/lvmi6tyz-nI

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/infty-f.c, bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/infty-w.c.

Számos módon hozhatunk és hozunk létre végtelen ciklusokat. Vannak esetek, amikor ez a célunk, például egy szerverfolyamat fusson folyamatosan és van amikor egy bug, mert ott lesz végtelen ciklus, ahol nem akartunk. Saját péláinkban ilyen amikor a PageRank algoritmus rázza az 1 liter vizet az internetben, de az iteráció csak nem akar konvergálni...

A **top** segitségével tudjuk monitorozni a programok CPU ki használását.

Egy mag 100 százalékban:

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

  return 0;
}
```

vagy az olvashatóbb, de a programozók és fordítók (szabványok) között kevésbé hordozható

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

```
  return 0;
}
```

```
top - 09:52:14 up 44 min,  1 user,  load average: 0,58, 0,50, 0,37
Tasks: 240 total,   2 running, 183 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0,6 us,  0,6 sy,  0,0 ni, 94,0 id,  0,0 wa,  0,0 hi,  4,7 si,  ↩
   0,0 s
%Cpu1  :  1,7 us,  0,7 sy,  0,0 ni, 97,7 id,  0,0 wa,  0,0 hi,  0,0 si,  ↩
   0,0 s
%Cpu2  :  1,0 us,  0,7 sy,  0,0 ni, 97,3 id,  0,0 wa,  0,0 hi,  1,0 si,  ↩
   0,0 s
%Cpu3  :100,0 us,  0,0 sy,  0,0 ni,  0,0 id,  0,0 wa,  0,0 hi,  0,0 si,  ↩
   0,0 s
KiB Mem :  3945424 total,   155968 free,  2224096 used,  1565360 buff/cache
KiB Swap:  2017304 total,  1829144 free,   188160 used.  1133028 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 4155 j         20   0    4372    768    704 R 100,0  0,0   0:20.92 l
```

Jól látszik, hogy a program csak egy magot használ a négyből.

Azért érdemes a `for(;;)` hagyományos formát használni, mert ez minden C szabvánnyal lefordul, másrészt a többi programozó azonnal látja, hogy az a végtelen ciklus szándékunk szerint végtelen és nem szoftverhiba. Mert ugye, ha a `while`-al trükközünk egy nem triviális `1` vagy `true` feltétellel, akkor ott egy másik, a forrást olvasó programozó nem látja azonnal a szándékunkat.

Egyébként a fordító a `for`-os és `while`-os ciklusból ugyanazt az assembly kódot fordítja:

```
$ gcc -S -o infty-f.S infty-f.c
$ gcc -S -o infty-w.S infty-w.c
$ diff infty-w.S infty-f.S
1c1
<   .file "infty-w.c"
---
>   .file "infty-f.c"
```

Egy mag 0 százalékban:

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

  return 0;
}
```

Ennel a feladatnál a program futatása során nem tapasztalunk semmi érdekeset a top segitségével, esetleg anyit hogy a sleeping programok száma meg ugrik eggyel.

Minden mag 100 százalékban:

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

A **gcc infty-f.c -o infty-f -fopenmp** parancssorral készíthető el a futtathatót program.

```
top - 20:09:06 up  3:35,  1 user,  load average: 5,68, 2,91, 1,38
Tasks: 329 total,   2 running, 256 sleeping,   0 stopped,   1 zombie
%Cpu0 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu1 : 99,7 us, 0,3 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu2 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu3 : 99,7 us, 0,3 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu4 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu5 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu6 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%Cpu7 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem :16373532 total,11701240 free, 2254256 used, 2418036 buff/cache
KiB Swap:16724988 total,16724988 free,       0 used. 13751608 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 5850 batfai    20   0   68360    932    836 R 798,3  0,0   8:14.23 infty-f
```

Werkfilm

- https://youtu.be/lvmi6tyz-nI

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:

```
        Változó csere
#include <iostream>

int main()
{
  int a = 4;
  int b = 6;
  std::cout << "a=" << a << " b=" << b << std::endl;

  //seged valtozo segitsegevel

  int c;

  c = a;
  a = b;
  b = c;
  std::cout << "a=" << a << " b=" << b << std::endl;



  //müveletek segitsegevel

  a = a + b;
```

```
  b = a - b;
  a = a - b;
  std::cout << "a=" << a << " b=" << b << std::endl;

  //EXOR segitsegevel
  a = a ^ b;
  b = a ^ b;
  a = a ^ b;
  std::cout << "a=" << a << " b=" << b << std::endl;




}
```

Tanulságok, tapasztalatok, magyarázat...

Két változó felcserélését több fajta képen végre lehet hajtani ezek ebből csak pár példát mutattam be. Az első példa egy segéd változót használ, a többi pedig műveletek segítségével cseréli fel a két változót.

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:

```
Labda patogtatas
{

#include <stdio.h>
#include <curses.h>
#include <unistd.h>

int main ( void )
{
    WINDOW *ablak;
    ablak = initscr ();

    int x = 1;
    int y = 1;

    int xnov = 1;
    int ynov = 1;
```

```
    int mx;
    int my;

  for (;; ) {

    getmaxyx(ablak, my, mx);

    mvprintw(y, x, "O");

    int direction[mx][my][2];//0 = az x ; 1 = az y irányal

    for (int i = 0; i < mx; i++) {
      for (int j = 0; j < my; j++) {
        direction[i][j][0] = 1;
        direction[i][j][1] = 1;
      }
    }

    for (int i = 0; i < my; i++) {
      direction[0][i][0] = -1;
      direction[mx - 1][i][0] = -1;
    }
    for (int i = 0; i < mx; i++) {
      direction[i][0][1] = -1;
      direction[i][my-1][1] = -1;
    }

        refresh ();
        usleep ( 100000 );
    clear();

        x = x + xnov;
        y = y + ynov;

    xnov = xnov * direction[x][y][0];
    ynov = ynov * direction[x][y][1];

    }

    return 0;
}
```

Tanulságok, tapasztalatok, magyarázat...

Az alap ötletet az adta hogy az if -es megoldásban xnov, ynov -1 gyel lett szorozva nem pedig egy szerü érték adással lett meg változtatva. Ebből ki folyólag kezdtem el gondolkodni azon, hogy hogyan tudnám meg forditani a az irányvektorokat az ablaki szélein. Egy "térképet" sikerült alkotni ami meg adja hogy adott pontba merre kell továb haladnija a labdának ez a térképem a 3D tömb, az első két dimenzió az adott pont x y kordi nátája a harmadik pedig a továb haladás irányát adja meg. A ciklus mag minden egyes

lefutásánál feltöltésre kerül a tömb ez nem feltétlenül optimális de így tud rea gálni az ablak méretének a meg változására (ha használhatnánk if-et akkor egy egyszerü fetétell meg lehetne vizsgálni hogy meg változott e az ablak mérete és csak akkor feltölteni ha igen).

2.5. Szóhossz és a Linus Torvalds féle BogoMIPS

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

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

Bemutató videó: https://youtu.be/YZe6VTHaTeM

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Turing/bogomips.c

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




int
main (void)
{

  int i=1;
  int j=1;


  printf ("int hosza..");
  fflush (stdout);


  while ((i<<=1))
    {
    j++;

  }

  printf("%u\n",j);
  return 0;



  }
```

Ez a kis kód meg adja hogy hány biten van tárolva az int típusú változó azaz a szó hosszt. Ez az én gépemen 32 bit volt. A program működése: az eltolás operátor rltolja az i-ben a biteke balra míg ez végre hajtható, ha már ez nem végrehajtható akkor ez azt jelenti, hogy szó hossznyiszor toltuk el

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:

```
        Helló, Google!
#include <stdio.h>
#include <math.h>

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

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

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

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

  long int i, j;
  i = 0; j = 0;
  double h;
```

```
  for (;;)
  {
    for (i = 0; i < 4; i++)
      PR[i] = PRv[i];
    for (i = 0; i < 4; i++)
    {
      double temp = 0;
      for (j = 0; j < 4; j++)
        temp += L[i][j] * PR[j];
      PRv[i] = temp;
    }

    if (tavolsag(PR, PRv, 4) < 0.00001)
      break;
  }
  kiir(PR, 4);
  return 0;

}
```

tav += (pagerank[i] - pagerank_temp[i])* (pagerank[i] - pagerank_temp[i]); a kordináták külömb ségének négyzet összege

```
        for (i = 0; i < 4; i++)
    {
      double temp = 0;
      for (j = 0; j < 4; j++)
        temp += L[i][j] * PR[j];
      PRv[i] = temp;
    }
```

A mátrix sorzás meg valósitása

Ezek a szorzások adják meg az egyes oldalak page rank értékét

2.7. A Monty Hall probléma

Írj R szimulációt a Monty Hall problémára!

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

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

Tanulságok, tapasztalatok, magyarázat...

2.8. 100 éves a Brun tétel

Írj R szimulációt a Brun tétel demonstrálására!

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

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

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

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

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

Majd vizsgáljuk meg az a sorozatot:

(n+1)!+2, (n+1)!+3,... , (n+1)!+n, (n+1)!+ (n+1) ez n db egymást követő azám, ezekre (a jól ismert bizonyítás szerint) rendre igaz, hogy

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

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

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

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

Ebben a példában egy olyan programot készítettünk, amely közelíteni próbálja a Brun konstans értékét. A repó `bhax/attention_raising/Primek_R/stp.r` mevű állománya kiszámolja az ikerprímeket, összegzi a reciprokaikat és vizualizálja a kapott részeredményt.

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

library(matlab)

stp <- function(x){

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

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

Soronként értelemezzük ezt a programot:

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

Kiszámolja a megadott számig a prímeket.

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

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

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

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

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

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

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

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

Kivette a primes-ból a párok első tagját.

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

A párok második tagját az első tagok kettő hozzáadásával képezzük.

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

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

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

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

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

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

```
> 1/t1primes+1/t2primes
[1] 0.5333333 0.3428571 0.1678322
```

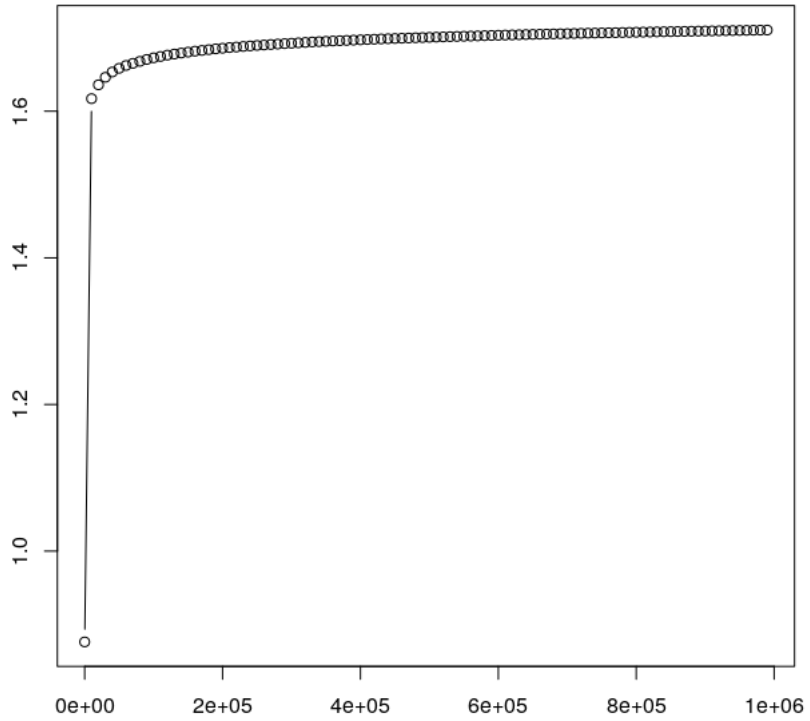
Nincs más dolgunk, mint ezeket a törteket összeadni a `sum` függvénnyel.

```
sum(rt1plust2)
```

```
>   sum(rt1plust2)
[1] 1.044023
```

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

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



2.1. ábra. A B_2 konstans közelítése

Werkfilm

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

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfjával megadva írd meg ezt a gépet!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

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

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

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:

```
<utasítás>:=
<címkézett_ utasítás>
<kifejezésutasítás>
<összetett_utasítás>
<kiválasztó_utasítás>
```

```
      <iterációs_utasítás>
      <vezérlésátadó_ utasítás>

<címkézett_utasítás>:=
      <azonosító> := utasítás
      <case állandó_kifejezés> := utasítás
      <default> := utasítás

<kifejezésutasítás>:=
      <kifejezés(opc)>;

<összetett_utasítás>:=
      { <deklarációs_listaopc> <utasítás_lista(opc)> }

<deklarációs_lista>:=
      <deklaráció>
      <deklarációs_lista> | <deklaráció>

<utasítás_lista>:=
      <utasítás>
      <utasítás_lista> | <utasítás>

<kiválasztó_utasítás>:=
      "if (" <kifejezés> ")" <utasítás>
      "if (" <kifejezés> ")" <utasítás> "else" <utasítás>
      "switch (" <kifejezés> ")" <utasítás>

<iterációs_utasítás>:=
      "while" ( <kifejezés> ) <utasítás>
      "do" <utasítás> "while" ( <kifejezés> ) ";"
      "for" <(kifejezés(opc)> ";" <(kifejezés(opc)> ";" <(kifejezés(opc)> " ↩
         )" <utasítás>

<Vezérlésátadó_utasítás>:=
      "goto" <azonosító> ";"
      "continue ;"
      "break ;"
      "return" <kifejezés(opc)> ";"
```

```
int main(){
//int x=2;
int b[4];
}
```

```
j@j-HP-250-G6-Notebook-PC:~/Desktop/proba$ gcc c89_99.c -o c89 -std=c89
c89_99.c: In function ‘main’:
c89_99.c:2:1: error: C++ style comments are not allowed in ISO C90
 //int x=2;
 ^
c89_99.c:2:1: error: (this will be reported only once per input file)
```

Az ilyen fajta commentelést a c99 ben hoztákbe.

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: https://youtu.be/9KnMqrkj_kU (15:01-től).

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/realnumber.l

```
%{
#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;
}
```

A program 3 részre osztható ezeket %-jelek választják el egymástól az első a definíciós rész a második a szabályok meg adására szolgál a harmadik pedig a programunk amit kibővít a lexer. {digit}+(\.{digit}+)? A számokat keresi meg. yytext: az épen vizsgált inputt. atof pedig stingből számot képez.

megoldottam vele a negatív számok keresését is

Bemutató videó: https://youtu.be/J2R-Ge-5v2A

```
%{
#include <stdio.h>
int realnumbers = 0;
%}
digit [0-9]
%%
(-{digit})?{digit}*(\.{digit}+)?  {++realnumbers;
    printf("[realnum=%s %f]", yytext, atof(yytext));}
%%
int
main ()
{
 yylex ();
```

```
 printf("The number of real numbers is %d\n", realnumbers);
 return 0;
}
```

3.5. Leetspeak

Lexelj össze egy l33t ciphert!

Megoldás videó: https://youtu.be/06C_PqDpD_k

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/l337d1c7.l

```
%{
  #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;
}
```

4 b3m3n3t1 szöV3g k4r4kt3r31'|' k1 cs3rél1 4 m3g4d0tt k4r4|{'|'3r3k v4l4m3ly1kér3 vél3tl3n sz3rű3n

(A bemeneti szöveg karaktereit ki cseréli a megadott karakterek valamelyikére véletlen szerűen)

Itt is mint az előző feladatokban óriások vállán állunk és a lexet használjuk ami meg könnyíti a program írását és utána a program meg értését. Egy a lex által értelmezhető kódot írunk amiből majd a lex készíti el a c forrást

A Leetspeak egy az internet közössége által használt szleng. Maga a program már ahogy fentebb is említettem a bemeneti szöveg karaktereit cseréli ki de nem teljesen véletlen szerűen az első megadott karakter 90%-kos esésjel az azt követő karakterek pedig egy re keveseb esélyel cserélődnek ki a szövegben.

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...

i 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.(megegyezik a példával)

ii 0-tól 4-ig egyelsével növelve (5ször fut le)

iii 0-tól 4-ig egyelsével növelve (5ször fut le)

iv tomb 0. elemét nem tudjuk az 1. ben 0 van 2. ban 1 és íg tovább

v i kisebb mint n és a d és s amire mutatt változó +1 egyenlő ek

vi ki írja az f fügvény viszatérési értékét a ,a+1 és a+1, a val

vii ki írja f fügvény viszatérési értékét a-val és az "a"-t

viii ki írja f fügvény viszatérési értékét a-val(az "a" referenc van átadva) és az "a"-t

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

Tanulságok, tapasztalatok, magyarázat...

minden számnál van egy nagyob prímszám

minden számhoz létezik olyan a számnál nagyob szám aminél a ketővel nagyob szám is prím

létezik egy olyan szám aminél minden prím kisebb

létezik egy olyan y szám aminél nagyob számok nem prímek

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:

Az utolsó két deklarációs példa demonstrálására két olyan kódot írtunk, amelyek összahasonlítása azt mutatja meg, hogy miért érdemes a **typedef** használata: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/fptr.c, bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Chomsky/fptr2.c.

```
#include <stdio.h>

int
sum (int a, int b)
{
    return a + b;
}

int
mul (int a, int b)
{
    return a * b;
}

int (*sumormul (int c)) (int a, int b)
{
    if (c)
        return mul;
    else
        return sum;

}

int
main ()
{

    int (*f) (int, int);

    f = sum;
```

```
    printf ("%d\n", f (2, 3));

    int (*(*g) (int)) (int, int);

    g = sumormul;

    f = *g (42);

    printf ("%d\n", f (2, 3));

    return 0;
}
```

```
#include <stdio.h>

typedef int (*F) (int, int);
typedef int (*(*G) (int)) (int, int);

int
sum (int a, int b)
{
    return a + b;
}

int
mul (int a, int b)
{
    return a * b;
}

F sumormul (int c)
{
    if (c)
        return mul;
    else
        return sum;
}

int
main ()
{

    F f = sum;

    printf ("%d\n", f (2, 3));

    G g = sumormul;

    f = *g (42);
```

```
    printf ("%d\n", f (2, 3));

    return 0;
}
```

Tanulságok, tapasztalatok, magyarázat...

4. fejezet

Helló, Caesar!

4.1. double ** háromszögmátrix

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

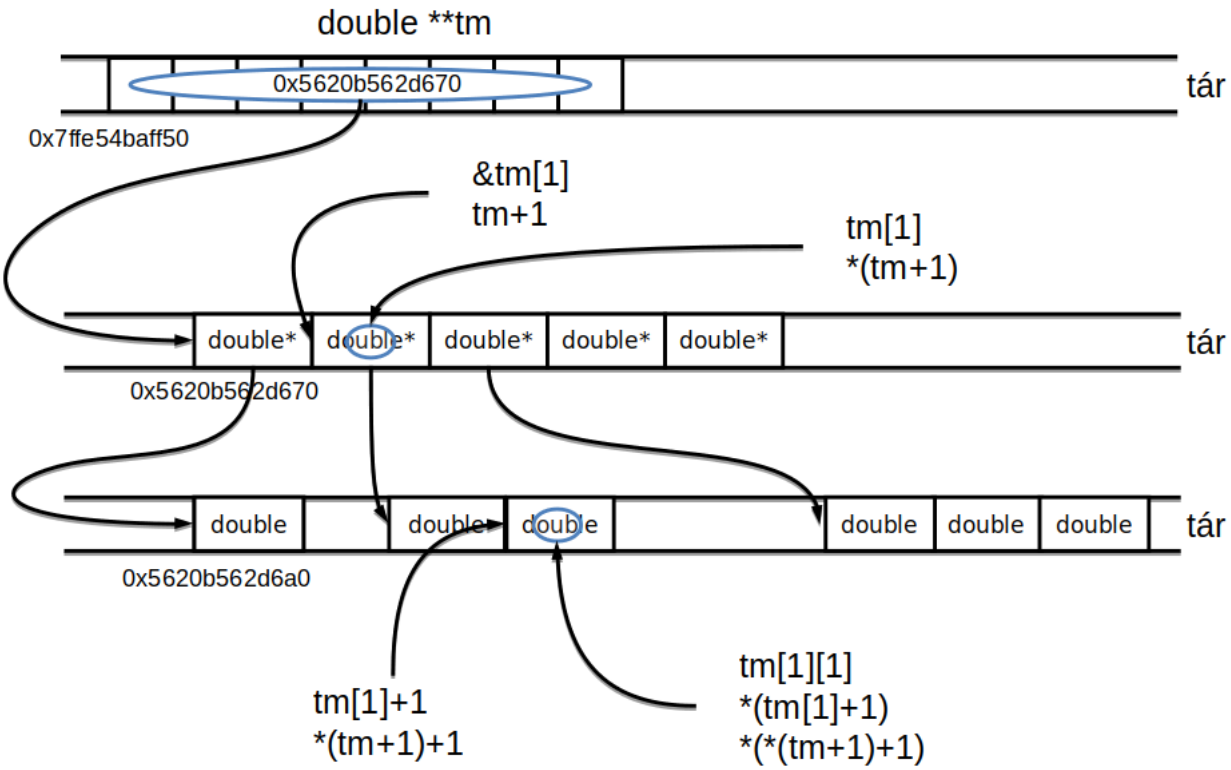
Megoldás videó: https://youtu.be/1MRTuKwRsB0, https://youtu.be/RKbX5-EWpzA.

Magyarázó videóm: https://www.youtube.com/watch?v=xEXzjmik9Do,

Megoldás forrása: bhax/thematic_tutorials/bhax_textbook_IgyNeveldaProgramozod/Caesar/tm.c

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

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

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

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

    }
```

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

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

    tm[3][0] = 42.0;
    (*(tm + 3))[1] = 43.0;  //mi van, ha itt hiányzik a küls˝o ()4.sor 0. ↩
       dik elemébe  irja bele
    *(tm[3] + 2) = 44.0;
    *(*(tm + 3) + 3) = 45.0;

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

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

    free (tm);

    return 0;
}
```



4.1. ábra. A double ** háromszögmátrix a memóriában

Tanulságok, tapasztalatok, magyarázat...

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 induláskor meg kapja kulcsot és a titkosítandó szöveget majd majd a kulcs meg és a titkosítandó szöveg között el végzi az exor műveletet (amikor kulcs végére ér újra kezdi azt míg a titkosítandó szöveg végére nem ér). Ez után pedig ki írja az immáron titkosított szöveget.

4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

Megoldás forrása: https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/ch01.html#exor_titkosito

Tanulságok, tapasztalatok, magyarázat...

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
```

```c
#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 törő szisztematikusan végig megy az összes lehetséges kulcson ebben az esetben egy 8 karakter hoszu szám sorozaton. Minden egyes kulccsal meg próbálja dekódolni és ez után a dekódolt szövegben meg nézi az átlagos szó hosszt és pár gyakori szót (ezek nyelv specifikus adatok). Ezek alapján válogatja ki a lehetséges helyes kulcsokat.

4.5. Neurális OR, AND és EXOR kapu

R

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

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

Tanulságok, tapasztalatok, magyarázat...

4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó: https://youtu.be/XpBnR31BRJY

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

