

# Univerzális programozás

---

**Írd meg a saját programozás tankönyvedet!**

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

Copyright © 2019 Bakos Bálint

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>	<i>SIGNATURE</i>
WRITTEN BY	Bakos, Bálint	2019. szeptember 19.	

## 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 <a href="https://gitlab.com/nbatfai/bhax">https://gitlab.com/nbatfai/bhax</a> repójába.	nbatfai
0.0.4	2019-02-19	Aktualizálás, javítások.	nbatfai

# Tartalomjegyzék

<b>I. Bevezetés</b>	<b>1</b>
<b>1. Vízió</b>	<b>2</b>
1.1. Mi a programozás?	2
1.2. Milyen doksikat olvassak el?	2
1.3. Milyen filmeket nézzek meg?	2
<b>II. Második felvonás</b>	<b>3</b>
<b>2. Helló, Berners-Lee!</b>	<b>5</b>
2.1. Olvasónapló: C++: Benedek Zoltán, Levendovszky Tihamér Szoftverfejlesztés C++ nyelven és Java: Nyékyné Dr. Gaizler Judit et al. Java 2 útikalauz programozóknak 5.0 I--II.II.	5
2.2. Olvasónapló: Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobil-programozásba. Gyors prototípus fejlesztés Python és Java nyelven	6
<b>3. Helló, Arroway!</b>	<b>7</b>
3.1. OO szemlélet	7
3.2. Homokozó	14
3.3. "Gagyí"	14
3.4. Yoda	14
3.5. Kódolás from scratch	14
<b>III. Irodalomjegyzék</b>	<b>15</b>
3.6. Általános	16
3.7. C	16
3.8. C++	16
3.9. Lisp	16

## 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]

# 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?

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 dlatex 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
dlatex 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 'dlatex' 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

### Mi a programozás?

### 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.

### 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!

**Olvasónapló: C++: Benedek Zoltán, Levendovszky Tihamér Szoftverfejlesztés C++ nyelven és Java: Nyékyné Dr. Gaizler Judit et al. Java 2 útikalauz programozóknak 5.0 I--II.II.**

A C++ valamint a Java nyelv is magasszintű programozási nyelvnek számít. Mindkettő objektumorientált, azonban mivel a C++ nyelv hamarabb alakult ki így kisebb-nagyobb eltérések találhatók a két nyelv között.

A C++ a C nyelvtől örökölt gépközel konstrukciókat, ebből adódik sebessége. A Java nyelv pedig a C++-ból vett át sok mindent. Azonban van egy lényeges eltérés, hogy az Java nyelv a mutatók helyett referenciákat használ, így biztonságosabb, megbízhatóbb programokat lehet írni. Valamint a programok hordozhatósága is eltér. Például ha egy Linuxon írt C++ kódot akarunk átvinni Windowsra az nem biztos, hogy működni fog, de ha egy Java kódot akarunk futtani máshol az jól fog szuperálni feltéve, hogy van JVM(Java Virtual Machine) a gépen. Mivel ez a Java bájt kódot fogja futtani, így ez platformfüggetlen lesz.

Szintaktikában is találunk eltéréseket a két nyelv között.

Hasonlítsuk össze mondjuk ezt a két Hello world! programot.

C++-ban:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

Java-ban:

```
public class HelloWorld {
    public static void main(String[] args) {
```

```
        System.out.println("Hello World!");  
    }  
}
```

Amint látható a Java nagyrészen osztályalapú. Az osztályoknak különböző elérése lehet: public, protected, private. Az osztályokon belül létrehozhatunk változókat, függvényeket.

## **Olvasónapló: Python: Forstner Bertalan, Ekler Péter, Kelényi Imre: Bevezetés a mobilprogramozásba. Gyors prototípus fejlesztés Python és Java nyelven**

A Python támogatja a funkcionális valamint az imperatív nyelveket is. Legfőbb jellemzője, ami teljesen eltér a többi magasszintű programozási nyelvtől, hogy behúzásalapú a szintaxisa, tehát semmilyen kapcsos zárójelre vagy explicit kulcsszóra nincs szükség. Valamint a sorok végén már megszokott pontosvessző sem kell.

```
if 1 < 2  
    print("Nagyobb!")  
else:  
    print("Kissebb!")
```

A Python egy objektumorientált nyelv, tehát minden adatot objektumok reprezentálnak. A változók típusának explicit megadására sincs szükség, a rendszer futási időben dönti el a változók típusát.

