Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Osztott rendszerek specifikációja és implementációja

IP-08bORSIG

Dokumentáció az *i.* beadandóhoz

`Gabriel Reyes`
`REAPER`

2017. szeptember 22.

1. Kitűzött feladat

A feladat részletes leírása. Nagyrészt a BEAD-on található kiírás (a körítés mellé opcionális), figyelve a formázásra, hogy a copy-paste miatt ne csússzon szét a szöveg, ne legyenek felesleges sortörések, értelmetlen, vagy rosszul elválasztott mondatok. Ha gondoljátok, egyéni megjegyzésekkel ki lehet egészíteni, ha szerintetek az átláthatóbbá teszi a feladat kiírását.

A bemeneti fájl – input.txt – első sorában, szóközzel elválasztva olvashatóak az n és m pozitív egészek, a következő n sorában pedig egy-egy m hosszú sorozat, azaz egy m dimenziós vektor szóközökkel elválasztott, valós ($\mathbb{R}$) elemei:

$n \quad m$
$x_{1,1} \quad x_{1,2} \dots x_{1,m}$ - az 1. vektor koordinátái
...
...
$x_{n,1} \quad x_{n,2} \dots x_{n,m}$ - az n. vektor koordinátái

A program olvassa be az adatokat, majd párhuzamosan számolja ki az n darab vektor hosszát (kettes norma). Az így kapott eredményeket – a vektorok sorszámának megfelelően – írja ki az output.txt kimeneti fájlba.

2. Felhasználói dokumentáció

Ide tartozik minden, amit közölni kell a felhasználóval ahhoz, hogy használni tudja a programot. Olyan módon kell fogalmazni, hogy a corvinusos vagy BTK-s hallgató is megértse, ha leül a gép elé, azaz ezt a részt programozni nem tudó emberek fogják olvasni.

2.1. Rendszer-követelmények, telepítés

Milyen függőségei vannak a programnak? Ide kerülhet pl. ha szükség van valamilyen keretrendszerre (.NET, JVM), vagy egyébre (pl. DirectX, OpenGL, VisualC++), ami nélkül a program hibásan vagy egyáltalán nem működne. Szükséges telepíteni a programunkat? Ha igen, annak a lépéseit részletezni kell.

A programunk több platformon is futtatható, dinamikus függősége nincsen, bármelyik, manapság használt PC-n működik. Külön telepíteni nem szükséges, elég a futtatható állományt elhelyezni a számítógépen.

2.2. A program használata

Hogyan indítjuk el a programot, parancssorból vagy intézőben? Ha több fájlból áll a program, melyiket kell pontosan elindítani? (Tehát ne a 'functions.dll'-t próbálja futtatni a felhasználó, hanem a 'kecske.exe'-t.) Ha kezdeti paramétereket kell átadni a programnak, azt is említsük meg. Milyen egyéb fájlokat vár/használ az alkalmazásunk, ezeket hova tegyük? Hogyan kell kinéznie a bemeneti fájlnak? Van rá külön névmegkötés? Ennek helyességét ellenőrzi a program, vagy felteszi, hogy a specifikációnak megfelelnek az adatok? Ha kimenetet is generál a szoftver, azt hol találhatjuk meg a futtatás végeztével? Hogyan kell értelmezni a kapott eredményt?

A program használata egyszerű, külön paramétereket nem vár, így intézőből is indítható. A futtatható állomány mellett kell elhelyezni az *input.txt* nevű fájlt, mely a bemeneti adatokat tartalmazza, a fenti specifikációnak megfelelően. Figyeljünk az ebben található adatok helyességére és megfelelő tagolására, mivel az alkalmazás külön ellenőrzést nem végez erre vonatkozóan. A futás során az alkalmazás mellett található *output.txt* fájl tartalmazza a kapott eredményt, ahol az i-ik sor a bemeneti fájl i-ik sorában található vektor hosszát jelenti valós számként ábrázolva.

3. Fejlesztői dokumentáció

Ez a rész azoknak szól, akik esetleg átnézik a forráskódodat, vagy később az ő feladatuk lesz tovább dolgozni a programon, felhasználva azt más projektben vagy csak szimplán fejleszteni/kiegészíteni új funkciókkal. Feltehetjük, hogy aki ezt olvassa, ért a C++hoz, és tisztában van a programozásban használatos fogalmakkal (tömbök, indexelés, függvények, szálak stb..).

3.1. Megoldási mód

A kitűzött feladatot hogy értelmezted? Milyen módot választottál ennek megoldására? Miért pont azt az algoritmust / reprezentációt használod, amelyiket? Mivel jobb ez, mint a többi lehetőség?

