ITKP102 Ohjelmointi 1 (6 op), arvosteluraportti

Tentaattori: Antti-Jussi Lakanen

21. huhtikuuta 2017

Yleistä

Tentti[1] oli pistekeskiarvon (14.0) perusteella vaikeudeltaan keskitasoa. Omasta tehtäväpaperista saa kopion Antti-Jussilta, huone C414.2. Jos en ole paikalla, laita sähköpostia tai soita 040 805 3276.

Uusintojen ajankohdat löydät kurssin Korppi-sivulta.

Tehtävä	Teki	Keskiarvo	Tarkastaja
T1	43	3.7	Mikko Häyrynen
T2	43	3.5	Mikko Häyrynen
T3	43	3.0	Mikko Häyrynen
T4	43	3.8	Mikko Häyrynen
Yht		**14.0**	

Arvosteluasteikko

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

[1] `http://users.jyu.fi/~anlakane/ohjelmointi1/tentit/2017-04-21-tentti2.pdf`

JYVÄSKYLÄN YLIOPISTO

Tehtävä 1 (6 p.)

Yleiset huomiot

Ensimmäisessä tehtävässä tuli kirjoittaa luokka, jonka sisällä on `Main`-pääohjelma ja `YmpyranAla`-aliohjelma. Pääohjelmassa luetaan käyttäjältä ympyrän sädettä kuvaava syöte ja parsitaan se `double`-muuttujaan `Double.Parse`-funktiolla. Vastauksessa sai olettaa, että saatu syöte on desimaaliluku, koska poikkeusten käsittely on ollut kurssilla melko vähäistä. Tämä muuttuja annetaan parametrina `YmpyranAla`-funktiolle, joka palauttaa säteen perusteella lasketun ympyrän pinta-alan. Pääohjelmassa tulostetaan tämä paluuarvo näytölle. Tehtävänannon esimerkkitulosteessa oli virheellisesti laskettu `5.1`-säteisen ympyrän alaksi `32.044`, mikä on todellisuudessa tämän ympyrän kehän pituus laskettuna kaavalla $2\pi r$. Todellisuudessa ympyrän pinta-alan kaava on πr^2. Virheellisen kaavan käytöstä ei vähennetty pisteitä, kunhan käytössä oli toinen edellä mainituista kaavoista. Esimerkkitulostetta oli lyhennetty `Math.Round`-funktiolla, mutta tätä ei vastauksessa vaadittu. Tehtävä osattiin yleisesti ottaen melko hyvin, joskin harva vastaus oli syntaksisesti täysin oikein. Yleisimmät virheet listattuna:

- Ei osattu käyttää `Console.ReadLine`- ja `Double.Parse`-funktioita oikein.
- Oli unohdettu, mistä piin arvo löytyy C#:ssa. `Math.PI`:n käytöstä väärällä kirjainkoolla ei vähennetty pisteitä. Myös omaa likiarvoa sai käyttää, jos se oli kirjoitettu syntaktisesti oikein. Pelkän matemaattisen symbolin käytöstä rokotettiin hieman.
- Oli unohdettu, mistä potenssifunktio löytyy C#:ssa, oli käytetty matemaattista potenssinotaatiota tai `^`-operaattoria. Koska potenssin pystyi tässä tapauksessa kiertämään helposti kertolaskulla, virheestä rokotettiin hivenen verran.

Mallivastaus

```
using System;

/// @author Mikko Häyrynen
/// @version 22.4.2017
/// <summary>
/// Ohjelma, joka kysyy säteitä ja tulostaa pinta-aloja.
/// </summary>
public class Tentti
{
    /// <summary>
    /// Pääohjelma, joka kysyy käyttäjältä säteen ja tulostaa ympyrän pinta-alan.
    /// </summary>
    public static void Main()
    {
        Console.Write("Anna ympyrän säde > ");
        string syote = Console.ReadLine();
        double sade = Double.Parse(syote);
        double ala = YmpyranAla(sade);
        Console.WriteLine("Ympyrän pinta-ala on " + Math.Round(ala, 3));
```

```
    }

    /// <summary>
    /// Funktio laskee ympyrän pinta-alan säteen perusteella.
    /// </summary>
    /// <param name="sade">Ympyrän säde</param>
    /// <returns>Ympyrän pinta-ala</returns>
    public static double YmpyranAla(double sade)
    {
        return Math.PI * sade * sade;
    }
}
```

