Univerzális programozás

Írd meg a saját programozás tankönyvedet!

Ed. BHAX, DEBRECEN,
2019. március 18., v. 0.1.2

COLLABORATORS			
	TITLE : Univerzális programozás		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Bátfai, Norbert Ã©s Pankotai, Kristóf	2019. március 18.	

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
0.0.5	2019-03-01	Munka elkezdése, adatok bevitele a szerzőkhöz. Touring csokor első 3 feladatának megoldása. Saját repository-ba való feltöltés.	pkristof1999
0.0.6	2019-03-04	Touring csokor következő 4 feladatának megoldása, első 3 finomítása, utolsó megnézése.	pkristof1999

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
0.0.7	2019-03-05	Touring csokor befejezése. Chomsky csokor feladatainak tanulmányozása.	pkristof1999
0.0.8	2019-03-08	Chomsky csokor kidolgozásának megkezdése.	pkristof1999
0.0.9	2019-03-11	Chomsky csokor kidolgozásának felfüggesztése/befejezése.	pkristof1999
0.1.0	2019-03-15	Caesar csokor elkezdése.	pkristof1999
0.1.1	2019-03-17	Caesar csokor feladatainak további tanulmányozása, megoldása.	pkristof1999
0.1.2	2019-03-18	Ceaser csokor befejezése legjobb tudás szerint.	pkristof1999

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

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

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.

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 forrása:

Minden szál 100%-on

Egy szál 100%-on

Egy szál altatása

Tanulságok, tapasztalatok, magyarázat...

Egy mag futtatása 100%-on egy egyszerű üres ciklussal megtehető. Több magnál a végtelen cikluson felül szükség van az OpenMP meghívására és fordításnál pedig az "-fopenmp" parancsra. Altatásnál ciklusban a Sleep() függvényre van szükség, amihez meg kell hívni az unistd.h könyvtárat.

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 forrása: tegyük fel, hogy akkora haxorok vagyunk, hogy meg tudjuk írni a `Lefagy` függvényt, amely tetszőleges programról el tudja dönteni, hogy van-e benne vlgtelen 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 épőlő Lefagy2 már nem tartalmaz feltételezett, csak 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...

Elkészíteni egy olyan programot, amely meg tudja nézni, hogy egy másik program lefagy-e (végtelen-e) nem lehetséges, mivel olyan kód nem létezik, amely egy másik program végtelen ciklusát "megelőzve" annak a végére érhetne, és valósan kimondhatná, hogy igen, ez egy olyan program, ami lefagyott (végtelen ciklus).

2.3. Változók értékének felcserélése

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

Megoldás forrása:

Változók felcserélése

Tanulságok, tapasztalatok, magyarázat...

Változók cseréjénél a felhasználótól 2 darab egész számot bekérünk, majd egy egyszerű összeadás/kivonás sorozattal ezeket megcseréljük, majd kiíratjuk.

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 forrása:

Labdapattogtatás if-fel

Labdapattogtatás if nélkül

Tanulságok, tapasztalatok, magyarázat...

Az if-es labdapattogtatásnál szükségünk van egy új ablak megnyitásához, az ablak méretének lekérdezéséhez, ahhoz hogy hol jár épp a labda és egy kezdőértékhez. A For cikluson belül valósul meg, hogy a program a "getmaxyx" segítségével lekérdezi az ablak méretét, az "mvprintw" függvény kirajzolja a labdát, majd az if elágazásoknál nézi a program, hogy mennyit kell adni a változókhoz. Az if nélküli labdapattogtatásnál előre megadott pályamérettel dolgozunk, tömbök és for ciklusok segítségével nézzük, hogy mikor éri el a labda a pálya szélét, és függően attól, hogy melyik oldalt éri el, úgy vált x vagy y plusz előjelről negatívra, amellyel az irányváltoztatás létrejön.

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 forrása:

Szóhossz

Tanulságok, tapasztalatok, magyarázat...

(Ratku Dániel segítségével) Ebben a feladatban az int típusnak a Bitwise operátorral megkapott értékét kell kiszámolnunk. Itt két különböző int-tel dolgozunk, az egyik értéke 0, a másik pedig 0x01, egy do-while ciklus segítségével a 0 értékű változót folyamatosan növeljük eggyel, egészen addig, amíg el nem érjük a 0x01 változót left shifteléssel.

2.6. Helló, Google!

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

Megoldás forrása: PageRank

Tanulságok, tapasztalatok, magyarázat...

