Programmeerimine keeles C++

Praktikum 7: C++ lisateegi kasutamine: Boost

Üldised nõuded

Boost (http://www.boost.org/) ei ole üksainus teek, vaid nende kogum. Selle koosseisu kuuluvad teegid aitavad programmeerijal töötada lõimede, failisüsteemi, konfiguratsioonifailide, arukate viitade ja palju muuga. Lisaks kuuluvad Boosti koosseisu ka mitmed teoreetiliselt huvitavad teegid, mis näitavad C++ keele võimsust.

Boost koosneb suures osas ainult päistest, sarnaselt meie universaalse geomeetria teegiga. Mõnede teekide jaoks on olemas ka objektkoodifailid. Alusprogrammi kompileerimiseks peate muutma `Makefile'i` ja märkima ära Boosti päiste (`BOOST_HEADERS`) ja teekide (`BOOST_LIBRARIES`) kausta kui ka kasutatavate teekide tüübi (`BOOST_FLAVOUR`). Lugege `Makefile`'i sees antud kommentaare ning vajadusel pöörduge õppejõu poole.

Boosti dokumentatsiooni leiate aadressil http://www.boost.org/doc/. Valige antud lehelt oma Boosti versioon ning leidke avanevalt lehelt vajalik teek. Selle pealkirjale vajutades viiakse teid konkreetse alamteegi dokumentatsiooni juurde.

Teegid, mida selle kodutöö lahendamiseks vaja läheb, on:

1) failisüsteemi teek **Boost::Filesystem**
2) programmi parameetrite kogumise teek **Boost::Program_Options**
3) lõimeteek **Boost::Thread**

Ülesanne 1 – Failide otsing failisisu järgi

Tuginedes alusprogrammile realiseerige käsurearakendus `findtext`, mis otsib etteantud kaustast ja kõigist selle alamkaustadest faile, mis sisaldavad etteantud otsisõna. Palun kasutage käsurealt parameetrite lugemiseks **program_options** teeki. Programmil on kaks parameetrit **folder** ja **text**, mõlemad on sõned ning mõlemad on kohustuslikud. Näide:

`findtext --folder . --text 123` otsib üles aktiivsest kaustast kõik failid, mis sisaldavad sõnet `123`. Kui nõutud nimega kausta ei ole, tuleb kasutajale anda viisakas veateade. Failisüsteemiga töötamiseks kasutage Boosti teeki nimega **filesystem**. Boosti dokumentatsioonis on alamjaotus „Two-minute tutorial“, mis on abiks selle ülesande lahendamisel. Failist teksti otsimise peate kirjutama ise. Proovige toetada ka teksti, milles on tühikuid (siis peab parameeter olema jutumärkides). Toimiv lahendus annab 2 punkti.

Ülesanne 2 – Failide sorteerimine

Tihti tekib vajadus sorteerida pilte, dokumente või muid faile kaustadesse mingi parameetri järgi. Kasutades **filesystem** teeki realiseerige käsurearakendus `sortfiles`, mis sorteerib (liigutab) lähtekaustas ja kõigis selle alamkaustades leiduvad failid nende viimase muutmise kuupäeva järgi loodud sihtkaustadesse. Selle käigus loob `sortfiles` sihtkausta iga kuupäeva jaoks uue kausta, mille nimeks on kuupäev kujul AAAA-KK-PP (aasta-kuu-päev).

Käsureaparameetreid on jälle kaks – **folder** ja **extension**. Neist esimene – töökaust – on kohustuslik parameeter. Teise parameetriga saab ette anda faililaiendi, mis tüüpi faile sorteerida tuleb. Kui seda parameetrit pole antud, siis sorteeritakse kõik failid. Näide:

`sortfiles --folder .` sorteerib kõik aktiivse kausta failid kuupäeva järgi

`sortfiles --folder pildid --extension jpg` sorteerib kaustas `pildid` kõik JPEG-failid

Lahenduse eest saab 3 punkti.

Märkus: Palun püüdke kinni ka käsureaparameetrite töötlemisel tekkivad erindid nii esimeses kui teises ülesandes.

Ülesanne 3 – Einestavate filosoofide probleem

Vaadake alusprogrammi **philosophers**. Tegemist on modifitseeritud variandiga einestavate filosoofide ülesandest (http://en.wikipedia.org/wiki/Dining_philosophers_problem). Meid ei huvita hetkel mitte filosoofide nälgimine ja hangumine (*starvation, deadlocks*), vaid olukord, kus mitu lõime töötavad samade andmetega ning tekivad olukorrad, kus mõni lõim jõuab teisest ette (*race condition*) ning muudab ära mingid andmed, mida teine just lugeda kavatses.

Olukorda näitlikustab alusprogramm, kus 50 filosoofi püüavad 50 kahvliga süüa. Programmi käivitades peaks see mõne aja pärast peatuma teatega, et mõni kahvel on katki või mõnest kahvlist on tekkinud koopia. Kui seda ei juhtu (erinevad arvutid käituvad erinevalt, siis proovige leida alusprogrammist kommentaar, kus on kolm hüüumärki (!!!) ning eemaldage kommentaar sellele järgneva rea eest. Ehitage programm ning proovige uuesti. Ebaedu korral suurendage väljakommenteeritud koodirea arvulist parameetrit (algselt 10, võib suurendada näiteks kuni 1000).

Teie ülesanne on teha programmile kaks olulist parandust:

1) muuta kahvli võtmine atomaarseks (st kahvli olemasolu kontroll ja võtmine ühe meetodiga)
2) välistada kahvli samaaegne võtmine ja tagasipanek kahe filosoofi poolt, kasutades mutexeid

Kokkuvõttes peaks filosoofide programm suutma 5 minutit töötada (ka sissekommenteeritud reaga), ilma et kahvlid kaoksid või juurde tekiksid. Kui lahendus sellisele tingimusele vastab ja on

mõistlikult saavutatud (mitte liiga esoteeriliste meetmetega), on ta väärt 4 punkti.

Kui soovite näiteid ja selgitusi lõimede kasutamise kohta, siis hea artikli Boosti lõimeteegi kohta on avaldanud Dr Dobbs: http://www.ddj.com/cpp/184401518. Boosti lõimeteegi omadusi selgitab ka see materjal: http://www.paulbridger.com/multithreading_tutorial/

Lisaülesanne – failide alamhulga määramine

Täiendage esimese ülesande programmi kolmanda parameetriga., milleks on mittekohustuslik sõneparameeter **pattern**. See parameeter määrab regulaaravaldise, millele failinimi vastama peab. Näiteks

```
findtext --folder . --pattern a* --text 123
```

otsib jooksvast kaustast kõik a-tähega algavad failid, mille sees on sõne `123`. Lahendus annab 2 lisapunkti.

Tähelepanu – küsimused ülesande kohta ja muud tingimused

Materjalid ja lahenduste üleslaadimise süsteem on aine veebilehel. Tähtaeg on praktikumi toimumisnädala pühapäeva õhtu kell 23:59 (aega on 7 päeva).