Univerzális programozás

Írd meg a saját programozás tankönyvedet!

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

COLLABORATORS			
	TITLE : Univerzális programozás		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Bátfai, Norbert Ã©s Orosz , Máté	2019. november 11.	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Ajánlás

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

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

Tartalomjegyzék

Ábrák jegyzéke

Előszó

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

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

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

Hogyan forgasd

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

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

Milyen nyelven nyomjuk?

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

Hogyan nyomjuk?

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

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

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

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

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

I. rész

Bevezetés

1. fejezet

Vízió

1.1. Mi a programozás?

1.2. Milyen doksikat olvassak el?

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

1.3. Milyen filmeket nézzek meg?

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

II. rész

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.

2. fejezet

Helló, Arroway!

2.1. OO szemlélet

A módosított polártranszformációs normális generátor beprogramozása Java nyelven. Mutassunk rá, hogy a mi természetes saját megoldásunk (az algoritmus egyszerre két normálist állít elő, kell egy példánytag, amely a nem visszaadottat tárolja és egy logikai tag, hogy van-e tárolt vagy futtatni kell az algot.) és az OpenJDK, Oracle JDK-ban a Sun által adott OO szervezés ua.! https://arato.inf.unideb.hu/batfai.norbert/UDPROG (16-22 fólia) Ugyanezt írjuk meg C++ nyelven is! (lásd még UDPROG repó: source/labor/polargen)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

```
import java.util.Random;
import java.io.*;
import java.lang.Math;
public class PolarGen {
  public final  static int RAND_MAX = 32767;
  private static boolean bExists;
  private double dValue;
  static Random cRandomGenerator = new Random();
  public PolarGen() {
  bExists = false;
  cRandomGenerator.setSeed(20);
  };
  public double PolarGet()  {
      if (!bExists)
  {
    double u1, u2, v1, v2, w;

    do{
      u1 = cRandomGenerator.nextInt (RAND_MAX) / (RAND_MAX + 1.0);
      u2 = cRandomGenerator.nextInt (RAND_MAX) / (RAND_MAX + 1.0);
      v1 = 2 * u1 - 1;
      v2 = 2 * u2 - 1;
```

```
      w = v1 * v1 + v2 * v2;
        }

    while (w > 1);
    double r = Math.sqrt ((-2 * Math.log (w)) / w);
    dValue = r * v2;
    bExists = !bExists;
    return r * v1;
  }
  else
  {
    bExists = !bExists;
    return dValue;
  }
  };
  public static void main(String args[])  {
    PolarGen cPolarGen = new PolarGen();
    double  dEredmeny = cPolarGen.PolarGet();
    System.out.println(dEredmeny);
  }
  </para>
    }
```

A program ellenőrzi, hogy van e tárolt érték, amennyiben nincs, létrehozza azt a következőképpen: 2 darab -1 és 1 közötti számot hoz létre, majd egyet elvesz, és aztán veszi a négyzetösszegüket. Ez addig ismétlődik, amíg s nagyobb vagy egyenlő mint 1, vagy s egyenlő 0. Ezután felveszzük a multiplier-t, ami az "s" -2-szeres logaritmus hányadosának négyzetgyöke. A tárolt érték multiplier szorozva v2-vel, utána pedig v1 szer multiplier értékét.

C++ -ban:

```
#include<cmath>
#include <cstdlib>
#include<ctime>
using namespace std;
class PolarGen {
public:
    PolarGen (){
        nincsTarolt = true;
        srand (time (NULL));
    }
    ~PolarGen(){
    }
    double kovetkezo ();
private:
    bool nincsTarolt;
    double tarolt;
};
double PolarGen::kovetkezo() {
    if (nincsTarolt) {
        double u1, u2, v1, v2, w;
```

```
        do {
            u1 = rand() / (RAND_MAX + 1.0);
            u2 = rand() / (RAND_MAX + 1.0);
            v1 = 2 * u1 - 1;
            v2 = 2 * u2 -1;
            w = v1 * v1 + v2 * v2;
        } while ( w > 1);
        double r = sqrt ((-2* log(w) / w));
        tarolt = r * v2;
        nincsTarolt = !nincsTarolt;
        return r * v1;
    }
     else {
        nincsTarolt = !nincsTarolt;
       return tarolt;
     }
}
int main(int argc, char** argv) {
    PolarGen pg;
    for (int i = 0; i < 10; i++)
        std::cout<<pg.kovetkezo()<<std::endl;
    return 0;
}
        }
```