A kódunkat logikailag két részre bonthatjuk, egy fő-, illetve több alfolyamatra. A fő folyamatunkat a `main()` függvény fogja megvalósítani, mely beolvassa az inputként kapott fájl tartalmát, majd egy $N \times M$ mátrixot tölt fel annak adataiból. Az alfolyamatokhoz ennek a mátrixnak egy-egy sorát társítjuk, melyek elvégzik a szükséges számításokat, majd az így kapott eredményeket a főfolyamat fogja a kimeneti fájlba írni.

3.2. Implementáció

A megoldási módban leírtakat hogyan valósítottad meg C++ban? Az ott leírt absztrakt reprezentációnak milyen típus felel meg a nyelvben? (pl. rendezett pár megvalósítása esetén std::pair<T,T'>-t használunk.) A megoldást hány fájlba szervezted? Ha többre, akkor melyik fájlban mi található, és mi alapján lett szétbontva? (pl. az osztályban található műveletek definíciói kerültek külön.) Nem kell azonban a másik végletbe sem esni, tehát túl részletezni, hogy a for ciklusban mire van az 'i' változó, vagy hogy mit jelent egy összeg kiszámítása előtt az 'int sum=0;' sor.

Az említett mátrixot `std::vector<std::vector<double>>` típussal fogjuk megvalósítani, míg az alfolyamatokat egy `std::future<double>` típusparaméterű vektorban fogjuk tárolni. Az egyes objektumok a kívánt visszatérési értéket is tartalmazni fogják, ezeknek külön memóriát nem kell foglalni. A szükséges N folyamatot az `std::async()` függvény segítségével, azonnal fogjuk új szálon indítani, paraméterül a végrehajtandó `vector_length()` függvényt, illetve a mátrix egy sorát fogjuk átadni. Ez a függvény az euklideszi normált, azaz a:
$||v||_2 = \sqrt{\sum_{i=1}^{n} |v_i|^2}$ $(v \in \mathbb{R}^n)$-t számolja ki, majd ezzel az értékkel tér vissza. A feladat egyszerűsége révén egyetlen forrásfájlban, a `main.cpp`-ben található a teljes implementációs kód.

3.3. Fordítás menete

Hogy kaphatunk a megadott forráskódból futtatható állományt? Milyen fordító szükséges ehhez, annak melyik verziója? Ha valamilyen extra flag-et kell használni ahhoz, hogy leforduljon a kód, akkor mi ez és miért van rá szükség?

A programunk forráskódját a `main.cpp` fájl tartalmazza. A fordításhoz elengedhetetlen egy `C++11` szabványt támogató fordítóprogram a rendszeren. Ehhez használhatjuk az *MSVC*, *g++* és *clang* bármelyikét. A fordítás menete (4.9.2-es verziójú g++ használata esetén) a következő: `'g++ main.cpp -std=c++11'`. A speciális, `-std=c++11` kapcsoló azért szükséges, mert alapértelmezés szerint ez a verziójú fordítóprogram még a régi, C++98-as szabványt követi, melyben a felhasznált nyelvi elemek még nem voltak jelen.

3.4. Tesztelés

Milyen tesztelést hajtottál végre a programon, hogy megbizonyosodj arról, hogy helyesen működik? Ha párhuzamosságot használ a program, hogyan győződtél meg arról, hogy így hamarabb lefutott, mint szekvenciálisan nézve? Ha gyorsabb, milyen mértékben, legalábbis mihez képest gyorsabb? Milyen architektúra alatt jött elő a gyorsulás (egy Xeon Phi-s szörnyetegen vagy egy Pentium 2-es gépen)?

A program tesztelése során különböző méretű bemeneti fájlokat generáltam egy python script segítségével. Az így kapott fájlok mindig a specifikációnak megfelelően, az átlagostól a szélsőséges esetekig terjedtek. A programom minden esetben a tőle elvárt kimenetet állította elő, így a tesztesetek alapján helyesnek gondolhatjuk a működését.

A számítógépemben található 4 magból 3-at kikapcsolva, (időben) szekvenciális lefutást tudtam előállítani, így egy átlagos méretű fájl esetében (10-50 vektor, 100-1000 elemmel) a futási idő $\sim$0.4 mp körüli volt. A magokat sorban visszakapcsolva ez az idő egészen $\sim$0.017 mp-ig csökkent, így megállapíthatjuk, hogy a párhuzamosított program tényleg gyorsabban futott, mint a szekvenciális változata (Intel i5-3570K processzorral).

Általánosságban figyeljetek a helyesírásra, és a szöveg formázására. A megadott szempontok alapján elvárt elemek mindenképp legyenek benne, ez a mintadokumentáció a kitett feladathoz is azért készült, hogy legyen egy irányelv, pontosan mit is szeretnénk látni.

Lehetőség szerint a fejezetcím és a hozzá tartozó információ között ne legyen oldaltörés, ekkor inkább a `\newpage` *paranccsal kezdjétek új oldalon az egészet, vagy szimplán egészítsétek ki / fogalmazzátok át előtte az információkat, hogy könnyebb legyen olvasni.*