Ilyen egyszerűen néz ki Pythonban egy deklarálás:

```
name = "Ádám"
```

A nyelvben definiálhatunk osztályokat is és ezeknek példányai az objektumok. Az osztályok attribútumai lehetnek objektumok vagy függvények is.

## 3. fejezet

# Helló, Arroway!

### OO szemlélet

A polártranszformációs generátor egy széles körben elterjedt random generátor. Olyannyira elterjedt formája ez a random szám generálásnak, hogy a `Java.util.Random` osztály is ezt a módszert alkalmazza.

Íme a Java kód teljes egészében:

```
public class PolarGenerator {

    boolean nincsTarolt = true;
    double tarolt;

    public PolarGenerator() {
        nincsTarolt = true;
    }

    public double kovetkezo() {
        if(nincsTarolt) {
            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);

            tarolt = r*v2;
            nincsTarolt = !nincsTarolt;
        }
    }
}
```

```
        return r*v1;
    } else {
        nincsTarolt = !nincsTarolt;
        return tarolt;
    }
}

public static void main(String[] args) {

    PolarGenerator g = new PolarGenerator();

    for(int i=0; i<10; ++i)
        System.out.println(g.kovetkezo());
}
```

A következő sorokban részenként magyarázom el mi történik a kódban.

```
boolean nincsTarolt = true;
double tarolt;
```

A PolárGenerátor classban létrehozunk két változót. Az egyik boolean típusú, amely azt fogja megmondani, hogy éppen van-e tárolt értékünk. A másik maga a tárolt érték tartalmazza, majd amit korábban kiszámítottunk.

```
public PolarGenerator() {
    nincsTarolt = true;
}
```

Ez a class publikus konstruktora, amely a nincsTarolt-at igazra állítja. Erre azért van szükség, hogy tudjuk éppen van-e tárolt érték vagy nincs.

```
public double kovetkezo() {
    if(nincsTarolt) {
        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);

    tarolt = r*v2;
    nincsTarolt = !nincsTarolt;

    return r*v1;
} else {
    nincsTarolt = !nincsTarolt;
    return tarolt;
}
}
```

Ez a program lelke ahol a kovetkezo() metódus végzi a meghatározó számítást. Ha a nincsTarolt értéke igaz, akkor számol két random értéket. Az egyiket elmenti a tarolt változóba, a másikat pedig visszaadja. Amennyiben a nincsTarolt értéke hamis, akkor pedig a tárolt értéket fogja visszaadni.

```
public static void main(String[] args) {

    PolarGenerator g = new PolarGenerator();

    for(int i=0; i<10; ++i)
        System.out.println(g.kovetkezo());
}
```

A main metódusul indul el a program. Itt példányosítjuk a PolárGenerátor classot. Ezután egy for ciklussal 10-szer kiíratjuk a meghívott kovetkezo() metódus értékét. Minden második érték a tarolt változóból visszaszárt érték lesz. Ezeket az értékeket generálta nekem:



```
bunyi@bunyi: ~/Documents
bunyi@bunyi:~/Documents$ javac PolarGenerator.java
bunyi@bunyi:~/Documents$ java PolarGenerator
-0.5518103598184344
-0.9108738027998466
-1.9560282490949241
0.5048225085719978
1.9118928849455419
-0.48283343220250585
1.7238209676208334
0.8937594137943287
-1.3956361879015262
0.8904870819477649
bunyi@bunyi:~/Documents$
```

Az érdekesség az még az itt, hogy a Java fejlesztők egy nagyon hasonló módon oldották meg a `java.util.Random` osztályban a Random szám generálást. Íme:

```
public double nextDouble() {
    return (((long) (next(26)) << 27) + next(27)) * DOUBLE_UNIT;
}

private double nextNextGaussian;
private boolean haveNextNextGaussian = false;

synchronized public double nextGaussian() {
    // See Knuth, ACP, Section 3.4.1 Algorithm C.
    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;
    }
}
```

