ITKP102 Ohjelmointi 1 (6 op), arvosteluraportti

Tentaattori: Antti-Jussi Lakanen

25. toukokuuta 2022

Yleistä

Tentti[1] oli pistekeskiarvon 11,8 (keskihajonta 6,4) perusteella tavanomaista vaikeampi. Niinpä pisterajat asettuivat tällä kertaa tavanomaista alemmas. Huomaa, että demopisteet on laskettu tentin päälle, ja arvosana lasketaan vasta sen jälkeen. Opiskelijan omat tehtävät ovat nähtävissä TIMissä. Viimeisen uusinnan ajankohdan löydät Sisusta.

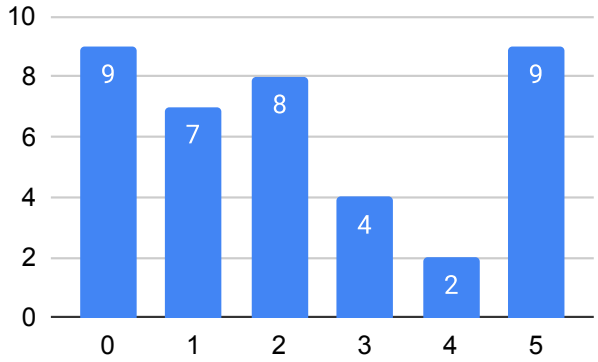
Tehtävä	Keskiarvo	Tarkastaja
T1	4,6	Antti-Jussi Lakanen
T2	2,9	Antti-Jussi Lakanen
T3	2,1	Antti-Jussi Lakanen
T4	2,2	Antti-Jussi Lakanen
Yht	**11,8**	(ennen demopisteitä)

[1] `http://users.jyu.fi/~anlakane/ohjelmointi1/tentit/2022-05-25-tentti1.pdf`

JYVÄSKYLÄN YLIOPISTO

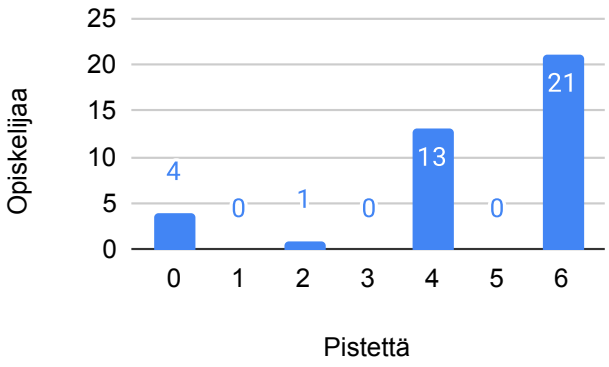
Arvosteluasteikko ja arvosanajakauma

Arvolause	Pistemäärä (alaraja)
5	22
4	19
3	16
2	13
1	10



Tentin arvosanajakauma.

Tehtävä 1 (6 p.)



Tehtävän 1 pistejakauma.

Tee funktio KierraTaulukkoa. Funktio ottaa parametrina kokonaislukutaulukon (int[]) ja palauttaa uuden taulukon (int[]), jossa kaikkia alkioita on siirretty yksi paikka vasemmalle päin. Ensimmäinen alkio siirretään viimeiseksi. Alkuperäistä argumenttina annettua taulukkoa ei saa muuttaa.

Esimerkki:

```
int[] luvut = { 1, 2, 3 };
int[] kierretty = KierraTaulukkoa(luvut);
```

Tämän seurauksena kierretty-taulukko sisältäisi luvut 2, 3, 1 .

Malliratkaisu

Tapa 1:

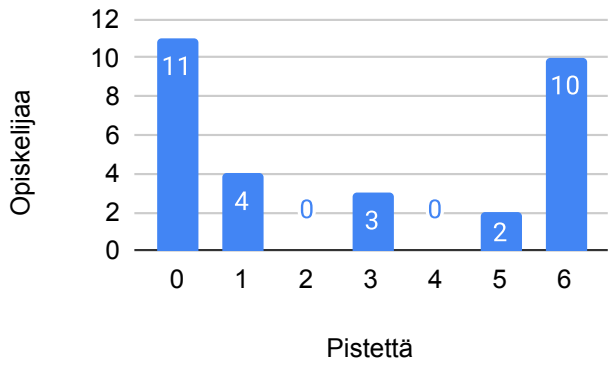
```
public static int[] KierraTaulukkoa(int[] taulukko)
{
  if (taulukko.Length == 0) return taulukko;
  int[] uusi = new int[taulukko.Length];
  int temp = taulukko[0];
  for (int i = 1; i < taulukko.Length; i++)
  {
    uusi[i - 1] = taulukko[i];
  }
  uusi[uusi.Length - 1] = temp;
  return uusi;
}
```

