ITKP102 Ohjelmointi 1 (6 op), arvosteluraportti

Tentaattori: Antti-Jussi Lakanen

22. huhtikuuta 2016

Yleistä

Tentti[1] oli pistekeskiarvon (15.1) perusteella vaikeudeltaan keskitasoa, ehkä hitusen viime vuotta vaikeampi. Omasta tehtäväpaperista saa kopion Antti-Jussilta, huone C414.2. Jos en ole paikalla, laita sähköpostia tai soita 040 805 3276. Papereihin on merkitty virheet, jotka voin käydä kanssasi läpi, tarvittaessa tarkastajan kanssa.

Uusintojen ajankohdat löydät kurssin Korppi-sivulta.

Tehtävä	Teki	Keskiarvo	Tarkastaja
T1	153	3.4	Emma Heikura
T2	155	4.2	Simo Rinne
T3	153	3.7	Teemu Natunen
T4	149	4.0	Emma Heikura
Yht		**15.1**	

Arvosteluasteikko

```
Arvolause   Pistemäärä (alaraja)
5   24
4   21
3   18
2   15
1   12
```

[1] `http://users.jyu.fi/~anlakane/ohjelmointi1/tentit/2016-04-22-tentti2.pdf`

JYVÄSKYLÄN YLIOPISTO

Tehtävä 1 (6 p.)

Yleiset huomiot

Ensimmäinen monivalintatehtävä meni yleisesti ottaen melko hyvin ja oli suoraviivainen arvosteltava. Ehkäpä ennakko-oletusten vastaisesti tämä monivalintatehtävä oli kuitenkin pistekeskiarvon perusteella tentin vaikein.

Parhaiten osattiin kohta 3 (jossa piti tunnistaa funktiolle sopiva esittelyrivi). Tehtävästä on vaikea tunnistaa selvästi eniten haasteita aiheuttanutta kohtaa, mutta ensimmäisessä ja kuudennessa kysymyksessä tehtiin hieman enemmän virheitä kuin muissa kohdissa. Lähes kaikki opiskelijat saivat tehtävästä pisteitä, mutta täysiin pisteisiin yllettiin harvoin.

Vastausavain

```
1B
2C
3C
4D
5B
6A
```

Pisteytys ja virheet

Tehtävän pisteytys oli yksinkertainen. Oikein menneestä kohdasta myönnettiin piste, väärin menneestä nolla pistettä. Miinuspisteitä ei annettu mistään syystä.

Tehtävä 2 (6 p.)

Yleiset huomiot

Tehtävä meni yleisesti ottaen melko hyvin. Lukujen jakojäännöksiä osattiin tutkia hyvin, mutta monella oli operaattorien merkit hukassa, joista pahimmat sekaannukset esitellään alempana. Vastauksessa piti olla luokka, jonka sisällä on pääohjelma. Muita aliohjelmia ei tarvittu. Luokka sekä pääohjelma piti olla dokumentoitu kurssilla esitettyjen dokumentaatiokäytäntöjen mukaisesti.

Malliratkaisu

```
using System;

/// @author  Simo Rinne
/// @version 25.4.2016
///
/// <summary>
/// Ohjelma tulostaa riveittäin luvut 1...100 siten, että kolmella
/// jaollisten lukujen perään tulostetaan myös "Hip" ja viidellä
```

```
/// jaollisten lukujen perään "Hei". Jos luku on jaollinen kolmella
/// sekä viidellä, niin luvun perään tulostetaan "HipHei".
/// </summary>
class HipHei
{
    /// <summary>
    /// Pääohjelma.
    /// </summary>
    public static void Main()
    {
        for (int i = 1; i <= 100; i++)
        {
            Console.Write(i + " ");
            if (i % 3 == 0)
            {
                Console.Write("Hip");
            }
            if (i % 5 == 0)
            {
                Console.Write("Hei");
            }
            Console.WriteLine(); // Rivinvaihto.
        }
    }
}
```

Pisteytys ja virheet

Monet olivat turhaan tarkastaneet erikseen onko luku kolmella sekä viidellä jaollinen. Tästä tuli yleensä pitkä `if`-ketju, josta saattoi helposti unohtua `else` välissä olevien `if`-lauseiden edestä:

```
if (i % 3 == 0 && i % 5 == 0) Console.WriteLine(i + " HipHei");
if (i % 3 == 0) Console.WriteLine(i + " Hip");
if (i % 5 == 0) Console.WriteLine(i + " Hei");
else Console.WriteLine(i);
```

Ongelmana tuossa on se, että kolmella sekä viidellä jaollisten lukujen kohdalla tulostuu

```
15 Hip
15 Hei
15 HipHei
```

