

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> | <i>SIGNATURE</i> |
| WRITTEN BY | Bátfai, Norbert | 2019. március 20. | |

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. Tematikus feladatok | 3 |
| 2. Helló, Turing! | 5 |
| 2.1. Végtelen ciklus | 5 |
| 2.2. Lefagyott, nem fagyott, akkor most mi van? | 6 |
| 2.3. Változók értékének felcserélése | 8 |
| 2.4. Labdapattogás | 8 |
| 2.5. Szóhossz és a Linus Torvalds féle BogomIPS | 11 |
| 2.6. Helló, Google! | 12 |
| 2.7. 100 éves a Brun tétel | 14 |
| 2.8. A Monty Hall probléma | 14 |
| 3. Helló, Chomsky! | 15 |
| 3.1. Decimálisból unárisba átváltó Turing gép | 15 |
| 3.2. Az $a^n b^n c^n$ nyelv nem környezetfüggetlen | 15 |
| 3.3. Hivatkozási nyelv | 15 |
| 3.4. Saját lexikális elemző | 16 |
| 3.5. l33t.1 | 16 |
| 3.6. A források olvasása | 16 |
| 3.7. Logikus | 17 |
| 3.8. Deklaráció | 17 |

| | |
|--|-----------|
| 4. Helló, Caesar! | 19 |
| 4.1. int *** háromszögmátrix | 19 |
| 4.2. C EXOR titkosító | 19 |
| 4.3. Java EXOR titkosító | 19 |
| 4.4. C EXOR törő | 19 |
| 4.5. Neurális OR, AND és EXOR kapu | 20 |
| 4.6. Hiba-visszaterjesztéses perceptron | 20 |
| 5. Helló, Mandelbrot! | 21 |
| 5.1. A Mandelbrot halmaz | 21 |
| 5.2. A Mandelbrot halmaz a std::complex osztállyal | 21 |
| 5.3. Biomorfok | 21 |
| 5.4. A Mandelbrot halmaz CUDA megvalósítása | 21 |
| 5.5. Mandelbrot nagyító és utazó C++ nyelven | 21 |
| 5.6. Mandelbrot nagyító és utazó Java nyelven | 22 |
| 6. Helló, Welch! | 23 |
| 6.1. Első osztályom | 23 |
| 6.2. LZW | 23 |
| 6.3. Fabejárás | 23 |
| 6.4. Tag a gyökér | 23 |
| 6.5. Mutató a gyökér | 24 |
| 6.6. Mozgató szemantika | 24 |
| 7. Helló, Conway! | 25 |
| 7.1. Hangyaszimulációk | 25 |
| 7.2. Java életjáték | 25 |
| 7.3. Qt C++ életjáték | 25 |
| 7.4. BrainB Benchmark | 26 |
| 8. Helló, Schwarzenegger! | 27 |
| 8.1. Szoftmax Py MNIST | 27 |
| 8.2. Szoftmax R MNIST | 27 |
| 8.3. Mély MNIST | 27 |
| 8.4. Deep dream | 27 |
| 8.5. Robotpszichológia | 28 |

| | |
|--|---------------|
| 9. Helló, Chaitin! | 29 |
| 9.1. Iteratív és rekurzív faktoriális Lisp-ben | 29 |
| 9.2. Weizenbaum Eliza programja | 29 |
| 9.3. Gimp Scheme Script-fu: króm effekt | 29 |
| 9.4. Gimp Scheme Script-fu: név mandala | 29 |
| 9.5. Lambda | 30 |
| 9.6. Omega | 30 |
| III. Második felvonás | 31 |
| 10. Helló, Arroway! | 33 |
| 10.1. A BPP algoritmus Java megvalósítása | 33 |
| 10.2. Java osztályok a Pi-ben | 33 |
| IV. Irodalomjegyzék | 34 |
| 10.3. Általános | 35 |
| 10.4. C | 35 |
| 10.5. C++ | 35 |
| 10.6. Lisp | 35 |

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 alkálni 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 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

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

Tematikus feladatok

**Bátf41 Haxor Stream**

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

DRAFT

2. fejezet

Helló, Turing!

2.1. Végtelen ciklus

Írj olyan C végtelen ciklusokat, amelyek 0 illetve 100 százalékban dolgoztatnak egy magot és egy olyat, amely 100 százalékban minden magot!

