

# A Prog2 tematika tartalmi megtöltése

A jelen dokumentumban konkrét labor és otthoni feladatokat rendelünk a heti bontású tematika tételeihez. A heti labormunka az adott héthez rendelt feladatok teljesítésén alapszik. A hallgatók a laboron is önállóan dolgoznak, az oktató rövid indító iránymutatása (például a feladatok pontosítása) után. A feladatok teljesítésére időben az adott hét áll rendelkezésre<sup>1</sup>, dolgozni rajtuk a laboron és természetesen otthon is lehet (kell)<sup>2</sup>. A heti előrehaladást egyénileg karbantartott laborjegyzőkönyvben kell vezetni. Ez a jegyzőkönyv rögzíti a hallgató előrehaladását az adott feladatokkal: sikeres megoldásról link a forrásokra, képernyőkép, YB videó link stb. (ha a megoldás nem sikerül, akkor arról is pár soros reflektálás, például az általam bemutatott megoldás már a környezet fejlődése miatt pocccintésre nem működik és a hallgatónak önállóan nem sikerült életre keltenie, mert...).

## Értékelési szempontok

A gyakorlati jegyet az utolsó harmadban a laborjegyzőkönyvre és a védésre adjuk.

- A jeles szükséges (de nem elégséges) feltétele, hogy a jegyzőkönyv DocBook XML 5.0 forrásban készüljön, abból dlatex-es pdf állományt kérünk<sup>3</sup>. Minden héten mind az 5 feladatra legyen megoldásunk bemutatva a jegyzőkönyvben. Legalább 3 hallgatótárs feltünteti a jegyzőkönyvében, hogy a szóban forgó hallgató tutorja volt.
- A jó szükséges (de nem elégséges) feltétele, hogy minden héten az 5 feladatból legyen 4 feladatra megoldásunk bemutatva a jegyzőkönyvben.
- A közepes szükséges (de nem elégséges) feltétele, hogy minden héten az 5 feladatból legyen 3 feladatra megoldásunk bemutatva a jegyzőkönyvben.
- A 9 heti bontásból 1-nél lehet három feladtnál kevesebb megoldás, ha 2 vagy több olyan hét van, ahol három feladtnál kevesebb megoldás van, az elégtelen gyakorlati jegyet eredményez.
- Ha bármely hétnél 1 megoldás van vagy egyetlen megoldás sincs, az elégtelen gyakorlati jegyet eredményez.

A félév utolsó harmada a védésé, amely a jegyzőkönyvből az oktató által kiválasztott feladat gép melletti bemutatásából áll. A jegyzőkönyv és a védés alapján adja a laborvezető a gyakorlati jegyet. Ezt időben támogatandó a tematika néhány párját összevontuk az alábbiak szerint.

## Háttér

Előadás nincs, ha nyelvi kérdésed van, akkor ezek forgatásával kezd:

- C++: Benedek Zoltán, Levendovszky Tihamér Szoftverfejlesztés C++ nyelven
- Java: Nyékyné Dr. Gaizler Judit et al. Java 2 útikalauz programozóknak 5.0 I-II.
- Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba. Gyors prototípus-fejlesztés Python és Java nyelven (35-51 oldal)

Feladatok kapcsán a tutorod, az UDPROG közösség (minden feladat már megoldott vagy a repóban, vagy az évkönyvben vagy a fészes csoportban) és a laborvezetőd tud segíteni.

---

<sup>1</sup> Az adott héten legalább el kell kezdeni a feladatot és ennek heti állapotát rögzíteni a jegyzőkönyvbe. A megkezdett feladat utólag egészen a leadásig karbantartható. A laborvezető a jegyzőkönyv előző hetét ellenőrizheti, ha egy feladat nincs megkezdve, az utólag már nem pótolható, a végelszámolásnál (gyakorlati jegy meghatározásánál) nem vehető figyelembe.

<sup>2</sup> Előre dolgozni, azaz későbbi hetek feladatait csinálni is lehet.

<sup>3</sup> A jegyzőkönyv formátuma bármikor változtatható.

