Ohjelmointikielten edistyneet piirteet

Toiset viikkoharjoitukset

1. Tee onKolmiollinen -aliohjelma, jolle menee parametrina sisään kolme sivun pituutta reaalilukuarvoina. Aliohjelma palauttaa paluuarvon tosi jos parametrina annetuista janoista voi muodostaa kolmion muutoin epätosiarvon. Aliohjelmalla on vielä yksi lisäparametri, jossa välitetään tieto siitä minkä tyyppinen kolmio on: epäsäännöllinen, tasakylkinen, tasasivuinen, suorakulmainen. Hyödynnä tässä parametrissa enum-määrittelyn avulla tekemääsi lueteltua tietotyyppiä.

   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 kulku voisi olla seuraava – käyttäjän syöte on merkitty harmaalla taustavärillä:

   ```
   Anna 1. sivun pituus: 3
   Anna 2. sivun pituus: 4.0
   Anna 3. sivun pituus: 5

   3:n, 4:n ja 5:n pituisista sivuista voidaan muodostaa
   suorakulmainen kolmio.
   ```

2. Vanha pienempi/isompi kokonaisluku ohjelma.
   Kirjoita ohjelma, joka lukee päätteeltä yhden luvun. Ohjelma tulostaa lukua edeltävän ja seuraavan kokonaisluvun lukualueelta. Ohjelman nimi olkoon valilla ja alla ajoesimerkki:

   ```
   Anna luku: 3.7
   Luku 3.7 sisaltyy valille [ 3, 4 ].
   ```

   Vihje: tutki valmismatematiikka-aliohjelmia, josko sieltä löytyisi sopiva käytettäväksi tässä !

3. Vanha kolme min() aliohjelmaa
   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);
   ```

4. Onko alkuluku aliohjelmavariantit

4. 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.

a. Kirjoita aliohjelma `prime`, joka päättelee, onko sille parametrina annettu luku alkuluku.

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

c. Tutkiessasi mielivaltaisen luvun X alkulukuominaisuutta voit käyttää seuraavia strategioita:

   a. tutkit, onko X jaollinen luvuilla 2, 3, …, X - 1
   b. tutkit, onko X jaollinen luvuilla 2, 3, …, X / 2
   c. tutkit, onko X jaollinen luvuilla 2, 3, …, neliöjuuri(X)

   Mikä 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 määrän tai niiden prosenttiosuuden kaikista tutkituista luvuista jne.).

d. Tutustu ja toteuta alkulukujen päättely tietyltä lukualueelta käyttäen hyväksesi Eratostheneen seula (engl. *Sieve of Eratoshenes*) –nimistä menetelmää; tämä menetelmä on vaihtoehtoinen tapa alkulukujen löytämiseen kohdissa a) – c) esitetylle tavalle. Eratostheneen seula -menetelmä toimii C++-ohjelmana seuraavasti:

   a. Luo C++ -taulukko, jonka alkiot ovat int-tyyppisiä ja jonka indeksi päättyy siihen lukuun, joka on suurin luku, jonka alkulukuominaisuutta haluat testata (esim. indeksit ovat välillä 0 – 10 000, mikäli testaat alkulukuja 10 000 :een asti). Aseta aluksi taulukon jokaisen alkion arvoksi 1 (true). Tämän algoritmin lopuksi kaikkiin niihin alkioihin jää jäljelle arvoksi 1, joiden indeksit ovat alkulukuja. Algoritmi nollaa kaikki muut taulukon alkiot.
   b. Käy lävitse taulukon alkiot indeksistä 2 alkaen loppuun asti seuraavalla tavalla: etsi aina seuraava sellainen taulukon alkio, jossa on arvona 1 (tämän alkion indeksi olkoon alkuindeksi) ja nollaa taulukossa kaikki ne alkiot, joiden indeksit ovat alkuindeksin monikertoja. Esim. 2-indeksin tapauksessa sinun tulee nollata ne alkiot, joiden indeksit ovat 4, 6, 8, 10, … ja 3-indeksin tapauksessa vastaavasti ne alkiot, joiden indeksit ovat 6, 9, 12, 15, … Miksi nämä alkiot voidaan nollata ? Miksi nollaamista ei tarvitse tehdä, kun alkuindeksin kertomassa alkiossa on arvona 0 ?
   c. Vaiheen 2 jälkeen alkulukuja ovat taulukossasi kaikki ne indeksiarvot, joita vastaavissa alkioissa on yhä arvona 1. Tulosta nämä indeksit pienimmästä suurimpaan; löytyvätkö samat alkuluvut kuin kohdissa a) – c) – kuvatulla menetelmällä. Huom: taulukon indeksiä 0 ei pidä käsitellä.

e. Tutki alkulukujen prosenttiosuutta tutkituista luvuista, kun tutkittavat alueet ovat: 1-100, 1-1000, 1-10000, 1-100000, 1-1000000 (= miljoona). Tee tämä ohjelmallisesti eli ohjelmasi ajotulokset kertovat osuuden.

5. sekunnitAjaksi

harjoitus [Kenttä]Kirjoita aliohjelma `sekunnit_ajaksi`, joka saa 1. parametrina sekunttien määrän vuorokauden alusta alkaen (kellonlyömästä 00:00:00 alkaen) ja palauttaa saman ajan tunteina, minuutteina ja sekunteina. Tee myös pääohjelma, jolla testaat em. aliohjelmaa. Esimerkiksi jos sekuntimäärä = 10000, niin vastaava aika tunteina, minuutteina ja sekunteina on 2 tuntia 46 minuuttia 40 sekuntia. Huomaa, että aliohjelmasta pitää palauttaa useampia tietoja; sovella tässä viittausparametreja.

Ajoesimerkki:

```
10000 sekuntia on 2 tuntia 46 minuuttia 40 sekuntia.
```