Megoldás videó:

Végtelen ciklus 1 mag 100%

```
#include <unistd.h>

int main ()
{
    for (;;) {} //for ciklus kilépési feltétel nélkül

    return 0;
}
```

Végtelen ciklus minden mag 100%

```
#include <stdbool.h> //while ciklusban található true miatt kell

//gcc vegtelenossz.c -fopenmp -o vegossz

int main()
{
    #pragma omp parallel //Kiosszuk a feladatot a magok között
    while (true) //while ciklus kilépési feltétel nélkül
    {
        ;
    }
}
```

```
return 0;
}
```

```
}
```

Végtelen ciklus proci 0%

```
#include <unistd.h>
```

```
int main ()
```

```
{
```

```
    for (;;) //for ciklus kilépési feltétel nélkül
```

```
        usleep (1); //az unistd.h -ban található usleep parancs "alvóba" teszi ↔  
        a processzort
```

```
    return 0;
```

```
}
```

```
}
```

Megoldás forrása:

Mindhárom esetben végtelen ciklusokat alkalmaztunk, mely lehet for vagy while is. Az érdekesebb része a több magra való feladat kiosztás jelentette "#pragma omp paralell", ezentúl ezen parancs segítségével meggyorsíthatjuk a programok futási idejét.

2.2. Lefagyott, nem fagyott, akkor most mi van?

Mutasd meg, hogy nem lehet olyan programot írni, amely bármely más programról eldönti, hogy le fog-e fagyni vagy sem!

Megoldás videó:

Megoldás forrása: tegyük fel, hogy akkora haxorok vagyunk, hogy meg tudjuk írni a Lefagy függvényt, amely tetszőleges programról el tudja dönteni, hogy van-e benne végtelen ciklus:

Program T100

```
{
```

```
    boolean Lefagy(Program P)
```

```
    {
```

```
        if(P-ben van végtelen ciklus)
```

```
            return true;
```

```
        else
```

```
            return false;
```

```
    }
```

```
main(Input Q)
{
    Lefagy(Q)
}
```

A program futtatása, például akár az előző v. c ilyen pszeudókódjára:

```
T100(t.c.pseudo)
true
```

akár önmagára

```
T100(T100)
false
```

ezt a kimenetet adja.

A T100-as programot felhasználva készítsük most el az alábbi T1000-set, amelyben a Lefagy-ra építő Lefagy2 már nem tartalmaz feltételezett, csak konkrét kódot:

```
Program T1000
{

    boolean Lefagy(Program P)
    {
        if(P-ben van végtelen ciklus)
            return true;
        else
            return false;
    }

    boolean Lefagy2(Program P)
    {
        if(Lefagy(P))
            return true;
        else
            for(;;);
    }

    main(Input Q)
    {
        Lefagy2(Q)
    }

}
```

Mit for kiírni erre a T1000 (T1000) futtatásra?

- Ha T1000 lefagyó, akkor nem fog lefagyni, kiírja, hogy true

- Ha T1000 nem fagyó, akkor pedig le fog fagyni...

akkor most hogy fog működni? Sehogy, mert ilyen `LeFagy` függvényt, azaz a T100 program nem is létezik.

Tanulságok, tapasztalatok, magyarázat...

2.3. Változók értékének felcserélése

Írj olyan C programot, amely felcseréli két változó értékét, bármiféle logikai utasítás vagy kifejezés használata nélkül!

Megoldás videó: https://bhaxor.blog.hu/2018/08/28/10_begin_goto_20_avagy_elindulunk

Csere

```
#include <stdio.h>

int main()
{
    int a = 0;  /a inicializáció
    int b = 0;  /b inicializáció
    printf("Adja meg az a számot: ");    //kiaratás consolera printf függvény ←
        segítségével
    scanf("%d" , &a );                    //olvasás a consolebol a scanf ←
        fuggveny segitsegevel, "%d" = kapott ertek integer legyen,
    printf("Adja meg a b számot: ");
    scanf("%d" , &b);
    b = b-a;
    a = a+b;
    b = a-b;
    printf("a=%d%s",a,"\\n");    // "\\n" = sortörés
    printf("b=%d%s",b,"\\n");
}
```

Megoldás forrása:

Megcseréltünk két változó értékét bármiféle logikai utasítás vagy kifejezés nélkül, kipróbálhattuk a "printf" "scanf" függvényeket.

