

Univerzális programozás

Írd meg a saját programozás tankönyvedet!

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

Copyright © 2019 Dr. Bátfai Norbert

Copyright (C) 2019, Norbert Bátfai Ph.D., batfai.norbert@inf.unideb.hu, nbatfai@gmail.com,

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

<https://www.gnu.org/licenses/fdl.html>

Engedélyt adunk Önnek a jelen dokumentum sokszorosítására, terjesztésére és/vagy módosítására a Free Software Foundation által kiadott GNU FDL 1.3-as, vagy bármely azt követő verziójának feltételei alapján. Nincs Nem Változtatható szakasz, nincs Címlapszöveg, nincs Hátlapszöveg.

<http://gnu.hu/fdl.html>

COLLABORATORS

	<i>TITLE :</i> Univerzális programozás	
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>
WRITTEN BY	Bátfai, Norbert	2019. szeptember 26.

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
0.0.1	2019-02-12	Az iniciális dokumentum szerkezetének kialakítása.	nbatfai
0.0.2	2019-02-14	Inciális feladatlisták összeállítása.	nbatfai
0.0.3	2019-02-16	Feladatlisták folytatása. Feltöltés a BHAX csatorna https://gitlab.com/nbatfai/bhax repójába.	nbatfai
0.0.4	2019-02-19	Aktualizálás, javítások.	nbatfai

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

I. Bevezetés	1
1. Vízió	2
1.1. Mi a programozás?	2
1.2. Milyen doksikat olvassak el?	2
1.3. Milyen filmeket nézzek meg?	2
II. Második felvonás	3
2. Helló, Berners-Lee!	5
2.1. JAVA, C++ összehasonlítás:	5
2.2. Python olvasó napló	6
3. Helló, Arroway!	7
3.1. OO szemlélet	7
3.2. Homokózó	7
3.3. „Gagyí”	7
3.4. Yoda	8
3.5. Kódolás from scratch	8
III. Irodalomjegyzék	9
3.6. Általános	10
3.7. C	10
3.8. C++	10
3.9. Lisp	10

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 (hogyan lássuk mást is) példával.

Hogyan nyomjuk?

Ránts 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 találsz 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.

DRAFT

2. fejezet

Helló, Berners-Lee!

2.1. JAVA, C++ összehasonlítás:

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

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

Osztályozásnak nevezzük azt a folyamatot, amelynek során a hasonló objektumokat közös csoportokba azaz osztályokba soroljuk. Az objektumorientált programokban közös tervezésre ad lehetőséget, hogy sok objektum hasonló jellemzőkkel rendelkezik. Nagy előnyt jelent az, ha sok hasonló objektumot közös "tervrajz" alapján tudunk elkészíteni. Ezeket az ún. tervdrajzokat hívjuk osztályoknak. Az osztály bizonyos fajta objektumok közös változóit és metódusait írja le. Az osztályok definiálhatnak példányváltozókat, osztályváltozókat és osztálymetódusokat is. Az osztályváltozók az összes objektumpéldány számára megosztott információkat tartalmaznak. Ha osztályváltozókat alkalmazunk akkor feleslegesek lesznek a példányváltozók. Az osztályok legnagyobb előnye az újrafelhasználhatóság. Az objektum változókból és kapcsolódó metódusokból felépített egység. Az objektum tulajdonságait célszerű elrejtetni tehát nem publikusként kezelni és csak a metódusokon keresztül befolyásolni. Az objektumok használatának legnagyobb előnye a modularitás és az információelrejtés. Modularitás: Az objektum forráskódja független marad más objektumok forráskódjától és ennek köszönhetően könnyen tud illeszkedni a rendszer különböző részeihez. Információ elrejtés: Az objektum a publikus interfészén kommunikál a többi objektum felé. Illetve gondoskodik a saját adatairól és csak a metódusain keresztül ad változtatási lehetőséget a külső objektumoknak. A külső objektumoknak igazából nem is kell tudnia arról, hogy az objektum állapota hogyan van reprezentálva, csak a kívánt viselkedést kell kérnie a metódusokon keresztül. A példányosításhoz a new operátort használjuk a Javában. Ez egy új példányt hoz létre az osztályból és foglal helyet a memóriában. Szükségünk van egy konstruktorra is ami a hívást írja elő, ennek a neve adja majd meg, hogy melyik osztályból kell létrehozni az új példányt és inicializálja az új objektumot. A new operátor egy hivatkozást ad vissza a létrehozott objektumra. Gyakran ezt a hivatkozást hozzárendeljük egy változóhoz. Ha a hivatkozás nincs hozzárendelve változóhoz, az objektumot nem lehet majd elérni, miután a new operátort tartalmazó utasítás végrehajtott. Az ilyen objektumot névtelen objektumnak is szoktuk nevezni. A java fordító egy bájtkódnak nevezett formátumra fordítja le a forráskódot amit a jvm önálló interpreterként fog érzékelni. Előnyös biztonsági szempontból de lassú, ezért minden jó jvm próbálja növelni a sebességet. A kódot fordítás előtt platformfüggő gépi kódra alakítja át. A nyelv szintaxisa c, c++ ból fejlődött ki, ez a szerkezetben jelenik meg de a java el is tér tőlük vagyis a hasonlóság nem egyenlő az azonossággal. C és c++-al szemben nincs alapértelmezett visszatérési érték, mindig meg kell adni, változókhöz "="-el lehet értéket rendelni kezdeti értékadás után nincs definiálva az érték. Még egy különbség, hogy név túlterhelés ugyan van a Javában

