Ohjelmoinnin perusteet

Ilkka Pokkinen

Päivitetty 21.9.2021

Sisällys

[1. Sanasto 3](#_Toc83850654)

[2. Johdanto 3](#_Toc83850655)

[3. Koodin kommentointi 3](#_Toc83850656)

[4. Muuttujat 3](#_Toc83850657)

[4.1 Tyyppi 4](#_Toc83850658)

[4.2 Elinaika 4](#_Toc83850659)

[5. Operaattorit 4](#_Toc83850660)

[5.1 Aritmeettiset operaattorit 5](#_Toc83850661)

[5.2 Vertailuoperaattorit 5](#_Toc83850662)

[5.3 Loogiset operaattorit 5](#_Toc83850663)

[5.4 Operaattoreiden ratkaisujärjestys 6](#_Toc83850664)

[6. Ehtolauseet 6](#_Toc83850665)

[7. Toistorakenteet 7](#_Toc83850666)

[7.1 While-loop 7](#_Toc83850667)

[7.2 Do-while-loop 8](#_Toc83850668)

[7.3 For-loop 8](#_Toc83850669)

[8. Taulukko ja lista 9](#_Toc83850670)

[9. Funktiot 9](#_Toc83850671)

[10. Luokat ja oliot 10](#_Toc83850672)

[11. Enum 12](#_Toc83850673)

# Sanasto

# Johdanto

# Koodin kommentointi

Kommenteilla dokumentoidaan koodin toiminnallisuutta, eikä niitä lueta ohjelman käännön tai suorituksen aikana. Koodin toiminnasta tai sen tarkoituksesta voi olla hyvinkin vaikea saada yleissilmäystä, jos sitä ei ole millään tavalla kommentoitu. Kommentointi kannattaa opetella varhain oman oppimisen vuoksi.



Koodiesimerkki 1 kommentit koodissa

# Muuttujat



Koodiesimerkki 2 Muuttujan esittely ja asettaminen

Muuttuja koostuu kahdesta osasta: datatyypistä ”int” ja nimestä ”kokonaisluku”. Rivin lopussa oleva puolipiste erottelee toisistaan ohjelman koodirivit. Seuraavan koodiesimerkin molemmat osat toimivat, mutta jälkimmäistä on helpompi lukea oikeanlaisen rivityksen takia.



Koodiesimerkki 3 rivinlopetusmerkki

Yhtäsuuruusmerkki on sijoitusoperaattori, jolla muuttujan arvo asetetaan. Merkki ei siis tarkoita samansuuruisuutta. kokonaisluku = kokonaisluku + 1; on aivan validi lause, jossa kokonaisluku -nimisen muuttujan uudeksi arvoksi asetetaan vanha arvo plus yksi.

Ohjelmoitaessa muuttujan nimetään sellaisiksi, että niiden nimistä voidaan päätellä muuttujien tarkoitus. Jos esitellään muuttuja, jonka arvoksi asetetaan ajoneuvon nopeus, se nimetään esimerkiksi sanalla ”nopeus”, ”ajoneuvonNopeus” tai ”kmph”. Kun muuttujan nimi sisältää enemmän kuin yhden sanan, on yleinen tapa aloittaa ensimmäinen sana pienellä ja sitä seuraavat isolla kirjaimella. Muuttujan nimi ei voi sisältää välilyöntejä.

## Tyyppi

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types>

Kaikki muuttujat ovat muistissa vain ykkösiä ja nollia peräkkäin. Tyypillä määritellään, miten näitä lukuja tulkitaan.

|  |  |  |
| --- | --- | --- |
| **Yleisiä alkeistyyppejä** | | |
| **Tyyppi** | **Koko** | **Kuvaus** |
| bool | 1 tavu | Sisältää totuusarvon true tai false |
| char | 2 tavua | Sisältää yhden UTF-16 merkin |
| int | 4 tavua | Kokonaisluku välillä -2147483648 ja 2147483647 |
| float | 4 tavua | Desimaaliluku 6–9 merkin tarkkuudella |
| string | 2 tavua per merkki | Merkkijono |

## Elinaika

Aaltosulkujen sisällä olevaa ohjelman osaa kutsutaan lohkoksi. Lohkon sisällä esitellyt muuttujat poistuvat muistista lohkon päätyttyä.

Text

Description automatically generated

Kuva 1 muuttujan elinajan päättyminen

# Operaattorit

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/>

Ohjelmointikielen dokumentaatiossa on määritelty operaattoreiden toiminnallisuus. Esimerkiksi kokonaisluvuilla + -operaattori laskee niiden summan, mutta merkkijonojen kesken tuloksena on niiden yhdistelmä.