Oikeanlaisen tulosteen olisi saanut laittamalla kahden jälkimmäisen `if` sanan eteen `else`, eli näin:

```
if (i % 3 == 0 && i % 5 == 0) Console.WriteLine(i + " HipHei");
else if (i % 3 == 0) Console.WriteLine(i + " Hip");
else if (i % 5 == 0) Console.WriteLine(i + " Hei");
else Console.WriteLine(i);
```

Vastauksissa oli aika sekaisin käytetty yhtä tai kahta `&`-merkkiä `if`-lauseiden ehdoissa. Loogisissa lauseissa ja-operaattorina on sallittua käyttää yhtä tai kahta `&`-merkkiä, mutta niiden toiminnassa on yksi pieni ero (jolla ei onneksi ole merkitystä tämän tehtävän kannalta). Yksittäin käytettynä se laskee kummankin puolen lopullisen arvon ja päättelee vasta sitten onko lopputulos `true` vai `false`. Kahdella `&`-merkillä lopputulos lasketaan minimaalisella evaluaatiolla, eli lopputulos on suoraan `false`, jos operaattorin vasemman puolen arvoksi tulee `false`.

Tyhjästä paperista, tai ohjelmasta, joka ei selvästi tee mitään järkevää tuli suoraan nolla pistettä. Pisteitä vähennettiin seuraavista asioista:

- Jokin operaattorimerkki kirjoitettu väärin, -0.5 p.
- Jaollisuuden testaus tehty väärin, -1 p.
- Dokumentaatio puuttuu kokonaan, -1 p.
- Dokumentaatio puuttuu osittain tai tehty väärin, -0.5 p.
- Luokka tai pääohjelma puuttuu, -1 p.
- Ratkaisu on turhan monimutkaisesti tehty (esim. käytetty taulukkoja tai liikaa ehtoja), -1 p.
- Ohjelma toimii väärällä tavalla ja tulostaa jotain muuta kuin mitä sen pitäisi. Toiminnan vääryydestä riippuen -0.5 p. – -2 p. (esim. rivinvaihto väärässä paikassa tai välilyönti puuttuu -0.5 p, samoja lukuja tulostetaan useasti tai Hip tai Hei tulostetaan liian monesti -1 p.)
- Tulostus alkaa väärällä luvulla tai päättyy väärään lukuun, -0.5 p.
- Nimeämiset tehty väärin (esim. luokka tai pääohjelma kirjoitettu pienellä alkukirjaimella), -0.5 p.
- Merkittäviä syntaksivirheitä (esim. puolipiste luokan tai pääluokan esittelyrivin päätteksi), -0.5 p.

Osa oli saanut jostain käsityksen, että silmukoita käyttäessä on pakko käyttää taulukoita myös. Turhasta taulukoiden käytöstä vähennettiin pisteitä. Yleisin ongelma jaollisuuden testaamisessa oli se, että ehdossa luki vain `i % 5`. Tuo laskee jakojäännöksen, mutta ei itsessään kelpaa ehdoksi. Jakojäännöksestä pitää vielä tutkia onko se tasan nolla, eli `i % 5 == 0`.

Esimerkkitulosteessa oli välilyönti numeron jälkeen. Koska sen laittamista ei erikseen vaadittu tehtävänannossa ja koska monelta se puuttui tai oli hankala tulkita onko välilyöntiä laitettu vai ei, niin päädyin lopulta siihen, että välilyönnin pois jättämisestä ei vähennetä pisteitä.

Dokumentaatiosta `@author` ja `@version` merkinnät puuttuivat monelta, mutta tästä ei vähennetty pisteitä. Pienistä syntaksivirheistä, esimerkiksi yksittäisestä puuttuvasta sulusta tai satunnaisista unohtuneista puolipisteistä ei vähennetty pisteitä. Hyvin monelta puuttui alusta `using System;` mutta sen laittamatta jättämisestä ei vähennetty pisteitä. Hyvin pienistä virheistä dokumentaatiossa ei vähennetty pisteitä.

Operaattorien merkit olivat monella hukassa. Yleisimmät virheet olivat seuraavanlaisia:

- Ehtolauseissa ja-operaattorina käytetty jotain muuta kuin `&` tai `&&` -merkkejä. Yleisin virheellinen vaihtoehto oli käyttää pilkkua.
- Pienempi tai yhtä suuri kuin -merkki oli kirjoitettu matemaattisessa muodossa, eli $\leq$. C#-kielessä pienempi tai yhtä suuri kuin vertailu tehdään pienempi kuin -merkillä ja yhtä suuri kuin -merkillä, jotka on kirjoitettu siten, että yhtä suuri kuin -merkki on jälkimmäisenä, eli näin: `<=`.
- Yhtäsuuruuden vertailussa oli käytetty vain yhtä =-merkkiä kahden sijaan.
- Jakojäännöstä oli yritetty laskea jollain muulla kuin %-operaattorilla. Yleisin virheellinen vaihtoehto oli käyttää kauttaviivaa.

