Ohjelmoinnin perusteet, syksy 2006

Esimerkkivastaukset 1. harjoituksiin.

Alkuperäiset esimerkkivastaukset laati Jari Suominen. Vastauksia muokkasi Jukka Stenlund.

*1. Esitä seuraavan algoritmin **tila** jokaisen rivin jälkeen. Algoritmille annetaan syötteet 1, 2 ja 3. (Huom: tämä **ei ole** Javaa!)*

```
a := -3; b := -2; c := -1;
lue(d);
a := (b-d)*a;
b := a+(b+(c+(d+1)*c)*b)*a;
lue(a); lue(c);
kirjoita(a-c);
c := c+c*c;
```

Alla algoritmin tila esitetään riveittäin. Jos kuvausrivillä on merkki '"', ei muuttujan tilaan kosketa kyseisellä rivillä. Se siis säilyy ennallaan. Merkki '?' tarkoittaa, ettei muuttujaa olla alustettu, eikä sen arvoa siis tiedetä. Viimeisellä sarakkeella kerrotaan vielä, mitä ohjelma tulostaa, vaikkei tätä tehtävänannossa pyydettykään. Jos sarakkeella ei lue mitään, ei ohjelma tulosta mitään...

	a	**b**	**c**	**d**	**tulostetaan**
a := 3; b := -2; c := 1;	3	-2	1	?	
lue(d);	"	"	"	1	
a := (b-d)*a;	-9	"	"	"	
b := a+(b+(c+(d+1)*c)*b)*a;	"	63	"	"	
lue(a); lue(c);	2	"	3	"	
kirjoita(a-c);	"	"	"	"	-1
c := c+c*c;	"	"	12	"	

*2. Mitä seuraava algoritmi tulostaa? Perustele. (Huom: tämä **ei ole** Javaa!)*

```
edellinen := 1; seuraava := 1;
while (edellinen < 25) {
  kirjoita(edellinen);
  apu := seuraava;
  seuraava := edellien + seuraava;
  edellinen := apu;
}
```

Simuloidaan algoritmin suoritusta rivi riviltä, muuttuja muuttujalta samaan tapaan kuin edellisessä tehtävässä.

	edellinen	**seuraava**	**apu**	**tulostetaan**
edellinen :=1; seuraava := 1;	1	1	?	
while (edellinen < 25) { // Ehto pätee!	"	"	?	
kirjoita(edellinen);	"	"	"	1
apu := seuraava;	"	"	1	
seuraava := edellinen+seuraava;	"	2	"	
edellinen := apu;	1	"	"	
}				
while (edellinen < 25) { // Ehto pätee!	"	"	"	

kirjoita(edellinen);	"	"	"	1
apu := seuraava;	"	"	2	
seuraava := edellinen+seuraava;	"	3	"	
edellinen := apu;	2	"	"	
}				
while (edellinen < 25) { // Ehto pätee!	"	"	"	
kirjoita(edellinen);	"	"	"	2
apu := seuraava;	"	"	3	
seuraava := edellinen+seuraava;	"	5	"	
edellinen := apu;	3	"	"	
}				
while (edellinen < 25) { // Ehto pätee!	"	"	"	
kirjoita(edellinen);	"	"	"	3
apu := seuraava;	"	"	5	
seuraava := edellinen+seuraava;	"	8	"	
edellinen := apu;	5	"	"	
}				
while (edellinen < 25) { // Ehto pätee!	"	"	"	
kirjoita(edellinen);	"	"	"	5
apu := seuraava;	"	"	8	
seuraava := edellinen+seuraava;	"	13	"	
edellinen := apu;	8	"	"	
}				
while (edellinen < 25) { // Ehto pätee!	"	"	"	
kirjoita(edellinen);	"	"	"	8
apu := seuraava;	"	"	13	
seuraava := edellinen+seuraava;	"	21	"	
edellinen := apu;	13	"	"	
}				
while (edellinen < 25) { // Ehto pätee!	"	"	"	
kirjoita(edellinen);	"	"	"	13
apu := seuraava;	"	"	21	
seuraava := edellinen+seuraava;	"	34	"	
edellinen := apu;	21	"	"	
}				
while (edellinen < 25) { // Ehto pätee!	"	"	"	
kirjoita(edellinen);	"	"	"	21
apu := seuraava;	"	"	34	
seuraava := edellinen+seuraava;	"	55	"	
edellinen := apu;	34	"	"	
}				
while (edellinen < 25) // Ehto ei päde!	"	"	"	

Algoritmi siis tulostaa Fibonaccin lukusarjaa: 1, 1, 2, 3, 5, 8, 13, 21.

