ITKP102 Ohjelmointi 1 (6 op), arvosteluraportti

Tentaattori: Antti-Jussi Lakanen

7. huhtikuuta 2017

Yleistä

Tentti[1] oli pistekeskiarvon (12.9) perusteella vaikeudeltaan tavallista 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	112	3.5	Mikko Häyrynen
T2	112	3.5	Otto Jahnukainen
T3	112	2.8	Simo Rinne
T4	112	2.9	Otto Jahnukainen
Yht		**12.9**	

Arvosteluasteikko

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

[1] http://users.jyu.fi/~anlakane/ohjelmointi1/tentit/2017-04-07-tentti1.pdf

JYVÄSKYLÄN YLIOPISTO

Tehtävä 1 (6 p.)

Yleiset huomiot

Ensimmäisessä tehtävässä tuli kirjoittaa `JononSumma`-funktio annettujen tietojen perusteella. Luokkaa, pääohjelmaa tai testejä ei pyydetty, mutta ne on kirjoitettu esimerkkivastaukseen mallin vuoksi. Tehtävä osattiin yleisesti ottaen melko hyvin. Oikeita ratkaisuja oli pääosin kahta eri tyyppiä. Valtaosa vastaajista kasasi summan suoraan yhteen muuttujaan silmukassa, kuten mallivastauksessa. Toiset taas muodostivat ensin parametrien avulla lukujonon taulukkoon tai listaan ja summasivat sisällön yhteen silmukassa. Yleisimmät virheet olivat kummassakin lähestymistavassa seuraavat:

- Silmukoitiin väärän muuttujan suhteen, aloitettiin tai lopetettiin summa väärästä paikasta tai lisättiin summaan väärää arvoa.
- Unohdettiin sijoittaa kasvatettu arvo takaisin muuttujaan. Esimerkiksi `for`-silmukan lopussa luki monella `i + askel`, eikä `i += askel`.
- Ei oltu huomioitu erityistapausta `askel == 0`.

Mallivastaus

```
public class Tentti
{
    /// <summary>
    /// Funktio palauttaa kokonaislukujen jonon summan.
    /// </summary>
    /// <param name="alku">Jonon ensimmäinen luku.</param>
    /// <param name="loppu">Jonon viimeinen luku.</param>
    /// <param name="askel">Askeleen pituus.</param>
    /// <returns>Summa parametrien määrittämästä jonosta.</returns>
    /// <example>
    /// <pre name="test">
    /// Tentti.JononSumma(2, 6, 2)   === 12;
    /// Tentti.JononSumma(3, 12, 3)  === 30;
    /// Tentti.JononSumma(10, 15, 2) === 36;
    /// Tentti.JononSumma(2, 2, 2)   === 2;
    /// Tentti.JononSumma(1, 0, 1)   === 0;
    /// Tentti.JononSumma(1, 5, 0)   === 0;
    /// Tentti.JononSumma(1, 5, -1)  === 0;
    /// Tentti.JononSumma(-2, 2, 1)  === 0;
    /// Tentti.JononSumma(-2, 4, 2)  === 4;
    /// Tentti.JononSumma(2, 5, 10)  === 2;
    /// </pre>
    /// </example>/
    public static int JononSumma(int alku, int loppu, int askel)
    {
        if (alku > loppu || askel <= 0) return 0;
```

```
        int summa = 0;
        for (int i = alku; i <= loppu; i += askel)
            summa += i;
        return summa;
    }

    public static void Main() {}
}
```

Pisteytys

- Dokumentaatio 1 p.
- Funktion esittelyrivi ja paluuarvo oikein 1 p.
- Kaikki erityistapaukset käsitelty oikein 1 p.
- Perustapaus käsitelty oikein toistorakenteella 3 p.

Tehtävässä ei pyydetty kirjoittamaan luokkaa, pääohjelmaa tai testejä, mutta niiden sisällyttämisestä ei vähennetty pisteitä, ellei toteutuksessa ei ollut merkittäviä virheitä.

Tehtävä 2 (6 p.)

Yleiset huomiot

Tehtävä 2 oli ensimmäinen kahdesta monivalintatehävästä jossa oli 6 kysymystä ja oli suoraviivainen arvosteltava. Keskimäärin tehtävä meni kohtalaisesti ja oli monivalintatehtävistä helpompi pistekeskiarvon perusteella, vaikkakin variaatiota vastauksissa oli hyvin paljon. Täysiä pisteitä oli jokunen mutta myös muutama nolla eksyi joukkoon. Parhaiten osattiin kysymys 2 (C# toiminnallisuus) ja eniten vaikeuksia oli kysymyksessä 5 (rekursio), jossa moni oli valinnut vaihtoehdon a, joka kertoi millainen on hallittu ikuinen silmukka. Rekursiivinen funktio ei kuitenkaan ole silmukka, eikä sillä ole ”runko-osaa” kuten `while`-, `for`-, jne. silmukoilla.

