Imperatiivisen ohjelmoinnin peruskäsitteet

- **muuttuja**
  - muuttujissa oleva data voi olla yksinkertaista eli primitiivistä (esim. luvut ja merkit) tai rakenteista jolloin puhutaan tietorakenteista.
  - puhuttaessa tietorakenteista tulee erottaa sen abstrakti malli (jolloin puhutaan abstraktista tietotyypistä eli ADT:stä) ja tallennusrakenne, joka kertoo millä ohjelmointikielen välineillä tietorakenne on toteutettu (taulukko, linkitetyt rakenteet, luokka, …)
- **lauseke** voi olla
  - muuttuja (arvo voi olla primit. tai rakent. tyyppiä)
  - aritmeettinen lauseke (lukuarvoinen)
  - looginen eli ehtolauseke (totuusarvoinen)
  - merkkijonolauseke (arvona merkkien jono)
  - funktionaalisen moduulin kutsu
- **lauseet:**
  - asetuslause
  - valintalause (IF)
  - toistolause (FOR, WHILE, REPEAT tai DO)
  - luku- / kirjoituslauseet
  - proseduraalisen moduulin kutsu

- **moduuli** eli metodi eli aliohjelma
  - erota moduulin esittely ja moduulin kutsu
  - moduulin parametrit ja niiden välitystapa

- muuttujien ja moduulien nimet ovat ohjelmoijan valitsemia kuvaavia **tunnisteita**
- algoritmi koostuu peräkkäisistä lauseista, joista keskeisessä asemassa on asetuslause
- algoritmin tila = muuttujien senhetkiset arvot
- jokaiseen lauseen suorituksen jälkeen siirrytään uuteen tilaan

ALKUTILA — Algoritmi → LOPPUTILA

algoritmissa käytettyjen muuttujien alkuarvot

muuttujien loppuarvot

Meidän käyttämän pseudokielen lauseiden syntaksi

- On myös kursssivuilla
- Huom. Lauseet sisältävät osinaan lauseita!

Lausejono:

yksi tai useampi lause peräkkäin
tai voi olla myös tyhjä

Asetuslause:

<muuttuja> := <lauseke>

Valintalauseet:

IF <totuusarvoinen lauseke> THEN <lausejono> ENDIF

ja

IF <totuusarvoinen lauseke> THEN
    <lausejono>
ELSE
    <lausejono>
ENDIF

Toistolauseet:

Alkuehtoinen toisto: toisto loppuu, kun <totuusarvoinen lauseke> on epätosi:

WHILE <totuusarvoinen lauseke> DO
    <lausejono>
ENDWHILE

Loppuehtoinen toisto: toisto loppuu, kun <totuusarvoinen lauseke> on tosi:

REPEAT
    <lausejono>
UNTIL <totuusarvoinen lauseke>

Askeltava toisto: suoritetaan kohdassa <lausejono> olevat lauseet yhden kerran kutakin listan alkiota kohti:

FOR <muuttuja>:=<listan ensimmäinen arvo>,...,
                <listan viimeinen arvo> DO
    <lausejono>
ENDFOR

Tätä monet ihmettelevät: Miksi algoritmien opettelussa käytettävät esimerkit ovat usein laskennallisia (matemaattisia)?

Esimerkki. Algoritmin muuttujat, lauseet ja (aritmeettiset/loogiset) lausekkeet? Mitä tulostuu? Tämä on vanha tenttitehtävä.

```
n := 2
k := 3
WHILE  n+k < 23  DO
    IF n < k THEN
            n := n + k
    ELSE
            k := k + 2
    ENDIF
    k := k+1
    tulosta n ja k
ENDWHILE
tulosta n ja k
```


Esimerkki. n! = 1*2 *3* … * n, kun n>0. 0!=1

Tehtävä 1: Aseta muuttujaan k lausekkeen 4! arvo.

Ratkaisu:

```
k := 1 * 2 * 3 * 4
```

jonka jälkeen muuttujassa k on arvo 24.

Tehtävä 2: Olkoon muuttujassa n ei-negatiivinen kokonaisluku. Aseta muuttujaan k lausekkeen n! arvo.

Ratkaisu: **Algoritmin pitää toimia olkoonpa n:ssä mikä tahansa ei-negat. kokonaisluku**. Oletetaan ensin, että n>0 ja tarkastellaan lopuksi erikoistapaus n=0. Asetuslause

```
k := 1 * 2 * 3 *…* n
```

on VÄÄRIN

Ratkaisu tulee perustaa kahden arvon kertomiseen keskenään:

- aluksi k =1
- kerrotaan k luvulla 1 ja sijoitetaan tulos k:hon (turha)
- kerrotaan k luvulla 2 ja sijoitetaan tulos k:hon
- kerrotaan k luvulla 3 ja sijoitetaan tulos k:hon
- …
- kerrotaan k arvolla n ja sijoitetaan tulos k:hon

Tämän jälkeen k:ssa on n-kertoman arvo.