# 1. hét

## 1. hét Az objektumorientált paradigma alapfoglamai. Osztály, objektum, példányosítás.

### 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/deprecated/Prog1\\_5.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog1_5.pdf) (16-22 fólia)

Ugyanezt írjuk meg C++ nyelven is! (lásd még UDPROG repó: source/labor/polargen)

### Homokózó

Írjuk 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!<sup>4</sup>

### „Gagyi”

Az ismert formális<sup>5</sup> „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, írd Java példaprogramot mely egyszer végtelen ciklus, más x, t értékekkel meg nem! A példát építsd a JDK Integer.java forrására<sup>6</sup>, hogy a 128-nál inkluzív objektum példányokat poolozza!

### Yoda

Írjunk olyan Java programot, ami java.lang.NullPointerException-el leáll, ha nem követjük a Yoda conditions-t! [https://en.wikipedia.org/wiki/Yoda\\_conditions](https://en.wikipedia.org/wiki/Yoda_conditions)

### Kódolás from scratch

Induljunk ki ebből a tudományos közleményből: <http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/bbp-alg.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-tanitok-javat/apbs02.html#pi\\_jegyei](https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apbs02.html#pi_jegyei) (mert ha csak lemásolod, akkor pont az a fejlesztői élmény marad ki, melyet szeretném, ha átélnél).

<sup>4</sup> Tavalyi prog2 első védés volt.

<sup>5</sup> [https://www.facebook.com/groups/udprog/permalink/437825193072042/?comment\\_id=437862206401674&reply\\_comment\\_id=437863669734861&comment\\_tracking=%7B%22tn%22%3A%22R3%22%7D](https://www.facebook.com/groups/udprog/permalink/437825193072042/?comment_id=437862206401674&reply_comment_id=437863669734861&comment_tracking=%7B%22tn%22%3A%22R3%22%7D)

<sup>6</sup> A JDK telepítési könyvtárában az src.zip-ben található.

## 2. hét

### 2. hét Öröklődés, osztályhierarchia. Polimorfizmus, metódustúlterhelés. Hatáskörkezelés. A bezárási eszközrendszer, láthatósági szintek. Absztrakt osztályok és interfészek.

#### 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](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/)

#### Szülő-gyerek

Írjunk Szülő-gyerek Java és C++ osztálydefiníciót, amelyben demonstrálni tudjuk, hogy az ősön keresztül csak az ős üzenetei küldhetők!

[https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_1.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_1.pdf) (98. fólia)

#### Anti OO

A BBP algoritmussal<sup>7</sup> 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/tartalom/tkt/javat-tanitok-javat/apas03.html#id561066>

#### 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!

#### 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](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_2.pdf) (77-79 fóliát)!

## 3. hét

### 3. hét Modellező eszközök és nyelvek. AZ UML és az UML osztálydiagramja.

#### 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.

---

<sup>7</sup>[https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apbs02.html#pi\\_jegyei](https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/apbs02.html#pi_jegyei)

## Forward engineering UML osztálydiagram

UML-ben tervezzünk osztályokat és generáljunk belőle forrást!

## Egy esettan

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

## BPMN

Rajzoljunk le egy tevékenységet BPMN-ben!

[https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_7.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_7.pdf) (34-47 fólia)

## BPEL Helló, Világ! - egy visszhang folyamat

Egy visszhang folyamat megvalósítása az alábbi teljes „videó tutorial” alapján:

[https://youtu.be/0OnlYWX2v\\_I](https://youtu.be/0OnlYWX2v_I)

## 4. hét

### 4. hét      Objektumorientált programozási nyelvek programnyelvi elemei: karakterkészlet, lexikális egységek, kifejezések, utasítások.

## Encoding

Fordítsuk le és futtassuk a Javat tanítók könyv MandelbrotHalmazNagyító.java forrását úgy, hogy a fájl neveiben és a forrásokban is meghagyjuk az ékezetes betűket!

<https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/adatok.html>

## OOCWC lexer

Vázzuk a <https://github.com/nbatfai/robocar-emulator/blob/master/justine/rcemu/src/carlexer.ll> lexert és kapcsolását a programunk OO struktúrájába!

## l334d1c4

Írj olyan OO Java vagy C++ osztályt, amely leet cipherként működik, azaz megvalósítja ezt a betű helyettesítést: <https://simple.wikipedia.org/wiki/Leet>

## Full screen

Készítsünk egy teljes képernyős Java programot!