*3. Millaisen "reaalimaailman" ongelman seuraava algoritmi ratkaisee? (Huom: tämä **ei ole** Javaa!)*

```
a := 0;
lue(b);
c := b;
while (c>0) {
  lue(d)
  a := a + d;
  c := c - 1;
}
if (b=0)
  kirjoita("heissulivei")
else
  kirjoita(a/b);
```

Kirjoita algoritmi uudelleen siten, että muuttujilla on nimet, joiden avulla algoritmin tehtävän voi ymmärtää helpommin! Täydennä algoritmia myös tulostusoperaatioin, jotka kertovat, mistä on kysymys. Ketä ensimmäinen parannus palvelee? Entä toinen?

Algoritmi laskee annettujen lukujen keskiarvon. Aluksi syötetään syötettävien lukujen määrä. Alla valistuneempi versio ohjelmasta.

```
kirjoita("Tämä ohjelma laskee lukujen keskiarvon.”);
kirjoita("Monenko luvun keskiarvo lasketaan?");
summa := 0;
lue(lukumäärä);
luettavaaJäljellä := lukumäärä;
while (luettavaaJäljellä > 0) {
  kirjoita("Syötä luku, kiitos:");
  lue(luku)
  summa := summa + luku;
  luettavaaJäljellä := luettavaaJäljellä - 1;
}
if (lukumäärä=0)
  kirjoita("Keskiarvo nollalle luvulle on määrittelemätön!");
else {
  kirjoita("Syötettyjen lukujen keskiarvo on:");
  kirjoita(summa/lukumäärä);
}
```

Kuvaavien muuttujanimien käyttö tekee koodista heti ymmärrettävämpää. Samalla koodi kommentoi itseään: tarve erillisten kommenttien kirjoittamiselle vähenee.

Tulostusoperaatiot tekevät koodista käyttäjäystävällisempää. Käyttäjän ei pitäisi tarvita nähdä ohjelmakoodia ymmärtääkseen, kuinka ohjelmaa käytetään.

4.

a) *Laadi algoritmi, joka lukee 10 lukua ja tulostaa niiden summan.*

Laaditaan algoritmi edellisten tehtävien tyyliin, ei siis Javalla.

```
kirjoita("Ohjelma lukee 10 lukua ja tulostaa niiden summan.");
summa := 0;
i := 0;
while (i < 10) {
  kirjoita("Syötä luku:");
```

```
    lue(apu);
    summa := summa + apu;
    i := i + 1;
  }
  kirjoita("Lukujen summa:");
  kirjoita(summa);
```

Ylläolevan kaltaisen while-ehdon määrittely on klassinen ohjelmointivirheen paikka. Pidä huoli, että lukuja algoritmissasi luetaan tosiaan kymmenen, ilman +/- yhden virhemarginaalia.

b) Laadi algoritmi, joka lukee ensin luvun n, sitten lukee n kappaletta lukuja ja tulostaa niiden summan.