2.2. Homokozó

rjuk át az első védési programot (LZW binfa) C++ nyelvről Java nyelvre, ugyanúgy működjön! Mutassunk rá, hogy gyakorlatilag a pointereket és referenciákat kell kiirtani és minden máris működik (erre utal a feladat neve, hogy Java-ban minden referencia, nincs választás, hogy mondjuk egy attribútum pointer, referencia vagy tagként tartalmazott legyen). Miután már áttettük Java nyelvre, tegyük be egy Java Servletbe és a böngészőből GET-es kéréssel (például a böngésző címsorából) kapja meg azt a mintát, amelynek kiszámolja az LZW binfáját!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Itt annyi dolgunk van, hogy a C++ szintaktikát használó pointereket és referenciákat át kell írni Java szintaktikának megfelelőekre.

```
            public class LZWBinFa {

  public LZWBinFa() {

    fa = gyoker;

  }
```

```
public void egyBitFeldolg(char b) {

  if (b == '0') {

    if (fa.nullasGyermek() == null)
    {

      Csomopont uj = new Csomopont('0');

      fa.ujNullasGyermek(uj);

      fa = gyoker;
    } else
    {

      fa = fa.nullasGyermek();
    }
  }
else {
    if (fa.egyesGyermek() == null) {
      Csomopont uj = new Csomopont('1');
      fa.ujEgyesGyermek(uj);
      fa = gyoker;
    } else {
      fa = fa.egyesGyermek();
    }
  }
}

public void kiir() {

  melyseg = 0;

  kiir(gyoker, new java.io.PrintWriter(System.out));

}

public void kiir(java.io.PrintWriter os) {
  melyseg = 0;
  kiir(gyoker, os);
}


class Csomopont {


  public Csomopont(char betu) {
    this.betu = betu;
```

```
      balNulla = null;
      jobbEgy = null;
    }

    ;



    public Csomopont nullasGyermek() {
      return balNulla;
    }


    public Csomopont egyesGyermek() {
      return jobbEgy;
    }


    public void ujNullasGyermek(Csomopont gy) {
      balNulla = gy;
    }


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


    public char getBetu() {
      return betu;
    }

    private char betu;

    private Csomopont balNulla = null;
    private Csomopont jobbEgy = null;

  };


  private Csomopont fa = null;

  private int melyseg, atlagosszeg, atlagdb;
  private double szorasosszeg;

  public void kiir(Csomopont elem, java.io.PrintWriter os) {

    if (elem != null) {
      ++melyseg;
      kiir(elem.egyesGyermek(), os);
```

```
    for (int i = 0; i < melyseg; ++i) {
      os.print("---");
    }
    os.print(elem.getBetu());
    os.print("(");
    os.print(melyseg - 1);
    os.println(")");
    kiir(elem.nullasGyermek(), os);
    --melyseg;
  }
}

protected Csomopont gyoker = new Csomopont('/');
int maxMelyseg;
double atlag, szoras;


public int getMelyseg() {
  melyseg = maxMelyseg = 0;
  rmelyseg(gyoker);
  return maxMelyseg - 1;
}

public double getAtlag() {
  melyseg = atlagosszeg = atlagdb = 0;
  ratlag(gyoker);
  atlag = ((double) atlagosszeg) / atlagdb;
  return atlag;
}

public double getSzoras() {
  atlag = getAtlag();
  szorasosszeg = 0.0;
  melyseg = atlagdb = 0;

  rszoras(gyoker);

  if (atlagdb - 1 > 0) {
    szoras = Math.sqrt(szorasosszeg / (atlagdb - 1));
  } else {
    szoras = Math.sqrt(szorasosszeg);
  }

  return szoras;
}

public void rmelyseg(Csomopont elem) {
  if (elem != null) {
    ++melyseg;
```

```
      if (melyseg > maxMelyseg) {
        maxMelyseg = melyseg;
      }
      rmelyseg(elem.egyesGyermek());

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

  public void ratlag(Csomopont elem) {
    if (elem != null) {
      ++melyseg;
      ratlag(elem.egyesGyermek());
      ratlag(elem.nullasGyermek());
      --melyseg;
      if (elem.egyesGyermek() == null && elem.nullasGyermek() == null) {
        ++atlagdb;
        atlagosszeg += melyseg;
      }
    }
  }

  public void rszoras(Csomopont elem) {
    if (elem != null) {
      ++melyseg;
      rszoras(elem.egyesGyermek());
      rszoras(elem.nullasGyermek());
      --melyseg;
      if (elem.egyesGyermek() == null && elem.nullasGyermek() == null) {
        ++atlagdb;
        szorasosszeg += ((melyseg - atlag) * (melyseg - atlag));
      }
    }
  }


  public static void usage() {
    System.out.println("Usage: lzwtree in_file -o out_file");
  }


  public static void main(String args[]) {

    if (args.length != 3) {

      usage();

      System.exit(-1);
    }
```

```
String inFile = args[0];

if (!"-o".equals(args[1])) {
  usage();
  System.exit(-1);
}


try {

  java.io.FileInputStream beFile =
          new java.io.FileInputStream(new java.io.File(args[0]));

  java.io.PrintWriter kiFile =
          new java.io.PrintWriter(
          new java.io.BufferedWriter(
          new java.io.FileWriter(args[2])));


  byte[] b = new byte[1];

  LZWBinFa binFa = new LZWBinFa();


  while (beFile.read(b) != -1) {
    if (b[0] == 0x0a) {
      break;
    }
  }

  boolean kommentben = false;


  while (beFile.read(b) != -1) {

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

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

    if (kommentben) {
      continue;
    }
```

```
        if (b[0] == 0x4e) // N betű
        {
          continue;
        }


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

          if ((b[0] & 0x80) != 0)
          {
            binFa.egyBitFeldolg('1');
          } else
          {
            binFa.egyBitFeldolg('0');
          }
          b[0] <<= 1;
        }

      }


      binFa.kiir(kiFile);


      kiFile.println("depth = " + binFa.getMelyseg());
      kiFile.println("mean = " + binFa.getAtlag());
      kiFile.println("var = " + binFa.getSzoras());

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

    } catch (java.io.FileNotFoundException fnfException) {
      fnfException.printStackTrace();
    } catch (java.io.IOException ioException) {
      ioException.printStackTrace();
    }

  }
}
```

2.3. „Gagyi”

Az ismert formális „ while (x <= t && x >= t && t != x); ” tesztkérdéstípusra adj a szokásosnál (miszerint x, t az egyik esetben az objektum által hordozott érték, a másikban meg az objektum referenciája) „mélyebb” választ, írj Java példaprogramot mely egyszer végtelen ciklus, más x, t értékekkel meg nem! A példát építsd a JDK Integer.java forrására3 , hogy a 128-nál inkluzív objektum példányokat poolozza!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

2.4. Yoda

Írjunk olyan Java programot, ami java.lang.NullPointerEx-el leáll, ha nem követjük a Yoda conditions-t! https://en.wikipedia.org/wiki/Yoda_conditions

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

```
class yoda {

public static void main(String args[]){
String yoda = null;
if ( yoda.equals("yoda")) { System.out.println("Nem jó");

}
}

}
```

2.5. Kódolás from scratch

Induljunk ki ebből a tudományos közleményből: http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/bbpalg.pdf és csak ezt tanulmányozva írjuk meg Java nyelven a BBP algoritmus megvalósítását! Ha megakadsz, de csak végső esetben: https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitokjavat/apbs02.html#pi_jegyei (mert ha csak lemásolod, akkor pont az a fejlesztői élmény marad ki, melyet szeretném, ha átélnél).

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3. fejezet

Helló, Liskov!

3.1. Liskov helyettesítés sértése

Írjunk olyan OO, leforduló Java és C++ kódcsipetet, amely megsérti a Liskov elvet! Mutassunk rá a megoldásra: jobb OO tervezés. https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf (93-99 fólia) (számos példa szerepel az elv megsértésére az UDPROG repóban, lásd pl. source/binom/Batfai-Barki/madarak/)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A Liskov elv azt mondja ki, hogy: Ha S osztály T osztály leszármazottja, akkor S szabadon behelyettesíthető minden olyan helyre (paraméter, változó, stb. . .), ahol T típust várunk. Példa sértésre:

public class Bird{ public void fly(){} } public class Duck extends Bird{}

Ebben az esetben még minden jól működik, hiszen a kacsa egy madár, azonban

```
public class Ostrich extends Bird{}
```

esetén a strucc már nem tudja használni a "fly" függvényt, annak ellenére hogy ő is madár.