2.4. Labdapattogás

Először if-ekkel, majd bármiféle logikai utasítás vagy kifejezés használata nélkül írj egy olyan programot, ami egy labdát pattogtat a karakteres konzolon! (Hogy mit értek pattogtatás alatt, alább láthatod a videókon.)

Megoldás videó: <https://bhaxor.blog.hu/2018/08/28/labdapattogas>

Megoldás forrása:

Labdaif

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

//gcc labdaif.c -o labda -lncurses      -igy kell futtatni kozolban

int
main ( void )    //void = nem ad vissza értéket
{
    WINDOW *ablak;      //ablak -ra mutató pointer
    ablak = initscr ();  // inicializáljuk a curses ablakot

    int x = 0;
    int y = 0;

    int xnov = 1;
    int ynov = 1;

    int mx;
    int my;

    for ( ;; ) {

        getmaxyx ( ablak, my , mx );    //jelenlegi kurzor koordinátákat ↔
        beleteszi az my, mx változókba

        mvprintw ( y, x, "O" );          // kiírja a formáott outputot a ↔
        curses ablakba

        refresh ();
        usleep ( 100000 );    // ennyi mikrosecig alszik a programunk
        clear();              //letakarítja az ablakot

        x = x + xnov;          //minden ciklusban hozzáad 1-et "xnov, ynov" ↔
        hozzáad a kurzor koordinátaíhoz
        y = y + ynov;

        if ( x>=mx-1 ) { // elerte-e a jobb oldalt?
            xnov = xnov * -1;
        }
        if ( x<=0 ) { // elerte-e a bal oldalt?
            xnov = xnov * -1;
        }
        if ( y<=0 ) { // elerte-e a tetejet?
            ynov = ynov * -1;
        }
    }
}
```

```
    }
    if ( y>=my-1 ) { // elerte-e a aljat?
        ynov = ynov * -1;
    }

}

return 0;

}
```

Labdanoif

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

#define SZEL 78          //definialjuk SZEL, MAG konstansokat
#define MAG 22

int putX(int x,int y)    //putX függvény
{
    int i;

    for(i=0;i<x;i++)
        printf("\n");    //sortörés

    for(i=0;i<y;i++)
        printf(" ");      //üres karakter

    printf("X\n");        //X majd sortörés

    return 0;
}

int main()
{
    int x=0,y=0;

    while(1)              //végtelen ciklus
    {
        system("clear");  //ablak letakarítása
        putX(abs(MAG-(x++%(MAG*2))),abs(SZEL-(y++%(SZEL*2)))); //abs = ↔
        //abszolútérték Meghívjuk a putX függvényt
        usleep(50000);    //alvás
    }

    return 0;
}
```

```
}
```

Mindkét esetben a kurzort pattogtattunk. Az első esetben egyszerű if-es feltétellel amely azt nézte elérte-e a határokat a kurzorunk. Második esetben már if-es feltétel nélkül pattogtattuk a kurzort függvényt használva.

2.5. Szóhossz és a Linus Torvalds féle BogoMIPS

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

Megoldás videó:

Szohossz

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

void
delay (unsigned long long int loops)      //delay függvényünk
{
    unsigned long long int i;
    for (i = 0; i < loops; i++);
}

int
main (void)
{
    unsigned long long int loops_per_sec = 1;      //inicializáljuk
    unsigned long long int ticks;      //definiáljuk

    printf ("Calibrating delay loop..");      //printf függvény kiírás
    fflush (stdout);      //az OS bufferelése miatt kell, így fixen ←
        megjelenik a konzolon a printfünk

    while ((loops_per_sec <= 1))
    {
        ticks = clock ();      //visszadja a program kezdete óta eltelt ←
            tick -ek számát
        delay (loops_per_sec);      //delay függvény meghívása
        ticks = clock () - ticks;

        printf ("%llu %llu\n", ticks, loops_per_sec);

        if (ticks >= CLOCKS_PER_SEC)
        {
            loops_per_sec = (loops_per_sec / ticks) * CLOCKS_PER_SEC;

            printf ("ok - %llu.%02llu BogoMIPS\n", loops_per_sec / 500000,
```

```
(loops_per_sec / 5000) % 100);  
  
return 0;  
}  
}  
  
printf ("failed\n");  
return -1;  
}
```

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

2.6. Helló, Google!

Írj olyan C programot, amely egy 4 honlaptól álló hálózatra kiszámolja a négy lap Page-Rank értékét!

Pagerank

