Ohjelmointikielet ja peruskäsitteet

Turun yliopisto
University of Turku

Viime luennon satoa - tutoriaaleista

- Tutoriaalien kone-/ohjelmistovaatimukset
  - Ei tarvita erillisiä ohjelmistoja
  - Moderni selain riittää
    - Internet Explorer ei toimi
- Tableteilla voi olla vaikeuksia ☹

Viime luennon satoa - algoritmiesimerkki

```
s := 0
k := 1
WHILE k <= T.length DO
  s := s + T[ k ]
  k := k + 1
ENDWHILE
keskiarvo := s / T.length
```

Toivottavasti ette vielä täysin ymmärrä ☺

Sisältö

- Ohjelmointikielet ja peruskäsitteet
- Muuttujat
- Valinta

Algoritmin ymmärrys

- Prosessorin tulee kyetä tulkitsemaan algoritmi
  - Ymmärtää esitysmuoto
  - Kyetä suorittamaan operaatiot
- Algoritmin esityksen *ymmärrys* jakautuu kahteen vaiheeseen:
  1. Ymmärtää symbolit joilla algoritmi esitetään
     - Käytetyn kielen aakkosto
     - Käytetyn kielen kielioppi
  2. Ymmärtää jokaisen askeleen merkitys

Kielen syntaksi – miten ohjelma kuvataan?

- Kielen ***syntaksi***:
  - Määrittää kielen lauseiden muodon eli kielioppisäännöt
  - Määrää miten kielen symboleja saa laillisesti käyttää
- ***Syntaksi siis määrää, millaisia ovat kielen oikein muodostetut lauseet.***
- Syntaktisesti oikea ohjelma noudattaa kielen syntaksia

Kielen semantiikka – mitä ohjelma tarkoittaa?

- Kielen semantiikka
  - Oppi lauseiden merkityksestä eli tiettyjen ilmaisujen merkitys kielessä
- ***Semantiikka määrää, mitä kielen lauseet tarkoittavat.***
- Semantiikka vastaa siis kysymykseen *Mitä?*

- Esimerkkejä:

Hauki on kala. (syntaktisesti ja semanttisesti oikein)

Delfiini on kala. (syntaktisesti oikein, semanttisesti väärin)

Ohjelmointikielten semantiikka

- Luonnollisiin kieliin verrattuna yksinkertaista
- Syntaksi ja semantiikka eroteltu ohjelmointikielissä
  - Algoritmin askeleet voivat olla syntaktisesti oikein, mutta semanttisesti merkityksettömiä

Ohjelmointikielen kielioppi

- Ohjemointikielten syntaksi nykyään tarkasti määritelty
- 60-luvulta lähtien syntaksi määritelty käyttäen ***kielioppia***
- Esim. Pascal-kielen kielioppi koostuu n 150 kielioppisäännöstä
  - Kielioppisäännöt määrittävät kielen syntaksin täsmällisesti
- Ohjelmointikielten semanttiikkaa ei (yleensä) kyetä määrittelemään täsmällisesti

Algoritmin suorittaminen

- Algoritmin *suorittaakseen* prosessorin tulee
  1. tuntea algoritmin kuvauksessa käytetyt symbolit
  2. osata liittää kuhunkin askeleeseen sen merkitys suoritettavien operaatioiden muodossa
  3. kyetä suorittamaan operaatiot

- Ohjelmointikielen *kääntäjä* suorittaa vaiheet 1 ja 2
- Kääntäjä muuttaa ohjelman jokaisen askeleen sopiviksi operaatioiksi
  - Prosessori suorittaa operaatiot

Syntaksivirheet

- Syntaksivirhe tarkoittaa kielen syntaksin rikkomista
- Virhe estää virheellisen lauseen suorittamisen
  - Usein yksikin virhe estää koko ohjelman suorittamisen

- Esimerkkejä:

```
// virheellinen esim. Pascal-kielessä, oikein C:ssä
c += d

// kirjoitusvirhe varatussa sanassa
funktion doSomething()
```

Virheiden havaitseminen

- Syntaksivirheet havaitaan vaiheessa 1 (kielen kuvauksen tunteminen)
- Jotkin semanttiset virheet vaiheessa 2 (askeleiden merkitys)
- Jotkin semanttiset virheet vaiheessa 3 (operaatioiden suoritus)

Virheiden havaitseminen - esimerkki

- Esimerkkilause: “Kirjoita viikon n:nnen päivän nimi.”