Pisteytys

- Yleinen rakenne kunnossa (luokka, pääohjelma, aliohjelma) 1 p.
- Osoitettu, että osataan käyttää dokumentaatiokommentteja oikein 1 p.
- Osattu lukea ja parsia merkkijono käyttäjältä 1 p.
- Osattu tuottaa tuloste, joka vastaa merkittäviltä osin tehtävänantoa 1 p.
- `YmpyranAla`-funktio on kirjoitettu kutsua myöten oikein 2 p.
- Huonosta yleisestä ilmeestä ja syntaksivirheistä vähennettiin 0-2 p.

Tehtävässä ei pyydetty kirjoittamaan testejä, eikä niiden sisällyttämisestä myönnetty tai vähennetty pisteitä.

Tehtävä 2 (6 p.)

Yleiset huomiot

Tehtävässä 2 tuli kirjoittaa **yksi** pääohjelma, jonka sisällä on kaksi erillistä silmukkarakennetta. Kummassakin tulostetaan kakkosen potenssit väliltä 1 - 128. Tulostuksen muodolla ei ollut merkitystä. Yleisin ratkaisu oli käyttää `for`- ja `while`-silmukoita. Osa myös hyödynsi taulukkoa ja `foreach`-silmukkaa. Jos taulukkoratkaisussa potenssit oli vain manuaalisesti sijoitettu taulukkoon ilman silmukkaa, pisteitä vähennettiin. Myös `do-while` oli käytössä muutamassa vastauksessa. Pistekeskiarvosta huolimatta tehtävä osattiin yleisesti ottaen valitettavan heikosti. Lähes kaikki vastaajat osasivat hyödyntää silmukkaa, mutta silmukan rakenne oli monella virheellinen. Yksikään vastaaja ei yltänyt täysiin pisteisiin. Yleisimmät virheet listattuna:

- Tulostettiin kaikki luvut väliltä 1 - 128.
- Oli käytetty %-operaattoria ja tulostettu kaikki 2:lla jaolliset tai jaottomat luvut.
- Silmukkamuuttujan arvoa oli kasvatettu, mutta muutettua arvoa ei oltu sijoitettu takaisin muuttujaan. Vertaa `i * 2` ja `i *= 2`.

- Eri silmukoissa oli käytetty saman nimisiä muuttujia. Muuttujat ovat samalla näkyvyysalueella, joten tämä aiheuttaa konfliktin.
- Käytettiin `Math.Pow`-funktion sijasta `^`-operaattoria.
- Yritettiin tulostaa taulukko tai lista antamalla se suoraan parametrina `Console.WriteLine`-aliohjelmalle.

Mallivastaus

```
/// <summary>
/// Pääohjelmassa tulostetaan kakkosen potensseja kahdessa silmukassa.
/// </summary>
public static void Main()
{
    for (int i = 1; i <= 128; i *= 2)
        Console.WriteLine(i);

    int j = 1;
    while (j <= 128)
    {
        Console.WriteLine(j);
        j *= 2;
    }
}
```

Pisteytys

- Dokumentaatio 0.5 p.
- Pääohjelman esittelyrivi oikein 0.5 p.
- Osattu käyttää kahta erilaista silmukkarakennetta 1p.
- Ensimmäinen silmukka toimii oikein 2 p.
- Toinen silmukka toimii oikein 2 p.

Tehtävässä ei pyydetty kirjoittamaan luokkaa tai testejä, eikä niiden sisällyttämisestä myönnetty tai vähennetty pisteitä.

Tehtävä 3 (6 p.)