Javított példa: public class Bird{ } public class FlyingBirds extends Bird{ public void fly(){} } public class Duck extends FlyingBirds{} public class Ostrich extends Bird{}

3.2. Szülő-gyerek

írjunk Szülő-gyerek Java és C++ osztálydefiníciót, amelyben demonstrálni tudjuk, hogy az ősön keresztül csak az ős üzenetei küldhetőek! https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf (98. fólia)4

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Itt két (egy Java és és egy C++) programot kellett elkészíteni, melyeken keresztül bemutatható, hogy, mindkét programban van egy "Parent", és egy "Child" osztály . Mindkét osztályon belül van egy-egy definiált metódus. A child, a parent leszármaozattja, így a parent tulajdonságai öröklődnek, emiatt a child osztály segítségével el tudjuk érni a parent-ben definiált tulajdonságokat, fordítva viszont nem. A példák ezt mutatják be Java példa:

```
class Szulo
{
  public static void szulo_uzen()
  {
    System.out.println("Ez a szülő üzenete");
  }
}

class Gyerek extends Szulo
{
  public static void gyerek_uzen()
  {
    System.out.println("Ez a gyerek üzenete");
  }
}

public class szulo_gyerek
{
  public static void main(String[] args)
  {
    Szulo p = new Gyerek();
    p.gyerek_uzen();
  }
}
```

Természetesen a Java-s példa hibaüzenettel fut. C++ példa:

```
#include <iostream>
using namespace std;

class Szulo {
public:
  void szulo_uzen()
  {
    cout << "Ez a szulő üzenete\n";
  }
};

class Gyerek : public Szulo {
  void gyerek_uzen()
  {
    cout << "Ez a gyerek üzenete\n";
  }
```

```
};

int main()
{
  Szulo * szulo = new Gyerek();
  szulo -> gyerek_uzen();
}
```

Itt egyszerűen egy olyan osztálydefiníciót kell írni, amely demonstrálja, hogy egy szülőn keresztül, csak a saját üzenete küldhető, egy hozzá tartozó gyereké már nem.

3.3. Anti OO

A BBP algoritmussal a Pi hexadecimális kifejtésének a 0. pozíciótól számított 10^6, 10^7, 10^8 darab jegyét határozzuk meg C, C++, Java és C# nyelveken és vessük össze a futási időket! https://www.tankonyvtar.hu/hu/tarta tanitok-javat/apas03.html#id561066

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.4. deprecated - Hello, Android!

Élesszük fel a https://github.com/nbatfai/SamuEntropy/tree/master/cs projektjeit és vessünk össze néhány egymásra következőt, hogy hogyan változtak a források!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.5. Hello, Android!

Élesszük fel az SMNIST for Humans projektet! https://gitlab.com/nbatfai/smnist/tree/master/forHumans/SMNIST Apró módosításokat eszközölj benne, pl. színvilág.

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.6. Hello, SMNIST for Humans!

Fejleszd tovább az SMNIST for Humans projektet SMNIST for Anyone emberre szánt appá! Lásd az smnist2_kutatasi_jegyzokonyv.pdf-ben a részletesebb hátteret!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.7. Ciklomatikus komplexitás

Számoljuk ki valamelyik programunk függvényeinek ciklomatikus komplexitását! Lásd a fogalom tekintetében a https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_2.pdf (77-79 fóliát)!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A Ciklomatikus komplexitás Thomas J. McCabe gráfelméletre alapuló számítása, amellyel a következőképpen fejezi ki számokkal egy forráskód komplexitását: M = E - N + 2P, ahol:

E: A gráf éleinek száma N: A gráfban lévő csúcsok száma P: Az összefüggő komponensek száma

A terminálban a "sudo apt-get install complexity" parancsal tudunk a komplexitás kiszámolására megfelelő programot letölteni, melyet utána "complexity --histogram --score --thresh=0 'file.c' " paranccsal tudunk használni. példa(torperdó játék kódját használva):

3.1. ábra. madarszulo

4. fejezet

Helló, Mandelbrot!

4.1. Reverse engineering UML osztálydiagram

UML osztálydiagram rajzolása az első védési C++ programhoz. Az osztálydiagramot a forrásokból generáljuk (pl. Argo UML, Umbrello, Eclipse UML) Mutassunk rá a kompozíció és aggregáció kapcsolatára a forráskódban és a diagramon, lásd még: https://youtu.be/Td_nlERlEOs. https://arato.inf.unideb.hu/batfai.norbert/UD (28-32 fólia)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A Visual Paradigm nevű program "Instant Reverse" funkcióját használva a C++ forráskódot UML osztálydiagrammá alakítottam:.

Az osztálydiagram az osztályok kapcsolatrendszerének összefoglalása. Az asszociációk két osztály, vagy két objektum közötti viszonyt fejeznek ki. Ebben az UML diagramban az aggregációt üres rombusz jelenti (gyenge tartalmazás), ebben az esetben, a rész az egészhez tartozik, de önmagában is létező entitás. A kompozíciót pedig a csúcsára állított teli rombusz jelöli (erős tartalmazás). Ebben az esetben a rész önmagában nem létezhet, csak az egész elemként.

4.2. Forward engineering UML osztálydiagram

UML-ben tervezzünk osztályokat és generáljunk belőle forrást!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Továbbra is a visual paradigm segítségével, az alábbi kódokat kaptam:

```
#include <exception>
using namespace std;
#include "LZWBinFa.h"
#include "Csomopont.h"
LZWBinFa::LZWBinFa() {
}
void LZWBinFa::_LZWBinFa() {
  throw "Not yet implemented";
}
void LZWBinFa::_<(char aB) {
  throw "Not yet implemented";
}
void LZWBinFa::kiir() {
  throw "Not yet implemented";
}
int LZWBinFa::getMelyseg() {
  return this->_melyseg;
}
double LZWBinFa::getAtlag() {
  return this->_atlag;
}
double LZWBinFa::getSzoras() {
  return this->_szoras;
}
void LZWBinFa::kiir(std::ostream& aOs) {
  throw "Not yet implemented";
}
LZWBinFa::LZWBinFa(const LZWBinFa& aUnnamed_1) {
}
LZWBinFa& LZWBinFa::_(const LZWBinFa& aUnnamed_1) {
  throw "Not yet implemented";
}
void LZWBinFa::kiir(Csomopont* aElem, std::ostream& aOs) {
  throw "Not yet implemented";
}
void LZWBinFa::szabadit(Csomopont* aElem) {
  throw "Not yet implemented";
}
void LZWBinFa::rmelyseg(Csomopont* aElem) {
  throw "Not yet implemented";
}
void LZWBinFa::ratlag(Csomopont* aElem) {
  throw "Not yet implemented";
}
void LZWBinFa::rszoras(Csomopont* aElem) {
  throw "Not yet implemented";
}
f
```

A header fileok szinte teljesen megegyeznek az eredetivel.