Vastausavain

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

Pisteytys ja virheet

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

Tehtävä 3 (6 p.)

Yleiset huomiot

Tehtävä oli ilmeisesti haastava ja monet eivät olleet ottaneet huomioon sitä, että merkkijonoja ei voi muokata samaan tyyliin kuin taulukoita (esim. `merkkijono[0] = 'K'`), vaan se pitää ensin muuttaa muuttaa `StringBuilder`-olioksi tai `char`-taulukoksi, mikäli noin haluaa tehdä.

Malliratkaisu

Tehtävän pystyi ratkaisemaan monella eri tavalla, jotka voidaan jaotella karkeasti kahteen kategoriaan: merkkijonon läpikäynti ja välilyönnin jälkeisten merkkien isoksi muuntaminen tai pilkkomalla merkkijono välilyöntien kohdalta ja muuttamalla jokaisen sanan ensimmäinen kirjain isoksi.

Alla malliratkaisu käymällä merkkijono merkki kerrallaan läpi ja käyttäen `StringBuilder`-oliota.

```
/// <summary>
/// Muuttaa merkkijonon jokaisen sanan ensimmäisen kirjaimen isoksi.
/// </summary>
/// <param name="merkkijono">Merkkijono jota muokataan</param>
/// <returns>Muutettu merkkijono</returns>
public static string MuutaIsoksi(string merkkijono)
{
    StringBuilder merkit = new StringBuilder(merkkijono);
    for (int i = 0; i < merkit.Length; i++)
    {
        if (i == 0 || merkit[i - 1] == ' ')
        {
            merkit[i] = Char.ToUpper(merkit[i]);
        }
    }
    return merkit.ToString();
}
```

`char`-taulukkoa voi käyttää myös, jolloin funktion toteutuksen ensimmäisestä rivistä tulisi `char[] merkit = jono.ToCharArray();` ja viimeisestä tulisi `return String.Join("", merkit);` koska taulukkoa ei voi muuttaa takaisin merkkijonoksi järkevästi `ToString`-metodilla.

Alla malliratkaisu pilkkomalla merkkijono sanoiksi ja muuttamalla jokaisen sanan ensimmäinen kirjain isoksi (dokumentaatio on jätetty pois, koska se olisi sama kuin ylemmässä malliratkaisussa):

```
public static string MuutaIsoksi(string merkkijono)
{
    // Pilkotaan merkkijono taulukkoon välilyöntien kohdalta.
    string[] sanat = merkkijono.Split(' ');
```

```
    for (int i = 0; i < sanat.Length; i++)
    {
        string sana = sanat[i];

        // Jätetään tyhjä sana huomiotta.
        if (sana.Length == 0)
            continue;

        //               'K'                    "issa"
        string isoSana = Char.ToUpper(sana[0]) + sana.Substring(1);
        sanat[i] = isoSana;
    }

    // Palautetaan yhdistetty sanat taulukko laittamalla jokaisen
    // sanan väliin välilyönti.
    return String.Join(" ", sanat);
}
```

Pisteytys ja virheet

ComTest-testejä ei vaadittu, eikä niiden lisäämisestä ollut haittaa eikä hyötyä. Aluksi tarkistettiin kuuluuko vastaus johonkin alla olevista kategorioista:

- Tyhjä paperi tai funktio, joka ei selkeästi tee mitään järkevää, 0 p.
- Funktio, joka toimii vain ja ainoastaan merkkijonoilla ”kissa” tai ”kissa istuu puussa”, 0 p.
- Funktio, josta näkee, että ollaan oikeilla jäljillä, mutta toteutus on erittäin puutteellinen, 1 – 2 p.
- Funktio, joka toimii vain jos merkkijonossa ei ole enempää kuin kolme sanaa, 2 p.