```
#include <iostream>
#include "mlp.hpp"
#include <png++/png.hpp>

int main(int argc, char **argv)
{
  png::image <png::rgb_pixel> png_image(argv[1]);

  int size = png_image.get_width() * png_image.get_height();

  Perceptron* p = new Perceptron(3, size, 256, 1);

  double* image = new double[size];

  for (int i = 0; i < png_image.get_width(); ++i) {
    for (int j = 0; j < png_image.get_height(); ++j) {
```

```
      image[i * png_image.get_width() + j] = png_image[i][j].red;
    }
  }

  double value = (*p) (image);

  std::cout << value << std::endl;

  delete p;
  delete[] image;
}
```

```
/**
* @brief JUDAH - Jacob is equipped with a text-based user interface
*
* @file ql.hpp
* @author  Norbert Bátfai <nbatfai@gmail.com>
* @version 0.0.1
*
* @section LICENSE
*
* Copyright (C) 2015 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 <http://www.gnu.org/licenses/>.
*
* @section DESCRIPTION
*
* JACOB, https://github.com/nbatfai/jacob
*
* "The son of Isaac is Jacob." The project called Jacob is an experiment
* to replace Isaac's (GUI based) visual imagination with a character ↩
   console.
*
* ISAAC, https://github.com/nbatfai/isaac
*
* "The son of Samu is Isaac." The project called Isaac is a case study
* of using deep Q learning with neural networks for predicting the next
* sentence of a conversation.
*
```

```
 * SAMU, https://github.com/nbatfai/samu
 *
 * The main purpose of this project is to allow the evaluation and
 * verification of the results of the paper entitled "A disembodied
 * developmental robotic agent called Samu Bátfai". It is our hope
 * that Samu will be the ancestor of developmental robotics chatter
 * bots that will be able to chat in natural language like humans do.
 *
 */
#include <iostream>
#include <cstdarg>
#include <map>
#include <iterator>
#include <cmath>
#include <random>
#include <limits>
#include <fstream>
class Perceptron
{
public:
  Perceptron(int nof, ...)
  {
    n_layers = nof;
    units = new double* [n_layers];
    n_units = new int[n_layers];
    va_list vap;
    va_start(vap, nof);
    for (int i{ 0 }; i < n_layers; ++i)
    {
      n_units[i] = va_arg(vap, int);
      if (i)
        units[i] = new double[n_units[i]];
    }
    va_end(vap);
    weights = new double** [n_layers - 1];
#ifndef RND_DEBUG
    std::random_device init;
    std::default_random_engine gen{ init() };
#else
    std::default_random_engine gen;
#endif
    std::uniform_real_distribution<double> dist(-1.0, 1.0);
    for (int i{ 1 }; i < n_layers; ++i)
    {
      weights[i - 1] = new double* [n_units[i]];
      for (int j{ 0 }; j < n_units[i]; ++j)
      {
        weights[i - 1][j] = new double[n_units[i - 1]];
        for (int k{ 0 }; k < n_units[i - 1]; ++k)
        {
```

```
          weights[i - 1][j][k] = dist(gen);
        }
      }
    }
  }
  Perceptron(std::fstream& file)
  {
    file >> n_layers;
    units = new double* [n_layers];
    n_units = new int[n_layers];
    for (int i{ 0 }; i < n_layers; ++i)
    {
      file >> n_units[i];
      if (i)
        units[i] = new double[n_units[i]];
    }
    weights = new double** [n_layers - 1];
    for (int i{ 1 }; i < n_layers; ++i)
    {
      weights[i - 1] = new double* [n_units[i]];
      for (int j{ 0 }; j < n_units[i]; ++j)
      {
        weights[i - 1][j] = new double[n_units[i - 1]];
        for (int k{ 0 }; k < n_units[i - 1]; ++k)
        {
          file >> weights[i - 1][j][k];
        }
      }
    }
  }
  double sigmoid(double x)
  {
    return 1.0 / (1.0 + exp(-x));
  }
  double operator() (double image[])
  {
    units[0] = image;
    for (int i{ 1 }; i < n_layers; ++i)
    {
#ifdef CUDA_PRCPS
      cuda_layer(i, n_units, units, weights);
#else
#pragma omp parallel for
      for (int j = 0; j < n_units[i]; ++j)
      {
        units[i][j] = 0.0;
        for (int k = 0; k < n_units[i - 1]; ++k)
        {
          units[i][j] += weights[i - 1][j][k] * units[i - 1][k];
        }
```

```
        units[i][j] = sigmoid(units[i][j]);
      }
#endif
    }
    return sigmoid(units[n_layers - 1][0]);
  }
  void learning(double image[], double q, double prev_q)
  {
    double y[1]{ q };
    learning(image, y);
  }


  void learning(double image[], double y[])
  {
    //( *this ) ( image );
    units[0] = image;
    double** backs = new double* [n_layers - 1];
    for (int i{ 0 }; i < n_layers - 1; ++i)
    {
      backs[i] = new double[n_units[i + 1]];
    }
    int i{ n_layers - 1 };
    for (int j{ 0 }; j < n_units[i]; ++j)
    {
      backs[i - 1][j] = sigmoid(units[i][j]) * (1.0 - sigmoid(units[i][j])) ↩
          * (y[j] - units[i][j]);
      for (int k{ 0 }; k < n_units[i - 1]; ++k)
      {
        weights[i - 1][j][k] += (0.2 * backs[i - 1][j] * units[i - 1][k]);
      }
    }
    for (int i{ n_layers - 2 }; i > 0; --i)
    {
#pragma omp parallel for
      for (int j = 0; j < n_units[i]; ++j)
      {
        double sum = 0.0;
        for (int l = 0; l < n_units[i + 1]; ++l)
        {
          sum += 0.19 * weights[i][l][j] * backs[i][l];
        }
        backs[i - 1][j] = sigmoid(units[i][j]) * (1.0 - sigmoid(units[i][j ↩
           ])) * sum;
        for (int k = 0; k < n_units[i - 1]; ++k)
        {
          weights[i - 1][j][k] += (0.19 * backs[i - 1][j] * units[i - 1][k ↩
             ]);
        }
      }
```

```
    }
    for (int i{ 0 }; i < n_layers - 1; ++i)
    {
      delete[] backs[i];
    }
    delete[] backs;
  }

  ~Perceptron()
  {
    for (int i{ 1 }; i < n_layers; ++i)
    {
      for (int j{ 0 }; j < n_units[i]; ++j)
      {
        delete[] weights[i - 1][j];
      }
      delete[] weights[i - 1];
    }
    delete[] weights;
    for (int i{ 0 }; i < n_layers; ++i)
    {
      if (i)
        delete[] units[i];
    }
    delete[] units;
    delete[] n_units;
  }
  void save(std::fstream& out)
  {
    out << " "
      << n_layers;
    for (int i{ 0 }; i < n_layers; ++i)
      out << " " << n_units[i];
    for (int i{ 1 }; i < n_layers; ++i)
    {
      for (int j{ 0 }; j < n_units[i]; ++j)
      {
        for (int k{ 0 }; k < n_units[i - 1]; ++k)
        {
          out << " "
            << weights[i - 1][j][k];
        }
      }
    }
  }
private:
  Perceptron(const Perceptron&);
  Perceptron& operator= (const Perceptron&);
  int n_layers;
  int* n_units;
```

```
  double** units;
  double*** weights;
};
```

Létre hozunk egy három rétegű hálót (az első réteg a kép pontjainak száma a második 256 az utolsó pedig 1 elemü) aminek az első rétege kép pontjainak rgb kódjának a red értékéből töltjük fel a következő rétegeket pedig az előző réteg pontjaiból és a hozzájuk való súly összegével. Az utolsó egy elemű rétegben ki alakuló értéket pedig ki íratjuk a képernyőre.

Az mlp.hpp-ben több példát látunk a double** és double*** használatára. A double** -ban a units-okat tároljuk először lefoglalunk benne annyi double * ahány rétegünk van azután pedig rétegenként lefoglaljuk a meg felelő mennyiségű dobule-t. A double*** weighst-ban a súlyokat tároljuk először lefoglalunk rétegek -1 dobule ** -ot azért kel ennyit mert csak a rétegek között vannak súlyok az ezután lefoglalt dobule* a meg felelő rétegekben található unitoknak a száma ezután pedig lefoglaljuk a megfelelő unitokhoz tartozó súlyt. Egy unithoz a következő rétegben lévő unitokkal meg egyező mennyiségű súlyt kell lefoglalni.

5. fejezet

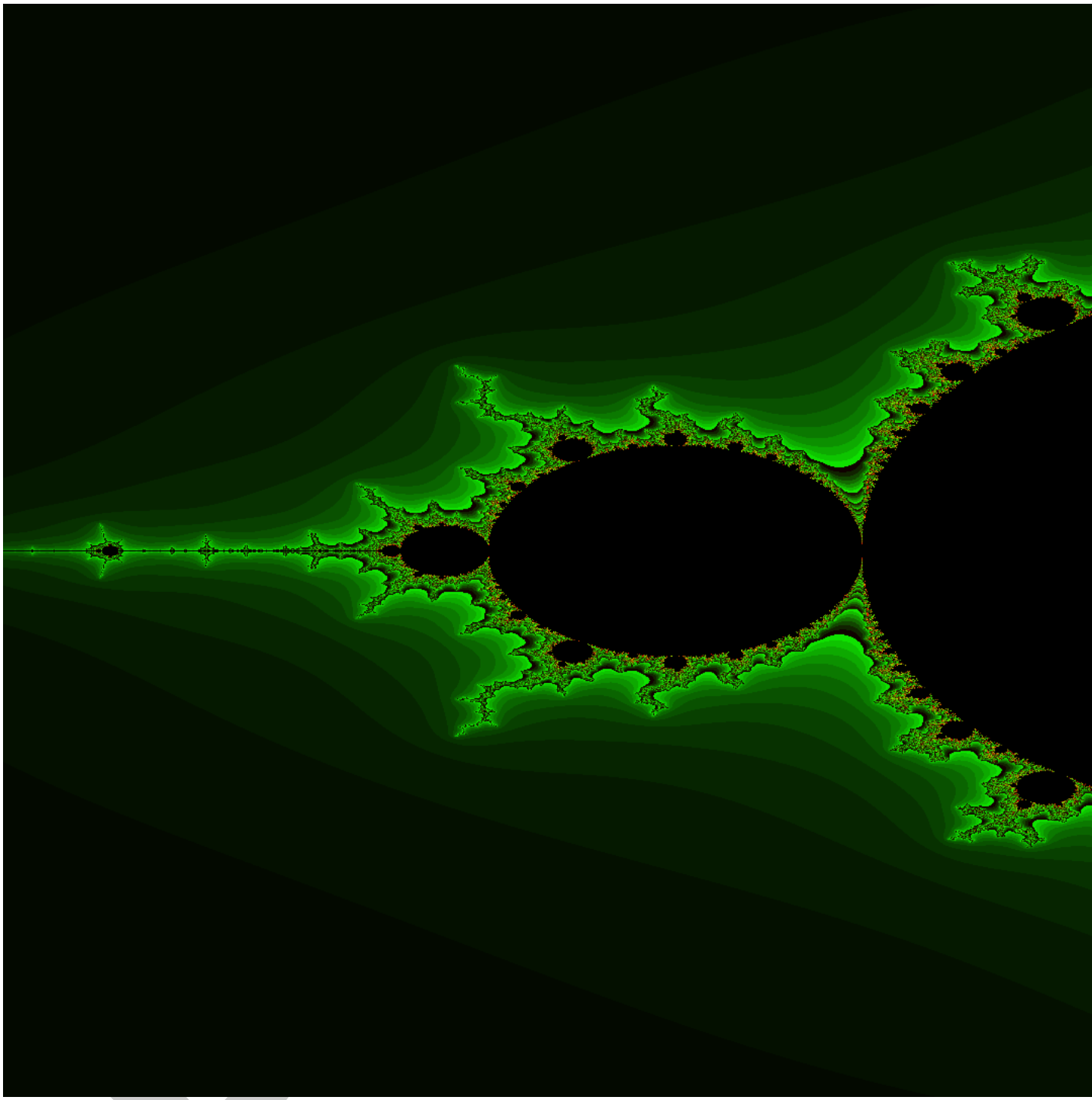
Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

Írj olyan C programot, amely kiszámolja a Mandelbrot halmazt!

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

Megoldás forrása: `bhax/attention_raising/CUDA/mandelpngt.c++` nevű állománya.



5.1. ábra. A Mandelbrot halmaz a komplex síkon

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

A Mandelbrot halmazt úgy láthatjuk meg, hogy a sík origója középpontú 4 oldalhosszúságú négyzetbe lefektetünk egy, mondjuk 800x800-as rácsot és kiszámoljuk, hogy a rács pontjai mely komplex számoknak

felelnek meg. A rács minden pontját megvizsgáljuk a $z_{n+1}=z_n{}^2+c$, (0<=n) képlet alapján úgy, hogy a c az éppen vizsgált rácspont. A z0 az origó. Alkalmazva a képletet a

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

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

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

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

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

Megoldás forrása:

A Mandelbrot halmaz pontban vázolt ismert algoritmust valósítja meg a repó `bhax/attention_raising/Mandelbrot/3.1.2.cpp` nevű állománya.

```
// Verzio: 3.1.2.cpp
// Forditas:
// g++ 3.1.2.cpp -lpng -O3 -o 3.1.2
// Futtatas:
// ./3.1.2 mandel.png 1920 1080 2040  ↩
   -0.019473810573093663922605855987058021128l8  ↩
   -0.019473810572541341845642648422654019668?  ↩
   0.798505756933826860155534177465597167611l  ↩
   0.798505756934379196110285192844457924366
// ./3.1.2 mandel.png 1920 1080 1020  ↩
   0.412765541820958925534057470940751954913l  ↩
   0.412765541824581805308014281763462349772?  ↩
   0.213538705176874649138696327099751215428l  ↩
   0.213538705180497528912653137922461610287?
// 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

A biomorfokra (a Julia halmazokat rajzoló bug-os programjával) rátaláló Clifford Pickover azt hitte természeti törvényre bukkant: https://www.emis.de/journals/TJNSA/includes/files/articles/Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf (lásd a 2307. oldal aljától).

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

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

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

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

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

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

            ++iteracio;
          }
```

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

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

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

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

A bimorfos algoritmus pontos megismeréséhez ezt a cikket javasoljuk: https://www.emis.de/journals/-TJNSA/includes/files/articles/Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf. Az is jó gyakorlat, ha magából ebből a cikkből from scratch kódoljuk be a sajátunkat, de mi a királyi úton járva a korábbi Mandelbrot halmazt kiszámoló forrásunkat módosítjuk. Viszont a program változóinak elnevezését összhangba hozzuk a közlemény jelöléseivel:

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

```
// See also https://www.emis.de/journals/TJNSA/includes/files/articles/ ↩
   Vol9_Iss5_2305--2315_Biomorphs_via_modified_iterations.pdf
//

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

}
```

5.4. A Mandelbrot halmaz CUDA megvalósítása

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

Megoldás forrása: `bhax/attention_raising/CUDA/mandelpngc_60x60_100.cu` nevű állománya.

5.5. Mandelbrot nagyító és utazó C++ nyelven

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

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

Megoldás forrása:

A program használatárol pár szó: A halmazban az egér segítségével lehet nagyítani az n lenyomásával kétszeresére növeljük az iterációs határt , az esc lenyomásával vissza ugrik az eredeti halmazra és a nyilakkal pedig lehet mozgatni a halmazt könnyebb pozicionálás érdekében(ez a része még egy kicsit bugos, de dolgozok rajta).

A program működéséről pár szó: A main-ban készitjük el a frakablak w1 et ez elkészit egy QImage-t a frakszal használatával.A fraklasz határozza meg a halmaz pontjait maj ezek a pontok alapjn a frakablak kiszinazi azokatmajd pedig a main kirajzolja a képet. Akkár hányszor mozgatjuk/nagyitjuk a halmazt az előző frakszal töröljük és egy újat hozunk létre az új kordináták segitségével.

Bemutató videó: https://youtu.be/TlL5aAfz7WY

```
// main.cpp
//
// Mandelbrot halmaz 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.
//
// Bár a Nokia Qt SDK éppen tartalmaz egy Mandelbrotos példát, de
// ezt nem tartottam megfelelõnek elsõ Qt programként ajánlani, mert elég
// bonyolult: használ kölcsönös kizárást stb. Ezért "from scratch" megírtam
// egy sajátot a Javát tanítokhoz írt dallamára:
// http://www.tankonyvtar.hu/informatika/javat-tanitok-2-2-080904-1
//

#include <QApplication>
#include "frakablak.h"

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

    FrakAblak w1;
    w1.show();
```

```
    return a.exec();
}
```

frakablak.h:

```
ifndef FRAKABLAK_H
#define FRAKABLAK_H

#include <QMainWindow>
#include <QImage>
#include <QPainter>
#include <QMouseEvent>
#include <QKeyEvent>
#include "frakszal.h"

class FrakSzal;

class FrakAblak : public QMainWindow
{
    Q_OBJECT

public:
    FrakAblak(double a = -2.0, double b = .7, double c = -1.35,
              double d = 1.35, int szelesseg = 600,
              int iteraciosHatar = 255, QWidget *parent = 0);
    ~FrakAblak();
    void vissza(int magassag , int * sor, int meret) ;
    void vissza(void) ;
    // A komplex sík vizsgált tartománya [a,b]x[c,d].
    double a, b, c, d;
    // A komplex sík vizsgált tartományára feszített
    // háló szélessége és magassága.
    int szelesseg, magassag;
    // Max. hány lépésig vizsgáljuk a z_{n+1} = z_n * z_n + c iterációt?
    // (tk. most a nagyítási pontosság)
    int iteraciosHatar;

protected:
    void paintEvent(QPaintEvent*);
    void mousePressEvent(QMouseEvent*);
    void mouseMoveEvent(QMouseEvent*);
    void mouseReleaseEvent(QMouseEvent*);
    void keyPressEvent(QKeyEvent*);

private:
    QImage* fraktal;
    FrakSzal* mandelbrot;
    bool szamitasFut;
```

```
    // A nagyítandó kijelölt területet bal felsõ sarka.
    int x, y;
    // A nagyítandó kijelölt terület szélessége és magassága.
    int mx, my;
   const double ca = -2.0;
   const double cb = .7;
   const double cc = -1.35;
   const double cd = 1.35;
   const double cszelesseg = 600;
   const double citeraciosHatar = 255;

};

#endif // FRAKABLAK_H
```

```
// frakablak.cpp
//
// Mandelbrot halmaz nagyító
// Programozó Páternoszter
//
// Copyright (C) 2011, Bátfai Norbert, nbatfai@inf.unideb.hu, nbatfai@gmail ↩
   .com
//// frakablak.cpp
//
// Mandelbrot halmaz nagyító
// 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; terjeszthetõ illetve módosítható a
// Free Software Foundation által kiadott GNU General Public License
// dokumentumában leírtak; akár a licenc 3-as, akár (tetszõleges) késõbbi
// 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    Bár a Nokia Qt SDK éppen tartalmaz egy Mandelbrotos példát, de
// ezt nem tartottam megfelelõnek elsõ Qt programként ajánlani, mert elég
// bonyolult: használ kölcsönös kizárást stb. Ezért "from scratch" megírtam
// egy sajátot a Javát tanítokhoz írt dallamára:
// http://www.tankonyvtar.hu/informatika/javat-tanitok-2-2-080904-1
//

#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;


    if (event->key() == Qt::Key_Escape)
    {
        this->a = ca;
        this->b = cb;
        this->c = cc;
        this->d = cd;
        this->szelesseg =cszelesseg ;
        this->iteraciosHatar = citeraciosHatar;
        magassag = (int)(szelesseg * ((d-c)/(b-a)));
    }

    if (event->key()==Qt::Key_Left)
    {
        this->a = a-((b-a)/10);
        this->b = b-((b-a)/10);

    }
    if (event->key()==Qt::Key_Right)
    {
        this->a = a+((b-a)/10);
        this->b = b+((b-a)/10);

    }
    if (event->key()==Qt::Key_Up)
    {

        this->c = c-((c-d)/10);
        this->d = d-((c-d)/10);
    }
```

```
    if (event->key()==Qt::Key_Down)
    {

        this->c = c+((c-d)/10);
        this->d = d+((c-d)/10);
    }


    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;
}
```

frakszal.h:

```
#ifndef FRAKSZAL_H
#define FRAKSZAL_H

#include <QThread>
#include <cmath>
#include <complex>
#include "frakablak.h"

class FrakAblak;

class FrakSzal : public QThread
{
    Q_OBJECT
```

```
public:
    FrakSzal(double a, double b, double c, double d,
             int szelesseg, int magassag, int iteraciosHatar, FrakAblak * ↩
                frakAblak);
    ~FrakSzal();
    void run();

protected:
    // A komplex sík vizsgált tartománya [a,b]x[c,d].
    double a, b, c, d;
    // A komplex sík vizsgált tartományára feszített
    // háló szélessége és magassága.
    int szelesseg, magassag;
    // Max. hány lépésig vizsgáljuk a z_{n+1} = z_n * z_n + c iterációt?
    // (tk. most a nagyítási pontosság)
    int iteraciosHatar;
    // Kinek számolok?
    FrakAblak* frakAblak;
    // Soronként küldöm is neki vissza a kiszámoltakat.
    int* egySor;

};

#endif // FRAKSZAL_H
```

```
// frakszal.cpp
//
// Mandelbrot halmaz 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; terjeszthetõ illetve módosítható a
// Free Software Foundation által kiadott GNU General Public License
// dokumentumában leírtak; akár a licenc 3-as, akár (tetszõleges) késõbbi
// 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    Bár a Nokia Qt SDK éppen tartalmaz egy Mandelbrotos példát, de
// ezt nem tartottam megfelelõnek elsõ Qt programként ajánlani, mert elég
// bonyolult: használ kölcsönös kizárást stb. Ezért "from scratch" megírtam
// egy sajátot a Javát tanítokhoz írt dallamára:
// http://www.tankonyvtar.hu/informatika/javat-tanitok-2-2-080904-1
//

#include "frakszal.h"