4.3. Egy esettan

A BME-s C++ tankönyv 14. fejezetét (427-444 elmélet, 445-469 az esettan) dolgozzuk fel!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4.4. BPMN

Rajzoljunk le egy tevékenységet BPMN-ben! https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog
(34-47 fólia)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Online elkészített ábra pizza rendelésről:

4.5. BPEL Helló, Világ! - egy visszhang folyamat

Egy visszhang folyamat megvalósítása az alábbi teljes „videó tutoriál” alapján: https://youtu.be/0OnlYWX2v_I

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4.6. TeX UML

Valamilyen TeX-es csomag felhasználásával készíts szép diagramokat az OOCWC projektről (pl. use case és class diagramokat).

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5. fejezet

Helló, Chomsky!

5.1. Encoding

Fordítsuk le és futtassuk a Javat tanítok könyv MandelbrotHalmazNagyító.java forrását úgy, hogy a fájl nevekben és a forrásokban is meghagyjuk az ékezetes betűket! https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/adatok.html

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A Mandelbort Halmaz nagyító programját kellett fordítani és futtatni úgy, hogy mind a kódban és a fájlnévben meghagyjuk az ékezetes karaktereket, ha azonban ezt módosítások nélkül próbáljuk meg, akkor hibaüzenetetet fogunk kapni, mivel az UTF8-as kódolás számára ismeretlen karaktereket tartalmaz a kód. Ezt a problémát a -encoding kapcsoló használatával tudjuk kiküszöbölni, mellyel a kódolást ISO/IEC 8859-2 (Latin-2)-re állítva a java.lang api-ban a következő eredményt kapjuk:

5.2. OOCWC lexer

Izzítsuk be az OOCWC-t és vázoljuk a https://github.com/nbatfai/robocaremulator/blob/master/justine/rcemu/src/ lexert és kapcsolását a programunk OO struktúrájába!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5.3. l334d1c45

Írj olyan OO Java vagy C++ osztályt, amely leet cipherként működik, azaz megvalósítja ezt a betű helyettesítést: https://simple.wikipedia.org/wiki/Leet (Ha ez első részben nem tetted meg, akkor írasd ki és magyarázd meg a használt struktúratömb memóriafoglalását!)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A leet osztályban található egy input string, ami tartalmazza a bekért szöveget, illetve egy output string vektort, ami a konvertált szöveget. A konvertálás a cipher függvényben zajlik. A bekért stringet nagybetűvé konvertáljuk, hogy ne kelljen különbséget tenni a kisbetűk és nagybetűk között. Ezután végimegyünk az input string minden karakterén és a karakternek mefelelő leet karaktert helyezünk az output vektorba. A main()-ben deklaráljuk a bekérés megszakításához szükséges stringet és boolt, ami minden fordtás után ellenőrzi, hogy szeretnék e még fordítani. A konvertálás, pedig a konstruktorral történik. meghívásával hajtódik végre.

```
          import java.util.Scanner;
public class Leet {
    private static StringBuilder transform(String input, StringBuilder sb)
    {
        input=input.toUpperCase();
        for(int i=0;i<input.length();i++)
        {
            switch (input.charAt(i)) {
                case 'A' : sb.append("4");
                    break;
                case 'B' : sb.append("|3");
                    break;
                case 'C' : sb.append("(");
                    break;
                case 'D' : sb.append("|)");
                    break;
                case 'E' : sb.append("3");
                    break;
                case 'F' : sb.append("|=");
                    break;
                case 'G' : sb.append("6");
                    break;
                case 'H' : sb.append("|-|");
                    break;
                case 'I' : sb.append("|");
                    break;
                case 'J' : sb.append(".]");
                    break;
                case 'K' : sb.append("|<");
                    break;
                case 'L' : sb.append("1");
                    break;
                case 'M' : sb.append("|Y|");
                    break;
                case 'N' : sb.append("/V");
                    break;
```

```
                case 'O' : sb.append("Ø");
                    break;
                case 'P' : sb.append("|O");
                    break;
                case 'Q' : sb.append("(,)");
                    break;
                case 'R' : sb.append("®");
                    break;
                case 'S' : sb.append("Ş");
                    break;
                case 'T' : sb.append("7");
                    break;
                case 'U' : sb.append("|_|");
                    break;
                case 'V' : sb.append("V");
                    break;
                case 'W' : sb.append("\\^/");
                    break;
                case 'X' : sb.append(" }{");
                    break;
                case 'Y' : sb.append("$\yen$");
                    break;
                case 'Z' : sb.append("2");
                    break;
                case ' ' : sb.append(" ");
                    break;
                    default: sb.append(input.charAt(i));
            }
        }
        return sb;
    }
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        Scanner scan = new Scanner(System.in);
        String input;
        System.out.println("Please enter the text that you want to l33tify" ↩
           );
        input=scan.nextLine();
        System.out.println("The 1337 version of your input is: \n");
        sb=transform(input,sb);
        System.out.print(sb);
        System.out.println("\n");
        }
    }
```

5.4. Full screen

Készítsünk egy teljes képernyős Java programot! Tipp: https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus_jatek

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A felhasznált labirintus program forrása: https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/javat-tanitok-javat.zip Használt displayek lekérdezése a getLocalGraphicsEnviroment() és a getDefaulScreenDevice() metódusokkal.

```
public class LabirintusJáték extends java.awt.Frame
        implements Runnable {
.
.
.
java.awt.GraphicsDevice graphicsDevice;
.
.
java.awt.GraphicsEnvironment graphicsEnvironment = java.awt. ↩
   GraphicsEnvironment.getLocalGraphicsEnvironment();
.
.
graphicsDevice = graphicsEnvironment.getDefaultScreenDevice();
.
.
teljesKépernyősMód(graphicsDevice);
```

A teljesKépernyősMód-hoz tartozó kód:

```
    public void teljesKépernyősMód(java.awt.GraphicsDevice graphicsDevice)  ↩
       {

        int szélesség = 0;
        int magasság = 0;
        setUndecorated(true);
        setIgnoreRepaint(true);
        setResizable(false);
        boolean fullScreenTamogatott = graphicsDevice.isFullScreenSupported ↩
           ();
        if(fullScreenTamogatott) {
            graphicsDevice.setFullScreenWindow(this);
            java.awt.DisplayMode displayMode
                    = graphicsDevice.getDisplayMode();
            szélesség = displayMode.getWidth();
            magasság = displayMode.getHeight();
            int színMélység = displayMode.getBitDepth();
            int frissítésiFrekvencia = displayMode.getRefreshRate();
```

```
            System.out.println(szélesség
                    + "x"  + magasság
                    + ", " + színMélység
                    + ", " + frissítésiFrekvencia);
            java.awt.DisplayMode[] displayModes
                    = graphicsDevice.getDisplayModes();
            boolean dm1024x768 = false;
            for(int i=0; i<displayModes.length; ++i) {
                if(displayModes[i].getWidth() == 1920
                        && displayModes[i].getHeight() == 1080
                        && displayModes[i].getBitDepth() == színMélység
                        && displayModes[i].getRefreshRate()
                        == frissítésiFrekvencia) {
                    graphicsDevice.setDisplayMode(displayModes[i]);
                    dm1024x768 = true;
                    break;
                }

            }

            if(!dm1024x768)
                System.out.println("Nem megy az 1024x768.");
```

