ITKP102 Ohjelmointi 1 (6 op), arvosteluraportti

Tentaattori: Antti-Jussi Lakanen

20. toukokuuta 2016

Yleistä

Tentti[1] oli pistekeskiarvon (11.6) perusteella vaikea. 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	27	2.6	Teemu Natunen
T2	30	3.1	Teemu Natunen
T3	31	4.1	Antti-Jussi Lakanen
T4	25	2.9	Teemu Natunen
Yht		**11.6**	

Arvosteluasteikko

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

[1] http://users.jyu.fi/~anlakane/ohjelmointi1/tentit/2016-05-20-tentti2.pdf

JYVÄSKYLÄN YLIOPISTO

Tehtävä 1 (6 p.)

Yleiset huomiot

Ensimmäisessä tehtävässä piti tehdä funktio VasenTayte ja luokka funktion ympärille. Tehtävä meni kohtalaisesti. Mitään tiettyä yleistä virhettä ei tehtävässä ollut mutta muutamia selkeästi erilaisia tapauksia nousi esiin:

- Lisättiin täytettä väärä määrä (korvattiin esimerkiksi koko alkuperäinen sana täytemerkillä).
- Lisättiin täytettä oikea määrä tulos-muuttujaan mutta unohdettiin lisätä alkuperäinen sana tulokseen.
- Kasattiin tulosta merkkijonomuuttujaan (String). Tässä luodaan turhia olioita.

Mallivastaus

```
public class Tentti2
{
    /// <summary>
    ///  Pääohjelma
    /// </summary>
    public static void Main()
    {
        String sana = "Teemu";
        String tulos = VasenTayte(sana, 8, '.');
        Console.WriteLine(tulos);
    }

    /// <summary>
    /// Aliohjelma täyttää merkkijonon alun halutulla merkillä.
    /// </summary>
    /// <param name="sana">Sana, jota ollaan muotoilemassa.</param>
    /// <param name="pituus">Tuloksen haluttu pituus.</param>
    /// <param name="tayte">Täytemerkki.</param>
    /// <returns>Merkkijono, johon on lisätty tarvittava
    ///  määrä täytemerkkejä.</returns>
    /// <example>
    /// <pre name="test">
    /// Tentti2.VasenTayte("aajii", 6, ' ') === " aajii";
    /// Tentti2.VasenTayte("aajii", 5, ' ') === "aajii";
    /// Tentti2.VasenTayte("aajii", 4, ' ') === "aajii";
    /// Tentti2.VasenTayte("aa", 6, ' ') ===    "    aa";
    /// Tentti2.VasenTayte("aa", 6, '.') ===    "....aa";
    /// Tentti2.VasenTayte("", 6, ' ') ===      "      ";
    /// Tentti2.VasenTayte("aajii", 0, ' ') === "aajii";
    /// Tentti2.VasenTayte("", 0, ' ') === "";
```

```
    /// </pre>
    /// </example>
    public static String VasenTayte(String sana, int pituus, char tayte)
    {
        /* TAPA 1
        StringBuilder tulos = new StringBuilder(sana);
        for (int i = 0; i < pituus - sana.Length; i++)
            tulos.Insert(0, tayte);
        return tulos.ToString();
        */

        /* TAPA 2 - huono, koska luodaan olioita
        String tulos = "";
        for (int i = 0; i < pituus - sana.Length; i++)
            tulos += tayte;
        tulos += sana;
        return tulos;
        */

        /* TAPA 3 */
        StringBuilder tulos = new StringBuilder(sana);
        while (tulos.Length < pituus)
            tulos.Insert(0, tayte);
        return tulos.ToString();
    }
}
```

Pisteytys ja virheet

Tehtävän pisteytys:

- Kommentit 1 p.
- Luokka ja main-metodi oikein $\frac{1}{2}$ p.
- VasenTayte-funktion esittelyrivi oikein $\frac{1}{2}$ p.
- VasenTayte-funktion toteutus 4 p.
  - Otetaan huomioon tapaus, jossa haluttuPituus < jononPituus 1 p.
  - Täytteen lisääminen jonon alkuun (oikea määrä, tulos-olio Stringbuilder-tyyppinen) 2 p.
  - Palautetaan oikeanmuotoinen tulos (String) 1 p.