FrakSzal::FrakSzal(double a, double b, double c, double d,
                   int szelesseg, int magassag, int iteraciosHatar,  ↩
                      FrakAblak *frakAblak)
{
    this->a = a;
    this->b = b;
    this->c = c;
    this->d = d;
    this->szelesseg = szelesseg;
    this->iteraciosHatar = iteraciosHatar;
    this->frakAblak = frakAblak;
    this->magassag = magassag;

    egySor = new int[szelesseg];
}

FrakSzal::~FrakSzal()
{
    delete[] egySor;
}

// A szál kódját a Javát tanítokhoz írt Java kódomból vettem át
// http://www.tankonyvtar.hu/informatika/javat-tanitok-2-2-080904-1
// mivel itt az algoritmust is leírtam/lerajzoltam, így meghagytam
```

```
// a kommenteket, hogy a hallgató könnyen hozzáolvashassa az "elméletet",
// ha érdekli.
void FrakSzal::run()
{
    // A [a,b]x[c,d] tartományon milyen sûrû a
    // megadott szélesség, magasság háló:
    double dx = (b-a)/szelesseg;
    double dy = (d-c)/magassag;
    double reC, imC, reZ, imZ, ujreZ, ujimZ;
    // Hány iterációt csináltunk?
    int iteracio = 0;
    // Végigzongorázzuk a szélesség x magasság hálót:
    for(int j=0; j<magassag; ++j) {
        //sor = j;
        for(int k=0; k<szelesseg; ++k) {
            // c = (reC, imC) a háló rácspontjainak
            // megfelelõ komplex szám
            reC = a+k*dx;
            imC = d-j*dy;
            // z_0 = 0 = (reZ, imZ)
      std::complex<double> c(reC, imC);

            reZ = 0;
            imZ = 0;
      std::complex<double> z_n(reZ, imZ);
            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+ std::atan(reZ*reZ - imZ*imZ) + std::sqrt(reC);
        ujimZ = 2*reZ*imZ+std::atan(2*reZ*imZ) + imC;

                reZ = ujreZ;
                imZ = ujimZ;

                ++iteracio;

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

```
            ++iteracio;
        }

                // ha a < 4 feltétel nem teljesült és a
                // iteráció < iterációsHatár sérülésével lépett ki, azaz
                // feltesszük a c-rõl, hogy itt a z_{n+1} = z_n * z_n + c
                // sorozat konvergens, azaz iteráció = iterációsHatár
                // ekkor az iteráció %= 256 egyenlõ 255, mert az esetleges
                // nagyítasok során az iteráció = valahány * 256 + 255

                iteracio %= 256;

                //a színezést viszont már majd a FrakAblak osztályban lesz
                egySor[k] = iteracio;
            }
            // Ábrázolásra átadjuk a kiszámolt sort a FrakAblak-nak.
            frakAblak->vissza(j, egySor, szelesseg);
        }
        frakAblak->vissza();

}
```

5.6. Mandelbrot nagyító és utazó Java nyelven

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

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

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 ElsoOsztajom
{
private:
    bool letezikKovVeletlen;
    double kovVeletlen;
public:
    ElsoOsztajom();
    ~ElsoOsztajom();

    double veletlen();
};

ElsoOsztajom::ElsoOsztajom()
{
    letezikKovVeletlen=false;
}

ElsoOsztajom::~ElsoOsztajom()
{
```

```
}

double ElsoOsztajom::veletlen()
{
    if (letezikKovVeletlen)
    {
        letezikKovVeletlen =false;
        return kovVeletlen;
    }
    else
    {
        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);

        kovVeletlen=r*v2;
        letezikKovVeletlen =true;

        return r*v1;
    }


}

int main(){
    double j=0;
    ElsoOsztajom r;
    for (int i=0; i<5; i++)
    {
        j=r.veletlen();
        std::cout<<j<<std::endl;
    }

}
```

Egy osztályt hoztam létre aminek a véletlen fugavénye vissza ad egy véletlen számot. De mind ezek mellet mivel ez egy költséges feladat így a helyet, hogy minden hívásnál újra számolnánk a függvény értékét az első hívásnál két értéket határozunk meg az egyiket vissza agyuk a másikat pedig el raktározzuk. A következő hívásnál a második értéket adjuk vissza így nem kel várnunk a számításra hanem egyből meg kapjuk a véletlen értéket. Ehhez nagyon hasonló módszert használtak a Sun programozói a JDK forrásban

```
            if (haveNextNextGaussian) {
```

```
            haveNextNextGaussian = false;
            return nextNextGaussian;
        } else {
            double v1, v2, s;
            do {
                v1 = 2 * nextDouble() - 1; // between -1 and 1
                v2 = 2 * nextDouble() - 1; // between -1 and 1
                s = v1 * v1 + v2 * v2;
            } while (s >= 1 || s == 0);
            double multiplier = StrictMath.sqrt(-2 * StrictMath.log(s)/s);
            nextNextGaussian = v2 * multiplier;
            haveNextNextGaussian = true;
            return v1 * multiplier;
        }
    }
```

6.2. LZW

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

Magyarázó videó: https://www.youtube.com/watch?v=W-z3OBj3gI0

Megoldás forrása:

```
          #include <iostream>
#include <random>
#include <functional>
#include <chrono>

class Unirand {

    private:
        std::function <int()> random;

    public:
        Unirand(long seed, int min, int max): random(
            std::bind(
                std::uniform_int_distribution<>(min, max),
                std::default_random_engine(seed)
            )
        ){}

   int operator()(){return random();}

};

template <typename ValueType>
class BinRandTree {
```

```
protected:
    class Node {

    private:
        ValueType value;
        Node *left;
        Node *right;
        int count{0};

        // TODO rule of five
        Node(const Node &);
        Node & operator=(const Node &);
        Node(Node &&);
        Node & operator=(Node &&);

    public:
        Node(ValueType value, int count=0): value(value), count(count),  ↩
           left(nullptr), right(nullptr) {}
        ValueType getValue() const {return value;}
        Node * leftChild() const {return left;}
        Node * rightChild() const {return right;}
        void leftChild(Node * node){left = node;}
        void rightChild(Node * node){right = node;}
        int getCount() const {return count;}
        void incCount(){++count;}
    };

    Node *root;
    Node *treep;
    int depth{0};

private:
    // TODO rule of five

public:
    BinRandTree(Node *root = nullptr, Node *treep = nullptr): root(root),  ↩
       treep(treep) {
        std::cout << "BT ctor" << std::endl;
    }

    BinRandTree(const BinRandTree & old) {
        std::cout << "BT copy ctor" << std::endl;

        root = cp(old.root, old.treep);

    }

    Node * cp(Node *node, Node *treep)
    {
```

```
        Node * newNode = nullptr;

        if(node)
        {
            newNode = new Node(node->getValue(),node->getCount());

            newNode->leftChild(cp(node->leftChild(), treep));
            newNode->rightChild(cp(node->rightChild(), treep));

            if(node == treep)
                this->treep = newNode;
        }

        return newNode;
    }


    BinRandTree & operator=(const BinRandTree & old) {
        std::cout << "BT copy assign" << std::endl;

        BinRandTree tmp{old};
        std::swap(*this, tmp);
        return *this;
    }

    BinRandTree(BinRandTree && old) {
        std::cout << "BT move ctor" << std::endl;

        root = nullptr;
        *this = std::move(old);
    }

    BinRandTree & operator=(BinRandTree && old) {
        std::cout << "BT move assign" << std::endl;

        std::swap(old.root, root);
        std::swap(old.treep, treep);

        return *this;
    }

    ~BinRandTree(){
        std::cout << "BT dtor" << std::endl;
        deltree(root);
    }
    BinRandTree & operator<<(ValueType value);
    void print(){print(root, std::cout);}
    void printr(){print(*root, std::cout);}
    void print(Node *node, std::ostream & os);
    void print(const Node &cnode, std::ostream & os);
```

```
    void deltree(Node *node);

    Unirand ur{std::chrono::system_clock::now().time_since_epoch().count(), ↩
        0, 2};

    int whereToPut() {

            return ur();
    }


};


template <typename ValueType>
class BinSearchTree : public BinRandTree<ValueType> {

public:
    BinSearchTree() {}
    BinSearchTree & operator<<(ValueType value);


};

template <typename ValueType, ValueType vr, ValueType v0>
class ZLWTree : public BinRandTree<ValueType> {

public:
    ZLWTree(): BinRandTree<ValueType>(new typename BinRandTree<ValueType>:: ↩
       Node(vr)) {
        this->treep = this->root;
    }
    ZLWTree & operator<<(ValueType value);


};

template <typename ValueType>
BinRandTree<ValueType> & BinRandTree<ValueType>::operator<<(ValueType value ↩
   )
{

    int rnd = whereToPut();

    if(!treep) {

        root = treep = new Node(value);

    } else if (treep->getValue() == value) {
```

```
        treep->incCount();

    } else if (!rnd) {

        treep = root;
        *this << value;

    } else if (rnd == 1) {

        if(!treep->leftChild()) {

            treep->leftChild(new Node(value));

        } else {

            treep = treep->leftChild();
            *this << value;
        }

    } else if (rnd == 2) {

        if(!treep->rightChild()) {

            treep->rightChild(new Node(value));

        } else {

            treep = treep->rightChild();
            *this << value;
        }

    }

    return *this;
}


template <typename ValueType>
BinSearchTree<ValueType> & BinSearchTree<ValueType>::operator<<(ValueType ↩
   value)
{
    if(!this->treep) {

        this->root = this->treep = new typename BinRandTree<ValueType>:: ↩
           Node(value);

    } else if (this->treep->getValue() == value) {

        this->treep->incCount();
```

```
    } else if (this->treep->getValue() > value) {

        if(!this->treep->leftChild()) {

            this->treep->leftChild(new typename BinRandTree<ValueType>:: ↩
               Node(value));

        } else {

            this->treep = this->treep->leftChild();
            *this << value;
        }

    } else if (this->treep->getValue() < value) {

        if(!this->treep->rightChild()) {

            this->treep->rightChild(new typename BinRandTree<ValueType>:: ↩
               Node(value));

        } else {

            this->treep = this->treep->rightChild();
            *this << value;
        }

    }

    this->treep = this->root;

    return *this;
}


template <typename ValueType, ValueType vr, ValueType v0>
ZLWTree<ValueType, vr, v0> & ZLWTree<ValueType, vr, v0>::operator<<( ↩
   ValueType value)
{

    if(value == v0) {

        if(!this->treep->leftChild()) {

            typename BinRandTree<ValueType>::Node * node = new typename  ↩
               BinRandTree<ValueType>::Node(value);
            this->treep->leftChild(node);
            this->treep = this->root;

        } else {
```

```
            this->treep = this->treep->leftChild();
        }

    } else {

        if(!this->treep->rightChild()) {

            typename BinRandTree<ValueType>::Node * node = new typename  ↩
               BinRandTree<ValueType>::Node(value);
            this->treep->rightChild(node);
            this->treep = this->root;

        } else {

            this->treep = this->treep->rightChild();
        }

    }

    return *this;
}

template <typename ValueType>
void BinRandTree<ValueType>::print(Node *node, std::ostream & os)
{
    if(node)
    {
        ++depth;
        print(node->leftChild(), os);

        for(int i{0}; i<depth; ++i)
            os << "---";
        os << node->getValue() << " " << depth << " " << node->getCount()  ↩
           << std::endl;

        print(node->rightChild(), os);
        --depth;
    }

}


template <typename ValueType>
void BinRandTree<ValueType>::print(const Node &node, std::ostream & os)
{

        ++depth;

        if(node.leftChild())
            print(*node.leftChild(), os);
```

```
        for(int i{0}; i<depth; ++i)
            os << "---";
        os << node.getValue() << " " << depth << " " << node.getCount() <<  ↩
           std::endl;

        if(node.rightChild())
            print(*node.rightChild(), os);

        --depth;

}


template <typename ValueType>
void BinRandTree<ValueType>::deltree(Node *node)
{
    if(node)
    {
        deltree(node->leftChild());
        deltree(node->rightChild());

        delete node;
    }

}


BinRandTree<int> bar()
{
    BinRandTree<int> bt;
    BinRandTree<int> bt2;

    Unirand r(0, 0, 1);

    bt << 0 << 0 << 0;
    bt2 << 1 << 1 << 1;
    bt.print();
    std::cout << " --- " << std::endl;
    bt2.print();


    return r()?bt:bt2;
}



BinRandTree<int> foo()
{
    return BinRandTree<int>();
```

```
}


int main(int argc, char** argv, char ** env)
{
  BinRandTree<int> bt;
    std::cout << " *** " << std::endl;
  BinRandTree<int> bt2{ bt };
    std::cout << " *** " << std::endl;
    bt2.print();
  bt2 = bt;



/*
    std::cout << std::endl;

    ZLWTree<char, '/', '0'> zt;

    zt <<'0'<<'1'<<'0'<<'0'<<'1' << '1';

    zt.print();

    ZLWTree<char, '/', '0'> zt2{zt};

    ZLWTree<char, '/', '0'> zt3;

    zt3 <<'1'<<'1'<<'1'<<'1'<<'1';

    std::cout << "***" << std::endl;
    zt = zt3;
    std::cout << "***" << std::endl;

    ZLWTree<char, '/', '0'> zt4 = std::move(zt2);

    BinSearchTree<std::string> bts;

    bts << "alma" << "korte" << "banan" << "korte";

    bts.print();
*/

}
```

7. fejezet

Helló, Conway!

7.1. Hangyaszimulációk

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

https://www.youtube.com/watch?v=uHbM36lrq4o

Main:

```
// BHAX Myrmecologist
//
// Copyright (C) 2019
// Norbert BÃ¡tfai, 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/>.
//
// https://bhaxor.blog.hu/2018/09/26/hangyaszimulaciok
// https://bhaxor.blog.hu/2018/10/10/myrmecologist
//

#include <QApplication>
#include <QDesktopWidget>
#include <QDebug>
#include <QDateTime>
```

```
#include <QCommandLineOption>
#include <QCommandLineParser>

#include "antwin.h"

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

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

    QApplication a ( argc, argv );

    QCommandLineOption szeles_opt ( {"w","szelesseg"}, "Oszlopok (cellakban) szama.", "szelesseg", "200" );
    QCommandLineOption magas_opt ( {"m","magassag"}, "Sorok (cellakban) szama.", "magassag", "150" );
    QCommandLineOption hangyaszam_opt ( {"n","hangyaszam"}, "Hangyak szama.", "hangyaszam", "100" );
    QCommandLineOption sebesseg_opt ( {"t","sebesseg"}, "2 lepes kozotti ido (millisec-ben).", "sebesseg", "100" );
    QCommandLineOption parolgas_opt ( {"p","parolgas"}, "A parolgas erteke.", "parolgas", "8" );
    QCommandLineOption feromon_opt ( {"f","feromon"}, "A hagyott nyom erteke.", "feromon", "11" );
    QCommandLineOption szomszed_opt ( {"s","szomszed"}, "A hagyott nyom erteke a szomszedokban.", "szomszed", "3" );
    QCommandLineOption alapertek_opt ( {"d","alapertek"}, "Indulo ertek a cellakban.", "alapertek", "1" );
    QCommandLineOption maxcella_opt ( {"a","maxcella"}, "Cella max erteke.", "maxcella", "50" );
    QCommandLineOption mincella_opt ( {"i","mincella"}, "Cella min erteke.", "mincella", "2" );
    QCommandLineOption cellamerete_opt ( {"c","cellameret"}, "Hany hangya fer egy cellaba.", "cellameret", "4" );
    QCommandLineParser parser;

    parser.addHelpOption();
    parser.addVersionOption();
    parser.addOption ( szeles_opt );
    parser.addOption ( magas_opt );
    parser.addOption ( hangyaszam_opt );
    parser.addOption ( sebesseg_opt );
    parser.addOption ( parolgas_opt );
    parser.addOption ( feromon_opt );
    parser.addOption ( szomszed_opt );
```

```
    parser.addOption ( alapertek_opt );
    parser.addOption ( maxcella_opt );
    parser.addOption ( mincella_opt );
    parser.addOption ( cellamerete_opt );

    parser.process ( a );

    QString szeles = parser.value ( szeles_opt );
    QString magas = parser.value ( magas_opt );
    QString n = parser.value ( hangyaszam_opt );
    QString t = parser.value ( sebesseg_opt );
    QString parolgas = parser.value ( parolgas_opt );
    QString feromon = parser.value ( feromon_opt );
    QString szomszed = parser.value ( szomszed_opt );
    QString alapertek = parser.value ( alapertek_opt );
    QString maxcella = parser.value ( maxcella_opt );
    QString mincella = parser.value ( mincella_opt );
    QString cellameret = parser.value ( cellamerete_opt );

    qsrand ( QDateTime::currentMSecsSinceEpoch() );

    AntWin w ( szeles.toInt(), magas.toInt(), t.toInt(), n.toInt(), feromon ↩
       .toInt(), szomszed.toInt(), parolgas.toInt(),
               alapertek.toInt(), mincella.toInt(), maxcella.toInt(),
               cellameret.toInt());
 /*   AntWin w ( szeles.toInt(), magas.toInt(), t.toInt(), n.toInt(),  ↩
    feromon.toInt(), szomszed.toInt(), parolgas.toInt(),
                     alapertek.toInt(), mincella.toInt(), maxcella.toInt() ↩
                        ,
                     cellameret.toInt() );
   */


//myrmecologist -w 250 -m 150 -n 400 -t 10 -p 5 -f 80 -d 0 -a 255 -i 3 -s 3 ↩
     -c 22
    w.show();

    return a.exec();
}
```

antwin.h:

```
          // BHAX Myrmecologist
//
// Copyright (C) 2019
// Norbert BÃ¡tfai, 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/>.
//
// https://bhaxor.blog.hu/2018/09/26/hangyaszimulaciok
// https://bhaxor.blog.hu/2018/10/10/myrmecologist
//

#ifndef ANTWIN_H
#define ANTWIN_H

#include <QMainWindow>
#include <QPainter>
#include <QString>
#include <QCloseEvent>
#include "antthread.h"
#include "ant.h"

class AntWin : public QMainWindow
{
    Q_OBJECT

public:
    AntWin(int width = 100, int height = 75,
           int delay = 120, int numAnts = 100,
           int pheromone = 10, int nbhPheromon = 3,
           int evaporation = 2, int cellDef = 1,
           int min = 2, int max = 50,
           int cellAntMax = 4, QWidget *parent = 0);

    AntThread* antThread;

    void closeEvent ( QCloseEvent *event ) {

        antThread->finish();
        antThread->wait();
        event->accept();
    }

    void keyPressEvent ( QKeyEvent *event )
    {
```

```
        if ( event->key() == Qt::Key_P ) {
            antThread->pause();
        } else if ( event->key() == Qt::Key_Q
                    ||  event->key() == Qt::Key_Escape ) {
            close();
        }

    }

    virtual ~AntWin();
    void paintEvent(QPaintEvent*);

private:

    int ***grids;
    int **grid;
    int gridIdx;
    int cellWidth;
    int cellHeight;
    int width;
    int height;
    int max;
    int min;
    Ants* ants;

public slots :
    void step ( const int &);

};

#endif
```

antwin.cpp:

```
          // BHAX Myrmecologist
//
// Copyright (C) 2019
// Norbert BÃ¡tfai, 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/>.
//
// https://bhaxor.blog.hu/2018/09/26/hangyaszimulaciok
// https://bhaxor.blog.hu/2018/10/10/myrmecologist
//

#ifndef ANTWIN_H
#define ANTWIN_H

#include <QMainWindow>
#include <QPainter>
#include <QString>
#include <QCloseEvent>
#include "antthread.h"
#include "ant.h"

class AntWin : public QMainWindow
{
    Q_OBJECT

public:
    AntWin(int width = 100, int height = 75,
           int delay = 120, int numAnts = 100,
           int pheromone = 10, int nbhPheromon = 3,
           int evaporation = 2, int cellDef = 1,
           int min = 2, int max = 50,
           int cellAntMax = 4, QWidget *parent = 0);

    AntThread* antThread;

    void closeEvent ( QCloseEvent *event ) {

        antThread->finish();
        antThread->wait();
        event->accept();
    }

    void keyPressEvent ( QKeyEvent *event )
    {

        if ( event->key() == Qt::Key_P ) {
            antThread->pause();
        } else if ( event->key() == Qt::Key_Q
                    ||  event->key() == Qt::Key_Escape ) {
            close();
        }

    }
```

```
    virtual ~AntWin();
    void paintEvent(QPaintEvent*);

private:

    int ***grids;
    int **grid;
    int gridIdx;
    int cellWidth;
    int cellHeight;
    int width;
    int height;
    int max;
    int min;
    Ants* ants;

public slots :
    void step ( const int &);

};

#endif
```

anthread.h:

```
          // BHAX Myrmecologist
//
// Copyright (C) 2019
// Norbert BÃ¡tfai, 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/>.
//
// https://bhaxor.blog.hu/2018/09/26/hangyaszimulaciok
// https://bhaxor.blog.hu/2018/10/10/myrmecologist
//
```

```
#ifndef ANTTHREAD_H
#define ANTTHREAD_H

#include <QThread>
#include "ant.h"

class AntThread : public QThread
{
    Q_OBJECT

public:
    AntThread(Ants * ants, int ***grids, int width, int height,
              int delay, int numAnts, int pheromone, int nbrPheromone,
              int evaporation, int min, int max, int cellAntMax);

    ~AntThread();

    void run();
    void finish()
    {
        running = false;
    }

    void pause()
    {
        paused = !paused;
    }

    bool isRunnung()
    {
        return running;
    }

private:
    bool running {true};
    bool paused {false};
    Ants* ants;
    int** numAntsinCells;
    int min, max;
    int cellAntMax;
    int pheromone;
    int evaporation;
    int nbrPheromone;
    int ***grids;
    int width;
    int height;
    int gridIdx;
    int delay;
```

```
    void timeDevel();

    int newDir(int sor, int oszlop, int vsor, int voszlop);
    void detDirs(int irany, int& ifrom, int& ito, int& jfrom, int& jto );
    int moveAnts(int **grid, int row, int col, int& retrow, int& retcol,
       int);
    double sumNbhs(int **grid, int row, int col, int);
    void setPheromone(int **grid, int row, int col);

signals:
    void step ( const int &);

};

#endif
```

anthread.cpp:

```
          // BHAX Myrmecologist
//
// Copyright (C) 2019
// Norbert BÃ¡tfai, 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/>.
//
// https://bhaxor.blog.hu/2018/09/26/hangyaszimulaciok
// https://bhaxor.blog.hu/2018/10/10/myrmecologist
//