Yleiset huomiot

Kolmas tehtävä koostui neljästä osatehtävästä, joihin oli tarkoitus vastata kattavasti muutamalla virkkeellä. Vastauksen pituudella ei kuitenkaan ollut merkitystä. Kuormittaminen ja tietotyyppien erot olivat selkeästi monelle vieraita asioita. Kommentointi osattiin yleisesti ottaen hyvin, joskin dokumentaatiokommenttien tageissa oli monilla pientä viilaamista.

Osatehtävien pistekeskiarvot

1. Maksimi 2 p, keskiarvo 0.74 p.
2. Maksimi 2 p, keskiarvo 0.76 p.
3. Maksimi 1 p, keskiarvo 0.67 p.
4. Maksimi 1 p, keskiarvo 0.81 p.

Yhteensä: Maksimi 6 p, keskiarvo 2.98 p.

Ratkaisut ja pisteytys

1. Alkeistietotyyppien ja oliotietotyyppien eroja käsittelevä osatehtävä osoittautui haastavaksi arvosteltavaksi. Koska monet eroista liittyvät kielen sisäiseen toteutukseen ja ohittavat siten tämän kurssin osaamistavoitteet, pisteitä myönnettiin osittain vääristäkin vastauksista, jos vastauksesta löytyi yksi tai useampia alla olevista asioista. Listan väittämät eivät ole absoluuttisia totuuksia.

   - Mainittiin, että alkeistietotyypit sijaitsevat arvoina suoraan järjestelmän pinomuistissa; oliotietotyyppien tapauksessa pinomuistissa on vain viite dynaamisen muistin muistipaikkaan 1 p.
   - Toisin kuin oliotietotyypit, alkeistietotyypit eivät voi käyttää metodeja, kuten ToString() tai Destroy() 1 p.
   - Oliotietotyyppi täytyy alustaa `new`-avainsanalla 1 p.
   - Alkeistietotyypillä on aina jokin arvo; oliotietotyyppi voi sisältää tyhjän viitteen 1 p.
   - Aliohjelmakutsussa alkeistietotyypin sisältö kopioidaan aliohjelman parametrimuuttujaan; oliotyyppinen muuttuja siirretään viitteenä 1 p.
   - Alkeistietotyyppejä ei voi luoda itse lisää 1 p.
   - Alkeistietotyypit varaavat aina vakiomäärän muistia; oliot eivät 1 p.
   - Oliotietotyypit määritellään alkeistietotyyppien avulla 0.5 p.
   - Alkeistietotyypit ovat muuttumattomia; oliot eivät 0.5 p.
   - Alkeistietotyypit sisältävät yhden arvon; oliot voivat sisältää monia 0.5 p.
   - Alkeistietotyyppien sisältöä voi vertailla operaattoreilla; olioiden sisältöä ei 0.5 p.

   Osatehtävästä sai kerättyä yhteensä enintään kaksi pistettä. Mainittakoon vielä monen unohtaneen, että myös `string` on olio C#:ssa. Vääristä vastauksista ei vähennetty pisteitä.

2. Aliohjelman kuormittamisella tarkoitetaan sitä, että saman niminen aliohjelma määritellään useaan kertaan eri parametreilla. Vastaukseksi riitti myös mainita, että samaa aliohjelmaa voidaan *kutsua* eri määrällä tai eri tyyppisillä parametreilla. Idean selittämisestä sai yhden pisteen ja havainnollistavasta käytännön esimerkistä toisen. Kokonaisia ohjelmia tai aliohjelmia täysiin pisteisiin ei tarvinnut

kirjoittaa. Täydet pisteet saattoi myös saada, jos esimerkissä oli käytössä oletusarvoiset parametrit, vaikka teknisesti tämä ei kuormittamista olekaan. Jos vastauksessa sanottiin olennaisesti vain, että aliohjelmasta kutsutaan toista aliohjelmaa, tai että aliohjelmalle annetaan ylimääräisiä parametreja, myönnettiin 0.5p. Esimerkkikoodin pienistä virheistä ei vähennetty pisteitä, jos idea oli oikein. Vastaajien pisteet polarisoituivat vahvasti joko kahteen tai nollaan.