Ez a kód felelős a fullscreen mód beállításáért. Az ablak kereteit egyszerűen kivesszük, és kikapcsoljuk az átméretezhetőséget. Ezután lekérdezzük, hogy a fullscreen mód támogatott-e, ha igen, átadjuk a képernyű tulajdonságait a setFullScreenWindow() metódusnak. A támogatott felbontásokat egy tömbbe helyezzük, majd összevetve a jelenlegi felbontással, kiderül hogy támogatott e (amennyiben nem, hibaüzenetet kapunk)

5.5. Paszigráfia Rapszódia OpenGL full screen vizualizáció

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5.6. Paszigráfia Rapszódia OpenGL full screen vizualizáció

Lásd vis_prel_para.pdf! Apró módosításokat eszközölj benne, pl. színvilág, textúrázás, a szintek jobb elkülönítése, kézreállóbb irányítás.

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5.7. Paszigráfia Rapszódia LuaLaTeX vizualizáció

Lásd vis_prel_para.pdf! Apró módosításokat eszközölj benne, pl. színvilág, még erősebb 3D-s hatás.

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5.8. Perceptron osztály

Dolgozzuk be egy külön projektbe a projekt Perceptron osztályát! Lásd https://youtu.be/XpBnR31BRJY

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

6. fejezet

Helló, Stroustrup!

6.1. JDK osztályok

Írjunk olyan Boost C++ programot (indulj ki például a fénykardból) amely kilistázza a JDK összes osztályát (miután kicsomagoltuk az src.zip állományt, arra ráengedve)!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A Boost C++ program segítségével kell a JDK összes osztályát az src.zip-ből kilistázni. Ehhez a fénykard programot hívtam segítségül annyi különbséggel, hogy itt .java fájlokat fogunk megkeresni, kiíratni és megszmálni.

```
#include <iostream>
#include <vector>
#include <string>
#include <stdio.h>
#include <boost/filesystem.hpp>
#include <boost/filesystem/fstream.hpp>
#define GetCurrentDir getcwd
using namespace std;
vector<string> searchRootFolders (vector<string> folders);
void readClasses (string path, vector<string>& classes);
string GetCurrentWorkingDir( void ) {
      char buff[FILENAME_MAX];
      GetCurrentDir( buff, FILENAME_MAX );
      string current_working_dir(buff);
      return current_working_dir;
}
int main(int argc, char const *argv[])
{
      vector<string> roots = {
        GetCurrentWorkingDir()+"/"+"src"
```

```
        };
        vector<string> classes = searchRootFolders ( roots );
        for(auto &i : classes)
        {
        cout << i << endl;
        }
        cout << "Összesen " << classes.size() << " osztály van a src-zip-ben\ ↩
           n";
        return 0;
}
void readClasses ( boost::filesystem::path path, vector<string>& classes)
{
        if ( is_regular_file ( path ) ) {
          std::string ext ( ".java" );
            if ( !ext.compare ( boost::filesystem::extension ( path ) ) ){
              classes.push_back(path.string());
        }
        } else if ( is_directory ( path ) )
          for ( boost::filesystem::directory_entry & entry : boost:: ↩
             filesystem::
          directory_iterator ( path ) )
        readClasses ( entry.path(), classes );
        }
        vector<string> searchRootFolders (vector<string> folders)
        {
        vector <string> classes;
        for ( const auto & path : folders)
        {
        boost::filesystem::path root ( path );
        readClasses ( root, classes);
        }
        return classes;
}
```

A futtatáshoz "sudo apt-get install libboost-all-dev" -el kell letöltenünk a megfelelő fileokat, aztán fordításnál "-lboost_system -lboost_filesystem -lboost_file_options -std=c++14" parancsot használunk A program futtatása után megkapjuk hogy mennyi java osztály van a jdk-ben amit használunk.

6.2. Másoló-mozgató szemantika

Kódcsipeteken (copy és move ctor és assign) keresztül vesd össze a C++11 másoló és a mozgató szemantikáját, a mozgató konstruktort alapozd a mozgató értékadásra!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A Prog 1-es LZW binfa-t használom a feladathoz.

A Binfa másoló konstruktora:

```
        LZWBinFa ( const LZWBinFa & regi ) {
            std::cout << "LZWBinFa copy ctor" << std::endl;
            gyoker = masol(regi.gyoker, regi.fa);
        }
```

Ezzel, a régi fa mintájára egy újat készítünk másolással, viszont nem default érték szerinti paraméter átadással, ugyanis abban az esetben a fa nem változna. Ahhoz hogy egy másik fát kapjunk saját konstruktorra van szükség:

```
        Csomopont *  masol ( Csomopont * elem, Csomopont * regifa ) {
        Csomopont * ujelem = NULL;
        if ( elem != NULL ) {
             ujelem = new Csomopont ( elem->getBetu() );
             ujelem->ujEgyesGyermek ( masol ( elem->egyesGyermek (),  ↩
                regifa ) );
             ujelem->ujNullasGyermek ( masol ( elem->nullasGyermek (),  ↩
                regifa ) );
             if ( regifa == elem )
                  fa = ujelem;
        }
        return ujelem;
    }
```

Paraméterként megadva a régi fa gyökerét és a fát, csomópontonként rekurzívan átmásoljuk a gyoker-be, viszont minden node-nál ujelem néven új Csomopont-ot hozunk létre melynek így már memóracíme eltérő lesz, ezáltal a fánk is máshogy néz ki.

A mozgató konstruktornak ezzel szemben az a lényege, hogy magát a régi fát mozgatjuk át egy másik memóriacímre, a régi címet üresen hagyva. Ehhez ezt a konstruktort használjuk:

```
        LZWBinFa (LZWBinFa && regi) {
            std::cout << "LZWBinFa move ctor" << std::endl;
            gyoker = nullptr;
            *this = std::move(regi);
        }
```

A gyökeret nullázzuk, ezután a move függvénynek átadjuk paraméterként a régi fát, azaz azt amit mozgatni szeretnénk és átmozgatjuk a regi-t az újba, a régit üresen hagyva.

6.3. Hibásan implementált RSA törése

Készítsünk betű gyakoriság alapú törést egy hibásan implementált RSA kódoló: https://arato.inf.unideb.hu/batfai.n
(71-73 fólia) által készített titkos szövegen.