Jos tehtävän oli toteuttanut ilman StringBuilder-oliota (käyttäen siis String-muuttujaa), vähennettiin kokonaispistemäärästä 1 p.. Merkkijonoja käytettäessä jokaisessa vastauksessa luotiin aina uusi olio, kun jonoon lisättiin täytemerkki. Turhien uusien olioiden luomista tulee välttää.

Tehtävä 2 (6 p.)

Yleiset huomiot

Tehtävä kaksi meni yleisesti ottaen hyvin. Selkeästi parhaiten osattiin vastata ensimmäiseen kohtaan eli kommentoinnin käyttöön ja tapoihin. Huonoiten osattiin esitellä sivuvaikutuksellisia operaattoreita. Tähän kohtaan moni oli jopa jättänyt vastaamatta kokonaan. Kolmannessa kohdassa kysyttiin aliohjelman esittelyrivin osia. Esittelyrivi osattiin selittää melkoisen hyvin, joskin suurella osalla jokin esittelyrivin osa jäi mainitsematta.

Malliratkaisu

Alla on lueteltu hyvässä vastauksessa esiintyviä seikkoja. Täysiä pisteitä varten kaikkia seikkoja ei ollut tarkoitus mainita, vaan vastauksen kokonaisuudella oli merkitystä.

1. Kommentointi

    - Kommentointi on koodin sekaan kirjoitettua tekstiä, jota kääntäjä ei huomioi
    - Kommentointi voidaan jakaa koodin sisäiseen kommentointiin joka näkyy vain lähdekoodin lukijalle ja sen muokkaajalle, sekä dokumentaatiokommentointiin (XML-muotoiset kommentit), joka on näkyvissä myös koodin "ulkopuolella" esimerkiksi API-dokumenteissa.
    - Sisäistä kommentointia voidaan tehdä seuraavasti: yksiriviset (// kommentti) ja moniriviset (/* monta riviä [rivinvaihto] kommentteja */)
    - Dokumentaatiokommentit: (/// <tägi> Kommentti... </tägi>)
    - Koodia on helpompi ymmärtää myös niiden, ketkä eivät ole ohjelmaa itse kirjoittaneet
    - Ajan myötä toiminta unohtuu, joten kommentointi helpottaa työskentelyyn palaamista myöhemmin
    - Dokumentaatiokommentit kertovat ohjelmasta / funktiosta kaiken oleellisen tiedon.
    - Koodia on helpompi päivittää ja käyttää uudelleen
    - Myös aliohjelmien, olioiden, muuttujien jne. nimet voidaan lukea osaksi kommentointia.

2. Sivuvaikutukselliset (tai arvonmuunto-) operaattorit

    - ++, lisää muuttujan arvoa (int, double, ...) yhdellä.
    - –, vähentää muuttujan arvoa (int, double, ...) yhdellä.
    - +=, lisäysoperaatio, esim. int luku = 0; luku += 2; lisää luku-muuttujan arvoa kahdella.
    - *=, kertolaskuoperaatio, esim. int luku = 2; luku *= 3; kertoo luku-muuttujan arvon kolmella, tuloksena luku-muttujan arvo on 6.
    - Muita ovat /= ja %=