is viszont operátor túlterhelés nincs benne. Illetve számos c++ kifejezés, utasítás szintaktikailag helyes Javában is sőt sokszor a jelenetése is megegyezik. A java mint nyelv szűkebb a c++-nál, de az osztálykönyvtárai miatt szélesebb az alkalmazhatósági területe. Támogatja például a GUI programozást, network programozást vagy éppen a perzisztenciát is. C++-ban is lehet ezeket csinálni de van amelyikhez külső könyvtárak segítségét kell igénybe vennünk. Valamint lehet benne forrás szinten hordozható programokat írni de a szerkesztett bináris kód már nem hordozható, mert a lefordított kód tartalmazza a helyi oprendszerre és hardverre vonatkozó feltételezéseket. A Java egyik fő célja és egyben egyik legnagyobb különbsége a c++-hoz képest az, hogy a kód teljes mértékben hordozható. Emiatt a Java szigorúbb előírásokat szab a típusok méretére, belső szerkezetére, a kifejezések kiértékelésére és a kivételek kiváltásának ellenőrzésére. A statikus változók inicializálása is futási időben történik Javában valamint sokkal kevesebb dolgot bíz az implementációra mint a c vagy a c++. Ezt azért teszi, hogy maga a kód amely hordozható ne függjön annyira a platformtól és az implementációtól. A Java nyelv nagyon odafigyel arra, hogy a kód a lehető legpontosabban forduljon le és működjön. Ezért az ellenőrzés során kitér olyan dolgokra is amire a c++ és a c nem. Ilyen például a lokális változók ellenőrzése, pontosabban annak ellenőrzése, hogy kapnak-e értéket.

2.2. Python olvaso naplo

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

A python programozási nyelv a C++-tól, c-től és a Javától eltérő módon arra lett inkább tervezve hogy ne a futási sebességet tegye előtérbe, hanem inkább a programozót segítse azzal, hogy könnyebben olvasható. Maga a Python egy nagyon magas szintű programozási nyelv melyet 1989 és 1991 között alkottak meg. Egy objektumorientált interpreteres nyelv (tehát rögtön futtatható, nincs különbség a forrás és a tárgykód között). Ahogy a könyv címe is sejteti az olvasóval a Python legelterjedtebb felhasználási területe a mobilprogramozás. A nyelv legfőbb jellemzője ami megkülönbözteti az általunk már jobban ismert nyelvektől és újdonság lehet a számunkra, hogy a szintaxisa behúzás alapú. Ez azt jelenti hogy az állításokat azonos szintű behúzásokkal tudjuk csoportosítani. Nem kell kapcsos zárójeleket és kulcsszavakat mint például a begin és az end használatba vennünk ehhez a feladathoz. Nagyon fontos, hogy az első utasítás a szkriptben nem lehet behúzás illetve ezeket egységesen kell kezelnünk. Továbbá az utasítások csak a sor végéig tartanak nem kell ezeket lezárni. Ha mégsem férne egy utasítás egy sorba akkor '/'-el tudjuk ezt folytatni a következő sorba.

3. fejezet

Helló, Arroway!

3.1. OO szemlélet

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 (16-22 fólia) Ugyanezt írjuk meg C++ nyelven is! (lásd még UDPROG repó: source/labor/polargen)

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.2. 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! 1

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.3. „Gagyí”

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, írd Java példaprogramot mely egyszer végtelen ciklus, más x, t értékekkel meg nem! A példát építsd a JDK Integer.java forrására 3, hogy a 128-nál inkluzív objektum példányokat poolozza!

Megoldás videó:

```
package gagyi;
class Gagyi {
    public static void main(String[] args) {

        Integer x=-128;
        Integer y=-128;

        System.out.println(x);
        System.out.println(y);

        while (x <= y && x >= y && y != x);

    }
}
```

Tanulságok, tapasztalatok, magyarázat...

3.4. 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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.5. 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 (mert ha csak lemásolod, akkor pont az a fejlesztői élmény marad ki, melyet szeretném, ha átélnél)

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

III. rész

Irodalomjegyzék

DRAFT

3.6. Általános

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

3.7. C

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

3.8. C++

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

3.9. 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.