Tehtävä 3 (6 p.)

Yleiset huomiot

Tehtävä meni yleisesti ottaen aika hyvin. Parhaiten osattiin `Tulosta`-metodi sekä tehtävän b-kohta. Eniten haasteita aiheutti eri alkioiden lisääminen uuteen listaan ja `eriLukujaEnintaan`-parametrin huomioiminen.

Malliratkaisu

```
public class T3
{
    public static void Main()
    {
        int[] luvut = { 1, 4, 3, 1, 1, 5, 4, 2, 10, 7 };
        int eriLukujaEnintaan = 5;
        int[] eriLuvut = EriLuvut(luvut, eriLukujaEnintaan);
        Tulosta(eriLuvut);
    }


    /// <summary>
    /// Etsii parametrina viedystä taulukosta kaikki eri alkiot
    /// ja palauttaa ne taulukkona. Eri alkioita etsitään korkeintaan
    /// eriLukujaEnintaan parametrin verran.
    /// </summary>
    /// <param name="luvut">Taulukko, josta etsitään</param>
    /// <param name="eriLukujaEnintaan">Eri alkioiden maksimimäärä</param>
    /// <returns></returns>
    /// <example>
    /// <pre name="test">
    /// int[] luvut  = {1, 2, 4, 8, 1, 2, 4, 8, 10};
    /// String.Join(" ", T3.EriLuvut(luvut, 2)) === "1 2"
    /// String.Join(" ", T3.EriLuvut(luvut, 4)) === "1 2 4 8"
    /// String.Join(" ", T3.EriLuvut(luvut, 5)) === "1 2 4 8 10"
```

```
/// String.Join(" ", T3.EriLuvut(luvut, 9)) === "1 2 4 8 10"
/// String.Join(" ", T3.EriLuvut(luvut, 0)) === ""
/// int[] luvut2  = {1};
/// String.Join(" ", T3.EriLuvut(luvut2, 1)) === "1"
/// String.Join(" ", T3.EriLuvut(luvut2, 9)) === "1"
/// String.Join(" ", T3.EriLuvut(luvut2, 0)) === ""
/// int[] luvut3  = {};
/// String.Join(" ", T3.EriLuvut(luvut3, 0)) === ""
/// String.Join(" ", T3.EriLuvut(luvut3, 100)) === ""
/// </pre>
/// </example>
public static int[] EriLuvut(int[] luvut, int eriLukujaEnintaan)
{
    List<int> eriLuvut = new List<int>();

    foreach (int luku in luvut)
        if (eriLuvut.Count < eriLukujaEnintaan && !eriLuvut.Contains(luku))
            eriLuvut.Add(luku);

    //foreach ( int luku in luvut)
    //{
    //    if (eriLukujaEnintaan <= eriLuvut.Count)
    //        break;
    //    if (eriLuvut.Contains(luku))
    //        continue;
    //    eriLuvut.Add(luku);
    //}

    return eriLuvut.ToArray();
}


/// <summary>
/// Tulostaa taulukon kaikki alkiot samalle riville välilyönnillä
/// eroteltuina.
/// </summary>
/// <param name="luvut">Tulostettava taulukko</param>
public static void Tulosta(int[] luvut)
{
    String vali = "";
    foreach (int luku in luvut) {
        Console.Write(vali + luku);
        vali = " ";
    }
}


/// <summary>
```

```
    /// Etsitään taulukon pisin sana.
    /// </summary>
    /// <param name="sanat">Käytettävä taulukko</param>
    /// <returns>Pisin sana</returns>
    /// <example>
    /// <pre name="test">
    /// string[] sanat = {};
    /// T3.PisinJono(sanat) === "";
    /// string[] sanat2 = {"kissa", "kani", "käärme"};
    /// T3.PisinJono(sanat2) === "käärme";
    /// string[] sanat3 = {"koira", "kissa", "lehmä"};
    /// T3.PisinJono(sanat3) === "koira";
    /// </pre>
    /// </example>
    public static string PisinJono(string[] sanat)
    {
        string pisin = "";
        foreach (string sana in sanat)
        {
            if (sana.Length > pisin.Length)
                pisin = sana;
        }
        return pisin;
    }
}
```