3. Funktion esittelyrivi esim: public static int Summa(int[] luvut, int raja)

    - Saantimääre, yllä public. Kertoo mistä funktiota voidaan kutsua. public-määreen tapauksessa karkeasti “mistä vain”. Muita vaihtoehtoja ovat private, internal ja protected internal. (ks. https://msdn.microsoft.com/en-us/library/ba0a1yw2.aspx)
    - static-sanalla ilmaistaan että jäsen (tässä siis kyseinen funktio) on osa tyyppiä (luokkaa), ja jättämällä static-sana pois jäsen on osa tyypistä tehtyä oliota. Kurssilla on myös käytetty sellaista ilmaisua, että jos funktio on staattinen, se pystyy toimimaan vain ja ainoastaan parametreina saatujen tietojen perusteella, eli se ei tarvitse esimerkiksi luokan attribuutteja tai muita olioita toimiakseen.
    - Paluuarvon tyyppi, yllä int. Tällä sanalla kerrotaan minkä tyyppisen arvon funktio palauttaa. Jos funktio ei palauta mitään arvoa, tyyppi on void.
    - Funktion nimi, yllä Summa.
    - Parametrit, yllä int[] luvut, int raja. Parametreja ei välttämättä tarvitse olla yhtään, jolloin sulkujen sisälle ei tule mitään.

Pisteytys ja virheet

Jokainen tehtävän kohta oli 2 pisteen arvoinen.

- Kommentointi 2 p.
    - Mainittiin, että kommentoinnilla on tarkoitus selventää ohjelman toimintaa joko itselle tai muille. 1 p.
    - Mainittiin vähintään kaksi erilaista kommentointitapaa. 1 p.
- Sivuvaikutukselliset operaattorit 2 p.
    - Operaattorin mainitseminen $\frac{1}{2}$ p ja esimerkki ko. operaattorin käytöstä $\frac{1}{2}$ p.
- Aliohjelman esittelyrivi 2p
    - Aliohjelman esittelyriviltä selitetty julkisuus (public, private), staattisuus, paluuarvo, aliohjelman nimi ja aliohjelman parametrit. 1.25 p.
    - Annettu esimerkki aliohjelman esittelyrivistä 0.75 p.

Tehtävä 3 (6 p.)

Malliratkaisu

Täyden pisteen vastauksissa esiintyi seuraavia asioita.

1. for-silmukan otsikon osat ovat alustus, loppuehto ja päivitys. Kunkin näiden osan välissä on puolipiste. Alla esimerkki for-silmukkarakenteen käytöstä.

    - Alustus, esimerkiksi int i = 0. Tämä suoritetaan aloitettaessa silmukkarakenteen suoritus, ja se suoritetaan kerran huolimatta siitä, suoritetaanko silmukan runko-osaa kertaakaan.

- Loppuehto, kurssilla käytetty myös nimitystä jatkamisehto. Esimerkiksi i < taulukko.Length, jonka saadessa arvon true silmukan runko-osa suoritetaan, ja arvon false silmukan suoritus lopetetaan.
- Päivitys, esimerkiksi `i++` suoritetaan runko-osan suorittamisen päätteeksi. Tämän jälkeen tarkastellaan taas loppuehtoa.

```
int[] taulukko = ... // taulukon alustus

for (int i = 0; i < taulukko.Length; i++)
{
  taulukko[i] = taulukko[i] + 1;
}
```

2. Kun int a = 3; ja int b = 5;, niin lausekkeen `b % a < b - a` arvo on false. Laskeminen tapahtuu seuraavasti.
   - Ensin lasketaan b % a, joka on 2. Perustelu: modulo-operaattorilla on korkein presedenssi.
   - Sitten b - a, joka on myöskin 2. Perustelu: vähennysoperaattorilla on pienempi-kuin-operaattoria korkeampi presedenssi.
   - Nyt ollaan saatu 2 < 2 joka on false.
3. Rekursiivinen aliohjelma kutsuu itseään.
4. !-operaattori (looginen negaatio) palauttaa annetun totuusarvon vastakkaisena. Esimerkki: lausekkeen !false arvo on true, lausekkeen !!true arvo on true, lausekkeen !(5==5) arvo on false, ja lausekkeen !(5 < 5 == !false) arvo on true.