#include "antthread.h"
#include <QDebug>
#include <cmath>
#include <QDateTime>

AntThread::AntThread ( Ants* ants, int*** grids,
                       int width, int height,
```

```
                        int delay, int numAnts,
                        int pheromone, int nbrPheromone,
                        int evaporation,
                        int min, int max, int cellAntMax)
{
    this->ants = ants;
    this->grids = grids;
    this->width = width;
    this->height = height;
    this->delay = delay;
    this->pheromone = pheromone;
    this->evaporation = evaporation;
    this->min = min;
    this->max = max;
    this->cellAntMax = cellAntMax;
    this->nbrPheromone = nbrPheromone;

    numAntsinCells = new int*[height];
    for ( int i=0; i<height; ++i ) {
        numAntsinCells[i] = new int [width];
    }

    for ( int i=0; i<height; ++i )
        for ( int j=0; j<width; ++j ) {
            numAntsinCells[i][j] = 0;
        }

    qsrand ( QDateTime::currentMSecsSinceEpoch() );

    Ant h {0, 0};
    for ( int i {0}; i<numAnts; ++i ) {

        h.y = height/2 + qrand() % 40-20;
        h.x = width/2 + qrand() % 40-20;

        ++numAntsinCells[h.y][h.x];

        ants->push_back ( h );

    }

    gridIdx = 0;
}

double AntThread::sumNbhs ( int **grid, int row, int col, int dir )
{
    double sum = 0.0;

    int ifrom, ito;
    int jfrom, jto;
```

```
    detDirs ( dir, ifrom, ito, jfrom, jto );

    for ( int i=ifrom; i<ito; ++i )
        for ( int j=jfrom; j<jto; ++j )

            if ( ! ( ( i==0 ) && ( j==0 ) ) ) {
                int o = col + j;
                if ( o < 0 ) {
                    o = width-1;
                } else if ( o >= width ) {
                    o = 0;
                }

                int s = row + i;
                if ( s < 0 ) {
                    s = height-1;
                } else if ( s >= height ) {
                    s = 0;
                }

                sum += (grid[s][o]+1)*(grid[s][o]+1)*(grid[s][o]+1);

            }

    return sum;
}

int AntThread::newDir ( int sor, int oszlop, int vsor, int voszlop )
{

    if ( vsor == 0 && sor == height -1 ) {
        if ( voszlop < oszlop ) {
            return 5;
        } else if ( voszlop > oszlop ) {
            return 3;
        } else {
            return 4;
        }
    } else if ( vsor == height - 1 && sor == 0 ) {
        if ( voszlop < oszlop ) {
            return 7;
        } else if ( voszlop > oszlop ) {
            return 1;
        } else {
            return 0;
        }
    } else if ( voszlop == 0 && oszlop == width - 1 ) {
        if ( vsor < sor ) {
            return 1;
```

```
        } else if ( vsor > sor ) {
            return 3;
        } else {
            return 2;
        }
    } else if ( voszlop == width && oszlop == 0 ) {
        if ( vsor < sor ) {
            return 7;
        } else if ( vsor > sor ) {
            return 5;
        } else {
            return 6;
        }
    } else if ( vsor < sor && voszlop < oszlop ) {
        return 7;
    } else if ( vsor < sor && voszlop == oszlop ) {
        return 0;
    } else if ( vsor < sor && voszlop > oszlop ) {
        return 1;
    }

    else if ( vsor > sor && voszlop < oszlop ) {
        return 5;
    } else if ( vsor > sor && voszlop == oszlop ) {
        return 4;
    } else if ( vsor > sor && voszlop > oszlop ) {
        return 3;
    }

    else if ( vsor == sor && voszlop < oszlop ) {
        return 6;
    } else if ( vsor == sor && voszlop > oszlop ) {
        return 2;
    }

    else { //(vsor == sor && voszlop == oszlop)
        qDebug() << "ZAVAR AZ EROBEN az iranynal";

        return -1;
    }

}

void AntThread::detDirs ( int dir, int& ifrom, int& ito, int& jfrom, int&  ↩
    jto )
{

    switch ( dir ) {
    case 0:
        ifrom = -1;
```

```
        ito = 0;
        jfrom = -1;
        jto = 2;
        break;
    case 1:
        ifrom = -1;
        ito = 1;
        jfrom = 0;
        jto = 2;
        break;
    case 2:
        ifrom = -1;
        ito = 2;
        jfrom = 1;
        jto = 2;
        break;
    case 3:
        ifrom =
            0;
        ito = 2;
        jfrom = 0;
        jto = 2;
        break;
    case 4:
        ifrom = 1;
        ito = 2;
        jfrom = -1;
        jto = 2;
        break;
    case 5:
        ifrom = 0;
        ito = 2;
        jfrom = -1;
        jto = 1;
        break;
    case 6:
        ifrom = -1;
        ito = 2;
        jfrom = -1;
        jto = 0;
        break;
    case 7:
        ifrom = -1;
        ito = 1;
        jfrom = -1;
        jto = 1;
        break;

    }
```

```
}

int AntThread::moveAnts ( int **racs,
                          int sor, int oszlop,
                          int& vsor, int& voszlop, int dir )
{

    int y = sor;
    int x = oszlop;

    int ifrom, ito;
    int jfrom, jto;

    detDirs ( dir, ifrom, ito, jfrom, jto );

    double osszes = sumNbhs ( racs, sor, oszlop, dir );
    double random = ( double ) ( qrand() %1000000 ) / ( double ) 1000000.0;
    double gvalseg = 0.0;


    for ( int i=ifrom; i<ito; ++i )
        for ( int j=jfrom; j<jto; ++j )
            if ( ! ( ( i==0 ) && ( j==0 ) ) )
            {
                int o = oszlop + j;
                if ( o < 0 ) {
                    o = width-1;
                } else if ( o >= width ) {
                    o = 0;
                }

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

                //double kedvezo = std::sqrt((double)(racs[s][o]+2));//( ↩
                   racs[s][o]+2)*(racs[s][o]+2);
                //double kedvezo = (racs[s][o]+b)*(racs[s][o]+b);
                //double kedvezo = ( racs[s][o]+1 );
                double kedvezo = (racs[s][o]+1)*(racs[s][o]+1)*(racs[s][o ↩
                   ]+1);

                double valseg = kedvezo/osszes;
                gvalseg += valseg;

                if ( gvalseg >= random ) {
```

```
                    vsor = s;
                    voszlop = o;

                    return newDir ( sor, oszlop, vsor, voszlop );

                }

            }

    qDebug() << "ZAVAR AZ EROBEN a lepesnel";
    vsor = y;
    voszlop = x;

    return dir;
}

void AntThread::timeDevel()
{

    int **racsElotte = grids[gridIdx];
    int **racsUtana = grids[ ( gridIdx+1 ) %2];

    for ( int i=0; i<height; ++i )
        for ( int j=0; j<width; ++j )
        {
            racsUtana[i][j] = racsElotte[i][j];

            if ( racsUtana[i][j] - evaporation >= 0 ) {
                racsUtana[i][j] -= evaporation;
            } else {
                racsUtana[i][j] = 0;
            }

        }

    for ( Ant &h: *ants )
    {

        int sor {-1}, oszlop {-1};
        int ujirany = moveAnts( racsElotte, h.y, h.x, sor, oszlop, h.dir );

        setPheromone ( racsUtana, h.y, h.x );

        if ( numAntsinCells[sor][oszlop] <cellAntMax ) {

            --numAntsinCells[h.y][h.x];
            ++numAntsinCells[sor][oszlop];

            h.x = oszlop;
            h.y = sor;
```

```
            h.dir = ujirany;

        }
    }

    gridIdx = ( gridIdx+1 ) %2;
}



void AntThread::setPheromone ( int **racs,
                        int sor, int oszlop )
{

    for ( int i=-1; i<2; ++i )
        for ( int j=-1; j<2; ++j )
            if ( ! ( ( i==0 ) && ( j==0 ) ) )
            {
                int o = oszlop + j;
                {
                    if ( o < 0 ) {
                        o = width-1;
                    } else if ( o >= width ) {
                        o = 0;
                    }
                }
                int s = sor + i;
                {
                    if ( s < 0 ) {
                        s = height-1;
                    } else if ( s >= height ) {
                        s = 0;
                    }
                }

                if ( racs[s][o] + nbrPheromone <= max ) {
                    racs[s][o] += nbrPheromone;
                } else {
                    racs[s][o] = max;
                }


            }

    if ( racs[sor][oszlop] + pheromone <= max ) {
        racs[sor][oszlop] += pheromone;
    } else {
        racs[sor][oszlop]  = max;
    }
```

```
}

void AntThread::run()
{
    running = true;
    while ( running ) {

        QThread::msleep ( delay );

        if ( !paused ) {
            timeDevel();
        }

        emit step ( gridIdx );

    }

}

AntThread::~AntThread()
{
    for ( int i=0; i<height; ++i ) {
        delete [] numAntsinCells[i];
    }

    delete [] numAntsinCells;
}
```

ant.h:

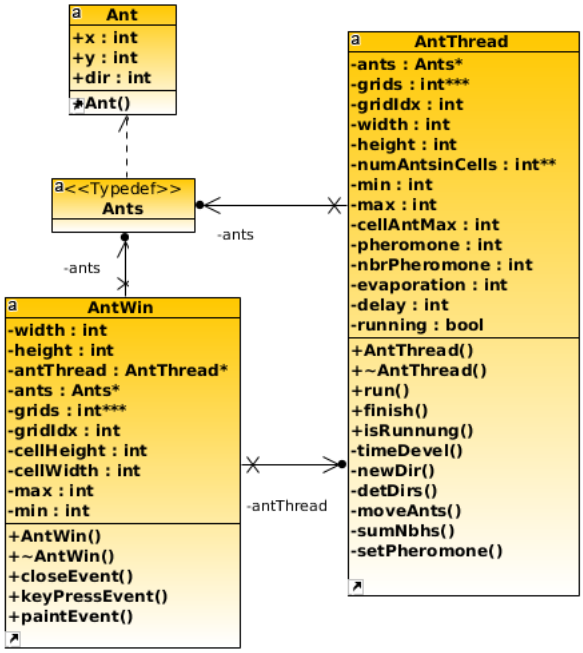
```
          // BHAX Myrmecologist
//
// Copyright (C) 2019
// Norbert BÃ¡tfai, batfai.norbert@inf.unideb.hu
//
//  This program is free software: you can redistribute it and/or modify
//  it under the terms of the GNU General Public License as published by
//  the Free Software Foundation, either version 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/>.
//
// https://bhaxor.blog.hu/2018/09/26/hangyaszimulaciok
// https://bhaxor.blog.hu/2018/10/10/myrmecologist
//

#ifndef ANT_H
#define ANT_H

class Ant
{

public:
    int x;
    int y;
    int dir;

    Ant(int x, int y): x(x), y(y) {

        dir = qrand() % 8;

    }

};

typedef std::vector<Ant> Ants;

#endif
```

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

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...



7.1. ábra. Hangyaszimuláció UML diagram

A program alapját a hangyák mozgásának és tájékozódásának a folyamata adja. Amikor a hangyák élelem után kutatnak feromonokat bocsátanak ki és ez által tudnak tájékozódni.

Az ant.h reprezentálja a hangyáinkat melyeket az antthread fog létre hozni és bele pakolni egy ant* típusu változóba. A hangyák létre hozásánál kapnak egy irányt és egy pozíciót. És ezek után a hangyák az anthread irányítására lesznek bízva ő agya meg hogy hova kell lépniük és hogy mi lesz a következő irányuk. Az irányokra azért van szükség, hogy ne forduljanak hátra a hangyák és így ne menjenek vissza a saját nyom vonalukon. A Mind ezek melet a feromonok elhelyezését és gyengítését is az antthread végzi a grids-eken. Az antwin az antthread által elő állatot adatokból elkészíti a látni kívánt képet amit a main-ben rakunk ki a képernyőre. A main nem anyira letisztult mint amit meg szokhatunk egy Qt programtól de ez a pár sor kód csak a rra szolgál, hogy képesek legyünk megadni pár paramétert a program inditásakor.

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:

Megoldás forrása:

main.cpp:

```
            #include <QApplication>
#include "sejtablak.h"
#include <QDesktopWidget>

int main(int argc, char *argv[])
{
  QApplication a(argc, argv);
  SejtAblak w(100, 75);
  w.show();

  return a.exec();
}
```

sejtablak.h

```
            #ifndef SEJTABLAK_H
#define SEJTABLAK_H

#include <QMainWindow>
#include <QPainter>
#include "sejtszal.h"

class SejtSzal;

class SejtAblak : public QMainWindow
{
  Q_OBJECT

public:
  SejtAblak(int szelesseg = 100, int magassag = 75, QWidget *parent = 0);
  ~SejtAblak();
  // Egy sejt lehet élõ
  static const bool ELO = true;
  // vagy halott
  static const bool HALOTT = false;
  void vissza(int racsIndex);

protected:
  // Két rácsot használunk majd, az egyik a sejttér állapotát
  // a t_n, a másik a t_n+1 idõpillanatban jellemzi.
  bool ***racsok;
  // Valamelyik rácsra mutat, technikai jellegû, hogy ne kelljen a
  // [2][][]-ból az elsõ dimenziót használni, mert vagy az egyikre
  // állítjuk, vagy a másikra.
  bool **racs;
```

```
  // Megmutatja melyik rács az aktuális: [rácsIndex][][]
  int racsIndex;
  // Pixelben egy cella adatai.
  int cellaSzelesseg;
  int cellaMagassag;
  // A sejttér nagysága, azaz hányszor hány cella van?
  int szelesseg;
  int magassag;
  void paintEvent(QPaintEvent*);
  void siklo(bool **racs, int x, int y);
  void sikloKilovo(bool **racs, int x, int y);

private:
  SejtSzal* eletjatek;

};

#endif // SEJTABLAK_H
```

sejtablak.cpp

```
            // 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; terjeszthetõ illetve módosítható a
// Free Software Foundation által kiadott GNU General Public License
// dokumentumában leírtak; akár a licenc 3-as, akár (tetszõleges) késõbbi
// 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 fõ szempontja az volt, hogy
// ne vagy alig különbözzön az elsõ 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 felsõ sarkának oszlopa
 * @param   y       a befoglaló tégla bal felsõ 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 lõ 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
 * mindkettõ tartalmaz két felesleges sejtet is.)
 *
 * @param   racs    a sejttér ahová ezt az állatkát helyezzük
 * @param   x       a befoglaló tégla bal felsõ sarkának oszlopa
 * @param   y       a befoglaló tégla bal felsõ 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;

}
```

sejtszal.h

```
            #ifndef SEJTSZAL_H
#define SEJTSZAL_H

#include <QThread>
#include "sejtablak.h"

class SejtAblak;

class SejtSzal : public QThread
{
    Q_OBJECT

public:
    SejtSzal(bool ***racsok, int szelesseg, int magassag,
             int varakozas, SejtAblak *sejtAblak);
    ~SejtSzal();
    void run();

protected:
    bool ***racsok;
    int szelesseg, magassag;
    // Megmutatja melyik rács az aktuális: [rácsIndex][][]
    int racsIndex;
    // A sejttér két egymást követõ t_n és t_n+1 diszkrét idõpillanata
    // közötti valós idõ.
    int varakozas;
    void idoFejlodes();
    int szomszedokSzama(bool **racs,
                        int sor, int oszlop, bool allapot);
    SejtAblak* sejtAblak;

};

#endif // SEJTSZAL_H
```

sejtszal.cpp

```
            // sejtszal.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; terjeszthetõ illetve módosítható a
// Free Software Foundation által kiadott GNU General Public License
// dokumentumában leírtak; akár a licenc 3-as, akár (tetszõleges) késõbbi
// 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 fõ szempontja az volt, hogy
// ne vagy alig különbözzön az elsõ 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 "sejtszal.h"

SejtSzal::SejtSzal(bool ***racsok, int szelesseg, int magassag, int  ↩
```

```
   varakozas, SejtAblak *sejtAblak)
{
    this->racsok = racsok;
    this->szelesseg = szelesseg;
    this->magassag = magassag;
    this->varakozas = varakozas;
    this->sejtAblak = sejtAblak;

    racsIndex = 0;
}

/**
   * Az kérdezett állapotban lévõ nyolcszomszédok száma.
   *
   * @param   rács    a sejttér rács
   * @param   sor     a rács vizsgált sora
   * @param   oszlop  a rács vizsgált oszlopa
   * @param   állapor a nyolcszomszédok vizsgált állapota
   * @return int a kérdezett állapotbeli nyolcszomszédok száma.
   */
int SejtSzal::szomszedokSzama(bool **racs,
                              int sor, int oszlop, bool allapot) {
    int allapotuSzomszed = 0;
    // A nyolcszomszédok végigzongorázása:
    for(int i=-1; i<2; ++i)
        for(int j=-1; j<2; ++j)
            // A vizsgált sejtet magát kihagyva:
            if(!((i==0) && (j==0))) {
        // A sejttérbõl szélének szomszédai
        // a szembe oldalakon ("periódikus határfeltétel")
        int o = oszlop + j;
        if(o < 0)
            o = szelesseg-1;
        else if(o >= szelesseg)
            o = 0;

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

        if(racs[s][o] == allapot)
            ++allapotuSzomszed;
    }

    return allapotuSzomszed;
}

/**
```

```
 * A sejttér idõbeli fejlõdése a John H. Conway féle
 * életjáték sejtautomata szabályai alapján történik.
 * A szabályok részletes ismertetését lásd például a
 * [MATEK JÁTÉK] hivatkozásban (Csákány Béla: Diszkrét
 * matematikai játékok. Polygon, Szeged 1998. 171. oldal.)
 */
void SejtSzal::idoFejlodes() {

    bool **racsElotte = racsok[racsIndex];
    bool **racsUtana = racsok[(racsIndex+1)%2];

    for(int i=0; i<magassag; ++i) { // sorok
        for(int j=0; j<szelesseg; ++j) { // oszlopok

            int elok = szomszedokSzama(racsElotte, i, j, SejtAblak::ELO);

            if(racsElotte[i][j] == SejtAblak::ELO) {
                /* Élõ élõ marad, ha kettõ vagy három élõ
             szomszedja van, különben halott lesz. */
                if(elok==2 || elok==3)
                    racsUtana[i][j] = SejtAblak::ELO;
                else
                    racsUtana[i][j] = SejtAblak::HALOTT;
            }  else {
                /* Halott halott marad, ha három élõ
             szomszedja van, különben élõ lesz. */
                if(elok==3)
                    racsUtana[i][j] = SejtAblak::ELO;
                else
                    racsUtana[i][j] = SejtAblak::HALOTT;
            }
        }
    }
    racsIndex = (racsIndex+1)%2;
}


/** A sejttér idõbeli fejlõdése. */
void SejtSzal::run()
{
    while(true) {
        QThread::msleep(varakozas);
        idoFejlodes();
        sejtAblak->vissza(racsIndex);
    }

}

SejtSzal::~SejtSzal()
{
```

```
}
```

A szabájokról pár szó :

Élõ sejt élõ marad(feketeként jelenikmeg), ha kettõ vagy három élõ szomszedja van, különben halott lesz.

Halott sejt halott marad, ha három élõ szomszedja van, különben élõ lesz.

A működéséről párszó:

A main ahogy eddig minden Qt programban a megjelenítésért felel a sejtablakban készítjük el a rácsot amin mozoghatnak a sejtek és magét a képet is a rácsról ezek mellet el helyez egy „sikló kilövőt” is a rácson ami folyamatosan fogja nekünk generálni a siklókat de ehez szükség van a sejtszalra ami alkalmazza a fentebb már említett szabályokat.

7.4. BrainB Benchmark

Megoldás videó:

main.cpp

```
          /**
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file main.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 <QApplication>
#include <QTextStream>
#include <QtWidgets>
#include "BrainBWin.h"

int main ( int argc, char **argv )
{
        QApplication app ( argc, argv );

        QTextStream qout ( stdout );
        qout.setCodec ( "UTF-8" );

        qout << "\n" << BrainBWin::appName << QString::fromUtf8 ( "  ↩
           Copyright (C) 2017, 2018 Norbert Bátfai" ) << endl;

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

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

        qout << QString::fromUtf8 ( "Ez a program szabad szoftver;  ↩
           terjeszthető illetve módosítható a Free Software" ) << endl;
        qout << QString::fromUtf8 ( "Foundation által kiadott GNU General  ↩
           Public License dokumentumában leírtak;" ) << endl;
        qout << QString::fromUtf8 ( "akár a licenc 3-as, akár (tetszőleges) ↩
            későbbi változata szerint.\n" ) << endl;

        qout << QString::fromUtf8 ( "Ez a program abban a reményben kerül  ↩
           közreadásra, hogy hasznos lesz, de minden" ) << endl;
        qout << QString::fromUtf8 ( "egyéb GARANCIA NÉLKÜL, az  ↩
           ELADHATÓSÁGRA vagy VALAMELY CÉLRA VALÓ" ) << endl;
        qout << QString::fromUtf8 ( "ALKALMAZHATÓSÁGRA való származtatott  ↩
           garanciát is beleértve. További" ) << endl;
        qout << QString::fromUtf8 ( "részleteket a GNU General Public  ↩
           License tartalmaz.\n" ) << endl;

        qout << "http://gnu.hu/gplv3.html" << endl;

        QRect rect = QApplication::desktop()->availableGeometry();
```

```
        BrainBWin brainBWin ( rect.width(), rect.height() );
        brainBWin.setWindowState ( brainBWin.windowState() ^ Qt:: ↩
           WindowFullScreen );
        brainBWin.show();
        return app.exec();
}
```

BrainBWin.h

```
            #ifndef BrainBWin_H
#define BrainBWin_H

/**
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBWin.h
 * @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 <QKeyEvent>
#include <QMainWindow>
#include <QPixmap>
#include <QPainter>
#include <QFont>
#include <QFile>
#include <QString>
```

```
#include <QCloseEvent>
#include <QDate>
#include <QDir>
#include <QDateTime>
#include "BrainBThread.h"

enum playerstate {
    lost,
    found
};

class BrainBWin : public QMainWindow
{
    Q_OBJECT

    BrainBThread *brainBThread;
    QPixmap pixmap;
    Heroes *heroes;

    int mouse_x;
    int mouse_y;
    int yshift {50};
    int nofLost {0};
    int nofFound {0};

    int xs, ys;

    bool firstLost {false};
    bool start {false};
    playerstate state = lost;
    std::vector<int> lost2found;
    std::vector<int> found2lost;