Pienellä muutoksella saadaan ylläolevasta algoritmista nyt haluttu.

```
  kirjoita("Ohjelma lukee halutun määrän lukuja ja tulostaa niiden summan.");
  summa := 0;
  kirjoita("Monenko luvun summa lasketaan?");
  lue(lukumäärä);
  i := 0;
  while (i < lukumäärä) {
    kirjoita("Syötä luku:");
    lue(apu);
    summa := summa + apu;
    i := i + 1;
  }
  kirjoita("Lukujen summa:");
  kirjoita(summa);
```

5. Kirjoita seuraava Java-ohjelma tekstitiedostoksi `Eka.java`:

```
public class Eka {
  public static void main(String[] args) {
    System.out.println("Hyvää päivää!");
    System.out.println("Minä olen tietokone.");
    System.out.println("          =========");
    System.out.println("Luulethan, että olen hyvin viisas?");
  }
}
```

Käännä ohjelma Java-kääntäjällä ja suorita ohjelma Java-tulkilla. Kokeile ohjelman ensimmäisen puolipisteen poistamista. Mitä kääntäjä sanoo? Kokeile ensimmäisen lainausmerkin poistamista. Mitä kääntäjä sanoo? Kirjoita ensimmäinen sana "System" pienellä alkukirjaimella. Mitä kääntäjä sanoo?

Poistetaan ensimmäinen puolipiste ja yritetään kääntää muokattu ohjelma. Tällöin kääntäjä herjaa:

```
Eka.java:4: ';' expected
        System.out.println("Minä olen tietokone.");
        ^
1 error
```

Virheilmoitus saadaan, koska riveillä 3 ja 4 olevien tulostuslauseiden välissä kuuluisi olla puolipiste. Hyvin muotoillussa ohjelmakoodissa tämä puolipiste kuuluu luonnollisesti rivin 3 loppuun eikä rivin 4 alkuun, mutta kääntäjän kannalta tällä ei ole merkitystä. Vasta riville 4 tullessaan kääntäjä tietää kaivata puolipistettä ja valittaa näin ollen rivistä 4, vaikka oikeastaan pitäisi korjata riviä 3. On monimutkaisempiakin tilanteita, joissa virhe koodissa aiheuttaa kääntäjän havaitseman kielioppivirheen vasta jollain myöhemmällä, sinänsä täysin virheettömällä rivillä.

Kokeillaan seuraavaksi, mitä tapahtuu, jos poistetaan ensimmäinen lainausmerkki. Saadaan kääntäjältä ilmoitus:

```
Eka.java:3: ')' expected
        System.out.println(Hyvää päivää!");
                                ^


Eka.java:3: unclosed string literal
        System.out.println(Hyvää päivää!");
                                           ^
2 errors
```

Kääntäjä on näkevinään kaksi virhettä. Koska aloitussulun ja `Hyvää`-merkkijonon välissä ei ole lainausmerkkiä, tulkitsee kääntäjä `Hyvää`-merkkijonon muuttujaksi. Kieliopillisesti oikein olisi kirjoittaa

```
        System.out.println(Hyvää);
```

ja jotain tällaista kääntäjä arvelee tulevan vastaan. Rivi ei kuitenkaan jatkukaan loppusulkeella, ja tämän kääntäjä tulkitsee virheeksi.

Toinen kääntäjää kummeksuttava asia on rivillä esiintyvä yksittäinen lainausmerkki. Kääntäjä ei kuitenkaan ole kyllin fiksu osatakseen näyttää, mista kohtaa lainausmerkin pari on jäänyt pois, joten se vain osoittaa paritonta, sinänsä ihan oikeassa paikassa olevaa lainausmerkkiä.

Kuten esimerkistä havaitaan, kääntäjä ei aina ymmärrä, mikä ongelma oikeasti on. Se ei siis välttämättä tarjoa suoraa, oikeaa ohjetta virheen korjaamiseen. Yksi ohjelmointivirhe voi myös kääntäjän herjoissa muuttua useammaksi.

Kirjoitetaan sitten yksi "System" pienellä:

```
Eka.java:3: package system does not exist
        system.out.println("Hyvää päivää!");
               ^
1 error
```

Tämän virheilmoituksen tarkkaa syytä ei tarvitse vielä tässä vaiheessa ymmärtää. Javassa merkkien koolla on merkitystä, joten "System" ei ole "system". (Tarkemmin: Rivillä käytetään PrintStream-luokan metodia java.lang-paketin System-luokan kautta. system-luokkaa tai -pakkausta ei Javasta puolestaan löydy.) Pakkauksiin (package) ja luokkiin perehdytään joskus tulevaisuudessa.