Jos pisteet menivät nollaan, niin hyvin tehdyllä dokumentaatiolla sai korotettua sen vielä takaisin yhteen pisteeseen. Mikäli vastaus ei kuulunut mihinkään edellä olevista kategorioista, niin vastaus oli tehty suurin piirtein oikein. Pisteytys aloitettiin silloin 6 pisteestä ja seuraavista asioista tuli vähennyksiä:

- Dokumentaatio puuttuu tai ei ole tehty XML-tyylisesti kurssilla opetettuun tyyliin, -1 p.
- Dokumentaatiossa on pieniä puutteita, esimerkiksi `<param>` tai `<returns>` puuttuu, -0,5 p.
- Parametrit tai paluuarvon tyyppi väärin, -1 p.
- Tehtävänannossa oli annettu vinkki, miten kirjain muunnetaan isoksi. Jos merkin isoksi muuntaminen oli silti tehty väärin ja/tai merkkijonoa yritettiin muokata suoraan indeksoimalla sitä kuin taulukkoa, -1 p.
- Jos isoksi kirjaimeksi muuntamisessa ainut vika oli se, että on kirjoittanut `jono[0].ToUpper()`, vaikka pitäisi olla `Char.ToUpper(jono[0])`, niin -0,5 p.

- Vakavista syntaksivirheistä -0,5 p. tai -1 p. jokaisesta erityyppisestä virheestä riippuen vakavuudesta. Saman virheen toistamisesta ei kumulatiivisesti vähennetty pisteitä.
  - Yksittäisten sulkujen tai puolipisteiden puuttumisesta tai `'`-merkin sijaan `"`-merkin käytöstä ei vähennetty pisteitä.
  - Jos yksittäinen vertailu tehtiin käyttämällä vain yhtä =-merkkiä, niin pisteitä ei vähennetty, mutta jos kaikki vertailut tehtiin vain yhdellä yhtäsuuri kuin -merkillä, niin -0,5 p.
  - Merkkijonojen tai taulukoiden indeksointi tavallisilla suluilla, -0,5 p.
  - StringBuilderin luonti ilman `new`-sanaa, esim `StringBuilder sb = "kissa"`, kun pitäisi olla `StringBuilder sb = new StringBuilder("kissa")`, -0,5 p.
  - Vertailu matemaattisella $\leq$-merkillä, kun pitäisi olla `<=`, -0,5 p.

Yleisimmät viat:

- Isoksi merkiksi muuntaminen tehty väärin. Esimerkiksi

  ```
  merkkijono[0].ToUpper();
  ```

  Tuossa on väärin se, että `merkkijono[0]` palauttaa merkin, jolla ei ole mitään `ToUpper`-metodia ja vaikka olisikin, niin se ei voisi muokata merkkijonon sisältöä suoraan. Se voisi korkeintaan palauttaa isoksi muutetun merkin, joka pitäisi sijoittaa jonnekin.

- Merkkijonon muokkaaminen taulukkomaisesti. Esimerkiksi

  ```
  merkkijono[0] = Char.ToUpper(merkkijono[0]);
  ```

  `string`-olioiden sisällöt ovat muuttumattomia, eikä niitä voi muokata kuin taulukoita. Tästä syystä merkkijonojen muokkausmetodit palauttavat aina uuden merkkijonon, joka pitää muistaa sijoittaa johonkin muuttujaan. Mikäli merkkijonoa haluaa käsitellä taulukkomaisesti, niin se pitää muuttaa `StringBuilder` olioiksi tai `char`-taulukoksi. Malliratkaisusta näkee sen voi tehdä.

- `char`-taulukkoa yritettiin muuttaa takaisin merkkijonoksi `ToString`illä, joka lopputuloksena tulee vain `"System.Char[]"`. Sen sijaan `String.Join("", merkkiTaulukko)` antaisi oikean lopputuloksen. `ToString`in käytöstä ei tosin vähennetty pisteitä, koska tätä esiintyi usealla eikä tämä ei ole mikään ilmiselvä vika.

- Dokumentaatiosta puutui `<returns>`-tagi.

- Monien funktiot kaatuisivat, jos merkkijonon alussa tai lopussa olisi välilyönti. Tästä ei vähennetty pisteitä, koska tehtävänannossa ei mainittu, että välilyöntejä olisi muuallakin kuin sanojen väleissä, eikä tehtävänannossa olleissa testitapauksissa ollut tälläistä tapausta.

Tehtävä 4 (6 p.)

Yleiset huomiot

Tehtävä neljä oli toinen monivalintakysymys tehtävä jossa oli myös 6 kysymystä. Kysymykset olivat selvästi haastavia ja useampi jäi pisteittä. Myös täydet pisteet olivat erittäin harvassa. Kysymyksessä 3 vaikeuksia oli int-muotoisten muuttujien jakamisessa. Kun jaetaan kaksi int-arvoa, saadaan int-arvo tulokseksi (desimaaliosa leikkautuu pois) vaikka se sijoitettaisiin double muuttujaan.

Myös kysymyksessä 4 return-lausetta *ei* voi korvata break-lauseella. Binääritehtävä oli osattu kohtuullisen hyvin ja sai eniten oikeita vastauksia.

Vastausavain

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

Pisteytys ja virheet

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