Haladó numerikus módszerek

1. gyakorlat

Tudnivalók

- Oktatók:
  - Kormányos Andor
  - Kaufmann Zoltán
  - Biricz András
  - Kunsági-Máté Sándor
  - Udvarnoki Zoltán

- Oktatás célja: A C programozási nyelv elsajátításával együtt megismerkedünk a fizikában, vagy a fizikában is használatos néhány numerikus módszerrel. Ezekre a fiznum1ben előre megírt függvényeket, operátorokat használtunk (pl. faktoriális, maximum keresés, sorrendezés, hisztogram készítés, mátrixszorzás, egyenletrendszer-megoldás, differenciál-egyenlet megoldása). Most megtanuljuk, hogy e függvényekbe rejtett algoritmusok hogyan működnek, és hogy hogyan kell hasonló bonyolultságú programot írni C-ben. A C előnyei leginkább menetközben lesznek világosak (optimális memóriaszervezés és a szigorúbb, alapszintűbb szintaxis révén gyorsabb működés) Ezek fontosak nagy mennyiségű adat feldolgozása esetén, ill. alapszintű program esetén, mint pl. egy operációs rendszer.
- Weboldalak
  - Elméleti weboldal: http://akormanyos.web.elte.hu (http://akormanyos.web.elte.hu) -> teaching -> progalap és halnum
  - (közvetlen link: http://akormanyos.web.elte.hu/teaching/progalap17va/progalap_and_halnum_2020.html (http://akormanyos.web.elte.hu/teaching/progalap17va/progalap_and_halnum_2020.html))
  - Innen van link a gyakorlat oldalára.
- Az elméleti oldalon vannak a progalap előadás fóliák, amik egyben a gyakorlat elméleti hátterének is használhatók. (meg is hallgathatók az előadás idejében) "00 Intro" mindjárt a tudnivalókat írja le.
- Teljesítés feltételei
  - 5 beadandó feladat megoldása. ezekből lesz alapszint és teljes szint. Legalább alapszinten kötelező mindet megoldani.
  - Beadás a kooplex megfelelő direktorijába másolással történik a megadott határidőig.
  - Jelenlét a gyakorlatokon kötelező.

Gyakorlat menete:

A gyakorlatok elején közös meetingben ismertetőt mondunk a tudnivalókról és a megoldandó feladatokhoz szükséges alapokat példákon mutatjuk be. Ez az első gyakokon több, azután kevesebb lesz, lehet hogy is nem mindig lesz. Utána önálló munka következik konzultációkkal. Konzultációt a hallgató is kezdeményezheti a konzultációs csatornába írva, elsősorban a kijelölt oktatónál, és megpróbáljuk aszerinti sorrendben hívni. Az oktató a gyakorlat oldalon a ponttáblázatban lesz kijelölve. Aki nem jelentkezik, annak az oktató keresésére kell válaszolnia, szükség esetén megmutatnia, hogy hol tart. Sokszor észreveszünk hibát, ha a hallgató nem is akart kérdezni. A konzultáció alapján kapja meg a hallgató a jelenléti OK-t.

Beadás utáni héten a konzultáció elsősorban a beadott megoldás értékeléséről szól, ekkor hasznos tanácsokat is próbálunk adni. Azon kívül pedig az új feladatról is lehet kérdezni.

Motiváció

Javasolt az elméleti oldalról a "01 Motiváció" c. rövid ismertetőt átolvasni.

Gyakorlati munka

Codeblocks kezelése:

- CodeBlocks elindítása
- A c programot a Codebloksban nem önálló fájlként, hanem egy projektfolder részeként kell létrehozni. Ennek lépései:
  - Create a new project (a nagy mezőben található linkkel, vagy a File menűből a New -> Projekt pontban
  - Console application kiválasztása 2x kattintással (ettől jelenik meg egy konzol ablak majd a program futtatásakor, amibe az ír és amin keresztül bekérhet adatokat)
  - language: C (nem C++)
  - Project title: pl. minta1. Utána a Foldert is meg kell adni, de a folder és a projekt neve ne tartalmazzon különleges karatereket (space, ékezetes betűk, írásjelek egy részétől megbolondul).
  - Compilert és beállításait ált. nem kell változtatni (gcc).
  - Ekkor egy folderpath/projectname folder jön létre és abban lesz egy main.c és egy projectname.cbp nevű projekt-definiáló file.
- fordítás
  - a sárga fogaskerékkel, ezt egyből ki is próbálhatjuk az alapmintaként kapott programon.
- futtatás
  - a zöld lejátszás gombbal, próbáljuk is ki!
  - fordítás és futtatás együtt a kombinált ikonnal
- ugyanez linux parancsablakban: gcc minta1.c -o minta1
- mentés, kilépés
  - A program mentése automatikusan megtörténik már a fordításkor. Kilépéskor rákérdez az állapotjelzések mentésére, amit érdemes elfogadni.
- program, projekt betöltése: vigyázzunk, hogy sose fájlként töltsük be a programot, hanem projektként! Vagy a codeblocksban a nagy mezőben (legörgetve látható) "Recent projects" listából, vagy a File menű "Recent projects" listájából, esetleg az "Open existing project" menűpontból a .cbp fájlra kattintva. vagy egy fájlkezelőben a cbp-re kattintva (ez elindítja a CodeBlocksot a projekttel)

Alapszintű programok

In [ ]:
```
// Legegyszerűbb programok
// ld. az elméleti weboldal "10, Első C program" c. dokumentumban is

- a legrövidebb szintaktikailag helyes program lényegében 2 soros:

int main()
{   return 0; }
```

In [ ]:
```
// "Hello world!" program (ezt kaptuk készen a CodeBlocksban)
// Itt magyarázatokkal bővítve látható:

#include <stdio.h>   // Ebben van a printf is. Csomagbetöltés a pythonban is
#include <stdlib.h>  // volt, de C-ben az alapvetőbb függvényekhez is kell.
                     // Láthatóan így lehet egysoros kommentet csinálni,
                     // az ilyen komment a sor végéig tart.
                     /* Ez is komment, ez a bezáró jelig tart,
                        és többsoros is lehet. */

int main()           // a főprogram is egy függvény
{
    printf("Hello world!\n");  // Ez magától értetődő, később magyarázzuk
    return 0;
}
```

In [ ]:
```
// változótípusok, egész és nemegész számok kiírása

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i,k;
    double g;

    printf("Hello world!\n");
    i=3;
    g=9.81;
    printf("%d %f\n",i,g);

    return 0;
}

Így bőbeszédűbb a kiírás:

    printf("egész számunk %d  g pedig %f\n",i,g);

Tehát a printf valójában az "" jelek közti részt, a format stringet írja ki,
behelyettesítve a %d és %f helyére az i és g értékét.
Egész típusú változó, kifejezés esetén %d használandó (decimális kiírás),
float, vagy double típus esetén a %f.

Hasonlítsuk össze a pythonnal:
  - C-ben a program fő részét is függvénybe kell írni,
  - pozicionálás helyett {} adja meg a fv. környezetet
  - utasítások után mindig kell ;
  - már az alap függvények használatához is szükséges csomagbeöltés
  - printf-be kell format string
```

In [ ]:
```
// feltételes utasítás
  // pl. ki szeretnáénk írni, hogy i egyenlő-e 42-vel
  // írhatjuk az előbbi printf alá ezeket a sorokat:

    if (i==42)       // Kötelező a feltételt () közé tenni
    {
        printf("i éppen 42\n");
    }
    else
    {
        printf("i nem 42\n");
    }
```