A PageRank nevezetű algoritmus a Google-höz kapcsolódik. Célja az, hogy azok a weboldalak, amelyekre több kattintás gyűlik össze, előrébb kerüljenek a ranglistán arra alapozva, hogy valószínűleg érdekesebb és relevánsabb tartalom van bennük, tehát a weboldalak "jóságát" veszi figyelembe. A program egy egyszerű 4 weboldalas esetet mutat be mátrixokkal és vektorokkal, habár még teljes mértékben nem látom át, ezért még nem tudom részletezni a kód jelentését.

2.7. 100 éves a Brun tétel

Írj R szimulációt a Brun tétel demonstrálására!

Megoldás forrása: Brun Tétel

Tanulságok, tapasztalatok, magyarázat...

(Ratku Dániel segítségével) A Brun tétel azt mondja ki, hogy az olyan egymást követő prímek, amelyek különbsége kettő, azoknak a reciprokösszege nem a végtelenbe, hanem egy ún. Brun konstanshoz tart. A programon belül deklaráljuk x-ig a prímszámokat 1-től, illetve 2-től is. Külön felvesszük azt a változót, amely azokat a prímeket gyűjti ki, amelyeknek a különbsége kettő, majd ezeket eltároljuk. Az eltárolt prímeknek vesszük a reciprokösszegét, majd az összeset összeadjuk egy változóba. Majd a matlab könyvtár segítségével ezt meg is jeleníthetjük.

2.8. A Monty Hall probléma

Írj R szimulációt a Monty Hall problémára!

Megoldás forrása: Monty Hall probléma

Tanulságok, tapasztalatok, magyarázat...

(Ratku Dániel segítségével) A feladat szituációja a következő, adott egy játékos és egy műsorvezető, plusz 3 ajtó, amelyből 2 vereséget és 1 nyereséget rejt. Ezek a "jutalmak" ajtók mögé vannak rejtve és a felhasználónak kell egyet kiválasztania a lehetőségek közül. A feladat során feltételezzük, hogy a műsorvezető pontosan tudja a győztes ajtó számát. A műsornak a felépítése olyan, hogy a játékosnak választania kell egy ajtót, majd a műsort levezető személynek választania kell a választott ajtótól egy különbözőt, amely nem tartalmazza a jutalmat. A probléma abban rejlik, hogy eredetileg 1/3-ad az esély a győzelemre, viszont miután a műsorvezető kinyitja az általunk választottól különböző ajtót, azt feltételezzük, hogy azért azt választotta, mert nem abban van a nyeremény, ezáltal pedig a győzelmi esélyünk már 2/3-ra nőtt. A kódunkban bármekkora számmal hozhatjuk létre a kísérletek számát, majd ezután deklaráljuk a kísérletek és játékosok nevezetű változót, amelyben ott van a lehetőségek száma és mellékeljük a replace=T -t, ami engedélyezi ezeknek az ismétlődését. A műsorvezető egy vektor lesz, aminek a hossza megegyezik a kísérletek számával, a műsor folyását egy for ciklussal kezdjük, ami átmegy az összes kísérleten, majd a beleépített if elágazásokkal megvizsgáljuk, hogy mit választott a játékos. Az eredeti ágnál megegyezik a játékos tippje a helyes ajtóval, ekkor a műsorvezető kivonja a 3 lehetőség közül azt amelyiket választott a játékos, egyéb esetben a házigazda már csak azt az ajtót tudja kiválasztani, ami mögött nem a jutalom van. Következő lépésben meghatározunk 2 esetet, az egyik amikor a játékos változtat, a másik amikor nem. Az utóbbi esetén a tipp megegyezik a győzelemmel. A következő for ciklus a változtatás esetén zajlik, amikor már arra az ajtóra váltunk, ami mögött a problémából adódóan feltételezzük a jutalom hollétét. Legvégül pedig kiíratjuk a kísérletek számát.

3. fejezet

Helló, Chomsky!

3.1. Decimálisból unárisba átváltó Turing gép

Állapotátmenet gráfjával megadva írd meg ezt a gépet!

Megoldás forrása:

Unárisba váltó

Tanulságok, tapasztalatok, magyarázat...

A program lényege, hogy decimálisból unárisba, azaz egyes számrendszerbe váltson át pozitív egész számokat. A program egy egyszerű for ciklus, amely a megadott számig vonásokat húz, amelyet minden 5. vonás után szóközzel választ el.