    QString statDir;

public:
    static const QString appName;
    static const QString appVersion;
    BrainBWin ( int w = 256, int h = 256, QWidget *parent = 0 );

    void closeEvent ( QCloseEvent *e ) {

        if ( save ( brainBThread->getT() ) ) {
            brainBThread->finish();
            e->accept();
        } else {
            e->ignore();
        }

    }
```

```
    virtual ~BrainBWin();
    void paintEvent ( QPaintEvent * );
    void keyPressEvent ( QKeyEvent *event );
    void mouseMoveEvent ( QMouseEvent *event );
    void mousePressEvent ( QMouseEvent *event );
    void mouseReleaseEvent ( QMouseEvent *event );

    double mean ( std::vector<int> vect ) {

        if ( vect.size() > 0 ) {
            double sum = std::accumulate ( vect.begin (), vect.end (), 0.0 ↩
               );
            return  sum / vect.size();
        } else {
            return 0.0;
        }
    }

    double var ( std::vector<int> vect, double mean ) {

        if ( vect.size() > 1 ) {

            double accum = 0.0;

            std::for_each ( vect.begin (), vect.end (), [&] ( const double ↩
               d ) {
                accum += ( d - mean ) * ( d - mean );
            } );

            return sqrt ( accum / ( vect.size()-1 ) );
        } else {
            return 0.0;
        }

    }

    void millis2minsec ( int millis, int &min, int &sec ) {

        sec = ( millis * 100 ) / 1000;
        min = sec / 60;
        sec = sec - min * 60;

    }

    bool save ( int t ) {

        bool ret = false;

        if ( !QDir ( statDir ).exists() )
```

```
            if ( !QDir().mkdir ( statDir ) ) {
                return false;
            }

        QString name = statDir + "/Test-" + QString::number ( t );
        QFile file ( name + "-screenimage.png" );
        if ( file.open ( QIODevice::WriteOnly ) ) {
            ret = pixmap.save ( &file, "PNG" );
        }

        QFile tfile ( name + "-stats.txt" );
        ret = tfile.open ( QIODevice::WriteOnly | QIODevice::Text );
        if ( ret ) {
            QTextStream textStremam ( &tfile );

            textStremam << appName + " " + appVersion << "\n";
            textStremam << "time      : " <<  brainBThread->getT() << "\n";
            textStremam << "bps       : " <<  brainBThread->get_bps() << "\ ↩
               n";
            textStremam << "noc       : " <<  brainBThread->nofHeroes() <<  ↩
               "\n";
            textStremam << "nop       : " <<  brainBThread->get_nofPaused() ↩
                << "\n";

            textStremam << "lost      : "  << "\n";
            std::vector<int> l = brainBThread->lostV();
            for ( int n : l ) {
                textStremam << n << ' ';
            }
            textStremam <<  "\n";
            int m = mean ( l );
            textStremam << "mean      : " <<  m << "\n";
            textStremam << "var       : " <<  var ( l, m ) << "\n";

            textStremam << "found     : " ;
            std::vector<int> f = brainBThread->foundV();
            for ( int n : f ) {
                textStremam << n << ' ';
            }
            textStremam <<  "\n";
            m = mean ( f );
            textStremam << "mean      : " <<  m << "\n";
            textStremam << "var       : " <<  var ( f, m ) << "\n";

            textStremam << "lost2found: "  ;
            for ( int n : lost2found ) {
                textStremam << n << ' ';
            }
            textStremam <<  "\n";
            int m1 = m = mean ( lost2found );
```

```
            textStremam << "mean      : " <<  m << "\n";
            textStremam << "var       : " <<  var ( lost2found, m ) << "\n" ↩
               ;

            textStremam << "found2lost: "  ;
            for ( int n : found2lost ) {
                textStremam << n << ' ';
            }
            textStremam <<  "\n";
            int m2 = m = mean ( found2lost );
            textStremam << "mean      : " <<  m << "\n";
            textStremam << "var       : " <<  var ( found2lost, m ) << "\n" ↩
               ;

            if ( m1 < m2 ) {
                textStremam << "mean(lost2found) < mean(found2lost)" << "\n ↩
                   ";
            }

            int min, sec;
            millis2minsec ( t, min, sec );
            textStremam << "time      : " <<  min  << ":"  << sec << "\n";

            double res = ( ( ( ( double ) m1+ ( double ) m2 ) /2.0 ) /8.0 ) ↩
                /1024.0;
            textStremam << "U R about " << res << " Kilobytes\n";

            tfile.close();
        }
        return ret;
    }

public slots :

    void updateHeroes ( const QImage &image, const int &x, const int &y );
    //void stats ( const int &t );
    void endAndStats ( const int &t );
};

#endif // BrainBWin
```

BrainBWin.cpp

```
            /**
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBWin.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 "BrainBWin.h"

const QString BrainBWin::appName = "NEMESPOR BrainB Test";
const QString BrainBWin::appVersion = "6.0.3";

BrainBWin::BrainBWin ( int w, int h, QWidget *parent ) : QMainWindow (  ↩
   parent )
{

//    setWindowTitle(appName + " " + appVersion);
//    setFixedSize(QSize(w, h));

        statDir = appName + " " + appVersion + " - " + QDate::currentDate() ↩
           .toString() + QString::number ( QDateTime:: ↩
           currentMSecsSinceEpoch() );

        brainBThread = new BrainBThread ( w, h - yshift );
        brainBThread->start();

        connect ( brainBThread, SIGNAL ( heroesChanged ( QImage, int, int ) ↩
            ),
                  this, SLOT ( updateHeroes ( QImage, int, int ) ) );

        connect ( brainBThread, SIGNAL ( endAndStats ( int ) ),
                  this, SLOT ( endAndStats ( int ) ) );

}
```

```
void BrainBWin::endAndStats ( const int &t )
{

        qDebug()  << "\n\n\n";
        qDebug()  << "Thank you for using " + appName;
        qDebug()  << "The result can be found in the directory " + statDir;
        qDebug()  << "\n\n\n";

        save ( t );
        close();
}

void BrainBWin::updateHeroes ( const QImage &image, const int &x, const int &
    y )
{

        if ( start && !brainBThread->get_paused() ) {

                int dist = ( this->mouse_x - x ) * ( this->mouse_x - x ) +
                    ( this->mouse_y - y ) * ( this->mouse_y - y );

                if ( dist > 121 ) {
                        ++nofLost;
                        nofFound = 0;
                        if ( nofLost > 12 ) {

                                if ( state == found && firstLost ) {
                                        found2lost.push_back ( brainBThread
                                            ->get_bps() );
                                }

                                firstLost = true;

                                state = lost;
                                nofLost = 0;
                                //qDebug() << "LOST";
                                //double mean = brainBThread->meanLost();
                                //qDebug() << mean;

                                brainBThread->decComp();
                        }
                } else {
                        ++nofFound;
                        nofLost = 0;
                        if ( nofFound > 12 ) {

                                if ( state == lost && firstLost ) {
                                        lost2found.push_back ( brainBThread
                                            ->get_bps() );
                                }
```

```
                                state = found;
                                nofFound = 0;
                                //qDebug() << "FOUND";
                                //double mean = brainBThread->meanFound();
                                //qDebug() << mean;

                                brainBThread->incComp();
                        }

                }

        }
        pixmap = QPixmap::fromImage ( image );
        update();
}

void BrainBWin::paintEvent ( QPaintEvent * )
{
        if ( pixmap.isNull() ) {
                return;
        }

        QPainter qpainter ( this );

        xs = ( qpainter.device()->width() - pixmap.width() ) /2;
        ys = ( qpainter.device()->height() - pixmap.height() +yshift ) /2;

        qpainter.drawPixmap ( xs, ys, pixmap );

        qpainter.drawText ( 10, 20, "Press and hold the mouse button on the ↩
            center of Samu Entropy" );

        int time = brainBThread->getT();
        int min, sec;
        millis2minsec ( time, min, sec );
        QString timestr = QString::number ( min ) + ":" + QString::number ( ↩
            sec ) + "/10:0";
        qpainter.drawText ( 10, 40, timestr );

        int bps = brainBThread->get_bps();
        QString bpsstr = QString::number ( bps ) + " bps";
        qpainter.drawText ( 110, 40, bpsstr );

        if ( brainBThread->get_paused() ) {
                QString pausedstr = "PAUSED (" + QString::number (  ↩
                   brainBThread->get_nofPaused() ) + ")";

                qpainter.drawText ( 210, 40, pausedstr );
        }
```

```
        qpainter.end();
}

void BrainBWin::mousePressEvent ( QMouseEvent *event )
{

        brainBThread->set_paused ( false );

}

void BrainBWin::mouseReleaseEvent ( QMouseEvent *event )
{

        //brainBThread->set_paused(true);

}

void BrainBWin::mouseMoveEvent ( QMouseEvent *event )
{

        start = true;

        mouse_x = event->pos().x() -xs - 60;
        //mouse_y = event->pos().y() - yshift - 60;
        mouse_y = event->pos().y() - ys - 60;
}

void BrainBWin::keyPressEvent ( QKeyEvent *event )
{

        if ( event->key() == Qt::Key_S ) {
                save ( brainBThread->getT() );
        } else if ( event->key() == Qt::Key_P ) {
                brainBThread->pause();
        } else if ( event->key() == Qt::Key_Q ||  event->key() == Qt:: ↩
           Key_Escape ) {
                close();
        }

}

BrainBWin::~BrainBWin()
{

}
```

BrainBThread.h

```
            #ifndef BrainBThread_H
#define BrainBThread_H

/**
 * @brief Benchmarking Cognitive Abilities of the Brain with Computer Games
 *
 * @file BrainBThread.h
 * @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 <QThread>
#include <QSize>
#include <QImage>
#include <QDebug>
#include <sstream>
#include <QPainter>
#include <cstdlib>
#include <ctime>
#include <vector>
#include "opencv2/opencv.hpp"
#include "opencv2/core/core.hpp"
#include "opencv2/imgproc/imgproc.hpp"

class Hero;
typedef std::vector<Hero> Heroes;

class Hero
```

```
{

public:
    int x;
    int y;
    int color;
    int agility;
    int conds {0};
    std::string name;

    Hero ( int x=0, int  y=0, int color=0, int agility=1,  std::string name ↩
        ="Samu Entropy" ) :
        x ( x ), y ( y ), color ( color ), agility ( agility ), name ( name ↩
            )
    {}
    ~Hero() {}

    void move ( int maxx, int maxy, int env ) {

        int newx = x+ ( ( ( double ) agility*1.0 ) * ( double ) ( std::rand ↩
            () / ( RAND_MAX+1.0 ) )-agility/2 ) ;
        if ( newx-env > 0 && newx+env < maxx ) {
            x = newx;
        }
        int newy = y+ ( ( ( double ) agility*1.0 ) * ( double ) ( std::rand ↩
            () / ( RAND_MAX+1.0 ) )-agility/2 );
        if ( newy-env > 0 && newy+env < maxy ) {
            y = newy;
        }

    }

};

class BrainBThread : public QThread
{
    Q_OBJECT


     //Norbi
    cv::Scalar cBg { 247, 223, 208 };
    cv::Scalar cBorderAndText { 47, 8, 4 };
    cv::Scalar cCenter { 170, 18, 1 };
    cv::Scalar cBoxes { 10, 235, 252 };


    /*
    //Matyi
    cv::Scalar cBg { 86, 26, 228 };
    cv::Scalar cBorderAndText { 14, 177, 232 };
```

```
    cv::Scalar cCenter { 232, 14, 103 };
    cv::Scalar cBoxes { 14, 232, 195 };
*/

    Heroes heroes;
    int heroRectSize {40};

    cv::Mat prev {3*heroRectSize, 3*heroRectSize, CV_8UC3, cBg };
    int bps;
    long time {0};
    long endTime {10*60*10};
    int delay {100};

    bool paused {true};
    int nofPaused {0};

    std::vector<int> lostBPS;
    std::vector<int> foundBPS;

    int w;
    int h;
    int dispShift {40};

public:
    BrainBThread ( int w = 256, int h = 256 );
    ~BrainBThread();

    void run();
    void pause();
    void set_paused ( bool p );
    int getDelay() const {

        return delay;

    }
    void setDelay ( int delay ) {

        if ( delay > 0 ) {
            delay = delay;
        }

    }

    void devel() {

        for ( Hero & hero : heroes ) {

            hero.move ( w, h, ( h<w ) ?h/10:w/10 );

        }
```

```
}

int nofHeroes () {

    return  heroes.size();

}

std::vector<int> &lostV () {

    return  lostBPS;

}

std::vector<int> &foundV () {

    return  foundBPS;

}

double meanLost () {

    return  mean ( lostBPS );

}

double varLost ( double mean ) {

    return  var ( lostBPS, mean );

}

double meanFound () {

    return  mean ( foundBPS );

}

double varFound ( double mean ) {

    return  var ( foundBPS, mean );

}

double mean ( std::vector<int> vect ) {

    double sum = std::accumulate ( vect.begin (), vect.end (), 0.0 );
    return  sum / vect.size();
```

```
}

double var ( std::vector<int> vect, double mean ) {

    double accum = 0.0;
    std::for_each ( vect.begin (), vect.end (), [&] ( const double d ) ↩
       {
        accum += ( d - mean ) * ( d - mean );
    } );

    return sqrt ( accum / ( vect.size()-1 ) );
}

int get_bps() const {

    return bps;

}

int get_w() const {

    return w;

}


bool get_paused() const {

    return paused;

}

int get_nofPaused() const {

    return nofPaused;

}

void decComp() {

    lostBPS.push_back ( bps );

    if ( heroes.size() > 1 ) {
        heroes.pop_back();
    }

    for ( Hero & hero : heroes ) {
        if ( hero.agility >= 5 ) {
            hero.agility -= 2;
        }
```

```
        }

    }

    void incComp() {

        foundBPS.push_back ( bps );

        if ( heroes.size() > 300 ) {

            return;

        }

        /*
        Hero other ( w/2 + 200.0*std::rand() / ( RAND_MAX+1.0 )-100,
                     h/2 + 200.0*std::rand() / ( RAND_MAX+1.0 )-100,
                     255.0*std::rand() / ( RAND_MAX+1.0 ), 11, "New Entropy "
                        " );
                     */

double rx = 200.0;
if(heroes[0].x - 200 < 0)
  rx = heroes[0].x;
else if(heroes[0].x + 200 > w)
  rx = w - heroes[0].x;

double ry = 200.0;
if(heroes[0].y - 200 < 0)
  ry = heroes[0].y;
else if(heroes[0].y + 200 > h)
  ry = h - heroes[0].y;

        Hero other ( heroes[0].x + rx*std::rand() / ( RAND_MAX+1.0 )-rx/2,
                     heroes[0].y + ry*std::rand() / ( RAND_MAX+1.0 )-ry/2,
                     255.0*std::rand() / ( RAND_MAX+1.0 ), 11, "New Entropy "
                        " );

        heroes.push_back ( other );

        for ( Hero & hero : heroes ) {

            ++hero.conds;
            if ( hero.conds == 3 ) {
                hero.conds = 0;
                hero.agility += 2;
            }

        }
```

```
}

void draw () {

    cv::Mat src ( h+3*heroRectSize, w+3*heroRectSize, CV_8UC3, cBg );

    for ( Hero & hero : heroes ) {

        cv::Point x ( hero.x-heroRectSize+dispShift, hero.y- ↩
           heroRectSize+dispShift );
        cv::Point y ( hero.x+heroRectSize+dispShift, hero.y+ ↩
           heroRectSize+dispShift );

        cv::rectangle ( src, x, y, cBorderAndText );

        cv::putText ( src, hero.name, x, cv::FONT_HERSHEY_SIMPLEX, .35, ↩
            cBorderAndText, 1 );

        cv::Point xc ( hero.x+dispShift , hero.y+dispShift );

        cv::circle ( src, xc, 11, cCenter, CV_FILLED, 8, 0 );

        cv::Mat box = src ( cv::Rect ( x, y ) );

        cv::Mat cbox ( 2*heroRectSize, 2*heroRectSize, CV_8UC3, cBoxes  ↩
           );
        box = cbox*.3 + box*.7;

    }

    cv::Mat comp;

    cv::Point focusx ( heroes[0].x- ( 3*heroRectSize ) /2+dispShift,  ↩
       heroes[0].y- ( 3*heroRectSize ) /2+dispShift );
    cv::Point focusy ( heroes[0].x+ ( 3*heroRectSize ) /2+dispShift,  ↩
       heroes[0].y+ ( 3*heroRectSize ) /2+dispShift );
    cv::Mat focus = src ( cv::Rect ( focusx, focusy ) );

    cv::compare ( prev, focus, comp, cv::CMP_NE );

    cv::Mat aRgb;
    cv::extractChannel ( comp, aRgb, 0 );

    bps = cv::countNonZero ( aRgb ) * 10;

    //qDebug()  << bps << " bits/sec";

    prev = focus;
```

```
        QImage dest ( src.data, src.cols, src.rows, src.step, QImage:: ↩
           Format_RGB888 );
        dest=dest.rgbSwapped();
        dest.bits();

        emit heroesChanged ( dest, heroes[0].x, heroes[0].y );

    }

    long getT() const {

        return time;

    }

    void finish () {

        time = endTime;

    }


signals:

    void heroesChanged ( const QImage &image, const int &x, const int &y );
    void endAndStats ( const int &t );

};

#endif // BrainBThread_H
```

BrainBThread.cpp

```
            /**
 * @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;

}
```

Megoldás forrása:

A BrainB egy tesztelő program ami azt figyeli hogy a játékos mennyire tud a saját objektumára figyelni. A saját objektuma felet kell tartania az egér mutatót és minél tovább sikerül rajtatartani annál több más objektumot pakol le a program ezzel nehezítve a játékos dolgát.

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/exar
https://progpater.blog.hu/2016/11/13/hello_samu_a_tensorflow-bol

Tanulságok, tapasztalatok, magyarázat...

8.2. Mély MNIST

Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.3. Minecraft-MALMÖ

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

[**?**]

Juhász István: Magas zintű programozási nyelvek (pici-könyv) 11-27. oldal 56-71:

A számítógépek programozására kialakult nyelveknek három szintjét különböztetjük meg:

gépi nyelv :Minden processzor rendelkezik saját gépi nyelvvel, és csak az adott gépi nyelven írt programokat tudja végrehajtani.

assembly szintő nyelv

magas szintő nyelv: Egy magas szintő programozási nyelvet szintaktikai és szemantikai szabályainak együttese határoz meg.

A fordítóprogram egy speciális program, amely a magas szintő nyelven megírt forrásprogramból gépi kódú tárgyprogramot állít elő.

Feladatai:

lexikális elemzés,szintaktikai elemzés, szemantikai elemzés, kódgenerálás

A lexikális elemzés során a forrásszöveget feldarabolja lexikális egységekre (l. 2.2. alfejezet), a szintaktikai elemzés folyamán ellenırzi, hogy teljesülnek-e az adott nyelv szintaktikaiszabályai.

A programnyelvek osztályozása:

Imperatív nyelvek (Algoritmikus nyelvek)

Deklaratív nyelvek (Nem algoritmikus nyelvek)

Karakterkészlet

Minden program forrásszövegének legkisebb alkotórészei a karakterek. Az eljárásorientált nyelvek esetén ezek a következők:

lexikális egységek, szintaktikai egységek, utasítások, programegységek, fordítási egységek, program

Minden nyelv definiálja a saját karakterkészletét.

Általában a karaktereket a következı módon kategorizálják:

betők, számjegyek, egyéb karakterek

Lexikális egységek

Lexikális egységek :

többkarakteres szimbólum, szimbolikus név, címke, megjegyzés, literál

UTASÍTÁSOK:

Értékadó utasítás: Feladata beállítani vagy módosítani egy (esetleg több) változó értékkomponensét a program futásának bármely pillanatában.

Üres utasítás: Jelentısége viszont általánosságban abban áll, hogy segítségével egyértelmőprogramszerkezet alakítható ki.

Ugró utasítás: Az ugró (vagy feltétel nélküli vezérlésátadó) utasítás segítségével a program egy adott pontjáról egy adott címkével ellátott végrehajtható utasításra adhatjuk át a vezérlést.

Kétirányú elágaztató utasítás (feltételes utasítás): A kétirányú elágaztató utasítás arra szolgál, hogy a program egy adott pontján két tevékenység közül válasszunk, illetve egy adott tevékenységet végrehajtsunk vagy sem

Többirányú elágaztató utasítás: A többirányú elágaztató utasítás arra szolgál, hogy a program egy adott pontján egymást kölcsönösen kizáró akárhány tevékenység közül egyet végrehajtsunk.

Ciklusszervező utasítások: A ciklusszervezı utasítások lehetıvé teszik, hogy a program egy adott pontján egy bizonyos tevékenységet akárhányszor megismételjünk.

Feltételes ciklus: Ennél a ciklusnál az ismétlıdést egy feltétel igaz vagy hamis értéke szabályozza

Előírt lépésszámú ciklus: Minden esetben tartozik hozzá egy változó, a ciklusváltozó. A változó által felvett értékekre fut le a ciklusmag. A változó az értékeit egy tartományból veheti föl. Ezt a tartományt a fejben adjuk meg kezdı- és végértékével.

Felsorolásos ciklus: A felsorolásos ciklus az elıírt lépésszámú ciklus egyfajta általánosításának tekinthetı. Van ciklusváltozója, amely explicit módon megadott értékeket vesz fel, és minden felvett érték mellett lefut a mag.

Végtelen ciklus: A végtelen ciklus az a ciklusfajta, ahol sem a fejben, sem a végben nincs információ az ismétlıdésre vonatkozóan.

Összetett ciklus: Az elızı négy ciklusfajta kombinációiból áll össze.

10.2. Programozás bevezetés

[KERNIGHANRITCHIE]

Kernighan+Ritchie: C programozási nyelv:

1 fejezet:

printf

%d a számot decimális egészként írja ki; %6d a számot decimális egészként, legalább hat karakter széles mezőbe írja ki; %f a számot lebegőpontosként írja ki; %6f a számot lebegőpontosként, legalább 6 karakter

széles mezőbe írja ki; %.2f a számot lebegőpontosként, két tizedessel írja ki; %6.2f a számot lebegőpontosként, legalább 6 karakter széles mezőbe, két tizedessel írja ki. szimbolikus állandók #define név helyettesítő szöveg

1.5. Karakteres adatok bevitele és kivitele

c = getchar( ) végrehajtása után a c változó a bemenő szöveg következő karakterét fogja tartalmazni. A karakterek általában a terminálról (billentyűzetről) érkeznek, az adatállományból történő beolvasással a 7. fejezetben fogunk foglalkozni. A putchar függvény minden egyes hívásakor kiír egy karaktert. A putchar(c) végrehajtása során a c egész típusú változó tartalma mint egy karakter íródik ki, általában a képernyőre. A putchar és a printf hívások felváltva is történhetnek, ilyenkor a kimenet a hívások sorrendjében fog megjelenni.

A függvénydefiníció általános alakja: visszatérési típus függvénynév (paraméter-deklarációk, ha vannak) { deklarációk utasítások }

Változónevek: A nevek betűkből és számjegyekből állhatnak és az első karakterüknek betűnek kell lenni. Az aláhúzás-karakter (_) betűnek számít, és alkalmazásával sokszor javítható a hosszú változónevek olvashatósága.

Deklarációk: A felhasználása előtt minden változót deklarálni kell, bár bizonyos deklarációk implicit módon, a programkörnyezet alapján is létrejöhetnek. A deklaráció egy típust határoz meg, és utána egy vagy több adott típusú változó felsorolása (listája) áll.

Vezérlési szerkezetek: Egy nyelv vezérlésátadó utasításai az egyes műveletek végrehajtási sorrendjét határozzák meg.

A C nyelvben a pontosvessző az utasításlezáró jel (terminátor), szemben a Pascal nyelvvel, ahol elválasztó szerepe van. A { } kapcsos zárójelekkel deklarációk és utasítások csoportját fogjuk össze egyetlen összetett utasításba vagy blokkba, ami szintaktikailag egyenértékű egyetlen utasítással.

Az if-else utasítást döntés kifejezésére használjuk.

A switch utasítás is a többirányú programelágaztatás egyik eszköze. Az utasítás úgy működik, hogy összehasonlítja egy kifejezés értékét több egész értékű állandó kifejezés értékével, és az ennek megfelelő utasítást hajtja végre.

A while (kifejezés) utasítás szerkezetben a program először kiértékeli a kifejezést. Ha annak értéke nem nulla (igaz), akkor az utasítást végrehajtja, majd a kifejezés újra kiértékelődik. Ez a ciklus mindaddig folytatódik, amíg a kifejezés nullává (hamissá) nem válik, és ilyen esetben a program végrehajtása az utasítás utáni helyen folytatódik.

A függvényekkel a nagyobb számítási feladatok kisebb egységekre oszthatók, így a programozó felhasználhatja a már meglévő egységeket és nem kell minden alkalommal elölről kezdeni a munkát. A függvények a működésük részleteit gyakran elrejtik a program többi része elől, de jól megírt függvények esetén nincs is szükség ezekre a részletekre.

A külső változók: A C nyelvű program külső objektumok – változók vagy függvények – halmaza. A külső jelzőt a belső ellentéteként használjuk, ami a függvények belsejében definiált argumentumok és változók leírására alkalmas. A külső változókat a függvényen kívül definiáljuk, így elvileg több függvényben is felhasználhatók.

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

10.3. Programozás

[BMECPP]

Benedek Zoltán, Levendovszky Tihamér: Szoftverfejlesztés C++ nyelven 6. oldalig + 487. oldal

A c és c++ nyelv

Függvény paraméterek és visszatérési érték

üres paraméter listával való függvény hívás :

c: tetszőleges számú paraméter

c++: void típusú paraméterrel hívható

c++ -ban a tetszőleges számu paraméterrel hívható fügvény:

```
void f(...)
{
//...
}
```

Vissza térési érték c-ben alapértelmezeten int típusu c++ ban nem támogatott az alapértelmezett visszatérés, a fordító hibát dob.

A main fügvény két típusa:

```
int main ()
{

  }
```

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

  }
```

argc: parancssor-argumentumok száma

argv: parancssori argumentumok

A bool típus

c++ nyelvben meg jelenik a bool típus értéke true vagy false át képezhető intre ahol a 0 jelenti a hamisat és minden más az igaz értéket.

wchar_t beépített típussá vált használata meg egyezik a c-beli használattal

Változódeklarálás mint utasítás c++ ban minden olyan helyen állhat változó deklaráció ahol utasítás állhat. A ható körük kezdete a deklaráció helye a vége pedig az adott blokk vége ha if ben vagy for ban deklarálunk akkor az adott utasítás vége a ható kör vége

Függvények túlterhelése a c++ nyelvben nem a csak a függvény nevével hanem a nevével és a paraméter listával lesznek azonosítva a függvények.

C++ ban a visza fele kompatibilitás miatt minden a c ben elő forduló utasítás használható az átállást segítő táblázatot lásd a könyv 487 oldalától.

10.4. Mobil programozás

Ekler Péter, Forstner Bertalan, Kelényi Imre: Bevezetés a mobilprogramozásba 35-49. oldal

Általános információk: A Python egy általános célú programozási nyelv. Guido van Rossum 1990-ben alkotta meg ezt a fejlesztők számára rengeteg pozitív tulajdonsággal rendel- kező nyelvet, amely magas szintű, dinamikus, objektumorientált és platform- független. A fejlesztés megkönnyítéséhez magas szintű típusokat is támogat, mint például különféle listák és szótárak.

Előnyei miatt az interpretert elkészítették a Symbian mobil operációs rendszer S60 platformja alá is, hogy megkönnyítsék az egyszerűbb szoftver és prototípusfejlesztést mobil készülékekre. A Python S60 implementáció nagy- részt lefedi az alap Pythont, azonban egy-két modult kihagytak belőle, amely nem kapcsolódik szorosan a mobileszközökhöz.

A Python nyelv jellemzői Amikor alkalmazásokat fejlesztünk, sok esetben szükség van olyan részek megírására is, amelyek az adott probléma szempontjából nem relevánsak, el- készítésük mégis sok időt vesz igénybe. Ilyenek például a különféle fájlkezelő metódusok, hálózatkezelés, GUI kialakítása stb. A szokásos programírás/fordítás/tesztelés/újrafordítás ciklust egy professzi- onális C, C++ vagy Java-fejlesztő is lassúnak találja egy idő után, márpedig er- re minden módosítás esetében szükség van. A Python ebben is segít, esetében ugyanis nincs szükség a fordítás fázisára, az értelmezőnek elegendő a Python- forr

A Python nyelv legfőbb jellemzője, hogy behúzásalapú a szintaxisa. A prog- ramban szereplő állításokat az azonos szintű behúzásokkal tudjuk csoportok- ba szervezni, nincs szükség kapcsos zárójelre vagy explicit kulcsszavakra (pl. begin, end). Egy adott blokk végét egy kisebb behúzású sor jelzi, tehát például üres sor lehet a blokkon belül. A szkript első utasítása nem lehet behúzott. Fontos, hogy a behúzásokat egységesen kezeljük, tehát vagy mindenhol tabot, vagy egységesen szóközt használjuk. Bizonyos szövegszerkesztőkben akár azt is beállíthatjuk, hogy a tab billentyű lenyomása 4 szóközt jelentsen. A nyelv további sajátossága, hogy a sor végéig tart egy utášítás, nincs szükség a megszokott ;’ használatára.

Pythonban minden adatot objektumok reprezentálnak. Az adatokon végezhe- tő műveleteket az objektum típusa határozza meg. Pythonban nincs szükség a változók típusainak explicit megadására, a rendszer futási időben, automa- tikusan „kitalálja" a változók típusát a hozzárendelt érték alapján. Az adattípusok a következők lehetnek: számok, sztringek, ennesek (tuples, n-es), listák, szótárak (dictionaries).

Változók és alkalmazásuk Pythonban a változók alatt az egyes objektumokra mutató referenciákat ér- tünk. Maguknak a változóknak nincsenek típusai, így egy szkript futása so- rán bármely, akár különböző típusú objektumra is hivatkozhatnak. Ameny- nyiben egy objektumra az utolsó hivatkozást is töröljük (pl. a változói már más objektumokra mutatnak), az automatikus garbage collector szabadítja fel a memóriaterületet. A nem létező változókra való hivatkozás futás közbeni kivételt okoz.

Pythonban függvényeket a def kulcsszóval definiálhatunk. A függvényekre mint értékekre is tekinthetünk, hiszen azok továbbadhatók más függvények- nek, illetve objektumkonstruktoroknak is. A függvények rendelkeznek para- méterekkel, amelyeknek, a szokásos megkötésekkel és szintaxissal, alapér- telmezett értéket is adhatunk. A paraméterek érték szerint adódnak át, kivé- ve az úgynevezett mutable típusok (pl. listák, szótárak), amelyek függvénybe- li megváltoztatása hatással van a hívókódrészletben lévő objektumra is. Az egyes paramétereknek a szokásos szintaxissal (paraméter értelmezett értéket is adhatunk.

A Python nyelv támogatja a klasszikus, objektumorientált fejlesztési eljárá- sokat, amelyeket ebben az alfejezetben röviden átnézünk. Definiálhatunk osztályokat, amelyek példányai az objektumok. Az osztályoknak lehetnek att- ribútumaik: objektumok, illetve függvények. Ez utóbbiakat metódusoknak vagy tagfüggvénynek is hívjuk. Ezenkívül az osztályok örökölhetnek más osz- tályokból is.

III. rész

Második felvonás

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

11. fejezet

Helló, Arroway!

11.1. 0.hét

Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba. Gyors prototípusfejlesztés Python és Java nyelven (35-51 oldal)

Általános információk: A Python egy általános célú programozási nyelv. Guido van Rossum 1990-ben alkotta meg ezt a fejlesztők számára rengeteg pozitív tulajdonsággal rendelkező nyelvet, amely magas szintű, dinamikus, objektumorientált és platform- független. A fejlesztés megkönnyítéséhez magas szintű típusokat is támogat, mint például különféle listák és szótárak. Előnyei miatt az interpreter elkészítették a Symbian mobil operációs rendszer S60 platformja alá is, hogy megkönnyítsék az egyszerűbb szoftver és prototípusfejlesztést mobil készülékekre. A Python S60 implementáció nagy- részt lefedi az alap Pythont, azonban egy-két modult kihagytak belőle, amely nem kapcsolódik szorosan a mobileszközökhöz. A Python nyelv jellemzői Amikor alkalmazásokat fejlesztünk, sok esetben szükség van olyan részek megírására is, amelyek az adott probléma szempontjából nem relevánsak, el- készítésük mégis sok időt vesz igénybe. Ilyenek például a különféle fájlkezelő metódusok, hálózatkezelés, GUI kialakítása stb. A szokásos programírás/fordítás/tesztelés/újra fordítás ciklust egy professzionális C, C++ vagy Java-fejlesztő is lassúnak találja egy idő után, márpedig erre minden módosítás esetében szükség van. A Python ebben is segít, esetében ugyanis nincs szükség a fordítás fázisára, az értelmezőnek elegendő a Python- forr A Python nyelv legfőbb jellemzője, hogy behúzásalapú a szintaxisa. A prog- ramban szerepl˝o állításokat az azonos szint˝u behúzásokkal tudjuk csoportok- ba szervezni, nincs szükség kapcsos zárójelre vagy explicit kulcsszavakra (pl. begin, end). Egy adott blokk végét egy kisebb behúzású sor jelzi, tehát például üres sor lehet a blokkon belül. A szkript els˝o utasítása nem lehet behúzott. Fontos, hogy a behúzásokat egységesen kezeljük, tehát vagy mindenhol tabot, vagy egységesen szóközt használjuk. Bizonyos szövegszerkeszt ˝okben akár azt is beállíthatjuk, hogy a tab billenty˝u lenyomása 4 szóközt jelentsen. A nyelv további sajátossága, hogy a sor végéig tart egy utášítás, nincs szükség a megszokott ;' használatára. Pythonban minden adatot objektumok reprezentálnak. Az adatokon végezhe- t˝o m˝uveleteket az objektum típusa határozza meg. Pythonban nincs szükség a változók típusainak explicit megadására, a rendszer futási id˝oben, automa- tikusan „kitalálja" a változók típusát a hozzárendelt érték alapján. Az adattípusok a következ˝ok lehetnek: számok, sztringek, ennesek (tuples, n-es), listák, szótárak (dictionaries). Változók és alkalmazásuk Pythonban a változók alatt az egyes objektumokra mutató referenciákat ér- tünk. Maguknak a változóknak nincsenek típusai, így egy szkript futása so- rán bármely, akár különböz˝o típusú objektumra is hivatkozhatnak. Amennyiben egy objektumra az utolsó hivatkozást is töröljük (pl. a változói már más objektumokra mutatnak), az automatikus garbage collector szabadítja fel a memóriaterületet. A nem létez˝o változókra való hivatkozás

futás közbeni kivételt okoz. Pythonban függvényeket a def kulcsszóval definiálhatunk. A függvényekre mint értékekre is tekinthetünk, hiszen azok továbbadhatók más függvények- nek, illetve objektumkonstruktoroknak is. A függvények rendelkeznek para- méterekkel, amelyeknek, a szokásos megkötésekkel és szintaxissal, alapér- telmezett értéket is adhatunk. A paraméterek érték szerint adódnak át, kivé- ve az úgynevezett mutable típusok (pl. listák, szótárak), amelyek függvénybe- li megváltoztatása hatással van a hívókódrészletben lév˝o objektumra is. Az egyes paramétereknek a szokásos szintaxissal (paraméter értelmezett értéket is adhatunk. A Python nyelv támogatja a klasszikus, objektumorientált fejlesztési eljárásokat, amelyeket ebben az alfejezetben röviden átnézünk. Definiálhatunk osztályokat, amelyek példányai az objektumok. Az osztályoknak lehetnek att- ribútumaik: objektumok, illetve függvények. Ez utóbbiakat metódusoknak vagy tagfüggvénynek is hívjuk. Ezenkívül az osztályok örökölhetnek más osz- tályokból is.

Összehasonlítás c++ java:

kifejezés fogalom ua.:Kifejezések olyan kód részletek amelyek valami műveletet végeznek és a művelet eredményét adják vissza, vagy pedig valami értéket reprezentálnak. , stb:Java esetén egy osztály definíciója és deklarációja,azaz az eljárások feje és megvalósítása nem szétválasztható, mint például C++-ban. Ahogy C++-ban, úgy a Javában is felsorolhatunk változókat,de a Java megengedi, hogy inicializáljuk is azokat,konstruktoron kívül konstruktor: Ennek neve – ahogy C++-ban is – megegyezik az osztály nevével, nincs visszatérési értéke. final: Végleges, azaz konstans érték (nem ugyanaz mint a C++ const! Hiszen itt a referencia az az érték, ami nem változtatható, de ettől még a hivatkozott objektum állapota változtatható) Javában minden metódus olyan, mint C++-ban a virtual kulcsszóval ellátottak.Itt a dinamikus kötés automatikusan létrejön! Ha a metódus az osztályban nem static jelzővel illetett, akkor nincs olyan momentum, amely egyértelműen az alaposztályához kötné (azaz példánymetódus). A kötés ekkor csakis dinamikus lehet, ami azért érdekes helyzet, mert előre senki sem tudhatja (legfőképpen a JVM nem), hogy éppen melyik példánymetódus fut, azaz a kötés csakis a futási idő alatt jöhet létre. Ezért illetik egyúttal a dinamikus kötést a következő jelzőkkel:futás alatti kötés:runtime binding, ,késői kötés - late binding, illetve dinamikus kötés alatti példánymetódus-keresés folyamatát dinamikus metóduskeresésnek (dynamic method lookup). A keresés végeredménye ebben az esetben csakis az lehet, hogy az éppen aktuálisan futó objektum kerül végrehajtásra. c++ ehhez a virtual jelzőt kell használni Javában minden objektum:referencia: klónozás A java a c++-al ellentétben platform független mivel egy virtuális gépen fut ez a JVM és JRE ezen felül van a JDK ami már a java programok fejlesztéséhez szükséges. Ebből következik, hogy a java programoknak rosszabb az optimalizáltságga a c+++ hoz képes amit a fordító programok gépi nyelvre fordítanak de szintén ebből következik hogy könnyebben hordozhatók és biztonságosabbak a java programok. C++ elsősorban rendszerprogramozásra használják míg A javat alkalmazások fejlesztésére. Széleskörben használják webes, mobilos alkalmazások fejlesztésére. Ennek fő okai a könnyű egységekre bontás és tagolása a programoknak. Goto utasítás a c6+ támogatja de a java nem . (de alapvetően egy jól struktúrát programban nincs is szükség a goto utasítás használatára) A c++ támogatja a többszörös öröklődést míg a java nem támogatja a többszörös öröklődést osztályok között.DE az interfacek használatával a jávában is át hidalható ez a probléma Az operátor túlterhelést nem támogatja a java a c++-al szemben ahol ez egy bevet metódus a javaban külön függvények segítségével oldható fel ez a probléma., Míg c++ teljesen támogatja a pointerek felhasználását a programokban addig a java csak belsőleg támogatja őket nem írható mutatókat explicite használó program. Ez azt jelenti, hogy a java korlátozta a mutató támogatását C++ ban mind érték mind referencia alapján lehet hívni addig java nem támogatja a referencia alapú hívást csak érték alapú hívások vannak a jávában. A java mindig egy öröklődési fát tartalmaz mert minden osztály az Object class gyermeke. A jávában mindig az Object Class a gyökere az öröklődési fának. Ezzel szemben a c++ mindig új öröklődési fát készít. A jdk JRE JVM felépitéséből adódóan a Java nem lép kapcsolatba közvetlenül a hardwarerel míg a c++ fordítási metódusából adódóan az abban írt programok igen.

11.2. 1.hét

Eljárás orientál vs Objektumorientált

Elsőként tisztázzuk a két fogalmat. Az eljárás orientált nyelvek alatt azokat értjük melyek struktúrát programozási elvek alapján működnek, mely eljárás híváson alapszik. Ezek az eljárások a program során bármikor meg hívhatóak, akár saját magukat is hívhatják. A feladatokat lépésről lépésre bontja le az eljárás orientált program változókra és eljárásokra melyek szekvenciális utasításokat tartalmaznak. Ezzel szemben az Objektumorientált nyelvek az objektumokon alapszanak melyeknek vannak tulajdonságuk és viselkedésük melyekkel képesek megváltoztatni a saját és más objektumok tulajdonságait. Az elmondható mind két paradigmáról, hogy kisebrészekre bontják fel a problémát ezen keresztül pedig a kódot is. A nagy különbség a kettő féle felbontásban az hogy az objektumorientáltnál sokall könnyebb utólagosan változtatni a programon egy jól meg tervezet programnál csak az adott objektumon kel választani. Míg az eljárás orientáltnál mivel az egész egy programként kezelődik így az egész program változik. Ebből következik az is, hogy az objektumorientált programok sokkal modulárisabbak egyes modulokat akár több programnál is fel lehet használni ezáltal és az által, hogy jobban tagolható az írása rövidül a fejlesztési idő is. Objektumorientált nyelvekben lehet szabályozni a láthatóságot (public, private, protected) mely segítségével elrejthetők az adatok ezáltal biztonságosabb mint az eljárásorientált nyelvek ahol erre nincs lehetőség. Az objektum orientált nyelveknél könnyebb bővítés az öröklődésnek köszönhetően. Erre az eljárás orientált nyelveknél nincs lehetőség.

Java Object metódusok :

protected Object clone(): Vissza tér egy másolatával az adott objektumnak. x.clone() != x igazat fog adni, x.clone().equals(x) igazat fog adni de ez nem elvárás. A használathoz felül kel definiálni, a protected miatt nem látszik. Ha az adot osztály nem implementálja a cloneable interface-t akkor metódus hívása CloneNotSupportedException fog okozni. Maga az interface nem tartalmazza a clone metódus törzsét ,így csak az implementálása nem elegendő a metódus használatához.

Boolean equals(Object obj) : megalapítja hogy egy objectum egyenlő-e azzal amelyből hívjuk. Abban az esetben kell felüldefiniálni ezt a metódust ha az adott objektum más objektumot is tartalmaz mer ebben az esetben az objektum == operátorral hasonlítja össze és így csak akkor ad igazat ha ugyan arra a helyre mutatnak. Abban az esetben ha felül definiáljuk a equals() metódust a hasCode() metódust is illik felüldefiniálni mivel az egyenlő objectumoknak ugyan annak kell lenije a has kódjának.

Class getClass() Visszatér az osztállyal futás időben mellyel különböző információkat nyerhetünk ki az osztályról.

Int hashCode() Vissza tér egy has értékel az objektumra. Egy futtatás alatt egy adott objektumnak nem szabad változnia a has kódjának de futások között már változhat ha két objectom egyenlő az equals(Object) metódus alapján akkor ugyan azt a has értéket kell hogy kapják az nem elvárás ha két objektum különbözik akkor külön hash kódot kapjanak

Void notify() Egyetlen szálat ébreszt fel, amely az objektum monitorára vár. Az felébresztett szál addig nem folytatódik, amíg az aktuális szál nem oldja fel az objektum zárolását csak az objektum „gazdája” adhatja ki ezt az utasítást. A fel ébresztett szál nem élvez semmilyen előjogot az objektumra

Void notifyAll() Minden szálat fel ébreszt , amely az objektum monitorára vár. A fel ébresztett szálak nem élveznek semmilyen előjogot az objektumra

String toString() Egy stringgle tér vissza ami meg adja az objektumot érdemes felül definiálni alapértelmezetten ezt a kifejezést adja vissza: getClass().getName() + ’@’ + Integer.toHexString(hashCode())

Void wait() A aktuális szál várását eredményezi míg fel nem ébresztik,vagy meg nem szakítják.

Void wait(long timeout, int nanos) aktuális szál várását eredményezi míg fel nem ébresztik,vagy meg nem szakítják vagy el nem telik a meg adott idő

Singleton Design Pattern:

Biztosítani kel, hogy csak egy objektum jöhessen létre az adott osztályból, ezt az objektumot mindenkinek el kel érnie. Ez azzal jár, hogy a lazán kell iniciálni az adott objektumot. Előállítása: összes konstruktort felül kell definiálni egy protecteddel vagy private függvénnyel és kel egy statikus függvény ami hívja a konstruktort ha még nem létezik az adott objektum Akkor érdemes használni ha egy objektumból pontosan egy darab kell pl.:facade objestc,State object.

Abstract Factory Design Pattern

A lényege a Abstract Factory-nek hogy interfacet adjon ahhoz hogy családokat hozzunk létre összetartozó objektumokból. A japán autó gyártásban használt modellt modellezi ahol egy prés kis változásokkal képes több típus adott elemét elő állítani. A programozásban a hordozhatóságra ad egy megoldást ahol az adott rendszerhez kell alkalmazkodni de ez sok #ifdef részel járhat a programban.(sokszor használ singletont az egyes objektumok hoz)

Builder Design Pattern

El választja az elkészítését egy összetett objektumnak a reprezentációjától és ezzel eléri, hogy ez az eljárás képes legyen különböző reprezentációkat el készíteni.