Pisteytys ja virheet

- a) kohdan `EriLuvut`-aliohjelma (yht. 2p)
  - Aliohjelma kommentoitu oikein, 0.5p
  - Aliohjelman esittelyrivi ja ohjelman runko kunnossa, 0.5p
  - Erilukujen lisääminen silmukassa uuteen listaan, 0.5p
  - `eriLukujaEnintaan`-parametrin huomioiminen tuloksessa, 0.5p
- a) kohdan `Tulosta`-aliohjelma (yht. 1p)
  - Aliohjelman esittelyrivi ja ohjelman runko kunnossa, 0.5p
  - Aliohjelma tulostaa taulukon oikein, 0.5p
- b) kohdan `PisinJono`-aliohjelma (yht. 3p)
  - Aliohjelma kommentoitu oikein, 0.5p
  - Aliohjelman esittelyrivi ja ohjelman runko kunnossa, 1p
  - Pisimmän sanan etsiminen silmukassa oikein, 1p
  - Aliohjelma toimii tyhjällä taulukolla, 0.5p

Kokonaispisteet koostuivat yllä olevien kriteerien mukaan mutta tehtävää katsottiin myös kokonaisuutena. Mikäli aliohjemissa oli selkeitä virheitä, siitä tuli miinuspisteitä joko 0.25 tai 0.5 pistettä verran riippuen virheen vakavuudesta. Yleisimpiä huomioita EriLuvut aliohjelmassa oli, että monikaan ei ollut käyttänyt listaa vaan taulukkoa. Uuden taulukon luonnissa käytettiin monesti kokona `eriLukujaEnintaan` muuttujaa, joka ei ole täysin väärin ajateltu. Siinä ei kuitenkaan oteta silloin huomioon sitä, jos alkuperäisestä taulukosta ei löydykään tarpeeksi montaa alkioita täyttämään uutta taulukkoa.

Joissakin ratkaisuissa `EriLuvut` aliohjelma oli toteutettu siten, että alkuperäinen taulukko järjestettiin kasvavaan järjestykseen ja siitä poimittiin alkiot uuteen taulukkoon / poistettiin ylimääräiset alkiot alkuperäisestä taulukosta. Mikäli tehtävässä meni muuttamaan alkuperäistä taulukkoa, tuli siitä pistevähennyksiä -0.5 pisteen verran.

`EriLuvut` aliohjelman testit olivat Bonustehtävänä ja niistä sai yhden pisteen, mikäli oli testannut useampaa erilaista taulukkoa ja verrannut tulosta oikein. Monikaan ei osannut verrata tulosta ja sai tällöin vain 0.5 p. bonusosioista. Mikäli testejä oli tehnyt vain samantyyppisille testitapauksille, sai korkeintaan 0.5 p. tehtävästä.

`PisinJono`-aliohjelma osattiin yleisesti ottaen hyvin. Joissakin vastauksissa oli mennyt taulukon indeksit sekaisin itse taulukon sisällön kanssa ja tämän seurauksena tehtävän pisteet jäivät vähäisiksi. Toinen selkeästi tässä tehtävässä esiinnoussut seikka oli aliohjelman toimiminen tyhjällä taulukolla. Useassa vastauksessa sijoitettiin heti alussa taulukon ensimmäinen alkio muuttujaan (esim. `String pisin = sanat[0];`) ja verrattiin jatkossa taulukon eri alkioiden pituuksia muuttujan `pisin` pituuteen. Mikäli taulukossa ei ole yhtään alkioita, tulisi sijoituksessa `IndexOutOfRangeException`.

Tehtävä 4 (6 p.)

Yleiset huomiot

Toinen monivalintatehtävä meni yleisesti ottaen hyvin ja oli suoraviivainen arvosteltava. Tehtävästä saatiin keskimäärin enemmän pisteitä kuin ensimmäisestä monivalinnasta. Virheet jakautuivat melko tasaisesti eri kohdille, joskin kohta kaksi osoittautui muita hiukan haastavammaksi. Kaksi opiskelijaa jätti lisäksi kokonaan vastaamatta kohtaan kolme, jossa tunnistettiin luvun -15 binäärilukuesitystä. Keskimäärin tehtävästä netottiin 4 pistettä.

Vastausavain

```
1A
2A
3A
4E
5B
6C
```

Pisteytys ja virheet

Tehtävän pisteytys oli yksinkertainen - oikein menneestä kohdasta myönnettiin piste, väärin menneestä nolla pistettä. Miinuspisteitä ei annettu mistään syystä.