- Syntaktisesti oikein
- Semanttinen oikeellisuus voidaan tarkastaa vasta suoritettaessa
  - Lause on semanttisesti oikein vain tietyille n:n arvoille
  - n:n arvo tiedetään vasta silloin

Loogiset virheet

- Syntaktisesti oikea, virheellisen tuloksen antava ohjelma sisältää ***loogisen virheen***
- Loogiset virheet voidaan havaita esim. vertaamalla algoritmin tuottamaa tulosta oikeaksi tiedettyyn tulokseen
  - Edellyttää oikein tuloksen tuntemista
  - Esim. Lukujen summan laskevan algoritmin tulos tunnetaan
- Loogiset virheet vaikeampi havaita kuin syntaktiset ja semanttiset
- Voidaan välttää tuomalla algoritmin semantiikka selvästi esille
  - Kuvaavien nimien käyttö
  - Ohjelmakoodin sijoitetut kommentit

Algoritmin suoritusjärjestys - peräkkäisyys

- Algoritmin lauseet suoritetaan algoritmin määräämässä järjestyksessä
- ***Peräkkäisyys*** (sequential) on yksi algoritmien perus***ohjausrakenteista*** (flow of control)
- Peräkkäisessä ohjelmassa lauseet suoritetaan
  - Yksi kerrallaan
  - Täsmälleen kerran
  - Siinä järjestyksessä kuin ne ovat algoritmissa
- Viimeisen lauseen suoritus päättää algoritmin suorituksen

Lause ja lauseke

- Algoritmin yksittäisiä käskyjä (tai komentoja) kutsutaan ***lauseiksi*** *(statement)*
- ***Lausekkeella*** *(expression)* tarkoitetaan
  - Luku- tai muita arvoja
  - Muuttujaviittauksia
  - Erilaisia laskutoimituksia (+, -, *, /) tai
  - Muita operaatioita sisältävää ilmausta
- Lausekkeella on sen osista määräytyvä yksikäsitteinen arvo

Lause ja lauseke

- Lauseen ja lausekkeen ero: lausekkeella on arvo, lauseella ei
- Lauseketta ei voi käyttää yksinään, sen on oltava lauseen osa
  - Lausekkeiden asema tosin vaihtelee eri ohjelmointikielissä

- Lausekkeita:

```
3 + 4
x + 7 * y
```

- Lauseita:

```
a := 3 + 4
b := 3 * a
```

Muuttujat

Muuttujat

- Algoritmin halutun tuloksen saavuttamiseen käytetään usein apuna ***muuttujia*** *(variables)*
- Muuttuja
  - On tietyn merkityksen omaava arvo
  - Arvo voi muuttua algoritmin suorituksen aikana
  - Tarkoittaa usein tietokoneen muistipaikkaa
- Algoritmin käskyt muuttavat muistipaikkojen sisältöä (eli muuttujien arvoja)

Muuttujan alustus ja arvon asetus

Asetuslause eli sijoituslause antaa muuttujalle arvon tai muuttaa sen arvoa

muuttuja := lauseke esim: x := 4 + 2

Muuttujan alustus ja arvon asetus

Asetuslause eli sijoituslause antaa muuttujalle arvon tai muuttaa sen arvoa

muuttuja := lauseke esim: x := 4 + 2

- *Muuttuja* on sen muuttujan nimi jonka arvo halutaan asettaa
- *Lauseke* on tälle muuttujalle sopivan arvon tuottava lauseke