Factory Method Design Pattern:

Meg ad egy felületet egy objektum létrehozásához, de hagyja, hogy az alosztályok döntsék el, melyik osztályt kell példányosítani. A Factory Method lehetővé teszi, hogy az osztály megengedi az alosztályok példányosítását. Definiál egy virtuális konstruktort. A new operátor ártalmas lehet.

Object Pool Design Pattern

Az object pool jelentős teljesítménynövekedést kínálhat; leghatékonyabb azokban a helyzetekben, amikor az osztálypéldány inicializálásának költsége magas, az osztály példányosítási aránya magas, és az egyszerre használt példányszám alacsony. Az Object Pool (más néven erőforráskészletek) az objektum-gyorsítótár kezelésére szolgálnak. Az Objektumkészlethez hozzáférő kliens elkerülheti az új Objektumok létrehozását, ha egyszerűen megkéri a készletet hogy adjon egy már példányosítottat. Általában a pool növekvő medence lesz, vagyis maga a pool új objektumokat hoz létre, ha a pool üres, vagy rendelkezhetünk pool-lal, ami korlátozza a létrehozott objektumok számát.

Prototype Design Pattern

létre hoz egy objektumot majd ezt az objektumot módosítja és clonozza ehez használnia kell a clone abel interfészt . A new operátor káros. Egy objektum az őse minden objekumnak

OO szemlélet

C++ ban:

```
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <ctime>
```

```
class ElsoOsztajom
{
private:
    bool letezikKovVeletlen;
    double kovVeletlen;
public:
    ElsoOsztajom();
    ~ElsoOsztajom();

    double veletlen();
};

ElsoOsztajom::ElsoOsztajom()
{
    letezikKovVeletlen=false;
}

ElsoOsztajom::~ElsoOsztajom()
{
}

double ElsoOsztajom::veletlen()
{
    if (letezikKovVeletlen)
    {
        letezikKovVeletlen =false;
        return kovVeletlen;
    }
    else
    {
        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);

        kovVeletlen=r*v2;
        letezikKovVeletlen =true;

        return r*v1;
    }


}
```

```
int main(){
    double j=0;
    ElsoOsztajom r;
    for (int i=0; i<5; i++)
    {
        j=r.veletlen();
        std::cout<<j<<std::endl;
    }

}
```

Javaban:

```
PolarGenerator:

package ooszemlelet;

import java.lang.Math;

public class PolarGenerator {

  private boolean isStored = false;
  private double stord;

  public double giveTheNext() {

    if (isStored) {

      isStored = false;
      return stord;


    } else {

      double u1, u2, v1, v2 ,w;

      do {
        u1 = Math.random();
        u2 = Math.random();
        v1 = 2 * u1 - 1;
        v2 = 2 * u2 - 1;
        w = v1 * v1 + v2 * v2;
      }
      while (w>1);

      double r =Math.sqrt((-2 * Math.log(w)) / w);
      stord=r * v2;
      isStored = true;
      return r*v1;
```

```
    }
  }



}


Main:

package ooszemlelet;



public class Main {

  public static void main(String[] args) {
    // TODO Auto-generated method stub

    PolarGenerator g = new PolarGenerator();
    for (int i=0; i<10 ; i++) {
      System.out.println(g.giveTheNext());
    }

  }

}
```

11.3. 2.hét

EPAM:Interfész evolúció Java ban

A java 8 élőt csak abstract metódust lehetet az interfészekben használni. Minden metódus public és abstract alapból. A java 8 tól kezdve lehet default és static metódusokat is használni. A default metódus lehetővé teszi a fejlesztőnek hogy anélkül agyon hozza az iterface hez egy metódust hogy azzal befolyásolná azokat az osztályokat amelyek használják az adott interfacet. Ezt a tulajdonságot nevezhetjük vissza felé kompatibilitásnak is mivel ezzel a képességet ki használva egy létező kódba tudunk új funkciót belerakni anélkül hogy tönkretennék a már létező kódot vagy teljesen újra kellene írnunk. A staticus metódusok hasonlóan viselkednek a default metódusokhoz azzal a különbséggel hogy ezeket a metódusokat nem lehet felülírni az osztályokban. Ez a két új funkció felveti a többszörös öröklődés kérdését is, ha van két interface amiben van egy egy ugyan olyan nevű metódus és egy osztály egyszere implementálja mind két interfacet nem lehet eldönteni hogy melyik metódus futsonle. Erre egy megoldás ha az ilyen metódusokat felül definiáljuk

EPAM:Liskov féle helyettesíthetőség elve, öröklődés

osztály definiciók:

```
 public class Vehicle {

    public Vehicle() {
        System.out.println("Creating vehicle!");
    }

    void start() {
        System.out.println("Vehicle is starting!");
    }

}
```

```
  public class Car extends Vehicle {

    public Car() {
        System.out.println("Creating car!");
    }

    @Override
    void start() {
        System.out.println("Car is starting!");
    }

}
```

```
 package com.epam.training;

public class Supercar extends Car {

    public Supercar() {
        System.out.println("Creating supercar!");
    }

    @Override
    void start() {
        System.out.println("Supercar is starting!");
    }

}
```

tesztelése:

```
Vehicle firstVehicle = new Supercar();
firstVehicle.start();
System.out.println(firstVehicle instanceof Car);
```

kimenet:

```
Creating vehicle!
Creating car!
```

```
Creating supercar!
Supercar is starting!
true
```

A firstVehicle létrehozásánál mivel Supercar-ból let létre hozva ezért lefut a Vehicle a Car és a Supercar konstruktora is de mivel a firstVehicle Vehicle típusú így ezen a változón nem érhetőek el sem a Car sem a Supercar azon metódusai amelyek nem egy Vehicle belli metódust írnak felül.

```
Car secondVehicle = (Car) firstVehicle;
secondVehicle.start();
System.out.println(secondVehicle instanceof Supercar);

System.out.println(firstVehicle.toString());
System.out.println(secondVehicle.toString());
```

kimenet:

```
Supercar is starting!
true
com.epam.training.Supercar@25618e91
com.epam.training.Supercar@25618e91
```

Az érték adásnál Car-t adunk árt Car-nak de mivel az átadot Car egy Supercar volt így átadás után is az marad mert csak a címét adta át az objektumnak.

```
Supercar thirdVehicle = new Vehicle();
thirdVehicle.start();
```

Itt mivel a Supercar bőveb mint a Vehicle nem tudja belőle létrehozni .

EPAM: Interfész, Osztály, Absztrak Osztály

Az interfacekkel teljes szintű absztrakció valósítható meg míg a abstract osztállyal csak részleges absztrakció valósítható meg. Az interface egy tervrajzhoz hasonlítható míg az abstract class csak azt engedi meg hogy az eredeti class egyes részeit mi mondhatjuk meg hogy definiáljuk

```
package abstractClassClass;

public abstract class IntegerStorage {

  int [] storage;
  int numberOfElement;
  final int MAX_NUMBER_OF_ELEMENT=10;

  public IntegerStorage() {
    super();
    this.storage =  new int [MAX_NUMBER_OF_ELEMENT];
    numberOfElement = -1;
  }

  abstract public void add(int value);
  abstract public int get();
```

```
  abstract public boolean contain(int value);


}
```

```
package abstractClassClass;

public class IntegerStorageFiFo extends IntegerStorage{

  @Override
  public void add(int value) {
    if (numberOfElement <MAX_NUMBER_OF_ELEMENT) {
      numberOfElement++;
      storage[numberOfElement] = value;
    }

  }

  @Override
  public int get() {
    numberOfElement--;
    return storage[numberOfElement+1];
  }

  @Override
  public boolean contain(int value) {
    int i=0;
    while ((i<=numberOfElement)&& (storage[i]!= value)) {
      i++;
    }

    if (i<=numberOfElement) {
      return true;
    }else {
      return false;
    }
  }


}
```

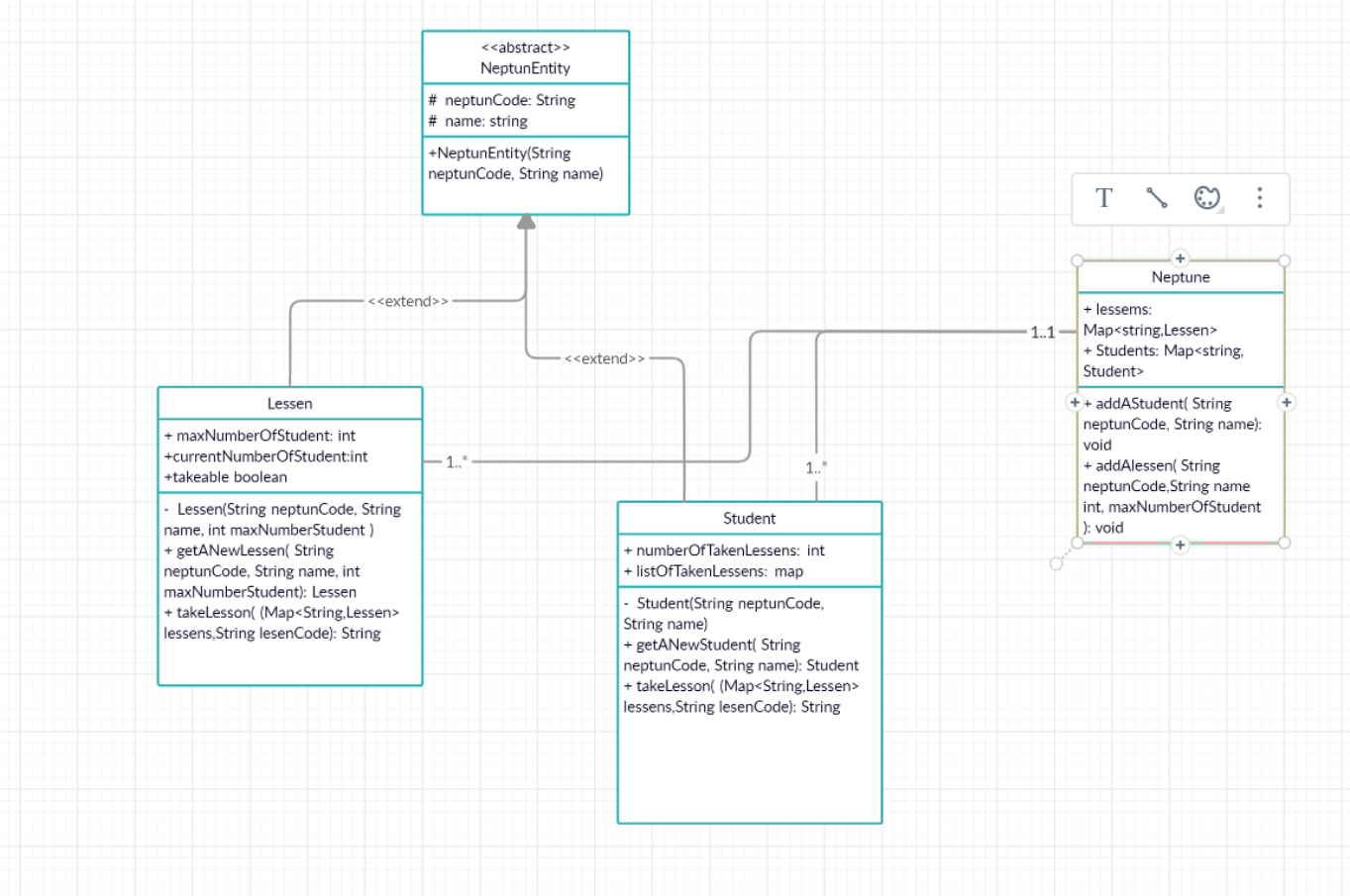
```
package interfaceClass;

public interface IntegerStorage {

  public void add(int value);
  public int get();
  public boolean contain(int value);

}
```

```
package interfaceClass;

public class IntegerStorageFiFo implements IntegerStorage {

  int [] storage;
  int numberOfElement;
  final int MAX_NUMBER_OF_ELEMENT=10;

  public IntegerStorageFiFo() {
    super();
    this.storage =  new int [MAX_NUMBER_OF_ELEMENT];
    numberOfElement = -1;
  }

  public void add(int value) {

    if (numberOfElement <MAX_NUMBER_OF_ELEMENT) {
      numberOfElement++;
      storage[numberOfElement] = value;
    }

  }

  public int get() {
    numberOfElement--;
    return storage[numberOfElement+1];
  }

  public boolean contain(int value) {
    int i=0;
    while ((i<=numberOfElement)&& (storage[i]!= value)) {
      i++;
    }

    if (i<=numberOfElement) {
      return true;
    }else {
      return false;
    }
  }

}
```

A példában bemutat kódban az abstrac class nál meg adjuk hogy milyen adat szerkezetet használunk de azt nem hogy ezzel hogyan dolgozunk, míg az interface nél ezt se adtuk meg teljesen független tőle csak azt adtuk meg hogy milyen metódusokat használjon. Az abstrakció szintjétől/igényétől függ hogy melyiket érdemes használni. Ha pedig egyáltalán nics absztrakcióra szükség akkor érdemes Class-t használni. Az interfacek e mellet arra is jók hogy egy azt implikáló osztályt tudjunk használni annak a belső működése ismerete nélkül. Ez elő segíti program kód tagolását is.

11.4. 3.hét

EPAM: Neptun tantárgyfelvétel modellezése UML ben



EPAM: Neptun tantárgyfelvétel UML diagram implementálása

Abstract class NeptunEntity

Azt az osztályt azért hoztam létre mert voltak azonos attribútumaik a Student-nek és a Lessen-nek

```
package targyfelvetel;

public abstract class NeptunEntity {
  protected String neptunCode;
  protected String name;




  public NeptunEntity(String neptunCode, String name) {
    super();
    this.neptunCode = neptunCode;
    this.name = name;
  }
  protected String getNeptunCode() {
    return neptunCode;
```

```
  }
  protected void setNeptunCode(String neptunCode) {
    this.neptunCode = neptunCode;
  }
  protected String getName() {
    return name;
  }
  protected void setName(String name) {
    this.name = name;
  }


}
```

Lessen class felelős az órák ért azoknak az adatainak a tárolásáért és beállításaiért

```
package targyfelvetel;

import java.util.Map;

public class Lessen extends NeptunEntity{

  boolean takeAble;
  int maxNumberStudent;
  int currentNumberOfStudent;
  //TODO linkidmap cserélni map re

  private Lessen(String neptunCode, String name, int maxNumberStudent) {
    super(neptunCode, name);
    this.takeAble = true;
    this.maxNumberStudent = maxNumberStudent;
    this.currentNumberOfStudent = 0;
  }

  static public void getANewLessen(Map<String, Lessen>lessens,String  ↩
     neptunCode, String name, int maxNumberStudent) {

    Lessen curentLessen=new Lessen(neptunCode, name, maxNumberStudent);
    lessens.put(neptunCode, curentLessen);
  }


  public boolean isTakeAble() {
    return takeAble;
  }


  public void setTakeAble(boolean takeAble) {
    this.takeAble = takeAble;
  }
```

```
  public int getMaxNumberStudent() {
    return maxNumberStudent;
  }


  public void setMaxNumberStudent(int maxNumberStudent) {
    this.maxNumberStudent = maxNumberStudent;
  }


  public int getCurrentNumberOfStudent() {
    return currentNumberOfStudent;
  }

  public boolean chekTakeAble() {
    if (currentNumberOfStudent<maxNumberStudent) {
      setTakeAble(true);
      return true;
    }else {
      setTakeAble(false);
      return(false);
    }
  }

  public void incrementCurrentNumberOfStudent() {
    this.currentNumberOfStudent++;
  }


  public void setCurrentNumberOfStudent(int currentNumberOfStudent) {
    this.currentNumberOfStudent = currentNumberOfStudent;
  }



}
```

Student class képes tárgyat felvenni (ezáltal változást is tud okozni a Lessen-ben ez felvethet némi problémát)

```
package targyfelvetel;

import java.util.Map;

public class Student extends NeptunEntity {

  LinkidMap<Integer, String> takenLessen;
  int numberOfTakenLessen;

  private Student(String neptunCode, String name) {
```

```
    super(neptunCode, name);
    this.takenLessen = new LinkidMap<>();
    numberOfTakenLessen = 0;
  }

  public static Student getANewStudent(String neptunCode, String name) {
    Student currentStudent = new Student(neptunCode, name);
    return currentStudent;
  }

  public void takeLesson(Map<String, Lessen> lessens, String lessenCode)  ↩
     throws Exception {

    if (lessens.containsKey(lessenCode)) {
      Lessen currentLessen;
      currentLessen = lessens.get(lessenCode);
      if (currentLessen.chekTakeAble()) {
        setNumberOfTakenLessen(getNumberOfTakenLessen() + 1);
        currentLessen.incrementCurrentNumberOfStudent();
        takenLessen.put(this.numberOfTakenLessen, lessenCode);

      }else {
        throw new Exception("Lessen is not take able "+ lessenCode);
        }

    } else {
      throw new Exception("Lessen dosen' t exists "+ lessenCode);
    }


  }

  public LinkidMap<Integer, String> getTakenLessen() {
    return takenLessen;
  }

  public void setTakenLessen(LinkidMap<Integer, String> takenLessen) {
    this.takenLessen = takenLessen;
  }

  public int getNumberOfTakenLessen() {
    return numberOfTakenLessen;
  }

  public void setNumberOfTakenLessen(int numberOfTakenLessen) {
    this.numberOfTakenLessen = numberOfTakenLessen;
  }

}
```

A neptun class ban vannak tárolva az órák és a hallgatók és itt tudjuk őket létre hozzni

```
package targyfelvetel;

public class Neptun {

    LinkidMap<String, Lessen> lessens = new LinkidMap<>();
    LinkidMap<String, Student> students = new LinkidMap<String, Student>();

    public Neptun() {


    }

    public void AddAStudent(String neptunCode,String name) {
      students.put(neptunCode,Student.getANewStudent( neptunCode, name));
    }

    public void AddALessen(String neptunCode,String name, int  ↩
       maxNumberStudent) {
      Lessen.getANewLessen(lessens, neptunCode, name, maxNumberStudent);
    }

}
```

EPAM:OO modellezés

SOLID

- Single-responsibility principle

  Egy osztálynak csak egyetlen felelőssége kell, hogy legyen, vagyis a szoftver specifikációjának csak egy részén végzett változtatások befolyásolhatják az osztály specifikációját.

- Open–closed principle

  A szoftvereknek nyitva kell lenniük a kiterjesztésre, de módosításra bezárva.

- Liskov substitution principle

  Egy objektumot bármikor helyettesíthet egy all osztálya az adott objektumnak

- Interface segregation principle

  Jobb mindenre külön külön interface csinálni ,mint egy univerzálisat használni

- Dependency inversion principle

  Az absztrakciónak kell az alapját szolgálnia a programoknak nem konkréttan megírt kódok

KISS

A KISS (keep it stupid simple/keep it simple, soldier) elv szerint a legtöbb rendszer akkor működik a legjobban, ha egyszerűbbé teszik, nem pedig bonyolulttá teszik őket; ezért a tervezésnél az egyszerűségnek kell kulcsfontosságúnak lennie, és kerülni kell a felesleges bonyolítást.

DRY

Don't repeat yourself

Minden információnak egy elfordulásának kell hogy legyen a rendszerben (kétszer ugyan azt a a kódot felesleges megírni csak hibaforrás lesz)

YAGIN

You ain't gonna need it

Addig nem kell létre hozni egy függvény míg nincs rá szükséged még akkor se ha azt sejted hogy szükséged lesz rá

11.5. 4.hét

EPAM: Order of everything

```
package com.epam.training;

import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.equalTo;

import java.util.Collection;
import java.util.Collections;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;

/**
 * # Order of everything
 *
 * Collection-ok rendezĂ©se esetĂ©n jellemzĹ‘en futĂ˘si idĹ‘ben derĂL'l ki,
 * ha olyan tĂpusĂş objektumokat prĂłbĂ˘lunk rendezni, amelyeken az ↩
    Ă¶sszehasonlĂtĂ˘s
 * nem Ă©rtelmezett (azaz 'T' tĂpus esetĂ©n nem implementĂ˘ljĂ˘k a ' ↩
    Comparable<T>' interface-t).
 * Pl. ClassCastException a [Collections.sort()](https://docs.oracle.com/ ↩
    javase/7/docs/api/java/util/Collections.html#sort(java.util.List)) eset ↩
    Ă©ben,
 * vagy ClassCastException a [Stream.sorted()](https://docs.oracle.com/ ↩
    javase/8/docs/api/java/util/stream/Stream.html#sorted--) esetĂ©ben.
 *
 * ĂŤrj olyan metĂłdust, amely tetszĹ‘leges Collection esetĂ©n vissza adja ↩
    az elemeket
 * egy List-ben nĂ¶vekvĹ‘en rendezve, amennyiben az elemek Ă¶sszehasonl ↩
    ĂthatĂłak velĂL'k azonos tĂpusĂş objektumokkal.
 * Ha ez a feltĂ©tel nem teljesĂL'l, az eredmĂ©nyezzen syntax error-t.
```

```
 *
 * PÃ©ldÃ¨ul:
 * ```
 *     List<Integer> actualOutput = createOrderedList(input);
 * ```
 * Ahol az `input` `Collection<Integer>` tÃpusÃş. TermÃ©szetesen mÃ¨s t ↩
    Ãpusokkal is mÍ\ensuremath{\pm}kÃ¶dnie kell,
 * feltÃ©ve, hogy implementÃ¨ljÃ¨k a Comparable interface-t.
 */
public class OrderOfEverythingTest {

    @Test(dataProvider = "collectionsToSortDataProvider")
    public void testOrderShouldReturnExpectedListWhenCollectionIsPassed( ↩
       Collection<Integer> input, List<Integer> expectedOutput) {
        // Given as parameters

        // When
        // createOrderedList(List.of(new OrderOfEverythingTest()));
        // ^ ez piros, az OrderOfEverythingTest nem implementÃ¨lja a  ↩
           Comparable<OrderOfEverythingTest> -et
        List<Integer> actualOutput = createOrderedList(input);

        // Then
        assertThat(actualOutput, equalTo(expectedOutput));
    }

    @DataProvider
    private Object[][] collectionsToSortDataProvider() {
        return new Object[][] {
            {Collections.emptySet(), Collections.emptyList()},
            {Set.of(1), List.of(1)},
            {Set.of(2,1), List.of(1,2)}
        };
    }

    private <T extends Comparable<T>> List<T> createOrderedList(Collection< ↩
       T> input) {
        return input.stream()
            .sorted()
            .collect(Collectors.toList());
    }
}
```