3. Kaksiparametrisen funktion dokumentaatiokommentista tuli löytyä `summary`-osio, kaksi `param`-osiota ja `returns`-osio. Osapisteitä myönnettiin jokaisesta oikein kirjoitetusta osiosta. Koska tehtävässä puhuttiin nimenomaan funktioaliohjelmasta, aliohjelman täytyi palauttaa jotain, joten `return`-osiota ei saanut jättää pois. Esimerkkivastaus:

```
/// <summary>
/// Aliohjelma laskee kahden kokonaisluvun summan.
/// </summary>
/// <param name="ekaluku">ensimmäinen kokonaisluku</param>
/// <param name="tokaluku">toinen kokonaisluku</param>
/// <returns>kokonaislukujen summa</returns>
```

4. C#:ssa on kolme erilaista kommenttityyppiä:

```
// yhden rivin kommentti
/* usean rivin kommentti */
/// <summary> dokumentaatiokommentti </summary>
```

   Yhden kommenttityypin esittelystä myönnettiin 0.5p ja kahdesta seuraavasta 0.25p.

Tehtävä 4 (6 p.)

Yleiset huomiot

Tehtävässä 4 tuli kirjoittaa `TulostaAika`-aliohjelma, joka ottaa parametrinaan yhden minuutteja kuvaavan kokonaisluvun väliltä 0 - 1439. Aliohjelma tulostaa parametrina saadut minuutit muodossa `tunnit:minuutit`, eikä palauta mitään. Yksinkertaisin tapa laskea minuutit ja tunnit erilleen oli suorittaa jakolaskut `/`- ja `%`-operaattoreilla. Osa käytti tässä myös silmukkaa. Jos tunnit tai minuutit olivat lopputuloksessa alle 10, tuli niiden eteen tulostaa 0, jotta tulosteessa on aina viisi merkkiä. Mallivastauksessa muotoilu onnistui parilla `if`-lauseella, mutta useamman ehtolauseen käytöstä ei vähennetty pisteitä, jos lähestymistapa oli muuten fiksu. Järkyttävästä ehtolausehässäkästä pisteitä vähennettiin. Luokkaa, pääohjelmaa tai testejä ei pyydetty kirjoittamaan. Yleisimmät virheet listattuna:

- Ajan laskemisessa oli käytetty silmukan sijasta pelkkää ehtolausetta.
- Tehtävää oli lähdetty ratkaisemaan kahden sivun ehtolausehässäkällä.
- Ehtolauseet eivät olleet toisiaan poissulkevia ja sama rivi tulostettiin monesti.
- Ehtolauseiden rajatapauksissa oli ongelmia, esim. kymmenen minuutin eteen tulostettiin vielä nolla.

Mallivastaus

```
/// <summary>
/// Tulostaa parametrina saadut minuutit muodossa tunnit:minuutit.
/// </summary>
/// <param name="aikaMinuutteina">Keskiyöstä kuluneet minuutit kokonaislukuna</param>
public static void TulostaAika(int aikaMinuutteina)
{
    // lasketaan tunnit ja minuutit erilleen parametrista
    // muutetaan luvut samalla merkkijonoiksi
    string tunnit = "" + aikaMinuutteina / 60;
    string minuutit = "" + aikaMinuutteina % 60;

    // lisätään tarvittaessa nolla tuntien tai minuuttien eteen
    if (tunnit.Length < 2) tunnit = "0" + tunnit;
    if (minuutit.Length < 2) minuutit = "0" + minuutit;

    // tulostetaan lopputulos
    Console.WriteLine(tunnit + ":" + minuutit);
}

// vaihtoehtoisesti aliohjelman saattoi kirjoittaa yhdellä
// rivillä, jos osasi käyttää muotoiltua tulostusta:
public static void TulostaAika(int min)
{
    Console.WriteLine("{0:00}:{1:00}", min/60, min%60);
}
```

Pisteytys

- Dokumentaatio 1 p.
- Aliohjelman esittelyrivi ja paluuarvo oikein 1 p.
- Osattu laskea tunnit ja minuutit erilleen parametrista 2 p.
- Osattu tulostaa lopputulos oikeassa muodossa 2 p.