Muuttujan alustus ja arvon asetus

muuttuja := lauseke esim: x := 4 + 2

- *Asetuslauseen* semantiikka:
  - Lasketaan asetuslauseen oikean puolen saama arvo
  - Asetetaan saatu arvo vasemman puolen muuttujan (uudeksi) arvoksi
  - Jos muuttujalla oli ennestään arvo, se häviää
- Oikealla puolella olevan lausekkeen jokaisella muuttujalla oltava arvo

Asetuslause ja ohjelmointikielet

- Kurssilla käytettävässä pseudokielessä asetusoperaattori on `:=`
- Useissa ohelmointikielissä käytetään yhtäsuuruusmerkkiä `=`
  - Esim Java ja C
  - Ongelma:
    - Arvojen yhtäsuuruuden vertailuun ei voida käyttää yhtäsuuruusmerkkiä
- Joissakin kielissä muuttujilla on jokin oletusarvo (esim. 0 lukuarvoille)
  - Kurssilla oletamme, että muuttujilla ei ole oletusarvoa

Asetuslause esimerkkejä

```
x := 4
```

- Muuttujan x arvo on lauseen suorittamisen jälkeen 4.

Asetuslause esimerkkejä

```
y := 4 + 3
```

- **Muuttujan y arvo lauseen suorittamisen jälkeen?**

Asetuslause esimerkkejä

```
y := 4 + 3
```

- **Muuttujan y arvo lauseen suorittamisen jälkeen?**
- Vastaus: 7

Asetuslause esimerkkejä

```
x  := 5
y  := x
```

- Muuttujan x arvo on ensimmäisen lauseen suorittamisen jälkeen 5.
- Muuttujan y arvo on toisen lauseen suorittamisen jälkeen sama kuin x:n arvo, eli 5.

Asetuslause esimerkkejä

```
x := 8
y := x + 2
x := 4
```

- **Muuttujien x ja y arvot ohjelman suorittamisen jälkeen?**

Asetuslause esimerkkejä

```
x  := 8
y  := x + 2
x  := 4
```

- **Muuttujien x ja y arvot ohjelman suorittamisen jälkeen?**
- Muuttujan x arvo on ensimmäisen lauseen suorittamisen jälkeen 8.
- Muuttujan y arvo on 2. lauseen suorittamisen jälkeen x:n arvo plus 2, eli 10.
- Muuttujan x arvo on viimeisen lauseen suorittamisen jälkeen 4. Lause ei muuta y:n arvoa, eli y:n arvo on edelleen 10.

Muuttujat ja algoritmin tila

- Algoritmin tila määräytyy muuttujien senhetkisten arvojen mukaan
- Muuttujan arvon asettaminen ja sen muuttaminen on keskeinen operaatio
- Koko ohjelma voidaan ajatella jonona muuttujien arvojen käyttöjä ja niiden arvojen asettamisia

Muuttujan tyyppi

- Muuttujiin ja lausekkeisiin liittyy ***tyypin*** käsite
- Tyyppi kertoo, millainen arvo muistipaikaan tallennetaan
- Tyyppi määrittää myös, arvon tallennusmuodon tietokoneen muistiin

Muuttujien päätyypit

Muuttujan tyyppi	Esimerkkejä arvoista
Kokonaisluku	4 -95 0
Desimaaliluku	3.14 -123.66
Merkki	’z’ ’C’
Totuusarvo	Tosi Epätosi
Merkkijono	”erkki merkkijono esimerkki” ””