3.2. Az $a^nb^nc^n$ nyelv nem környezetfüggetlen

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

Megoldás forrása:

Tanulságok, tapasztalatok, magyarázat...

A feladat nem kristály tiszta, még megoldás/kigondolás alatt áll, később, ha megértem a lényegét akkor visszatérek rá.

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.

Tanulságok, tapasztalatok, magyarázat...

A C nyelv C99-es kiadása a C89-hez képest sok újítást hozott, viszont az egyik legfontosabb, illetve legegyszerűbb újítás az, hogy C89-hez képest a C99-es nyelvben lehet C++ típusú kommentelést használni.

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 forrása:

Lexikális elemző

Tanulságok, tapasztalatok, magyarázat...

Ez egy olyan program, amelyet lefuttatva lex-el készít nekünk egy új c forrást, amely bármely karaktersor közül képes felismerni a valós számokat.

3.5. Leetspeak

Lexelj össze egy l33t ciphert!

Megoldás forrása:

Leetspeak

Tanulságok, tapasztalatok, magyarázat...

Ez a program l nyelvből készít egy c forrást, amely felismeri a karaktereket és kódolja őket más hasonlóakkal. Az l kód lényegében annyit csinál, hogy minden betűhöz és számhoz hozzárendel egy 4 karaktert tartalmazó tömböt, amelyekből betűnként véletlenszerűen választ majd cseréli ki.

3.6. A források olvasása

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

```
if(signal(SIGINT, jelkezelo)==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 jelkezelo 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 megy ránézésre, elkapja valamelyiket esetleg a splint vagy a frama?

i.
```
if(signal(SIGINT, SIG_IGN)!=SIG_IGN)
    signal(SIGINT, jelkezelo);
```

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);
```

Tanulságok, tapasztalatok, magyarázat...

Első: Ha eddig nem volt figyelmen kívül hagyva a SIGINT, akkor a jelkezelő függvény kezelje, ha figyelmen kívül hagyva, akkor maradjon is így.

Második: Egyszerű for ciklus, az i-t 0-tól kezdve addig növelje 1-el amíg el nem éri a 4-et.

Harmadik: Mivel for ciklusban nem számít, hogy i++ vagy ++i, így ez megegyezik az előzővel.

Az ezt követő algoritmusokat nem igazán értem, de a félév további részében, amint megértem visszatérek és kiegészítem.

3.7. Logikus

Hogyan olvasod természetes nyelven az alábbi Ar nyelvű formulákat?

```
$(\forall x \exists y ((x<y)\wedge(y \text{ prím})))$
$(\forall x \exists y ((x<y)\wedge(y \text{ prím})\wedge(SSy \text{ prím})) ↩
   )$
$(\exists y \forall x (x \text{ prím}) \supset (x<y)) $
$(\exists y \forall x (y<x) \supset \neg (x \text{ prím}))$
```

Megoldás forrása:

Logikus

Tanulságok, tapasztalatok, magyarázat...

Ez a feladat több elsőrendű logikai állítást fogalmaz meg ar nyelven, melyekből először vázoljuk, hogy melyik kifejezés mit jelent. Két univerzális kvantor van: az "exist" az azt jelenti, hogy létezik olyan..., a "forall" pedig, hogy bármely...; Továbbá a "neg" a negációt, a "supset" a konjunkciót és a "wedge" pedig az implikációt jelöli.

Az első kifejezés: Bármely x esetén létezik olyan y, ahol ha x kisebb, akkor y prímszám.

Második kifejezés: Bármely x esetén létezik olyan y, ahol ha x kisebb, akkor y prímszám és ha y prímszám, akkor az azt követő utáni szám is prím.

Harmadik kifejezés: Van olyan y, ahol bármely x esetén az x prím és kisebb, mint y.

Negyedik kifejezés: Van olyan y, ahol bármely x-nél az x nagyobb, mint y és x nem prím.

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
- függvénymutató 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 forrása:

Deklaráció

Tanulságok, tapasztalatok, magyarázat...

Nem teljesen látom át a programot, bár a kódot megkaptam és a későbbiekben ezt még kidolgozom segítséggel.

4. fejezet

Helló, Caesar!

4.1. double ** háromszögmátrix

Írj egy olyan `malloc` és `free` párost használó C programot, amely helyet foglal egy alsó háromszög mátrixnak a szabad tárban!

