Taulukot
Merkkijonot

Olio-ohjelmointi

Lapin AMK / EM

Viitetietotyypit

- Viitetyyppejä ovat merkkijono, taulukko, luokka, rajapinta ja delegaatti
- Viitetyyppisten muuttujien arvo on null tai määrittelemätön tai ne viittaavat pysyvällä muistialueella keossa (heap) olevaan olioon

Taulukot

- Taulukko luodaan new-operaattorilla. Taulukkoa luotaessa on ilmoitettava sen koko.
- Esim. kokonaislukutaulukko:

```
int[] taulu = new int[5];
```

- Koko voidaan määrätä myös vasta ajon aikana, esim.

```
int koko;
koko = 10;
int[] taulu = new int[koko];
```

Taulukot

- Taulukot voivat olla myös moniulotteisia:

```
double[,] taulu = new double[3,5];
```

- Taulukko voidaan luoda myös alkuarvolistan avulla:

```
int[] taulukko = {1,2,3};
```

- Huomaa ero C-kieleen: hakasulut ovat aina tietotyypin eikä muuttujan perässä
- Taulukon koko saadaan Length-ominaisuudesta. Indeksit alkavat nollasta, joten taulukon viimeisen alkion indeksi on Length-1

Taulukoiden käsittely

- System.Array-luokka sisältää staattisia metodeja taulukoiden käsittelyyn:
  - Haku: BinarySearch, Find, IndexOf
  - Lajittelu: Sort
  - Tyhjentäminen: Clear
  - Kopiointi: Copy
  - Käänteinen järjestys: Reverse
  - ym.

foreach – in

- Tavallisen for-silmukan lisäksi C#:ssa on foreach-in –silmukkarakenne, jolla voidaan selata taulukoiden ja kokoelmien/säiliöolioiden (collection) sisältö läpi
- Syntaksi:

```
foreach (tyyppi muuttuja in lauseke)
{
  // koodia
}
```

foreach – in

- Syntaksissa mainittu ***`lauseke`*** tulee olla joko enumerable-tyyppinen kokoelma tai taulukko
- Kun silmukkaa suoritetaan, kullakin kierroksella sijoitetaan seuraava kokoelman jäsen ***`muuttujan`*** arvoksi

foreach – in

- Esimerkki, tulostetaan komentoriviparametrit:

```
using System;
namespace ParametrienTulostus
{
  class Parametrit {
    static void Main(string[] args) {
       foreach (string s in args) {
         Console.WriteLine(s);
       }
    }
  }
}
```

Merkkijonoluokka string

- Merkkijonoja käsitellään string (System.String) -tyypin avulla
  - string (pieni alkukirjain) on alias-nimi System.String -luokalle
  - String-olio on muuttamaton (immutable): olemassaolevan merkkijonon sisältöä ei voi muuttaa
  - String-luokassa on lukuindeksoija, jonka avulla merkkijonon yksittäiseen merkkiin voidaan viitata taulukkomaisesti hakasuluilla, esim. s[0] on merkkijonon 1. merkki. Merkkejä voidaan vain lukea, ei muuttaa
  - Merkkijonon pituus saadaan Length-muuttujasta

Merkkijonojen vertailu

- Merkkijonojen yhtäsuuruutta voidaan verrata usealla eri tavalla:
  - CompareTo(string) vertaa kutsuvaa merkkijonoa argumenttina annettuun (oliometodi)
    - Palauttaa negatiivisen kokonaisluvun, jos metodia kutsuva merkkijono on aakkosissa ennen kuin parametrina annettu merkkijono
    - Palauttaa positiivisen kokonaisluvun, jos metodia kutsuva merkkijono on aakkosissa parametrina annetun merkkijonon jälkeen
    - Palauttaa nollan, jos merkkijonot ovat samat
  - Compare(string, string) vertaa kahden argumenttina annetun merkkijonon yhtäsuuruutta (luokkametodi).
  - Equals(string) palauttaa arvon true (tosi), jos merkkijonot ovat yhtäsuuret, muutoin false (epätosi)
  - Javasta poiketen == ja != operaattorit on kuormitettu ja ne vertaavat merkkijonojen sisällön yhtäsuuruutta, eivät viitemuuttujien yhtäsuuruutta.
- Myös + ja += operaattorit on kuormitettu: merkkijonoja voidaan liittää yhteen (hidas, jos yhdistämisiä useita, ei pidä käyttää silmukassa)

Muutettava merkkijono: StringBuilder

- System.Text.StringBuilder määrittelee merkkijonon, jonka sisältöä voi muuttaa
- Luokassa on sekä luku- että kirjoitusindeksoija: hakasulkuviittauksella voidaan myös muuttaa yksittäistä merkkiä merkkijonossa, esim. s[0]=’A’;
- Append-metodilla voidaan lisätä merkkejä loppuun

String vai StringBuilder

- Pienten merkkijonojen kohdalla StringBuilderin käyttö vie stringiin verrattuna enemmän muistia ja prosessoriaikaa
- Käytä string-oliota, kun kyse on
  - metodin parametrista
  - vakiomerkkijonosta
  - tulostettavasta merkkijonosta
  - tietokantaan tallennettavasta merkkijonosta
- Käytä StringBuilderia, jos sinun täytyy
  - käsitellä erittäin suuria merkkijonoja
  - yhdistää merkkijonoja neljä tai enemmän
  - manipuloida merkkijonojen sisältöä

StringBuilder stringiksi

- Monet BCL:n metodit hyväksyvät argumentiksi vain viitteen string-tyyppisen olioon
- StringBuilderin ToString-metodi palauttaa sisällön string-tyyppisenä viitteenä
- toString-metodin käyttö on turvallista, koska se ei luo uutta string-oliota, vaan palauttaa stringin osoitteen StringBuilder-olion sisällä

Merkkijonon pilkkominen

- Merkkijono (string-olio) voidaan pilkkoa osiin kutsumalla Split-metodia
- Split-metodille annetaan parametrina osien erotinmerkki/-merkit merkkitaulukkona, esim.:
  ```
  char[] delimiters = {'+', '-'};
  ```
- Jos osien erotin on sekin merkkijono, Split-metodille annetaan parametriksi erotinmerkkijonot string-taulukkona, esim.:
  ```
  string[] delimiters = new string[]{"://"};
  ```
  - Toiseksi parametriksi tulee antaa tässä tapauksessa StringSplitOptions.None tai StringSplitOptions.RemoveEmptyEntries
- Split-metodille voidaan antaa parametrina myös osien lukumäärä. Jos lukumääräparametria ei anneta, Split palauttaa kaikki osat
- Split palauttaa merkkijonon osat string-taulukkona

Esimerkki: merkkijonon pilkkominen

```
public class StringSplit {
   public static void Main()
   {
      string delimStr = " ,.:";
      char[] delimiter = delimStr.ToCharArray();

      string words = "one two,three:four.";
      string[] split = null;

      Console.WriteLine("The delimiters are {0}", delimStr);

      split = words.Split(delimiter);
      foreach (string s in split)
      {
         Console.WriteLine("{0}", s);
      }
   } // Main
} // Class StringSplit
```