```
#include <stdio.h>  
#include <math.h>  
  
void  
kiir (double tomb[], int db)    //kiir függvényünk  
{  
    int i;  
    for (i=0; i<db; i++)  
        printf("PageRank [%d]: %lf\n", i, tomb[i]);  
}  
  
double tavolsag(double pagerank[], double pagerank_temp[], int db)    // ←  
    tavolsag függvényünk  
{  
    double tav = 0.0;  
    int i;  
    for (i=0; i<db; i++)  
    {  
        if ((pagerank[i] - pagerank_temp[i])<0)  
        {  
            tav += (-1*(pagerank[i] - pagerank_temp[i]));  
        }  
        else  
        {  
            tav += (pagerank[i] - pagerank_temp[i]);  
        }  
    }  
}
```

```
}
return tav;          //visszaadja a távolságot
}

int main(void)
{
double L[4][4] = {          //4x4 -es mátrix feltöltése
{0.0, 0.0, 1.0 / 3.0, 0.0},
{1.0, 1.0 / 2.0, 1.0 / 3.0, 1.0},
{0.0, 1.0 / 2.0, 0.0, 0.0},
{0.0, 0.0, 1.0 / 3.0, 0.0}
};

double PR[4] = {0.0, 0.0, 0.0, 0.0};          /x4 -es tömb feltöltése, ez ←
        fogja a végeredményt tárolni
double PRv[4] = {1.0 / 4.0, 1.0 / 4.0, 1.0 / 4.0, 1.0 / 4.0};    //Az ←
        oldalak jósága számositva

long int i,j,h;    //definiálunk
i=0; j=0; h=5;    //értéket adunk

for (;;)          //végtelen for ciklus
{
for(i=0;i<4;i++)
PR[i] = PRv[i];    //PR[i] feltöltése PRv[i] értékeivel
for (i=0;i<4;i++)
{
double temp=0;
for (j=0;j<4;j++)
temp+=L[i][j]*PR[j];    //temp=L[i][j]*PR[j]
PRv[i]=temp;
}

if ( tavolsag(PR,PRv, 4) < 0.00001)    //if feltétele tavolsag függvény
break;    //ha feltétel teljesül kilé a ciklusból
}
kiir (PR,4);    //meghivjuk a kiir függvényt
return 0;
}
```

Megoldás videó:

Megoldás forrása:

A Google ezzel az algoritmussal futott be a piacra. Ez az algoritmus kiszámolja az oldalak jóságát, ez alapján helyezi sorrendbe őket. Olyan algoritmus mely a hiperlinkekkel összekötött oldalakhoz hozzárendel egy számot, ezt a számot befolyásolja, hogy hány db oldal mutat az adott oldalra és figyelembe veszi az adott oldalra mutató oldalak jóságát is.

2.7. 100 éves a Brun tétel

Írj R szimulációt a Brun tétel demonstrálására!

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

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

2.8. A Monty Hall probléma

Írj R szimulációt a Monty Hall problémára!

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

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

Tanulságok, tapasztalatok, magyarázat...

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet grájával megadva írd meg ezt a gépet!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.2. Az $a^n b^n c^n$ nyelv nem környezetfüggetlen

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

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.3. Hivatkozási nyelv

A [KERNIGHANRITCHIE] könyv C referencia-kézikönyv/Utasítások melléklete alapján definiáld BNF-ben a C utasítás fogalmát! Majd mutass be olyan kódcsipeteket, amelyek adott szabvánnyal nem fordulnak (például C89), mással (például C99) igen.

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.4. Saját lexikális elemző

Írj olyan programot, ami számolja a bemenetén megjelenő valós számokat! Nem elfogadható olyan megoldás, amely maga olvassa betűnként a bemenetet, a feladat lényege, hogy lexert használjunk, azaz óriások vállán álljunk és ne kispályázzunk!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.5. l33t.l

Lexelj össze egy l33t ciphert!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

3.6. A források olvasása

Hogyan olvasod, hogyan értelmezed természetes nyelven az alábbi kódcsipeteket? Például

