# Ohjelmointikielten edistyneet piirteet

## Toiset viikkoharjoitukset

1. Tee onKolmiollinen -aliohjelma, jolle menee parametrina sisään kolme sivun pituutta reaalilukuarvoina. Aliohjelma palauttaa paluuarvon tosi jos parametrien pituisista janoista voi muodostaa kolmion; muutoin epätosiarvon. Aliohjelmalla on vielä yksi lisäparametri, jossa välitetään tieto siitä minkä tyyppinen kolmio voidaan muodostaa (jos se onnistuu): epäsäännöllinen, tasakylkinen, tasasivuinen, suorakulmainen. Hyödynnä tässä parametrissa enum-määrittelyn avulla tekemääsi lueteltua tietotyyppiä jossa arvoina ovat juuri erilaiset kolmiotyypit.

Tee lisäksi pääohjelma, joka kysyy kolmen sivun pituudet käyttäjältä. Pääohjelma päättelee em. aliohjelman avulla voidaanko kolmio tehdä ja minkä tyyppinen vai eikö voida. Pääohjelma tulostaa analyysin tuloksen näytölle.

Ohjelman ajo voisi olla seuraavan kaltainen – käyttäjän syöte on merkitty harmaalla taustavärillä:

**Anna 1. sivun pituus: 3**

**Anna 2. sivun pituus: 4.0**

**Anna 3. sivun pituus: 5**

**Annetuista sivuista voidaan muodostaa suorakulmainen kolmio.**

1. Kirjoita ohjelma, joka lukee päätteeltä yhden reaaliluvun. Ohjelma tulostaa syötettyä lukua edeltävän ja seuraavan kokonaisluvun lukualueelta. Alla ajoesimerkki:

Anna luku: 3.7

Luku 3.7 sisaltyy valille [ 3, 4 ].

Vihje: tutki C++-valmiskirjastoista löytyviä matematiikka-aliohjelmia, josko sieltä löytyisi sopivia sovellettavaksi tässä ! (esim. google-haulla ”C++ math library” pitäisi löytyä sopivia. Kiinnitä huomiota siihen, mitä header-tiedostoja pitää inkludoida jotta valitsemasi matematiikka-aliohjelman käyttö on mahdollinen. Artikkelit sisältävät usein toimivia koodiesimerkkejä).

1. Toteuta seuraavat 3 min()-funktiota, jotka kaikki palauttavat annettujen kokonaislukuparametrien pienimmän arvon. 3- ja 4-parametrisissa versioissa on hyödynnettävä 2-parametrista versiota, eli näistä **min()**-funktioista on kutsuttava edellä tehtyjä **min()**-funktioita (= aliohjelmien uudelleenkäyttöä). Tee lisäksi pääohjelma, jossa kutsutaan tehtyjä aliohjelmia niiden toimivuuden varmistamiseksi.

int min(int x, int y);

int min(int x, int y, int z);

int min(int x, int y, int z, int w);

1. Kokonaisluvun sanotaan olevan alkuluku (engl. prime number), jos se on jaollinen ainoastaan 1:llä ja itsellään. Esimerkiksi 2, 3, 5 ja 7 ovat alkulukuja. Sitä vastoin luvut 4, 6, 8 ja 9 eivät ole alkulukuja.

* + 1. Kirjoita aliohjelma **isPrime**, joka päättelee, onko sille parametrina annettu luku alkuluku ja palauttaa arvon **true** jos on; muutoin **false**:n.

* + 1. Käytä edellä tekemääsi aliohjelmaa hyväksi pääohjelmassa, joka päättelee sen, mitkä luvut väliltä 1-10000 ovat alkulukuja. Pääohjelma tulostaa nämä alkuluvut, niiden lukumäärän sekä niiden prosenttiosuuden kaikista tutkituista luvuista.

* + 1. Tutkiessasi mielivaltaisen luvun X alkulukuominaisuutta voit käyttää seuraavia strategioita:
       - 1. tutkit, onko luku X jaollinen luvuilla 2, 3, …, X - 1
         2. tutkit, onko luku X jaollinen luvuilla 2, 3, …, X / 2
         3. tutkit, onko luku X jaollinen luvuilla 2, 3, …, neliöjuuri( X )

Mikä edellisistä ratkaisuista on tehokkain ? Miksi 2 viimeistä strategiaa ovat toimivia ? Tee a) –kohdan aliohjelmasta versiot **prime\_all**, **prime\_half** sekä **prime\_square\_root**, jotka hyödyntävät edellisiä strategioita nimiensä mukaisesti. Varmista, että niillä kaikilla löytyvät samat alkuluvut, kun suoritat etsinnän luvuista 1, 2, … , 10000 b) –kohdan mukaisesti (esim. tulostamalla kaikki löytyvät alkuluvut tai laskemalla niiden määrät tai niiden prosenttiosuuden kaikista tutkituista luvuista jne.).