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Az RSA algoritmust 1976-ban fejlesztette ki Ron Rivest, Adi Shamir és Len Adleman, az RSA elvenezés a nevük kezdőbetűiből ered. Napjainkban is az egyik legelterjedtebb titkosító eljárás nagy biztonságának hála. A lényege, hogy két darab kulcs létezik, egy kódoló és egy dekódoló, melyek mindegyike egy-egy számpár. Az egyik a modulus és ez 512-4096 bit hosszú lehet, minnél nagyobb annál biztonságosabb a titkosítás. Egy elegendő nagyságú kulcs feltörésének jelenleg nincs ismert módja. A kodoló kulcs publikus viszont a dekódoló privát.

A kívánt szöveg karaktereit ASCII kóddá alakítjuk a modPow fügvénnyel, amiben e-edikre (exponent) emelünk m-mel (modolus) osztva és ennek a műveletnek a maradékát kapjuk. Amit fontos megjegyezni, hogy az e exponensünk, a kulcsunk egyik párja az nem más minta d kulcsunk inverze lesz, ez fogja nekünk biztosítani, hogy tudjunk majd dekódolni is. A visszafejtés szinte ugyanez lesz, csak nem e-edikre emelünk, hanem d-edikre, a modolus marad ugyanaz, majd a kapott ASCII kódot visszaalakítjuk.

```
import java.util.Scanner;
public class RSA {
    public static void main(String[] args) {
        KulcsPar jSzereplo = new KulcsPar();
        String szoveg;
        Scanner sc = new Scanner(System.in);
        System.out.println("Adja meg a titkosítani kívánt szöveget: ");
        szoveg=sc.nextLine();
        byte[] buffer = szoveg.getBytes();
        java.math.BigInteger[] titkos = new java.math.BigInteger[buffer. ↩
           length];
        for (int i = 0; i < titkos.length; ++i) {
            titkos[i] = new java.math.BigInteger(new byte[]{buffer[i]});
            titkos[i] = titkos[i].modPow(jSzereplo.e, jSzereplo.m);
        }
        for (java.math.BigInteger t : titkos) {
            System.out.print(t);
            System.out.println();
        }
        for (int i = 0; i < titkos.length; ++i) {
            titkos[i] = titkos[i].modPow(jSzereplo.d, jSzereplo.m);
            buffer[i] = titkos[i].byteValue();
        }
        System.out.println("\n" + new String(buffer));
    }
}
class KulcsPar {
```

```
    java.math.BigInteger d,e,m;
    public KulcsPar() {
        int meretBitekben = 700 * (int) (java.lang.Math.log((double) 10)
                / java.lang.Math.log((double) 2));
        java.math.BigInteger p = new java.math.BigInteger(meretBitekben,
           100, new java.util.
                Random());
        java.math.BigInteger q = new java.math.BigInteger(meretBitekben,
           100, new java.util.
                Random());
        m = p.multiply(q);
        java.math.BigInteger z = p.subtract(java.math.BigInteger.ONE).
           multiply(q.subtract(java.
                math.BigInteger.ONE));
        do {
            do {
                d = new java.math.BigInteger(meretBitekben, new java.util.
                   Random());
            } while (d.equals(java.math.BigInteger.ONE));
        } while (!z.gcd(d).equals(java.math.BigInteger.ONE));
        e = d.modInverse(z);
    }
}
```

6.1. ábra. RSA

6.4. Változó argumentumszámú ctor

Készítsünk olyan példát, amely egy képet tesz az alábbi projekt Perceptron osztályának bemenetére és a Perceptron ne egy értéket, hanem egy ugyanakkora méretű „képet” adjon vissza. (Lásd még a 4 hét/Perceptron osztály feladatot is.)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

6.5. Összefoglaló

Az előző 4 feladat egyikéről írj egy 1 oldalas bemutató „”esszé szöveget!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7. fejezet

Helló, Gödel!

7.1. Gengszterek

Gengszterek rendezése lambdával a Robotautó Világbajnokságban https://youtu.be/DL6iQwPx1Yw (8:05-től)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A Robotautó Világbajnokság forráskódjában szereplő lambda kifejezések használatával történo std::sort-ról lesz szó. A lambda függvény segítségével többsoros névtelen függvények definiálhatók. Általános alakja: [] {} () Ahol a szögletes zárójel jelzi, hogy lambda kifejetés következik, a kapcsos zárójelbe kerül a függvény törzse, a kerek zárójel pedig függvényhívást jelenti. Ha ezek közül valamelyik hiányzik az autómatikusan kitöltésre kerül.A forráskódban szereplő kód:

```
        std::sort ( gangsters.begin(), gangsters.end(), [this, cop] ( ↩
           Gangster x, ←-
        Gangster y )
        {
        return dst ( cop, x.to ) < dst ( cop, y.to );
        } ) ;
        f
```

A fenti lambda függvény csak az std::vector beépítése után működoképes. Az std::sort segítségével folyik a sorbarendezés, melynek paramétere a lambda függvény. A .begin() és a .end() függvények alkalmazásával mondjuk meg, hogy mettől meddig szeretnénk számításba venni a sorbarendezést. Mi most a gangsters vektort használjuk erre, azaz a vektor első és utolsó indexét adjuk meg. A szögletes zárójelben két változó van: a this segítségével érhetjük el az osztály tagjait, és a cop objektumára lenne szükségünk. Az std::sort() {} kapcsos zárójelében annak az értékét kapjuk meg, hogy melyik gangster van közelebb.

7.2. C++11 Custom Allocator

https://prezi.com/jvvbytkwgsxj/high-level-programming-languages-2-c11-allocators/ a CustomAlloc-os példa, lásd C forrást az UDPROG repóban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7.3. STL map érték szerinti rendezése

Például: https://github.com/nbatfai/future/blob/master/cs/F9F2/fenykard.cpp#L180

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Az STL a C++ nyelv Szabványos Sablonkönyvtára (Standard Template Library). Használatával fontos és elterjedt adatstruktúrákat és algoritmusokat építhetünk be a programunkba. 3 csoportja: konténerek, alogritmusok és iterátoroa. A map konténerekben adatpárok tárolódnak, ahol az első tag a kulcs, a második pedig az adat. A kódcsipet a fenykard.cpp programból van.

```
        std::vector<std::pair<std::string, int>> sort_map ( std::map <std:: ↩
           string, ←-
int> &rank )
{
std::vector<std::pair<std::string, int>> ordered;
for ( auto & i : rank ) {
if ( i.second ) {
std::pair<std::string, int> p {i.first, i.second};
ordered.push_back ( p );
}
}
std::sort (
std::begin ( ordered ), std::end ( ordered ),
[ = ] ( auto && p1, auto && p2 ) {
return p1.second > p2.second;
}
);
return ordered;
}
        f
```

Itt az látható, hogy a sort_map két paramétert vár: egy string és egy integer típusút. Ebből fogunk csinálni egy vektort, mégpedig úgy, hogy a két paramétert összepárosítjuk az std::pair függvénnyel, majd ezek összességét elnevezzük ordered-nek. A pair lehetové teszi, hogy egyetlen objektumban két különböz ˝ o˝ objektumot tároljunk, amelyek közül az elsőre a first, míg a másodikra a second névvel hivatkozhatunk. A push_back segítségével tesszük bele a kulcs/érték párokat az ordered vektorba. Ekkor jön a már megismert sorbarendezés. A szögletes zárójelben lévő egyenlőségjel segítségével az összes helyi változót érték szerint fogjuk elkapni és ez alapján rendezzük majd a vektort. A zárójelben lévő auto kulcsszóval pedig automatikusan fog megtörténni a típus-meghatározás.