Tapa 2:

```
public static int[] KierraTaulukkoa(int[] taulukko)
{
  int pituus = taulukko.Length;
```

```
  int[] kierretty = new int[pituus];
  for(int i = 0; i < pituus; i++)
  {
    kierretty[i] = taulukko[(i + 1) % pituus];
  }
  return kierretty;
}
```

Tehtävä 2 a (3 p.)



Tehtävän 2 (sekä a- että b-kohdat) pistejakauma.

Tee aliohjelma TulostaLuvut. Aliohjelma ei ota parametreja eikä palauta arvoa. Tee myös luokka, pääohjelma ja tarvittavat using-lauseet. Kutsu aliohjelmaasi yhden kerran pääohjelmassa.

Vaatimukset

1. Aliohjelman tulee tulostaa silmukoita hyväksi käyttäen seuraavanlainen kuvio:

```
1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

2. Jos ratkaisussa ei käytetä silmukoita, tulos on 0 pistettä.

3. Jos ratkaisussa kirjoitetaan useita tulostuskomentoja putkeen (ts. tehdään tulostus manuaalisesti), tulos on 0 pistettä.

4. Yhtään if-lausetta ei saa käyttää. Silmukan toistoehtoa ei luonnollisesti lasketa tässä kielletyksi if-lauseeksi.

5. Aliohjelma ei saa tulostaa mitään muuta.

Malliratkaisu

Dokumentaatiot on jätetty tässä pois tilan säästämiseksi.

```
using System;

public class Tulostus
{
  public static void Main()
  {
    TulostaLuvut();
  }

  public static void TulostaLuvut()
  {
    int alku = 1; int loppu = 5;
    for (int i = alku; i <= loppu; i++)
    {
      for (int j = alku; j <= i; j++)
      {
        Console.Write($"{j} "); // tai ...Write(j + " ");
      }
      for (int j = i - 1; j >= alku; j--)
      {
        Console.Write($"{j} ");
      }
      Console.WriteLine();
    }
  }
}
```

Tehtävä 2 b (3 p.)

Tee aliohjelma TulostaLuvut, joka ottaa parametreina kaksi kokonaislukua. Ensimmäinen parametri on pienin tulostettava luku ja toinen parametri on suurin tulostettava luku. Aliohjelma ei palauta arvoa. Tee myös luokka, pääohjelma ja tarvittavat using-lauseet. Kutsu aliohjelmaasi kaksi kertaa pääohjelmassa seuraavasti:

```
TulostaLuvut(1, 5);
TulostaLuvut(-2, 2);
```

Aliohjelman tulee toimia kaikenlaisilla kokonaislukusyötteillä. Muut vaatimukset ovat kuten a-kohdan vaatimukset 2–5.

Esimerkkejä:

- Kutsuttaessa TulostaLuvut(1, 5) tuloste on kuten a-kohdassa.
- Kutsuttaessa TulostaLuvut(-2, 2) tuloste on:

```
-2
-2 -1 -2
-2 -1 0 -1 -2
-2 -1 0 1 0 -1 -2
-2 -1 0 1 2 1 0 -1 -2
```

- Kutsuttaessa TulostaLuvut(3, 3) tuloste on 3.

Huomautus: Voit olettaa, että funktiota kutsutaan aina kokonaisluvulla; vääränlaisen syötteen käsittelyä ei tarvitse tehdä.

Malliratkaisu

Luokka ja using-lauseet kuten a-kohdassa.