C++-ban így néz ki a teljes kód:

```
#include <iostream>
#include <tgmath.h>
#include <cstdlib>
```

```
#include <time.h>

using namespace std;

class PolarGenerator {
private:
    bool nincsTarolt;
    double tarolt;

public:
    PolarGenerator() {
        nincsTarolt = true;
        srand (time(NULL));
    }

    double 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) {
    PolarGenerator g;

    for (int i = 0; i < 10; ++i)
        cout << g.kovetkezo() << endl;
```

```
    return 0;
}
```

A következő sorokban részenként magyarázom el mi történik a kódban.

```
private:
    bool nincsTarolt;
    double tarolt;
```

Ez a Polárgenrátor class private része. Ez tartalmaz egy bool és egy double típusú változót. Ezek a változók csak az osztályon belül lesznek elérhetőek.

```
public:
    PolarGenerator() {
        nincsTarolt = true;
        srand (time(NULL));
    }

    double 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;
        }
    }
};
```

Ez pedig a class public része, amelyben a változók és metódusok példányosítás után elérhetőek az osztályon kívül is.

```
PolarGenerator() {  
    nincsTarolt = true;  
    srand (time(NULL));  
}
```

Az osztály nevével megegyező metódust konstruktornak nevezzük. Az ebben lévő kódok példányosításkor hajtódnak végre.

```
double 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;  
    }  
}
```

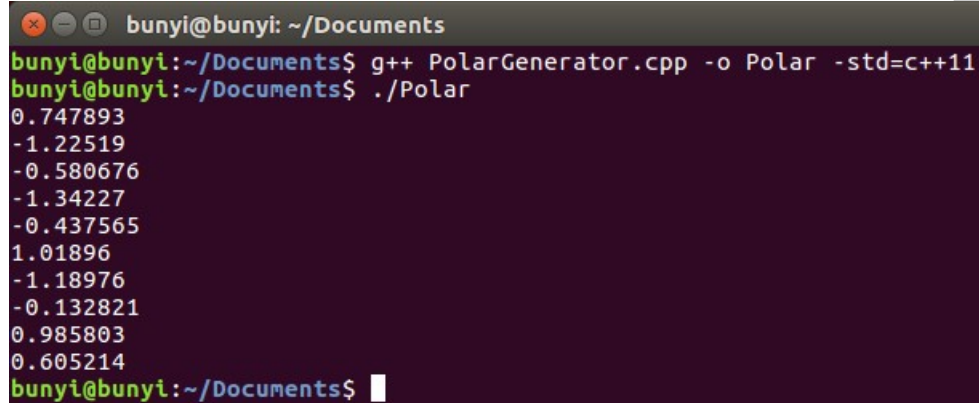
A kovetkezo metódusban szinte semmilyen lényegi eltérés nincs a Java kódhoz képest. Ugynaz történik ha nincsTarolt igaz akkor generál randomot, ha pedig hamis, akkor visszaadja az eltárolt értéket.

```
int main(int argc, char** argv) {  
    PolarGenerator g;  
  
    for (int i = 0; i < 10; ++i)  
        cout << g.kovetkezo() << endl;
```

```
    return 0;  
}
```

A main metódusban szintén, mint a Javanál példányosítunk és egy for ciklussal 10-szer visszadjuk a következő() metódus értékét.

C++-ban generált értékek:



```
bunyi@bunyi: ~/Documents  
bunyi@bunyi:~/Documents$ g++ PolarGenerator.cpp -o Polar -std=c++11  
bunyi@bunyi:~/Documents$ ./Polar  
0.747893  
-1.22519  
-0.580676  
-1.34227  
-0.437565  
1.01896  
-1.18976  
-0.132821  
0.985803  
0.605214  
bunyi@bunyi:~/Documents$
```

## Homokozó

valami

## "Gagyi"

valami

## Yoda

valami

## Kódolás from scratch

valami

# **III. rész**

## **Irodalomjegyzék**

DRAFT

## Általános

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

## C

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

## C++

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

## Lisp

[METAMATH] Chaitin, Gregory, *META MATH! The Quest for Omega*, [http://arxiv.org/PS\\_cache/math/pdf/0404/0404335v7.pdf](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.