7.4. Alternatív Tabella rendezése

Mutassuk be a https://progpater.blog.hu/2011/03/11/alternativ_tabella a programban a java.lang Interface Comparable T szerepét!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7.5. Prolog családfa

Ágyazd be a Prolog családfa programot C++ vagy Java programba! Lásd para_prog_guide.pdf!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7.6. GIMP Scheme hack

Ha az előző félévben nem dolgoztad fel a témát (például a mandalás vagy a króm szöveges dobozosat) akkor itt az alkalom!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8. fejezet

Helló, !

8.1. FUTURE tevékenység editor

Javítsunk valamit a ActivityEditor.java JavaFX programon! https://github.com/nbatfai/future/tree/master/cs/F6
Itt láthatjuk működésben az alapot: https://www.twitch.tv/videos/222879467

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Az "ActivityEditor" futtatásához szükséges parancsok:

```
$ git clone https://github.com/nbatfai/future.git
$ cd future/cs/F6
$ javac ActivityEditor.java
$ java ActivityEditor --city=Debrecen --props=me.props,gaming.props, ↩
   programming.props
```

Az editort CSS stíluslappal lehet szépíteni, ehhez azonban az alábbi kódra is szükség van:

```
public class ActivityEditor extends javafx.application.Application {
    private static String path;
    public static void main(String[] args) {
  path="style.css";
        javafx.application.Application.launch(args);
    }
```

Az editor dobozainak fő háttérszíne a világoszöld, a tevékenységek és tulajdonságok sötétkékek lesznek. Ha rákattintunk egy dobozra, akkor annak szegélye is sötétkékké válik. A címkék háttere halvány-szürke. Az alábbi kód a CSS stíluslapé:

```
.tree-cell{
  -fx-background-color: #8ed49d;
  -fx-text-fill: #2200ff;
}
.vbox{
```

```
  -fx-background-color: #2f00ff;
}
.label{
  -fx-text-fill: #2f00ff;
  -fx-background-color: #eee9e1;
}
.content{
  -fx-background-color: #8ed49d;
}
```

(ide még fog jönni egy kép)

8.2. OOCWC Boost ASIO hálózatkezelése

Mutassunk rá a scanf szerepére és használatára! https://github.com/nbatfai/robocaremulator/blob/master/justine/ro

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

Tutor: Kovács Ferencz

Az OOCWC (Robocar World Championship) egy páréves platform, melynek célja volt a forgalomirányítási algoritmusok kutatása és a robotautók terjedésének vizsgálata. A "https://github.com/nbatfai/robocar-emulator" Robocar City Emulator lehetővé tette volna fejlesztők számára új modellek és elméletek tesztelését. A debreceni Justine prototípus része a CarLexer, melynek sscanf függvényét kell feldolgoznunk.

A sscanf függvény addig dolgozza fel a formázott stringből az adatokat, amíg meg nem kapja a Gangster négy argumentumát. A %n az olvasott karakterek számát rögzíti, az nn változóba kerülnek tehát az összesen beolvasottak száma. A data segítségével tudjuk olvasni a még beolvasatlan adatokat. Ezt a pointert a beolvasott karakterek méretével toljuk el, így a data+nn az olvasatlan rész elejére fog mutatni. Az <OK %d %u %u %u> alak teljesülése esetén egy gengszter összes adata beolvasásra került, tehát létrehozunk egy új objektumot és belehelyezzük a gangster vektorba.

```
while (std::sscanf (data+nn, "<OK %d %u %u %u>%n", &idd, &f, &t, &s, &n) == ↩
    4)
    {
        nn += n;
        gangsters.push_back (Gangster {idd, f, t, s});
    }
```

8.3. SamuCam

Mutassunk rá a webcam (pl. Androidos mobilod) kezelésére ebben a projektben: https://github.com/nbatfai/SamuC

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A "https://github.com/nbatfai/samucam" linken található Samucam kísérlet lényege, hogy a program képes felismerni, tanulmányozni az emberi arcokat videók és fotók alapján. Ugyan az emberi arcok megkülönböztetése nehézkessé válik, az arcfelismerés jól működik az OpenCV könyvtár segítségével. A videók és fotók megmutatásához használhatjuk Androidos készülékünk kameráját, de jelen esetben én egy otthoni laptop kameráját használtam.

A SamuCam.cpp-ben látható, hogy a videoCapture open függvénye nyitja meg a VideoStream-et. Mivel alapból telefonos használatra íródott a program, ezért a VideoStream helyett 0-t kell írnunk, ha webkamerával szeretnénk dolgozni. Ezután beállítjuk a kamerakép szélességét, magasságát és FPS értékét.

```
SamuCam::SamuCam ( std::string videoStream, int width = 176, int height =  ↩
    144 )
  : videoStream ( videoStream ), width ( width ), height ( height )
{
  openVideoStream();
}
SamuCam::~SamuCam ()
{
}
void SamuCam::openVideoStream()
{
  videoCapture.open ( 0 );
  videoCapture.set ( CV_CAP_PROP_FRAME_WIDTH, width );
  videoCapture.set ( CV_CAP_PROP_FRAME_HEIGHT, height );
  videoCapture.set ( CV_CAP_PROP_FPS, 10 );
}
```

A CascadeClassifier alapvetően tárgyak felismerését segíti, jelen esetben pedig ez teszi lehetővé a kameraképen látható arcok feldolgozását. A helyes működés érdekében le fogok szedni a GitHub-ról egy frontal face XML-t, melyben az értékek a kamerával szemben elhelyezett arcképek felismerését biztosítja.

```
void SamuCam::run()
{
  cv::CascadeClassifier faceClassifier;
  std::string faceXML = "lbpcascade_frontalface.xml"; // https://github.com ↩
     /Itseez/opencv/tree/master/data/lbpcascades
  if ( !faceClassifier.load ( faceXML ) )
    {
      qDebug() << "error: cannot found" << faceXML.c_str();
      return;
    }
```