```
public static void Main()
{
  TulostaLuvut(1, 5);
  TulostaLuvut(-2, 2);
}

public static void TulostaLuvut(int alku, int loppu)
{
  for (int i = alku; i <= loppu; i++)
  {
    for (int j = alku; j <= i; j++)
    {
      Console.Write($"{j} "); // tai ...Write(j + " ");
    }
    for (int j = i - 1; j >= alku; j--)
    {
      Console.Write($"{j} ");
    }
    Console.WriteLine();
  }
}
```

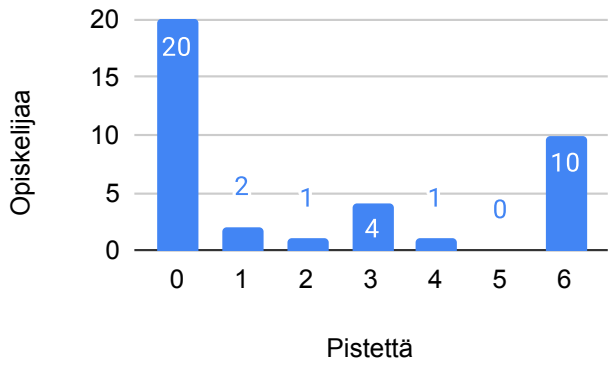
Tehtävä 3 (6 p.)

Tee funktio YhteisiaMerkkeja.

Vaatimukset: Funktio ottaa parametreina kaksi merkkijonoa, ja palauttaa kokonaislukuna tiedon, kuinka monta yhteistä merkkiä näissä jonoissa on.

Esimerkkejä:

- Argumenteilla “mikki” ja “pukki” tulos on 3, koska molemmista jonoista löytyy merkit k, k ja i.
- Argumenteilla “kauppa” ja “soppakauppa” tulos on 6, sillä kummastakin löytyy merkit k, a, u, p, p ja a. Toisaalta soppakauppa-jonosta löytyisi enemmänkin merkkejä, mutta niitä ei löydy ensimmäisestä jonosta.



Tehtävän 3 pistejakauma.

Ohje/vinkki: Tehtävässä voi hyödyntää Dictionary-tietorakennetta (sanakirja), jossa avaimena on merkki (char) ja arvona jonosta löytyvien merkkien lukumäärä. Alla on hyödyllisiä sanakirja-olion metodeja:

- Add(avain, arvo): Lisää sanakirjaan uuden avaimen ja sille arvon. (ks. C#-dokumentaatio[2])
- ContainsKey(avain): Sisältääkö sanakirja argumenttina annettua avainta. Palauttaa true tai false. (ks. C#-dokumentaatio[3])

Sanakirja-olion avain-arvo-parien läpikäyminen onnistuu esimerkiksi seuraavasti:

```
foreach (KeyValuePair<char, int> avainJaArvo in sanakirja)
{
    // avainta voit käsitellä: avainJaArvo.Key
    // arvoa voit käsitellä: avainJaArvo.Value
}
```

Muitakin tapoja tehtävän ratkaisuun on olemassa, eikä sanakirjan käyttö ole pakollista.

Malliratkaisu

Ratkaisutapoja on useita. Tässä kolme esimerkkiä, joista ensimmäinen perustuu sanakirjan käyttöön. Kahdessa jälkimmäisessä muokataan merkkijonoja joko muuttamalla ne listamuotoon tai sitten suoraan string-olioita.
Tapa 1:

```
public static int YhteisiaMerkkeja(string s1, string s2)
{
    Dictionary<char, int> merkitJaMaarat = new Dictionary<char, int>();

    for (int i = 0; i < s1.Length; i++)
```

[2] https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2.add?view=net-6.0
[3] https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2.containskey?view=net-6.0

```
    {
        if (s2.Length == 0) break;
        char merkkiNyt = s1[i];
        if (s2.Contains(merkkiNyt))
        {
            if (merkitJaMaarat.ContainsKey(merkkiNyt))
            {
                merkitJaMaarat[merkkiNyt]++;
            }
            else
            {
                merkitJaMaarat.Add(merkkiNyt, 1);
            }
            s2 = s2.Remove(s2.IndexOf(merkkiNyt), 1);
        }
    }

    int summa = 0;
    foreach (KeyValuePair<char, int> kv in merkitJaMaarat)
    {
        summa += kv.Value;
    }
    return summa;
}
```

Tapa 2:

```
public static int YhteisiaMerkkeja(string jono1, string jono2)
{
  int yhteisia = 0;
  foreach (char merkki in jono1)
  {
    if (jono2.Contains(merkki))
    {
        int paikka = jono2.IndexOf(merkki);
        jono2 = jono2.Remove(paikka, 1);
        yhteisia++;
    }
  }
  return yhteisia;
}
```

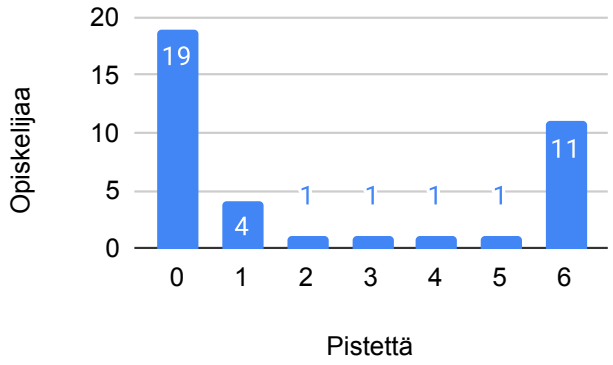
Tapa 3:

```
public static int YhteisiaMerkkeja(string jono1, string jono2)
{
  int yhteisia = 0;
  List<char> jono1Listana = jono1.ToList();
```

```
  List<char> jono2Listana = jono2.ToList();

  foreach (char merkki in jono2Listana)
  {
    if (jono1Listana.Contains(merkki))
    {
      jono1Listana.Remove(merkki);
      yhteisia++;
    }
  }
  return yhteisia;
}
```

Tehtävä 4 (6 p.)



Tehtävän 4 pistejakauma.

Toteuta seuraava ohjelma (luokka, pääohjelma, tarvittavat using-lauseet, tarvittavat funktiot, dokumentaatiokommentit).

Vaatimukset

1. Luo pääohjelmassa taulukko, jonka pituus on satunnainen väliltä 10..20. Taulukko sisältää satunnaisia kokonaislukuja väliltä -10..10.

2. Tee funktio Jarjesta joka ottaa kyseisen taulukon ja järjestää sen nousevaan järjestykseen (pienemmät luvut ennen suurempia lukuja) alla olevan algoritmin mukaisesti.

Algoritmi funktiolle Jarjesta

1. Käydään taulukko läpi alusta loppuun.

2. Mikäli kaksi perättäistä alkiota eivät ole järjestyksessä (vasemmalla oleva alkio on suurempi kuin oikealla), vaihdetaan näiden alkioiden paikkoja keskenään.

3. Käsitellään jokainen perättäisten alkioiden pari kohdassa 2 kuvatulla tavalla.

4. Jos tehtiin alkioiden vaihtoja (ks. kohta 2) aloitetaan uudestaan kohdasta 1.
5. Kun järjestäminen on tehty, funktio palauttaa järjestetyn taulukon.
6. Huom! Parametrina saatua taulukkoa EI saa muuttaa vaan funktion tulee tehdä uusi taulukko.

Malliratkaisu

Algoritmi tunnetaan nimellä kuplalajittelu (engl. bubble sort).

```
/// @author Antti-Jussi Lakanen
/// @version 25.5.2022
///
/// <summary>
/// Tehtävä 4. Lukujen järjestäminen.
/// </summary>
public class T4_BubbleSort
{
  /// <summary>
  /// Luodaan taulukko ja kutsutaan funktiota.
  /// </summary>
  public static void Main()
  {
    Random r = new Random();
    int[] luvut = new int[r.Next(10, 21)];
    for (int i = 0; i < luvut.Length; i++)
    {
      luvut[i] = r.Next(-10, 11);
    }

    // Alla olevaa lausetta ei kirjaimellisesti ottaen
    // tehtävässä vaadittu.
    int[] jarjestetty = Jarjesta(luvut);
  }

  public static int[] Jarjesta(int[] luvut)
  {
    int[] jarjestetty = (int[])luvut.Clone();

    // Tai seuraavasti:
    // int[] jarjestetty = new int[luvut.Length];
    // for (int i = 0; i < luvut.Length; i++)
    //   jarjestetty[i] = luvut[i]

    int i = 1;
    while (i < jarjestetty.Length)
    {
      if(jarjestetty[i-1] > jarjestetty[i])
```

```
      {
        int temp = jarjestetty[i];
        jarjestetty[i] = jarjestetty[i - 1];
        jarjestetty[i - 1] = temp;
        i = 1;
      } else
      {
        i++;
      }
    }
    return jarjestetty;
  }
}
```