Tipp: [https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus\\_jatek](https://www.tankonyvtar.hu/en/tartalom/tkt/javat-tanitok-javat/ch03.html#labirintus_jatek)

## Perceptron osztály

Dolgozzuk be egy külön projektbe a projekt Perceptron osztályát!

Lásd <https://youtu.be/XpBnR31BRJY>

## 5. hét

**5. hét      Objektumorientált programozási nyelvek  
típusrendszere (pl.: Java, C#) és 6. hét      Típusok tagjai:  
mezők, (nevesített) konstansok, tulajdonságok, metódusok,  
események, operátorok, indexelők, konstruktorok,  
destruktorok, beágyazott típusok.**

Összevonva.

### 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)!

### Másoló-mozgató szemantika

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

### Hibásan implementált RSA törése

Készítsünk betű gyakoriság alapú törést egy hibásan implementált RSA kódoló:

[https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2\\_3.pdf](https://arato.inf.unideb.hu/batfai.norbert/UDPROG/deprecated/Prog2_3.pdf) (71-73 fólia) által készített titkos szöveget.

### 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.)

### Összefoglaló

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

## 6. hét

**7. hét      Interfészek. Kollekcíók. és 8. hét Funkcionális nyelvi  
elemek. Lambda kifejezések.**

Összevonva.

### Gengszterek

Gengszterek rendezése lambdával a Robotautó Világbajnokságban

<https://youtu.be/DL6iQwPx1Yw> (8:05-től)

## 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!

## STL map érték szerinti rendezése

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

## Alternatív Tabella rendezése

Mutassuk be a [https://progpater.blog.hu/2011/03/11/alternativ\\_tabella](https://progpater.blog.hu/2011/03/11/alternativ_tabella) a programban a java.lang Interface Comparable<T> szerepét!

## Összefoglaló

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

## 7. hét

## 9. hét      Adatfolyamok kezelése, streamek és 11. hét      I/O, állománykezelés. Szerializáció.

Összevonva.

## 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>

## OOCWC Boost ASIO hálózatkezelése

Mutassunk rá a scanf szerepére és használatára!

<https://github.com/nbatfai/robocar-emulator/blob/master/justine/rcemu/src/carlexer.ll>

## SamuCam

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

## BrainB

Mutassuk be a Qt slot-signal mechanizmust ebben a projektben: <https://github.com/nbatfai/esport-talent-search>

## OSM térképre rajzolása

Debrecen térképre dobunk 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/nbatfai/robocar-emulator/tree/master/justine/rcwin> is kiindulhatsz, mondjuk az komplexebb, mert ott időfejlődés is van...)

## 8. hét

### 10. hét Kivételkezelés. és 12. hét Reflexió. A fordítást és a kódgenerálást támogató nyelvi elemek (annotációk, attribútumok).

Összevonva.

#### Port scan

Mutassunk rá ebben a port szkennelő forrásban a kivételkezelés szerepére!

<https://www.tankonyvtar.hu/hu/tartalom/tkt/javat-tanitok-javat/ch01.html#id527287>

#### AOP

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

#### Android Játék

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

#### Junit teszt

A [https://progater.blog.hu/2011/03/05/labormeres\\_otthon\\_avagy\\_hogyan\\_dolgozok\\_fel\\_egy\\_pedat](https://progater.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).

#### Összefoglaló

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

## 9. hét

### 13. hét Multiparadigmás nyelvek és 14. hét Programozás multiparadigmás nyelveken.

Összevonva.

#### MNIST

Az alap feladat megoldása, +saját kézzel rajzolt képet is ismerjen fel,  
[https://progater.blog.hu/2016/11/13/hello\\_samu\\_a\\_tensorflow-bol](https://progater.blog.hu/2016/11/13/hello_samu_a_tensorflow-bol)

#### Deep MNIST

Mint az előző, de a mély változattal.

#### CIFAR-10

Az alap feladat megoldása, +saját fotót is ismerjen fel,  
[https://progater.blog.hu/2016/12/10/hello\\_samu\\_a\\_cifar-10\\_tf\\_tutorial\\_peldabol](https://progater.blog.hu/2016/12/10/hello_samu_a_cifar-10_tf_tutorial_peldabol)

## **Android telefonra a TF objektum detektálója**

Telepítsük fel, próbáljuk ki!

### **Összefoglaló**

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

Debrecen, 2018-09-10

Dr. Bátfai Norbert