Muuttujan tyyppi pseudokielessämme

- Pseudokielessämme emme ilmoita muuttujan tyyppiä
  - Muuttujan tyyppi määräytyy muuttujaan tallennetun arvon perusteella
  - Näin on myös esim. Python-ohjelmointikielessä
- Esim Java on ns. vahvasti tyypitetty kieli:
  - Muuttujan tyyppi on määriteltävä muuttujan esittelyn yhteydessä
  - Esim. int x määrittelee kokonaislukutyyppisen muuttujan x
  - Muuttujaan voidaan tallentaa vain sen tyyppisiä arvoja

Aritmeettinen lauseke

Aritmeettinen lauseke

Aritmeettinen lauseke on lauseke, jonka arvo on luku

esim. 3 * x + y (muuttujat x ja y ovat lukuarvoisia)

- Käytössä matematiikasta tutut operaattorit:
  - +
  - -
  - * (kertolasku)
  - / (jakolasku)
  - % (jakojäännös)

Aritmeettinen lauseke

- Lauseketta laskettaessa:
  - Arvo lasketaan vasemmalta oikealle
    - Aritmeettiset operaattorit * ja / ovat vahvempia (lasketaan ensin)
    - Operaattorit + ja – ovat heikompia
  - Voidaan käyttää sulkeita laskujärjestyksen osoittamiseksi

Aritmeettinen lauseke - esimerkkejä

```
3 * x + 2
```

- Lasketaan ensin 3*x, johon lisätään 2.
- Esim jos x:n arvo on 4, tulee lausekkeen arvoksi 14.

Aritmeettinen lauseke - esimerkkejä

```
3 * (x + 2)
```

- Lasketaan ensin x + 2, jonka tulos kerrotaan 3:lla.
- Esim jos x:n arvo on 4, tulee lausekkeen arvoksi 18.

Aritmeettinen lauseke - esimerkkejä

```
z := 4 * (x + y - 1)
```

- Lasketaan ensin x + y, jonka tuloksesta vähennetään 1.
- Seuraavaksi kerrotaan tämä 4:llä.
- Lopuksi sijoitetaan saatu arvo muuttujaan z.
- **Mikä tulee z:n arvoksi jos x:n arvo on 4 ja y:n -2?**

Aritmeettinen lauseke - esimerkkejä

```
z := 4 * (x + y - 1)
```

- Lasketaan ensin x + y, jonka tuloksesta vähennetään 1.
- Seuraavaksi kerrotaan tämä 4:llä.
- Lopuksi sijoitetaan saatu arvo muuttujaan z.
- **Mikä tulee z:n arvoksi jos x:n arvo on 4 ja y:n -2?**
- Vastaus: lausekkeen arvoksi tulee 4 eli 4 * (4 + (-2) - 1.

- **Huom!** Aritmeettinen lauseke on sijoituslauseen oikea puoli

Looginen lauseke

Looginen eli totuusarvoinen lauseke

Looginen tai totuusarvoinen tai ehtolauseke on lauseke, jonka arvo on totuusarvo tosi (true) tai epätosi (false)

esim. $a < b + 1$

Vertailuoperaattorit

- Lausekkeissa voidaan käyttää ***vertailuoperaattoreita***

Vertailuoperaattori	Esimerkki
Suurempi >	a > 4
Pienempi <	a < -2
Suurempi tai yhtäsuuri >=	a >= b + 2
Pienempi tai yhtäsuuri <=	b <= 0
Yhtäsuuret =	a = b
Erisuuret <>	a <> c

- **Huom!** Esimerkkilauseiden arvot riippuvat muuttujien arvoista.
- Vertailuoperaattorit vaihtelevat ohjelmointikielissä.

Totuusarvoiset lausekkeet - esimerkkejä

```
a < 4
```

- Lausekkeen arvo on tosi, mikäli muuttujan a arvo on pienempi kuin 4. Muulloin lausekkeen arvo on epätosi.

Totuusarvoiset lausekkeet - esimerkkejä

```
a >= b + 2
```

- Lausekkeen arvo on tosi, mikäli muuttujan a arvo on suurempi tai yhtäsuuri kuin aritmeettisen lausekkeen b + 2. Muulloin lausekkeen arvo on epätosi.