In [ ]:
```
// egyszerű ciklusok:
// for ciklus
//   Pl. az 1--10 számok négyzeteinek kiírása
//   írjuk az eddigiek után a return elé, hogy azok mintaként megmaradjanak

    printf("\n");    // üres sort írunk ki, hogy áttekinthetőbb legyen
    int n=10;
    for (i = 1; i <= n; i++)
    {
        printf("i %i i^2 %i\n", i, i*i);
    }

// a for ciklus sorában lehetnek boynolultabb utasítások
// pl. ha nem az i értékre akarunk feltételt szabni,
// hanem a négyzetszámokat kérjük 150-ig:

    int n=150;
    for (i = 1; i*i <= n; i++)
    {
        printf("i %i i^2 %i\n", i, i*i);
    }

// itt is a pythonhoz hasonlíthatjuk:
// A sorok betolása (indent) nem kötelező, de ettől áttekinthető a program.
// Ehelyett {} fogja össze, ha több utasításos blokkot akarunk futtatni.
// Ez érvényes a függvényekre (ld. main()), if, for
// (Egy utasítás esetén egyébként általában a {} elhagyható,
//    de áttekinthetőség kedvéért meggondolandó meghagyni)

// while ciklus:
//   pl. azt szeretnénk kiszámolni, hogy hány lépésben jutunk el
//   i=1-től 50-ig, ha mindig annyival növeljük i-t,
//   amennyi a négyzetgyökének az egész része

// matematikai függvények (mint most az sqrt) használata esetén
// be kell toldanunk egy újabb csomagbetöltést, pl. a 3. sorba:
    #include <math.h>

// A főprogramrészt pedig így folytatjuk:

    printf("\n");
    i=1;           // kezdő pozíció
    k=0;           // megtett lépések száma
    while (i<50)
    {
        i+=sqrt(i);
        k++;
        printf("i %i k %i\n", i,k);
    }

// Ha parancssorban fordítjuk, akkor
// a matematikai függvények esetén szükséges a -lm opció,
// és általában hasznos az összes figyelmeztetés (Warning all) bekapcsolása:
   gcc -Wall minta1.c -o minta1 -lm
```

- Hibajelzések és figyelmeztetések: kétféle problémára utaló üzenetet kaphatunk a CodeBlocks alsó, üzenet ablakában, ill. a gcc parancssori használatakot a parancsablakban:
  - warning (figyelmeztetés): Ilyenkor lefordul a program és futtatható, de érdemes meggondolni, hogy nem egy olyan hiba okozza-e a problémát, ami elronthatja a program eredményeit. Pl. ha a figyelmeztetés szerint létrehoztunk olyan változót, amit nem használunk, az lehet szándékos, vagy lehet amiatt, hogy véletlen más változót használunk helyette.
  - error (hiba): Le sem fordul a program, meg kell keresnünk a hibát. Lehet komoly hiba is, vagy csak egy elírás (gyakori kezdeti hiba a bezáró ",), } vagy ; lehagyása, vagy név, utasítás elgépelése).
  - Mindkét esetben látjuk, hogy melyik sorban lép fel a probléma és annak tömör leírását, amit kis gyakorlattal jól fel tudunk használni.pl. ha lehagyunk egy " jelet, vagy ;-t, vagy létrehozunk egy k változót, de nem használjuk

---- A gyakorlaton önálló munkaként ezekből ajánlott választani ----

- Másodfokú egyenlet megoldása
- Ellenőrizzük, hogy a szomszédos négyzetszámok különbsége páratlan és számtani sorozatot alkot (így minden páratlan számhoz van megfelelő i)! Érdekesség, hogy ha e különbség négyzetszám is, akkor pitagoraszi számhármast alkot a két négyzetszámmal.
- Számok faktoriálisának kiszámítása

---- Ismertető folytatása ----

- másodfokú egyenlet megoldása megtalálható a "10, Első C program" c. dokumentumban
- négyzetek különbsége a gyakorlaton bemutatva

In [ ]:
```c
// faktoriális számítás megoldása és binomiális együtthatók kiszámítása
// érdemes új projektet kezdeni pl. "binom" néven

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n=5;
    int f=1;
    int i;
    for (i = 2; i <= n; i++)
        f=f*i;               // Ez is rövidíthető így: f*=i;
    printf("%d\n",f);
    return 0;
}
```

In [ ]:
```c
// függvény írás: faktoriális
// ha pl. a binomiális együtthatót akarjuk kiszámolni, 3x kell a faktoriális
// ne kelljen annyiszor leírni,
// ezért függvénybe rakjuk ki a faktoriális kiszámolását:

#include <stdio.h>
#include <stdlib.h>

int factorial(int n) {
    int f=1;
    int i;
    for (i = 2; i <= n; i++)
        f*=i;
    return f;
}

int main()
{
    int n=5;
    int k=2;

    int binom=factorial(n)/factorial(k)/factorial(n-k);
    printf("%d\n",binom);

    return 0;
}
```

Szorgalmi Házi Feladat

- Fibonacci számok előállítása
- pitagoraszi számhármasok keresése a gyakorlat weboldalon megjelenő útmutatás szerint