```
if(signal(SIGINT, jelkezelő)==SIG_IGN)
    signal(SIGINT, SIG_IGN);
```

Ha a SIGINT jel kezelése figyelmen kívül volt hagyva, akkor ezen túl is legyen figyelmen kívül hagyva, ha nem volt figyelmen kívül hagyva, akkor a jelkezelő függvény kezelje. (Miután a **man 7 signal** lapon megismertem a SIGINT jelet, a **man 2 signal** lapon pedig a használt rendszerhívást.)



Bugok

Vigyázz, sok csipet kerülendő, mert bugokat visz a kódba! Melyek ezek és miért? Ha nem meggyránzésre, elkapja valamelyiket esetleg a splint vagy a frama?

i.

```
if(signal(SIGINT, SIG_IGN)!=SIG_IGN)
    signal(SIGINT, jelkezelő);
```

ii.

```
for(i=0; i<5; ++i)
```

iii.

```
for(i=0; i<5; i++)
```

- iv. `for(i=0; i<5; tomb[i] = i++)`
- v. `for(i=0; i<n && (*d++ = *s++); ++i)`
- vi. `printf("%d %d", f(a, ++a), f(++a, a));`
- vii. `printf("%d %d", f(a), a);`
- viii. `printf("%d %d", f(&a), a);`

Megoldás forrása:

Megoldás videó:

Tanulságok, tapasztalatok, magyarázat...

3.7. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

`$(\texttt{\textbackslash forall } x \texttt{\textbackslash exists } y ((x<y)\texttt{\textbackslash wedge}(y \texttt{\textbackslash text{ prím}})))$`

`$(\texttt{\textbackslash forall } x \texttt{\textbackslash exists } y ((x<y)\texttt{\textbackslash wedge}(y \texttt{\textbackslash text{ prím}})\texttt{\textbackslash wedge}(SSy \texttt{\textbackslash text{ prím}})) \leftrightarrow)$`

`$(\texttt{\textbackslash exists } y \texttt{\textbackslash forall } x (x \texttt{\textbackslash text{ prím}}) \texttt{\textbackslash supset } (x<y))$`

`$(\texttt{\textbackslash exists } y \texttt{\textbackslash forall } x (y<x) \texttt{\textbackslash supset } \texttt{\textbackslash neg } (x \texttt{\textbackslash text{ prím}}))$`

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

Megoldás videó: <https://youtu.be/ZexiPy3ZxsA>, https://youtu.be/AJSXOQFF_wk

Tanulságok, tapasztalatok, magyarázat...

3.8. Deklaráció

Vezesd be egy programba (forduljon le) a következőket:

- egész
- egészre mutató mutató
- egész referenciája

- egészek tömbje
- egészek tömbjének referenciája (nem az első elemé)
- egészre mutató mutatók tömbje
- egészre mutató mutatót visszaadó függvény
- egészre mutató mutatót visszaadó függvényre mutató mutató
- egészet visszaadó és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó függvény
- függvényt mutató egy egészet visszaadó és két egészet kapó függvényre mutató mutatót visszaadó, egészet kapó függvényre

Mit vezetnek be a programba a következő nevek?

- `int a;`
- `int *b = &a;`
- `int &r = a;`
- `int c[5];`
- `int (&tr)[5] = c;`
- `int *d[5];`
- `int *h ();`
- `int *(*l) ();`
- `int (*v (int c)) (int a, int b)`
- `int ((*z) (int)) (int, int);`

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4. fejezet

Helló, Caesar!

4.1. int *** háromszögmátrix

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4.4. C EXOR törő

Írj egy olyan C programot, amely megtöri az első feladatban előállított titkos szövegeket!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

4.5. Neurális OR, AND és EXOR kapu

R

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

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

Tanulságok, tapasztalatok, magyarázat...

4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

5. fejezet

Helló, Mandelbrot!

5.1. A Mandelbrot halmaz

Megoldás videó:

Megoldás forrása:

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

Megoldás videó:

Megoldás forrása:

5.3. Biomorfok

Megoldás videó: <https://youtu.be/IJMbgRzY76E>

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

Tanulságok, tapasztalatok, magyarázat...

5.4. A Mandelbrot halmaz CUDA megvalósítása

Megoldás videó:

Megoldás forrása:

5.5. Mandelbrot nagyító és utazó C++ nyelven

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

Megoldás forrása:

Megoldás videó:

Megoldás forrása:

5.6. Mandelbrot nagyító és utazó Java nyelven

DRAFT

6. fejezet

Helló, Welch!

6.1. Első osztályom