Képkockánként kerül beolvasásra a kamerakép a read függvényben. Ha kap egy képkockát, akkor első lépésként átméretezi, majd szürkére átszínezi. Az equalizeHist függvény kiegyenlíti a szürke képkocka hisztogramját.

```
while ( videoCapture.read ( frame ) )
        {
          if ( !frame.empty() )
            {
```

```
            cv::resize ( frame, frame, cv::Size ( 176, 144 ), 0, 0, cv:: ↩
               INTER_CUBIC );
            std::vector<cv::Rect> faces;
            cv::Mat grayFrame;
            cv::cvtColor ( frame, grayFrame, cv::COLOR_BGR2GRAY );
            cv::equalizeHist ( grayFrame, grayFrame );
```

A detectMultiScale függvény segítségével történik a képkockán lévő arcok keresése. Ha talált egy arcot, akkor az alapján létrehozunk egy QImage-t. A faceChanged egy signal, bekövetkezte után az arc köré rajzolunk egy keretet, és egy újabb QImage-t készítünk. Ha pedig a webcamChanged signal bekövetkezett, akkor 80 ms-t követően következhet egy újabb képkocka beolvasása.

```
            faceClassifier.detectMultiScale ( grayFrame, faces, 1.1, 3,  ↩
               0, cv::Size ( 60, 60 ) );
            if ( faces.size() > 0 )
              {
                cv::Mat onlyFace = frame ( faces[0] ).clone();
                QImage* face = new QImage ( onlyFace.data,
                                            onlyFace.cols,
                                            onlyFace.rows,
                                            onlyFace.step,
                                            QImage::Format_RGB888 );
                cv::Point x ( faces[0].x-1, faces[0].y-1 );
                cv::Point y ( faces[0].x + faces[0].width+2, faces[0].y + ↩
                    faces[0].height+2 );
                cv::rectangle ( frame, x, y, cv::Scalar ( 240, 230, 200 ) ↩
                    );
                emit  faceChanged ( face );
              }
            QImage*  webcam = new QImage ( frame.data,
                                           frame.cols,
                                           frame.rows,
                                           frame.step,
                                           QImage::Format_RGB888 );
            emit  webcamChanged ( webcam );
          }
        QThread::msleep ( 80 );
      }
```

A SamuCam futtatásához először le kell szednünk a GitHub projektet, majd pedig a korábban említett XML fájlt. A Qt keretrendszer segítségével létrehozzuk a Makefile-t, ezután pedig indíthatjuk is a programot az alábbi módon:.

```
git clone https://github.com/nbatfai/SamuCam.git
cd SamuCam/
wget https://github.com/Itseez/opencv/raw/master/data/lbpcascades/ ↩
   lbpcascade_frontalface.xml
git checkout vInitialHack
 ~/Qt/5.9.8/gcc_64/bin/qmake SamuLife.pro
 make
 ./SamuCam
```

(A végeredményről itt is lesz kép)

8.4. BrainB

Mutassuk be a Qt slot-signal mechanizmust ebben a projektben: https://github.com/nbatfai/esporttalent-search

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.5. OSM térképre rajzolása

Debrecen térképre dobjunk rá cuccokat, ennek mintájára, ahol én az országba helyeztem el a DEAC hekkereket: https://www.twitch.tv/videos/182262537 (de az OOCWC Java Swinges megjelenítőjéből: https://github.com emulator/tree/master/justine/rcwin is kiindulhatsz, mondjuk az komplexebb, mert ott időfejlődés is van...)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9. fejezet

Helló, Schwarzenegger!

9.1. Port scan

Mutassunk rá ebben a port szkennelő forrásban a kivételkezelés szerepére! https://www.tankonyvtar.hu/hu/tartalom
tanitok-javat/ch01.html#id527287

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9.2. AOP

Szőj bele egy átszövő vonatkozást az első védési programod Java átiratába! (Sztenderd védési feladat volt korábban.)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9.3. Android Játék

Írjunk egy egyszerű Androidos „játékot”! Építkezzünk például a 2. hét „Helló, Android!” feladatára!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9.4. Junit teszt

A https://progpater.blog.hu/2011/03/05/labormeres_otthon_avagy_hogyan_dolgozok_fel_egy_pedat poszt kézzel számított mélységét és szórását dolgozd be egy Junit tesztbe (sztenderd védési feladat volt korábban).

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9.5. OSCI

Készíts egyszerű C++/OpenGL-es megjelenítőt, amiben egy kocsit irányítasz az úton. A kocsi állapotát minden pillanatban mentsd le. Ezeket add át egy Prolog programnak, ami egyszerű reflex ágensként adjon vezérlést a kocsinak, hasonlítsd össze a kézi és a Prolog-os vezérlést. Módosítsd úgy a programodat, hogy ne csak kézzel lehessen vezérelni a kocsit, hanem a Prolog reflex ágens vezérelje!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

10. fejezet

Helló, Calvin!

10.1. MNIST

Az alap feladat megoldása, +saját kézzel rajzolt képet is ismerjen fel, https://progpater.blog.hu/2016/11/13/hello_s
bol Háttérként ezt vetítsük le: https://prezi.com/0u8ncvvoabcr/no-programming-programming/

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

10.2. Deep MNIST

Mint az előző, de a mély változattal. Segítő ábra, vesd össze a forráskóddal a https://arato.inf.unideb.hu/batfai.norb
8. fóliáját!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

10.3. CIFAR-10

Az alap feladat megoldása, +saját fotót is ismerjen fel, https://progpater.blog.hu/2016/12/10/hello_samu_a_cifar-
10_tf_tutorial_peldabol

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

10.4. Android telefonra a TF objektum detektálója

Telepítsük fel, próbáljuk ki!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

10.5. SMNIST for Machines

Készíts saját modellt, vagy használj meglévőt, lásd: https://arxiv.org/abs/1906.12213

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

10.6. Minecraft MALMO-s példa

A https://github.com/Microsoft/malmo felhasználásával egy ágens példa, lásd pl.: https://youtu.be/bAPSu3Rndi8,
https://bhaxor.blog.hu/2018/11/29/eddig_csaltunk_de_innentol_mi, https://bhaxor.blog.hu/2018/10/28/minecraft_

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

11. fejezet

Helló, Berners-Lee!

11.1. C++ és Java összehasonlítás

C++: Benedek Zoltán, Levendovszky Tihamér Szoftverfejlesztés C++ nyelven

Java: Nyékyné Dr. Gaizler Judit et al. Java 2 útikalauz programozóknak 5.0 I-II

11.2. Python

Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba. Gyors prototípus-fejlesztés Python és Java nyelven (35-51 oldal), a kijelölt oldalakból élmény-olvasónapló

III. rész

Irodalomjegyzék

11.3. Általános

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

11.4. C

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

11.5. C++

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

11.6. Lisp

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

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

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