Totuusarvoiset lausekkeet - esimerkkejä

```
a - 1 > b
```

- **Kun a := 3 ja b := 2, mikä on lausekkeen arvo?**

Totuusarvoiset lausekkeet - esimerkkejä

```
a – 1 > b
```

- **Kun a := 3 ja b := 2, mikä on lausekkeen arvo?**
- Vastaus:
  - Vasen puoli: a – 1 eli 3 – 1 = 2
  - Oikea puoli: b eli 2
  - Vertailu on siis 2 > 2, mikä ei pidä paikkaansa joten lausekkeen arvo Epätosi

Loogiset operaattorit

- Vertailuoperaattorien käyttö ohjelmointikielissä on rajattua
  - Esim vertailu a < b < c ei ole mahdollista (paitsi harvoissa kielissä)
- Totuusarvoinen lauseke voikin sisältää ***loogisia operaattoreita*** vertailujen yhdistämiseen
- Pseudokielemme loogiset operaattorit ovat **AND**, **OR** ja **NOT**
  - Esim yo. vertailu voidaan kirjoittaa: a < b AND b < c

Loogiset operaattorit

lauseke1	lauseke2	lauseke1 AND lauseke2	lauseke1 OR lauseke2	NOT lauseke1
epätosi	epätosi	epätosi	epätosi	tosi
tosi	epätosi	epätosi	tosi	epätosi
epätosi	tosi	epätosi	tosi	tosi
tosi	tosi	tosi	tosi	epätosi

Totuusarvoiset lausekkeet - esimerkkejä

```
a < 0 AND b < 0
```

- Lausekkeen arvo on tosi, mikäli muuttujien a ***ja*** b arvo on pienempi kuin 0. Mikäli edes toisen arvo on nolla tai sitä suurempi, lausekkeen arvo on epätosi.

Totuusarvoiset lausekkeet - esimerkkejä

```
a < 0 OR b < 0
```

- **Koska lausekkeen arvo on tosi?**

Totuusarvoiset lausekkeet - esimerkkejä

```
a < 0 OR b < 0
```

- **Koska lausekkeen arvo on tosi?**
- Lausekkeen arvo on tosi, mikäli muuttujan a ***tai*** b arvo on pienempi kuin 0. Mikäli *molempien* arvo on nolla tai sitä suurempi, lausekkeen arvo on epätosi.

Valinta- eli ehtolause

https://www.flickr.com/photos/sniegowski/8489818744

Ohjausrakenteet

- Peräkkäisyys on perusohjausrakenne
- Algoritmin on kuitenkin kyettävä toimimaan erilaisissa tilanteissa -> pelkkä peräkkäisyys ei riitä
- Tarvitaan siis ***ohjausrakenteita*** eli ***kontrollirakenteita***:
  - ***Valinta*** *(branching)*
  - ***Toisto*** *(repetition)*
- Ohjausrakenteet kuvaavat järjestyksen, jossa algoritmin toimenpiteet suoritetaan
- Algoritmi koostetaan ohjausrakenteita yhdistelemällä
- Peräkkäisyys, valinta ja toisto riittävät minkä tahansa algoritmin esittämiseen

Valinta- eli ehtolause

- Saman ongelman ratkaisemiseksi eri tilanteissa algoritmin toimittava eri tavoin
- Algoritmin varauduttava vaihtoehtoisiin toimintoihin
- Algoritmin on siis tehtävä ***valinta*** eri toimintojen välillä
- Valinta ei saa olla satunnainen vaan sen tulee perustua yksiselitteiseen *valintaehtoon*

Valinta- eli ehtolause

- IF...THEN-lause on yksinkertaisin valinnan muoto jonkin toiminnon tekemiseen tai sen tekemättä jättämiseen.

```
IF ehto THEN
   toiminto
ENDIF
```