Valósítsd meg C++-ban és Java-ban az módosított polártranszformációs algoritmust! A matek háttér teljesen irreleváns, csak annyiban érdekes, hogy az algoritmus egy számítása során két normálist számol ki, az egyiket elspájzold és egy további logikai taggal az osztályban jelzed, hogy van vagy nincs eltérve kiszámolt szám.

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat... térj ki arra is, hogy a JDK forrásaiban a Sun programozói pont úgy csinálták meg ahogyan te is, azaz az OO nemhogy nem nehéz, hanem éppen természetes neked!

6.2. LZW

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

Megoldás videó:

Megoldás forrása:

6.3. Fabejárás

Járd be az előző (inorder bejárású) fát pre- és posztorder is!

Megoldás videó:

Megoldás forrása:

6.4. Tag a gyökér

Az LZW algoritmust ültesd át egy C++ osztályba, legyen egy Tree és egy beágyazott Node osztálya. A gyökér csomópont legyen kompozícióban a fával!

Megoldás videó:

Megoldás forrása:

6.5. Mutató a gyökér

Írd át az előző forrást, hogy a gyökér csomópont ne kompozícióban, csak aggregációban legyen a fával!

Megoldás videó:

Megoldás forrása:

6.6. Mozgató szemantika

Írj az előző programhoz mozgató konstruktort és értékadást, a mozgató konstruktor legyen a mozgató értékadásra alapozva!

Megoldás videó:

Megoldás forrása:

7. fejezet

Helló, Conway!

7.1. Hangyaszimulációk

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

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

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7.2. Java életjáték

Írd meg Java-ban a John Horton Conway-féle életjátékot, valósítsa meg a sikló-kilövőt!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7.3. Qt C++ életjáték

Most Qt C++-ban!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

7.4. BrainB Benchmark

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

DRAFT

8. fejezet

Helló, Schwarzenegger!

8.1. Szoftmax Py MNIST

aa Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.2. Szoftmax R MNIST

R

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.3. Mély MNIST

Python

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.4. Deep dream

Keras

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

8.5. Robotpszichológia

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

DRAFT

9. fejezet

Helló, Chaitin!

9.1. Iteratív és rekurzív faktoriális Lisp-ben

Megoldás videó:

Megoldás forrása:

9.2. Weizenbaum Eliza programja

Éleszd fel Weizenbaum Eliza programját!

Megoldás videó:

Megoldás forrása:

9.3. Gimp Scheme Script-fu: króm effekt

Írj olyan script-fu kiterjesztést a GIMP programhoz, amely megvalósítja a króm effektet egy bemenő szövegre!

Megoldás videó: https://youtu.be/OKdAkI_c7Sc

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

Tanulságok, tapasztalatok, magyarázat...

9.4. Gimp Scheme Script-fu: név mandala

Írj olyan script-fu kiterjesztést a GIMP programhoz, amely név-mandalát készít a bemenő szövegből!

Megoldás videó: https://bhaxor.blog.hu/2019/01/10/a_gimp_lisp_hackelese_a_scheme_programozasi_nyelv

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

Tanulságok, tapasztalatok, magyarázat...

9.5. Lambda

Hasonlítsd össze a következő programokat!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

9.6. Omega

Megoldás videó:

Megoldás forrása:

DRAFT

III. rész

Második felvonás

DRAFT

**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

10. fejezet

Helló, Arroway!

10.1. A BPP algoritmus Java megvalósítása

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

10.2. Java osztályok a Pi-ben

Az előző feladat kódját fejleszd tovább: vizsgáld, hogy Vannak-e Java osztályok a Pi hexadecimális kifejtésében!

Megoldás videó:

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

IV. rész

Irodalomjegyzék

10.3. Általános

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

10.4. C

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

10.5. C++

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

10.6. 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.