Pisteytys ja virheet

Huomioita kohdittain.

1. Vastauksissa oli paljon hajontaa. Pyrin arvioimaan kohtaa ja ymmärtämistä kokonaisuutena. Pienistä epätarkkuuksista en sakottanut. Selkeistä käsitteellisistä virheistä vähensin kuitenkin varsin herkästi pisteitä.
2. Tyypillisin virhe tässä kohdassa oli jättää lopullinen tulos false kirjoittamatta paperille. Laskut osattiin aika hyvin, mutta laskujen jälkeen lausekkeen 2 < 2 evaluointi tuotti monille harmaita hiuksia: tämän tulos on siis todellakin false, sillä <-operaattori tuottaa bool-tyyppisen arvon.
3. Moni antoi esimerkkinä rekursiivisista aliohjelmista fraktaalien piirtämisen – esimerkkejähän ei tässä tehtävässä pyydetty. On hyvä muistaa, että rekursiota voi käyttää ja käytetään paljon muuhunkin kuin fraktaaleihin.
4. Tyypillinen virhe oli liittää operaattori erisuuruuden vertailuun. Tähän on kuitenkin oma operaattorinsa, !=. Huomattava määrä muitakin virheitä esiintyi. Selkeästi tämä oli tämän tehtävän vaikein kohta.

Tehtävä 4 (6 p.)

Yleiset huomiot

Neljäs tehtävä oli tentin tehtävistä selkeästi vaikein. Tehtävässä vaadittiin kokonaislukutaulukon moodin selvittämistä. Helpoin tapa oli ensin järjestää taulukko ja tämän jälkeen etsiä "pisin pätkä"samaa alkiota taulukossa. Tähän tarvittiin useampi muuttuja, joihin pituuksia laskettiin.

Mallivastaus

```
/// <summary>
/// Laskee kokonaislukutaulukon moodin
/// </summary>
/// <param name="luvut">Käytettävä taulukko</param>
/// <returns>Taulukon moodi</returns>
/// <example>
/// <pre name="test">
/// int[] luvut = {1, 2, 8, 2, 4, 3, 4, 2, 2, 9, 8, 7};
/// Tentti2.Moodi(luvut) === 2;
/// int[] luvut5 = {};
/// Tentti2.Moodi(luvut5) === Int32.MinValue;
/// int[] luvut2 = {2};
/// Tentti2.Moodi(luvut2) === 2;
/// int[] luvut3 = {1, 1, 1, 1, 1, 1, 1};
/// Tentti2.Moodi(luvut3) === 1;
/// int[] luvut4 = {1, 2, 8, 7, 5, 3};
/// Tentti2.Moodi(luvut4) === 1;
/// </pre>
/// </example>
public static int Moodi(int[] luvut)
{
    if (luvut.Length == 0) return Int32.MinValue;
    if (luvut.Length == 1) return luvut[0];
    Array.Sort(luvut);

    int pisinPituus = 0;
    int pituus = 1;
    int tulos = luvut[0];

    for (int i = 0; i < luvut.Length - 1; i++)
    {
        if (luvut[i] == luvut[i + 1]) pituus++;
        else
        {
            if (pituus > pisinPituus)
            {
                tulos = luvut[i];
```

```
                pisinPituus = pituus;
                pituus = 1;
            }
        }
    }
    return tulos;
}
```

Pisteytys ja virheet

- Kommentit 1p
- Aliohjelman esittelyrivi oikein 1p
- Aliohjelman toiminta 4p
  - Taulukon järjestäminen 1p
  - Muuttujien määrittely (Mallivastauksen nykyinenPituus, pisinPituus, moodi) sekä niiden käyttäminen koodissa järkevästi 1p
  - Loogiset operaatiot moodin selvittämiseksi 1p
  - Aliohjelma toimii erilaisilla taulukoilla 1p