Valinta- eli ehtolause


```
IF ehto THEN
  toiminto
ENDIF
```

- *ehto* on totuusarvoinen lauseke (eli arvo on joko tosi tai epätosi)
- Ehto määrää tilanteen jossa toiminto suoritetaan
  - Jos ehto on tosi, toiminto suoritetaan muuten ei

Valinta- eli ehtolause


```
IF ehto THEN
  toiminto
ENDIF
```

- *toiminto* voi olla yksittäinen lause tai joukko lauseita
- Se voi siis koostua useasta toiminnosta, tai olla myös tyhjä
- Voi siis sisältää asetuslauseita, valintalauseita, toistolauseita, jne.

Ehtolause - esimerkki

```
IF a < b + 2 THEN
   a := b
ENDIF
```

- Mikäli a:n arvo on pienempi kuin b:n arvo + 2, suoritetaan toiminto.
- Tässä toiminto asettaa a:n arvoksi b:n arvon.

IF…THEN…ELSE -lause

IF...THEN...ELSE-lause on yleisempi valinta, jossa valitaan kahden vaihtoehtoisen toiminnon välillä:

```
IF ehto THEN
  toiminto1
ELSE
  toiminto2
ENDIF
```

IF…THEN…ELSE -lause

```
IF ehto THEN
  toiminto1
ELSE
  toiminto2
ENDIF
```

- Mikäli *ehto* on tosi, suoritetaan *toiminto1*. Muussa tapauksessa suoritetaan *toiminto2*.
- Vain toinen toiminnoista suoritetaan

IF…THEN…ELSE - esimerkki

```
IF a > 0 AND b > 0 THEN
  print “Molemmat nollaa suurempia”
ELSE
  print “Molemmat ei nollaa suurempia”
ENDIF
```

- Huomaa lauseiden sisennys siten, että eri haarat tulevat selvästi esille.
  - IF, ELSE ja ENDIF kirjoitetaan yleensä eri riveille ja sisennetään alkamaan samasta sarakkeesta
  - ***Kierroksen tehtävät vaativat että koodi sisennetään näin!***

IF..THEN..ELSE - esimerkki

```
IF x > y THEN
  max := x
ELSE
  max := y
ENDIF
```

- Mikäli muuttujan x arvo on suurempi kuin muuttujan y, sijoitetaan muuttujaan max x:n arvo. Muussa tapauksessa (ELSE-haarassa) sijoitetaan max:iin y:n arvo.

Sisäkkäiset ehtolauseet

- Monimutkaiset loogiset lausekkeet voivat tehdä ymmärtämisestä hankalaa
- Algoritmi voidaan kirjoittaa yksinkertaisimmilla loogisilla lausekkeilla käyttäen sisäkkäisiä ehtolauseita

Sisäkkäiset ehtolauseet - esimerkki

```
IF a > b AND a > c THEN
  tulosta "a on suurin”
ENDIF
```

- Tämä voidaan kirjoittaa sisäkkäisiä ehtolauseita käyttäen:

```
IF a > b THEN
  IF a > c THEN
    tulosta "a on suurin"
  ENDIF
ENDIF
```

- Huomaa lauseiden sisennys!

Valintalause ohjelmointikielissä

- Valintalause on käytettävissä kaikissa ohjelmointikielissä
- Syntaksi vaihtelee

Java	Python

`if (ehto) {`
`  toiminto1;`
`} else {`
`  toiminto2;`
`}`
	`if ehto:`
	`  toiminto1`
	`else:`
	`  toiminto2`

Ensi viikolla

- Tutoriaali tämän päivän aiheesta ti 9.9. klo 12:15
- Seuraava luento to 11.9. klo 14:15
  - Toisto
  - Johdanto modulaarisuuteen

- **1. kotitehtäväkierroksen määräaika ti 9.9. klo 11:00**