Megoldás forrása:

Háromszögmátrix

Tanulságok, tapasztalatok, magyarázat...

A program egy olyan négyzetes mátrix (négyzetes mátrix: sorainak és oszlopainak a száma megegyezik) alsó háromszögét számítja ki, melynek a főátló felett kizárólag nulla áll. A program a memóriában 40 bájtot foglal le a malloc fügvény segítségével.

4.2. C EXOR titkosító

Írj egy EXOR titkosítót C-ben!

Megoldás forrása:

C EXOR titkosító

Forrás (SourceForge, UDPROG)

Tanulságok, tapasztalatok, magyarázat...

Ez a titkosító program az EXOR-ra, azaz a kizáró vagyra épít. A program lényege, hogy egy felhasználó által megadott 8 karakteres int kulcs segítségével (azért 8, mert a későbbiekben a visszafordító program 8 karakteres kulccsal dolgozik) egy szöveget lekódoljunk, amit majd a későbbiekben ugyanazzal a kóddal dekódolni is tudunk. A program két konstans értékkel dolgozik: a kulcs és a buffer maximális méretével, amit a program elején kell definiálni. A program fő ágában deklaráljuk magát a kulcsot és a buffert, és azokat a változókat, amelyek tárolják a kulcs hosszúságát és a beolvasott bájtok mennyiségét. A továbbiakban pedig rögzítjük a felhasználó által megadott kulcsnak a méretét, ez lesz az első parancssori argumentum is. Majd egy while ciklus segítségével megyünk végig a beolvasott bájtokon és EXOR séma segítségével titkosítjuk őket. Lefuttatni pedig a következővel tudjuk: először is a forrást "gcc eClean.c -o ec"-vel fordítjuk majd a következőkben "./ec "kulcs" <tisztaszöveg.txt >titkosszöveg.txt".

4.3. Java EXOR titkosító

Írj egy EXOR titkosítót Java-ban!

Megoldás forrása:

Java EXOR titkosító

Forrás: www.tankonyvtar.hu

Tanulságok, tapasztalatok, magyarázat...

Ez az EXOR törő a fentebb megoldot feladat java-ban megírt megfelelője. Működése elvben és gyakorlatban is megegyezik. Lefordítása pedig a következő: "javac -encoding UTF-8 ExorTitkosító.java" majd futtatása pedig: "java ExorTitkosító "kulcs" < titkosított.txt".

4.4. C EXOR törő

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

Megoldás forrása:

C EXOR törő

Forrás (SourceForge, UDPROG)

Tanulságok, tapasztalatok, magyarázat...

Ez a program a fent EXOR-ral lekódolt szöveget kódolja vissza a megfelelő 8 int hosszúságú kulcs segítségével. Ebben a programban is definiáljuk, hogy mennyi konstansokkal dolgozzon. Kezdetnek az első for ciklusnál kiszámoljuk, hogy mennyi az átlagos szóhossz szóközök segítségével, ami azért fontos, hogy könnyebben válassza el a szavakat. Ezután jöhet az EXOR végrehajtása bájtonként. Amíg van karakter a szöveges fájlban, addig fut a program, majd amint elfogynak kiürítjük a buffert is. Majd a for ciklusokkal minden lehetséges kulcsot előállítunk. Lefuttatni pedig a következővel tudjuk: először is a forrást "gcc tClean.c -o tc"-vel fordítjuk majd a következőkben "./tc "kulcs" <titkosszöveg.txt >tisztaszöveg.txt".

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...

A programot még nem látom át/nem értem teljesen de dolgozom a megoldásán.

4.6. Hiba-visszaterjesztéses perceptron

C++

Megoldás videó: https://youtu.be/XpBnR31BRJY

Megoldás forrása: https://github.com/nbatfai/nahshon/blob/master/ql.hpp#L64

Tanulságok, tapasztalatok, magyarázat...

A programot még nem látom át/nem értem teljesen de dolgozom a megoldásán.

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 iteréció bejárta z_n komplex számokat!

Megoldás forrása:

Megoldás videó:

Megoldás forrása:

5.6. Mandelbrot nagyító és utazó Java nyelven

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 elspájzolod és egy további logikai taggal az osztályban jelzed, hogy van vagy nincs eltéve 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...

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...

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:

III. 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.

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. Ã©s Ritchie, Dennis M., *A C programozási nyelv*, Bp., Műszaki, 1993.

10.5. C++

[BMECPP] Benedek, Zoltán Ã©s 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.