Operaattorit Valinta- ja toistorakenteet

Olio-ohjelmointi

Lapin AMK / EM

Operaattorit

- Operaattorien avulla tehdään laskutoimituksia, verrataan arvoja toisiinsa ja sijoitetaan arvoja muuttujiin
- Ovat pitkälti samat kuin C-kielessä
- Sijoitusoperaattori on =. Esim. **`int i = 10;`**
- Yhtäsuuruuden vertailuoperaattori on **==**. Esim. **`if (i==10)`**
- Vertailuoperaattorit ovat yhtäsuuruus **==**, erisuuruus **!=**, suurempi kuin **>**, pienempi kuin **<**, suurempi tai yhtäsuuri kuin **>=**, pienempi tai yhtäsuuri kuin **<=**
- Käytössä on normaalit aritmeettiset operattorit +, -, /, %, * jne.

Operaattorit

- Loogisia operaattoreita ovat:
  - ! EI, esim. **`if !(x==3) on sama kuin if (x!=3)`**
  - || TAI, esim. **`if (x==2 || x==3)`**
  - && JA, esim. **`if (x==1 && y==2)`**
- Bittitason operaattorit ovat
  - ~ EI-operaatio, kääntää luvun kaikkien bittien arvot vastakkaisiksi
  - & JA-operaatio, käytetään esim. sammuttamaan halutut bitit tai testaamaan, onko tietty bitti päällä
  - | TAI-operaatio, käytetään esim. sytyttämään tietyt bitit
  - ^ XOR-operaatio, kuin TAI, mutta 1 ^ 1 on 0. Esim. on/off –kytkin
- Bittien siirto-operaattoreita <<, >> ja >>> voidaan käyttää bittien vyöryttämiseen vasemmalle tai oikealle, mikä vastaa joko kertomista tai jakamista kahden potensseilla

Operaattorit

- Arvonmuunto-operaattoreita ovat:
  - a++ lisää a:han yksi
  - a-- vähennä a:sta yksi
  - a+=b lisää a:han b
- Arvonmuunto-operaattorit, jatkoa
  - a -= b vähennä a:sta b
  - Samat myös muille peruslaskutoimituksille
- Olioiden ja taulukoiden käsittelyssä tarvittaviin operaattoreihin palataan myöhemmin

Operaattorit

- Sijoitusoperaattorit ovat operaattorien suoritusjärjestyksessä alimmalla tasolla
- Jos olet epävarma operattorien suoritusjärjestyksestä, käytä sulkumerkkejä varmistamaan haluttu järjestys. Usein sulkujen käyttäminen myös selkeyttää ohjelmaa

Valintarakenteet

- if-lause on muotoa

```
if (ehto)
   lause1;
else
   lause2;
```

- Lause1 suoritetaan vain, jos annettu ehto on tosi. Muutoin suoritetaan lause2. Else-osa ei ole pakollinen.

Valintarakenteet

- Käyttämällä aaltosulkuja if-lauseessa voidaan suorittaa yhden lauseen sijaan useampia lauseita. Esimerkiksi:

```
if (a==1)
{
   a=0;
   b=2;
   c=4;
}
```

Valintarakenteet

- switch-lauseen syntaksi on seuraava:

```
switch (muuttuja)
{
  case arvo1:
    lause1;
    break;
  case arvo2:
    lause2;
    break;
  default:
    lause3;
}
```

Valintarakenteet

- switch-lause tutkii annetun muuttujan arvon ja haarautuu vastaavaan case-haaraan. Mikäli minkään case-haaran arvo ei vastaa muuttujan arvoa, suoritetaan default-haara
- Jokainen case-haara tulee päättää break-lauseeseen ellei suorituksen haluta valuvan seuraavaan case-haaraan
- switch-lausetta voidaan käyttää vain kokonaislukumuuttujille

Toistorakenteet

- for-lause muodostuu kolmesta osasta, ensimmäisessä osassa alustetaan laskuri, toisessa annetaan toistoehto ja kolmannessa lisätään laskuria
- for-lauseen syntaksi on seuraava:

```
for (alkuarvo; ehto; lisäys)
  lause;
```

Toistorakenteet

- for-rakennetta seuraavaa lausetta tai aaltosuluilla rajattua lohkoa suoritetaan niin monta kertaa kuin annettu ehto on voimassa. Esimerkki:

```
for (int i =0; i < 10; i++)
{
  Console.WriteLine(i);
}
```

Toistorakenteet

- while-lause muodostuu pelkästä ehdosta ja sitä seuraavasta lauseesta tai lohkosta. while-lausetta toistetaan niin kauan kuin ehto on tosi
- Jos ehto on epätosi jo while-lauseeseen tultaessa, sen sisältämää lausetta tai lohkoa ei suoriteta kertaakaan
- while-lauseen syntaksi on seuraava:

```
while(ehto)
   lause;
```

Toistorakenteet

- Esimerkki:

```
int i = 0;
while (i < 10)
{
   Console.WriteLine(i);
   i++;
}
```

Toistorakenteet

- do-while -rakenne poikkeaa while-rakenteesta siten, että sen sisältämä lause suoritetaan aina vähintään yhden kerran, vaikka ehto olisi epätosi
- do-while –rakenteen syntaksi on seuraava:

```
do
   lause;
while (ehto);
```

Toistorakenteet

- Erona C-kieleen C# sisältää foreach –toistorakenteen
- foreach –rakennetta voidaan käyttää taulukon tai säiliöolion sisällön läpikäymiseen elementti kerrallaan
- foreach mahdollistaa vain elementtien luvun, ei muokkaamista, esimerkki:

```
String[] nimet = {"Ville","Vilma","Velmu"};
foreach (String nimi in nimet)
{
    Console.WriteLine(nimi);
}
```