Tartalomjegyzék

Tobb file-ban dolgozas

3(4) fele file-unk lesz:

- main cpp file, amiben a main fuggveny van (ebbol csak 1 lehet programonkent/projectenkent)
- header file (.h kiterjesztesu), ebben lesznek a headerek (a fuggveny/osztaly deklaraciok)
- cpp file (.cpp kiterjesztes), ebben lesznek a hozza tartozo headerben deklaraltak definicioi

Egy header file pl igy nezne ki:

test.h:

```
#pragma once

int example(int, float);
```

A pragma once parancs azt biztositja, hogy a file a forditott kodban csak egyszer lesz behuzva. Regi megoldas meg erre az ifndef define hasznalata, de ezt manapsag mar csak nagyon regi kodban lehet latni. Joforman ugyanazt csinaljak. (Ezt include guard-nak hivjak.)

A cpp file-ban include-olni kell a .h file-jat. Lokalisan ugy tudunk include-olni, ha a <>-ek helyett "http://wiki.math.bme.hu"http://wiki.math.bme.hu-t hasznalunk. Ekkor az ugyanabban a mappaban levo file-okat latja.

test.cpp:

```
#include "test.h"

int example(int a, float b) {
  if(b > 0.5) {
    return a;
  }
  return -1 * a;
}
```

A main file-ba eleg csak a main fuggvenyt irni. De ne felejtsuk el hogy ide kellene az include-ok amiket hasznalunk a main-ben.

main.cpp:

```
#include<iostream>
#include "test.h"
```

```
using namespace std;

int main(void) {
  int c = example(5, -2.5);
  cout << c << endl;
  return 0;
}
```

Akik g++-t hasznalnak a forditasra, annyi dolguk van csak, hogy az osszes cpp file-t beirjak a forditasnal pl:

```
g++ main.cpp String.cpp
```

Akik IDE-t hasznalnak (codeblocks/visual studio): csinaljatok uj projectet es ebbe hozzatok letre uj file-okat. Az IDE ezutan mar megoldja a file-ok forditasat.

String kivagas

Keszitsunk .h es .cpp file-okat a String osztalynak amit gyakran hasznaltunk mostanaban. Irjunk egy rovid main-t is (vagy hasznaljuk az eredeti feladat main-jet). A vegen 3 file-unk legyen: main.cpp, String.h, String.cpp.

Konyvtarak

Sok cpp konyvtar van. Most csak gyorsan nezzunk ra az std-re amit eddig is hasznaltunk. Nezzunk meg egy tarolot (mint a lancolt lista volt):

https://en.cppreference.comhttp://wiki.math.bme.hu/w/cpp/container/vector

Paratlan szamok vector-al

Irjunk egy fuggvenyt ami parameterkent egy std::vector<int>-et kap es visszaad egy uj vectort, amiben csak a paratlan szamok vannak benne a bemeneti vector-bol.

Paros/paratlan szamok vector-al

Egeszitsuk ki a fentit, hogy adja vissza a paratlan szamok vector-at is. (Probaljuk megoldani pointerekkel.)

Numerikus integral

Irjunk fuggvenyt ami tudja egyvaltozos fuggveny integraljat kozeliteni. Ehhez hasznalhatjatok a definicio szerinti fuggveny alatti teglalapos kozelitest.

Mondjuk ha 0.1-es pontossaggal dolgozunk, akkor pl kiszamolhatjuk a fuggvenyerteket f(2)-n es keszithetunk ezzel teglalapot, a kovetkezo sarkokkal: (2, f(2)), (2.1, f(2)), (2.1, 0), (2, 0). Ezt ha megcsinaljuk az intervallumon 0.1-es (azaz a pontossagos lepeskozzel) akkor ezek szummaja jo kozelites lesz az integralra.

A numerikus integral fuggveny parameterei:

- A pontossag (float)
- A bal oldala az intervallumnak (float)
- A jobb oldala az intervallumnak (float)

A visszateresi ertek legyen float. A fuggveny aminek az integraljat akarjuk szamolni legyen beleepitve az

integral szamolas fuggvenybe. A kovetkezo feladatnal ezt javitjuk. Hasznalhatunk pl negyzet fuggvenyt a tesztelesre.

Altalanos numerikus integral

Lehet fuggveny parametert is csinalni:

```
float integral(float (*f)(float), float from, float to, float precision) {
```

Igy az elso parameter egy fuggveny ami float-ot ad vissza es 1 float parametere van. Igy nezne ki pl egy ilyen fuggveny:

```
float func(float x) {
  return x * x;
}
```

Es igy kene meghivni a fo fuggvenyt:

```
float val = integral(func, -5, 5, 0.1);
```

Modositsuk az integralos feladatot, hogy be tudjuk neki adni a fuggvenyt is amit integralni akarunk.