EPAM: Bináris keresés és Buborék rendezés implementálása

```
package IntegerStorage;

import java.util.Arrays;

public class IntegerStorage {
  private int[] storage;
```

```
private int index = 0;

public IntegerStorage(int size) {
  super();
  this.storage = new int[size];
}

public IntegerStorage(int[] storage) {
  super();
  this.storage = storage;
  this.index = storage.length;
}

@Override
public int hashCode() {
  final int prime = 31;
  int result = 1;
  result = prime * result + Arrays.hashCode(storage);
  return result;
}

@Override
public boolean equals(Object obj) {
  if (this == obj)
    return true;
  if (obj == null)
    return false;
  if (getClass() != obj.getClass())
    return false;
  IntegerStorage other = (IntegerStorage) obj;
  if (!Arrays.equals(storage, other.storage))
    return false;
  return true;
}

@Override
public String toString() {
  return "IntegerStorage [storage=" + Arrays.toString(storage) + "]";
}

public void add(Integer value) {
  storage[index++] = value;
}

public boolean contains(Integer value) {
  int u = 0;
  int v = index-1;
  boolean l = false;

  sort();
```

```
    while ((l == false) && (u <= v)) {
      int i=(u+v)/2;

      if (storage[i]==value)
        l =true;

      if(storage[i]<value)
        u=i+1;

      if(storage[i]>value)
        v=i-1;

    }

    return l;

  }

  public int[] sort() {

    for (int i = index - 1; i >= 0; i--) {
      for (int j = 0; j < i; j++) {
        if (storage[j] > storage[i]) {
          int s;
          s = storage[j];
          storage[j] = storage[i];
          storage[i] = s;
        }
      }

    }

    return storage;
  }

}
```

EPAM: Saját HashMap implementáció

Egy Hash map kialakításához egty ArrayList-et használtam melybe Map-eket hoztam létre (ez a saját magam által meg írt linkedmap volt)ez álltan láncolt listával oldom meg az ugyan olyan hash kódú elemek tárolását de igazából a HashMap szempontjából/működéséből következően annak nincs jelentősége hogy milyen mappal lesz ez megoldva.

```
package sajatHasMap;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.Map;
```

```
import java.util.Set;
import java.util.Map.Entry;



public class HashMap2<K,V> implements Map<K, V> {

  private final int  PRIME=7;
  ArrayList<LinkidMap<K,V>> rows;
  int size;

  private int hash(Object key) {
    int result;
    result =key.hashCode() % PRIME;

    return result;
  }




  public HashMap2() {
    super();
    this.rows = new ArrayList<LinkidMap<K,V>>(PRIME);
    for(int index=0; index<PRIME; index++) {
      rows.add(new LinkidMap<K,V>());
    }
    this.size = 0;
  }




  @Override
  public int size() {
    // TODO Auto-generated method stub
    size=0;
    for(int index=0; index<PRIME; index++) {
      size+=rows.get(index).size();
    }
    return size;
  }

  @Override
  public boolean isEmpty() {
    // TODO Auto-generated method stub

    for (int index=0; index<PRIME; index++) {
```

```
    if(! rows.get(index).isEmpty())
      return false;
  }

  return true;
}

@Override
public boolean containsKey(Object key) {
  // TODO Auto-generated method stub

  int index= hash(key);
  return rows.get(index).containsKey(key);

}

@Override
public boolean containsValue(Object value) {
  // TODO Auto-generated method stub
  for(int index=0; index<PRIME;index++)
    if (rows.get(index).containsValue(value))
      return true;
  return false;
}

@Override
public V get(Object key) {
  // TODO Auto-generated method stub
  int index= hash(key);
  return rows.get(index).get(key);
}

@Override
public V put(K key, V value) {
  // TODO Auto-generated method stub
  int index= hash(key);
  return rows.get(index).put(key, value);
}

@Override
public V remove(Object key) {
  // TODO Auto-generated method stub
  int index= hash(key);
  return rows.get(index).remove(key);
}

@Override
public void putAll(Map<? extends K, ? extends V> m) {
  // TODO Auto-generated method stub
  Set<? extends K> keys = m.keySet();
```

```
    for ( K key:keys) {
      this.put(key, m.get(key));
    }

  }

  @Override
  public void clear() {
    // TODO Auto-generated method stub
    for (int index=0; index<PRIME; index++)
      rows.get(index).clear();
  }

  @Override
  public Set<K> keySet() {
    // TODO Auto-generated method stub
    Set<K> keys =  new HashSet<K>();
    for (int index=0; index<PRIME; index++)
      keys.addAll(rows.get(index).keySet());
    return keys;
  }

  @Override
  public Collection<V> values() {
    // TODO Auto-generated method stub
    Collection<V> collection =new ArrayList<>();
    for (int index=0; index<PRIME; index++)
      collection.addAll(rows.get(index).values());
    return collection;
  }

  @Override
  public Set<Entry<K, V>> entrySet() {
    // TODO Auto-generated method stub
    Set<Entry<K, V>> result = new HashSet<>();
    for (int index=0; index<PRIME; index++)
      result.addAll(rows.get(index).entrySet());
    return result;
  }

}
```

```
package sajatHasMap;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Map;
import java.util.Set;
import java.util.HashSet;
```

```
public class LinkidMap<K, V> implements Map<K,V> {

  private Node<K, V> root;

  public LinkidMap() {
    root = null;
  }

  private static class Node<K,V> implements Map.Entry<K, V>{

    private K key;
    private V value;
    private Node<K, V> next;

    public Node(K key, V value, Node<K, V> next) {
      super();
      this.key = key;
      this.value = value;
      this.next = next;
    }

     @Override
    public String toString() {
      // TODO Auto-generated method stub

       String string ="Key: "+ getKey() +" "+ "Value: "+ getValue() ;

       return string;
    }

    @Override
    public int hashCode() {
      final int prime = 31;
      int result = 1;
      result = prime * result + ((key == null) ? 0 : key.hashCode());
      result = prime * result + ((value == null) ? 0 : value.hashCode());
      return result;
    }

    @Override
    public boolean equals(Object obj) {
      if (this == obj)
        return true;
      if (obj == null)
        return false;
      if (getClass() != obj.getClass())
        return false;
      Node<K,V> other = (Node<K,V>) obj;
      if (key == null) {
        if (other.key != null)
```

```
        return false;
      } else if (!key.equals(other.key))
        return false;
      if (value == null) {
        if (other.value != null)
          return false;
      } else if (!value.equals(other.value))
        return false;
      return true;
    }

    public K getKey() {
      return key;
    }

    public void setKey(K key) {
      this.key = key;
    }

    public V getValue() {
      return value;
    }

    public V setValue(V value) {
      this.value = value;
      return  value;
    }

    public Node<K, V> getNext() {
      return next;
    }

    public void setNext(Node<K, V> next) {
      this.next = next;
    }

  }

  @Override
  public String toString() {
    Node<K,V> currentNode = root;
    String string="LinkidMap [root]" ;
    while (currentNode != null) {
      string= string +" "+ currentNode.toString();
      currentNode=currentNode.getNext();
    }
    return string;
  }

  @Override
```

```
public int hashCode() {
  final int prime = 31;
  int result = 1;
  Node<K,V> current=root;
  while(current != null) {
    result = result +current.hashCode();
    current=current.next;
  }
  result = prime * result;
  return result;
}

@Override
public boolean equals(Object obj) {
  if (this == obj)
    return true;
  if (obj == null)
    return false;
  if (getClass() != obj.getClass())
    return false;
  LinkidMap<K,V> other = (LinkidMap) obj;
  if (root == null) {
    if (other.root != null)
      return false;
  } else {
    if (other.root == root)
      return true;//Ezz nincs tesztelve elméletileg nem is lehetne ilyen ↩
         eset maximum a node setNext()-jével Null nál true-t add vissza
    if (this.root.key.getClass() != other.root.key.getClass())
      return false;
    if (this.root.value.getClass() != other.root.value.getClass())
      return false;
    Node<K, V> current =root ;
    Node<K, V> otherCurrent =other.root;
    boolean isItEquals = true;
    while ((current.next != null)&&(otherCurrent.next != null) &&  ↩
       isItEquals) {
      isItEquals = current.equals(otherCurrent);
      current = current.next;
      otherCurrent =otherCurrent.next;
    }
    if ((current.next == null) && (otherCurrent.next != null) || (( ↩
       current.next != null) && (otherCurrent.next == null) ) )
      return false;
    if (!isItEquals)
      return false;
  }

  return true;
}
```

```
@Override
public int size() {
  int size = 0;
  Node<K,V> node =root;
  while (node != null) {

    node=node.getNext();
    size++;

  }
  return size;
}

@Override
public boolean isEmpty() {

  if (root == null)
    return true;

  return false;
}

@Override
public boolean containsKey(Object key) {
  // TODO Auto-generated method stub
  Node<K,V> node =root;
  while (node != null) {
    if (node.getKey() == key) {
      return true;
    }else {
      node=node.getNext();
    }
  }

  return false;
}

@Override
public boolean containsValue(Object value) {
  // TODO Auto-generated method stub

  Node<K,V> node =root;
  while (node != null) {
    if (node.getValue() == value) {
      return true;
    }else {
      node=node.getNext();
    }
  }
```

```
  return false;
}

@Override
public V get(Object key) {
  // TODO

  Node<K,V> node =root;
  while (node != null) {
    if (node.getKey() == key) {
      return node.getValue();
    }else {
      node=node.getNext();
    }
  }

  return null;
}

@Override
public V put(K key, V value) {
  // TODO Auto-generated method stub
  if (!containsKey(key)) {
  Node<K, V> node =new Node<>(key,value,root);
  root = node;
  return value;
  }else {
    return null;
  }
}

@Override
public V remove(Object key) {
  // TODO Auto-generated method stub első elem???
  Node<K,V> befor =root;
  Node<K,V> current=root;

  while (current != null) {
    if (current.getKey() == key) {
      befor.setNext(current.getNext());
      return current.getValue();
    } else {
      befor=current;
      current=current.getNext();
    }

  }

  return null;
```

```
}

@Override
public void putAll(Map<? extends K, ? extends V> m) {
  // TODO Auto-generated method stub
  //Sorendet nem tartja

  Set<? extends K> keys = m.keySet();
  for ( K key:keys) {
    this.put(key, m.get(key));
  }

}

@Override
public void clear() {
  // TODO Auto-generated method stub
  root=null;
}

@Override
public Set<K> keySet() {
  // TODO Auto-generated method stub
  Set<K> keys =  new HashSet<K>();
  Node<K,V> currentNode =root;

  while (currentNode != null) {
    keys.add(currentNode.getKey());
    currentNode = currentNode.getNext();
  }

  return keys;
}

@Override
public Collection<V> values() {
  // TODO Auto-generated method stub

  Node<K,V> currentNode =root;
  Collection<V> collection =new ArrayList<>();
  while (currentNode != null) {
    collection.add(currentNode.getValue());
    currentNode=currentNode.getNext();
  }
  return collection;
}

@Override
public Set<Entry<K, V>> entrySet() {
  // TODO Auto-generated method stub
```

```
    Set<Entry<K, V>> result = new HashSet<>();
    Node<K,V> currentNode =root;
    while (currentNode != null) {
      result.add(currentNode);
      currentNode=currentNode.getNext();
    }

        return result;
  }
}
```

11.6. 5. hét

EPAM: It's gone. Or is it?

```
public class BugousStuffProducer {
  private final Writer writer;

  public BugousStuffProducer(String outputFileName) throws IOException {
    writer = new FileWriter(outputFileName);
  }

  public void writeStuff() throws IOException {
    writer.write("Stuff");
  }

  @Override
  public void finalize() throws IOException {
    writer.close();
  }
}
```

Ha csak a writeStuff() eljárást hívjuk meg és a finalize()-t nem hívódik meg a program vége előtt akkor nem fog ki íródni a file-ba a bele írt szöveg. A lapból a garbage collector hívja a finalize()-t de amíg ez nem történik meg nem záródik le a file egyes esetekben még a program le futása végén se hívódik meg.

Erre egy megoldást jelenthet ha implementálja az adott osztály az AutoCloseable interface-t és annak a close metódusában zárja le a filet.

EPAM: Kind of equal

```
// Given
  String first = "...";
  String second = "...";
  String third = new String("...");
  // When
  boolean firstMatchesSecondWithEquals = first.equals(second);
  boolean firstMatchesSecondWithEqualToOperator = first == second;
  boolean firstMatchesThirdWithEquals = first.equals(third);
  boolean firstMatchesThirdWithEqualToOperator = first == third;
```

A JVM optimalizációs célból fent tart egy string pool-t. Ha literálként hozzuk létre a string-et akkor először végig nézi a pool-ban hogy már létezik e az adott string ha igen akkor csak az adott string referenciájával tér vissza ha nem létezik létre hozza és bele rakja a pool-ba.

Ezzel szembe ha a new operátorral végezzük ell a string létrehozását akkor mindig egy új objektum jön létre.

Ebből következik az is hogy a first==second az true-val tér vissza mert ténylegesen ugyan azt az objektumot kapták meg a JVM-től, míg a Third.-nél új objektum jött létre ugyan azzal a tartalommal

EPAM: Java GC

Serial Garbage Collector

Ez a GC legegyszerűbb megvalósítása. Alapvetően egyszálas környezetre tervezték. Ez a GC megvalósítás fagyasztja az összes alkalmazásszálat, amikor fut. Egyetlen szálat használ a garbage collection-höz. Ezért nem jó ötlet több szálon futó alkalmazásokban használni, például szerver környezetben.

Parallel Garbage Collector

Ez a GC a z előzővel ellentétben több szálon fut de ugyan úgy meg állítja az alkalmazást mint az előző. Be alíthatjuk a szálak számát, az rendelkezésére álló időt, maximum throughput-ot és a maximum heap footprintet.

CMS Garbage Collector

Az előzőhöz hasonlóan több szálat használ. Megkeresi é meg jelöli a heap memory-ban a törlendő objektumokat majd ki törli őket. olyan alkalmazásoknál célszerű használni melyek több kisebb meg állást preferálnak egy hosszabb helyet és képesek CPU időt biztositani a GC-nek.

G1 Garbage Collector

Arra tervezték hogy több CPU-s környezetbn nagy memória terület mellet dolgozzon jól. Felosztja egyenlő részekre a memóriát és párhuzamosan gyűjt belőlük. Kettő fázisa van Marking és Sweeping. Mindig annál a memória résznél kezdi a gyűjtést ami a leginkább üres

Epsilon Garbage Collector

Lefoglalja a memóriát a programnak de nem gyűjt szemetet ha tesztelni akarjuk a programot hogy menyi memóriát használ egy jóeszköz vagy ha ki akarjuk sajtolni a legjobb teljesítményt az adott programból.

Z garbage collector

A ZGC minden költséges munkát egyidejűleg végez, anélkül, hogy 10 ms-nál hosszabb időre leállítaná az alkalmazásszálak végrehajtását, ami alkalmassá teszi alacsony késleltetést igénylő alkalmazásokhoz és / vagy nagyon nagy heap memorit használó alkalmazásokhoz. Az Oracle dokumentációja szerint több terabájtos memóriát képes kezelni. A Z szemétgyűjtő a szálaiban végzi ciklusait. Átlagosan 1 ms-ig szünetelteti az alkalmazást. A G1 és a Parallel kollektorok átlagosan nagyjából 200 ms-ot tesznek ki.

Shenandoah Garbage Collector

Shenandoah memóriaterületeket használ annak kezelésére, hogy mely objektumok már nincsenek használatban, melyek élnek és készek a tömörítésre. Shenandoah emellett továbbít mutatót minden kupac objektumhoz, és az objektumhoz való hozzáférés ellenőrzésére használja. Shenandoah tervezése párhuzamos CPU-ciklusokat és helyet biztosít a szünetidő javítására. A továbbító mutató megkönnyíti az objektumok mozgatását, de az agresszív mozgások azt jelentik, hogy Shenandoah több memóriát használ, és több párhuzamos munkát igényel, mint más GC-k. De a rendkívüli munkát a világ rövid megállásaival végzi.

out of memory

Lehetséges ha elkapjuk az errot valahol és tudunk felszabadítani memóriát de nem lehetünk benne biztosak hogy a JVM milyen álapotban van éppen és menyire tudjuk vissza álitani. Jobb eljárás lenne inkább csak meg próbálni lezárni az erőforrásokat esetleg ki loggolni a hibát kiváltó tényzőket

11.7. 6. hét

EPAM: Mátrix szorzás Stream API val

```
public static int[][] matrixszorzas(int[][] a, int[][] b) {

    if (a[0].length == b.length) {
      int bNumberOfCloumns = b[0].length; // oszlopok száma
      int bNumberOfRows = b.length; // szorzások száma

      int[][] c = Arrays.stream(a)// c.length-szer fut le
          .map(r -> IntStream.range(0, bNumberOfCloumns)
              .map(i -> IntStream.range(0, bNumberOfRows).map(j -> r[j] * b ↩
                 [j][i]).sum()).toArray())
          .toArray(int[][]::new);
      return c;
    } else {

      return null;
    }
  }
```

Egy m × t-s A és egy t × nes B mátrix szorzatán azt az AB-vel jelölt m × n-es C mátrixot értjük,

A mátrix i-edik sorának és a B mátrix j-edik oszlopának skaláris szorzata, azaz cij = $a_{i*} \cdot b_{*}j$.

EPAM: Refactoring

```
public class LegacyRefactoring {
  private Calculator creatCalculators() {
    return number -> number * number;
  }

  private Runnable creatRunnable() {
    return () -> System.out.println("Runnable!");
  }

  private  Consumer<Integer> creatConsumer()  {
    return number -> System.out.println(number);

  }

  public void legacy() {

    Runnable runnable = creatRunnable();
```

```
    runnable.run();

    Calculator calculator = creatCalculators();

    Integer result = calculator.calculate(3);
    System.out.println("Calculation result: " + result);

    List<Integer> inputNumbers = Arrays.asList(1, null, 3, null, 5);
    List<Integer> resultNumbers = new ArrayList<>();
    resultNumbers = inputNumbers.stream()
        .filter(x -> x != null)
        .map(x -> calculator.calculate(x))
        .collect(Collectors.toList());

    Consumer<Integer> method =creatConsumer();
    System.out.println("Result numbers: ");
    resultNumbers.forEach(method);

    Formatter formatter = creatFormatter();

    System.out.println("Formatted numbers: " + formatter.format( ↩
       resultNumbers));
  }

  private Formatter creatFormatter() {
    return numbers -> numbers.stream()
        .map(x -> String.valueOf(x))
        .collect(Collectors.joining());
  }
```

EPAM: LinkedList vs ArrayList

LinkidListet

LinkidList-tet akkor érdemes használni ha gyorsan változik a tárolt adatok mennyisége vagy sok törlés beszúrás történik ellenben a gyors hozzáférést nem támogatja végig kel menni az egész láncon a keresendő elemig ezzel szemben az arrayList-nél közvetlen hozzáférést biztosit így gyorsabb az adattok olvasása de ha törlünk akkor vagy "lyukakat" hagyunk a memóriában vagy átlagosan az elemek felét másolnunk kell. Ha pedig bővítjük a listát és ki futunk az előzetesen lefoglalt területből akkor a teljes listát új helyre kell másolni. Akkor érdemes használni ha előre tudjuk a tárolni kívánt adatok mennyiséget (ilyenkor már a konstruktorban meg adható a tárolandó elemek száma) vagy fontosabb az adatok gyors elérése.

```
package linkidlistvsarraylist;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class Main {
```

```
static final int NUMBER_OF_ELEMET=200000;

static void addingElementsToArrayList() {
  List<Integer> list = new ArrayList<>();
  long time = System.currentTimeMillis();
  for (int i = 0; i < NUMBER_OF_ELEMET; i++)
    list.add(i);
  System.out.println("Adding "+NUMBER_OF_ELEMET +" elemet to arrayList: " ↩
      + (System.currentTimeMillis() - time));
}
static void removeElementFromArrayList() {
  List<Integer> list = new ArrayList<>();
  for (int i = 0; i < NUMBER_OF_ELEMET; i++)
    list.add(i);
  long time = System.currentTimeMillis();
  while(list.size()!=0)
    list.remove(list.size()/2);
  System.out.println("Removing "+NUMBER_OF_ELEMET +" elemet from  ↩
     arrayList: " + (System.currentTimeMillis() - time));
}

static void getElementFromArrayList() {
  List<Integer> list = new ArrayList<>();
  for (int i = 0; i < NUMBER_OF_ELEMET; i++)
    list.add(i);
  long time = System.currentTimeMillis();
  for (int i = 0; i < NUMBER_OF_ELEMET; i++)
    list.get(i);
  System.out.println("Geting "+NUMBER_OF_ELEMET +" elemet from arrayList: ↩
      " + (System.currentTimeMillis() - time));
}

static void addingElementsToLinkedList() {
  long time = System.currentTimeMillis();
  List<Integer> list = new LinkedList<>();
  for (int i = 0; i < NUMBER_OF_ELEMET; i++)
    list.add(i);
  System.out.println("Adding "+NUMBER_OF_ELEMET +" elemet to LinkedList:  ↩
     " + (System.currentTimeMillis() - time));
}

static void removeElementFromLinkedList() {
  List<Integer> list = new LinkedList<>();
  for (int i = 0; i < NUMBER_OF_ELEMET; i++)
    list.add(i);
  long time = System.currentTimeMillis();
  while(list.size()!=0)
    list.remove(list.size()/2);
  System.out.println("Removing "+NUMBER_OF_ELEMET +" elemet from  ↩
     LinkedList: " + (System.currentTimeMillis() - time));
```

```
  }

  static void getElementFromLinkedList() {
    List<Integer> list = new LinkedList<>();
    for (int i = 0; i < NUMBER_OF_ELEMET; i++)
      list.add(i);
    long time = System.currentTimeMillis();
    for (int i = 0; i < NUMBER_OF_ELEMET; i++)
      list.get(i);
    System.out.println("geting "+NUMBER_OF_ELEMET +" elemet from LinkedList ↩
       : " + (System.currentTimeMillis() - time));
  }

  public static void main(String[] args) {
    // TODO Auto-generated method stub

    addingElementsToArrayList();
    addingElementsToLinkedList();
    removeElementFromArrayList();
    removeElementFromLinkedList();
    getElementFromArrayList();
    getElementFromLinkedList();
  }

}
```

kimenet:

```
Adding 200000 elemet to arrayList: 23
Adding 200000 elemet to LinkedList: 26
Removing 200000 elemet from arrayList: 1219
Removing 200000 elemet from LinkedList: 31024
Geting 200000 elemet from arrayList: 3
geting 200000 elemet from LinkedList: 28911
```

11.8. 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

11.9. Általános

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

11.10. C

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

11.11. C++

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

11.12. 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.