Koodiesimerkki 4 operaattoreita eri datatyypeillä

## Aritmeettiset operaattorit

|  |  |
| --- | --- |
| \* | Kertolasku |
| / | Jakolasku |
| % | Jakojäännös (modulo) |
| + | Yhteenlasku |
| - | Vähennyslasku |

## Vertailuoperaattorit

|  |  |
| --- | --- |
| < | Pienempi kuin |
| > | Suurempi kuin |
| <= | Pienempi tai yhtä suuri kuin |
| >= | Suurempi tai yhtä suuri kuin |
| == | Yhtä suuri kuin |
| != | Eri suuri kuin |

## Loogiset operaattorit

|  |  |
| --- | --- |
| && | AND, lopputulos true, jos molemmat puolet ovat true |
| || | OR, lopputulos true, jos edes yksi on true |
| ! | Negaatio, kääntää arvon |



Koodiesimerkki 5 loogiset operaattorit

## Operaattoreiden ratkaisujärjestys

<https://docs.microsoft.com/en-us/cpp/c-language/precedence-and-order-of-evaluation?view=msvc-160>

Aritmeettiset operaattorit seuraavat samaa ratkaisujärjestystä kuin matematiikassa tavallisestikin ja suoritetaan ennen muita operaattoreita. Näiden jälkeen suoritetaan vertailuoperaattorit ja viimeisenä loogiset, AND-operaattori ennen OR-operaattoria.



Koodiesimerkki 6 operaattoreiden ratkaisujärjestys

Sulut toimivat myös samalla tavalla kuin matematiikassa. Koodiesimerkissä kuusi ei ole pakko käyttää sulkuja, mutta monimutkaisemmissa lauseissa voidaan vähentää virheiden riskiä ja helpottaa koodin lukevuutta käyttämällä sulkuja.

# Ehtolauseet

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/if-else>

Ehtolauseella voidaan eristää osa ohjelmasta. Lausetta seuraava lohko suoritetaan vain, jos ehtolauseen vertailu tuottaa true boolean arvon. Ehtolauseessa voidaan tehdä peräkkäisiä vertailuja, joista ensimmäisen true arvon antavan lohko suoritetaan.



Koodiesimerkki 7 if-else lause

# Toistorakenteet­

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/while>

Toistorakenteilla voidaan suorittaa ohjelman lohko useita kertoja jonkin ehdon ollessa tosi. Kun toistorakenteesta poistutaan, ohjelman suoritus jatkuu lohkon jälkeen.

Break -sanalla voidaan poistua toistorakenteesta sen suorituksen keskellä.

## While-loop

While-loop on yksinkertaisin toistorakenne ja muut toistorakenteet voitaisiin korvata sillä. While-loopin ehto tarkastetaan lohkon alussa.



Koodiesimerkki 8 kertoma while-loopilla

## Do-while-loop

Do while loopin ehto ratkaistaan lohkon päätyttyä.



Koodiesimerkki 9 do-while loop

## For-loop

For-loop on yleisin toistorakenne. Sen alussa voidaan esitellä muuttujia lohkon sisään, määritellä ehto, jolla poistutaan lohkosta ja määritellä asioita tapahtumaan jokaisen kierroksen jälkeen.



Koodiesimerkki 10 tyypillinen for-loop

# Taulukko ja lista

Taulukko on tietorakenne, jolla varataan tilaa usealle muuttujalle, jotka ovat muistissa peräkkäin. Taulukosta haetaan muuttuja sen indeksillä, joka kertoo sen etäisyyden taulukon alusta. Ensimmäinen muuttuja on etäisyydellä nolla, toinen etäisyydellä yksi jne. Taulukko on hyvin tehokas tietorakenne, mutta sen pituutta ei voi muuttaa kirjoittamatta koko taulukkoa uudelleen muistiin.

Lista on tietorakenne, jonka pituus voi muuttua dynaamisesti, mutta sen lukeminen ei ole yhtä nopeaa kuin taulukon. Listan alkiot eivät välttämättä sijaitse peräkkäin muistissa, joten alkion sijaintia ei voida laskea sen indeksistä.



Koodiesimerkki 11 taulukon ja listan luonti

# Funktiot

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/local-functions>

Funktiolla vältetään saman koodin uudelleenkirjoittamista ja selkeytetään ohjelman rakennetta.



Koodiesimerkki 12 funktio

# Luokat ja oliot

Luokalla (class) määritellään tyyppi. Luokan muistiin luotua instanssia kutsutaan olioksi (object).



Koodiesimerkki 13 esimerkki luokasta



# Enum