Miten algoritmin suunnittelu aloitetaan:

- Mitä muuttujia tarvitaan?
- Toistorakenteen laatiminen (tässä järjestyksessä):
    - kirjoitetaan ensin silmukan rungon lauseet
    - kirjoitetaan muuttujien alkuarvot
    - kirjoitetaan WHILEn ehto (totuusarvoinen lauseke)

Seuraavassa ovat ratkaisut, jotka eivät muuta n:n sisältöä:

kertoja := 1
k := 1
WHILE kertoja <= n DO
  k := kertoja * k
  kertoja := kertoja + 1
ENDWHILE
	
	k := 1
	FOR kertoja := 1,2,…,n DO
	  k := kertoja * k
	ENDFOR

Alkutila, lopputila? Mistä n?

Mitä algoritmi laskee, jos WHILE lauseen rungon lauseiden järjestys vaihdetaan? Algoritmi voidaan kuitenkin muuttaa oikeaksi myös tällöin. Miten?

Miten voidaan tutkia onko WHILE-rakenteen lopetusehto oikein?

Toimiiko algoritmi kun n=0 ja mitä tällöin tapahtuu.

MODUULIT JA PARAMETRIT

- (algoritmi)moduuli = aliohjelma = metodi = rutiini
- yleensä ohjelma koostuu ns. **päämoduulista (=pääohjelma)**, joka suoritetaan. Päämoduuli (main-moduuli) sisältää tyypillisesti moduulien kutsuja, jotka puolestaan voivat sisältää muiden moduulien kutsuja.
- moduulilla tulee olla yksi selkeä tehtävä ja se pyritään yleistämään parametrien avulla mahdollisimman yleiseksi, jotta se olisi monikäyttöinen
- moduulien ja parametrisoinnin avulla pyritään siihen, että
    - sama koodinpätkä tarvitsee kirjoittaa vain kerran (uudelleenkäytettävyys)
    - ohjelmakokonaisuus muodostuisi hyvin määritellyistä komponenteista niin, että koko ohjelma on selkeä ja että ohjelman muuttaminen olisi mahdollisimman helppoa (ylläpito helpottuu)
    - tietorakenteiden käsittely ja niiden käytös (ominaisuudet) tulevat täsmällisesti määriteltyä niihin kohdistettavien metodien välityksellä (olio-ohjelmointi ja valmiit luokkakirjastot)
- **pitää erottaa moduulin määrittely ja sen kutsu** (käyttö)

- Moduulin kirjoittajan tulee kirjoittaa moduulille **alkuehto** kommentin muodossa: ne ehdot parametreille, jolloin moduuli toimii oikein. Esim. n-kertoman moduulissa alkuehto on: n >= 0.
- **rekursiivinen moduuli** = moduuli, joka sisältää itsensä kutsun. Rekursiivisessa moduulissa tulee olla
  - ns. rekursion kanta (triviaalitapaus), jossa tehtävälle annetaan ratkaisu suoraan
  - rekursioaskel, jossa palautetaan tehtävä saman tehtävän pienemmäksi tehtäväksi, jonka tulee lähestyä rekursion kantaa
- **moduuleja on kahta tyyppiä: proseduureja ja funktioita**
  - funktionaalisen moduulin kutsu on lauseke (sillä on siis arvo) ja proseduraalisen moduulin kutsu on lause (käsky).
  - kirjoitamme proseduurin -> lisäämme kieleen (parametrisoidun) uuden lauseen eli käskyn
  - kirjoitamme funktion -> lisäämme kieleen uuden funktion, jonka tuloksena on arvo (useissa kielissä arvo voi olla myös rakenteinen)
  - jos moduuli ei palauta mitään, on se proseduuri. Jos taas moduuli palauttaa jotain, se on funktio.

Funktionaalisen moduulin määrittely:

MODULE <moduulin nimi> (*<muodolliset parametrit>*)
RETURNS <palautusarvo>
<moduulin lauseet>
ENDMODULE

Moduulissa tulee olla RETURN-lause!

Proseduraalisen moduulin määrittely:

MODULE <moduulin nimi> (*<muodolliset parametrit>*)
<moduulin lauseet>
ENDMODULE

Funktion kutsu (on lauseke ja sillä on arvo):

<moduulin nimi> (*<todelliset parametrit>*)

Proseduurin kutsu (on lause eli käsky):

<moduulin nimi> (*<todelliset parametrit>*)

Esimerkkejä.

- Kirjoita moduuli, joka palauttaa kolmesta luvusta pienimmän (suoraan ilman moduulia min2). Tehdään yhdessä. Moduulin määrittely vs. käyttö, muod. ja tod. parametrit? main-moduuli

- **Muodolliset parametrit ja todelliset parametrit**
- Parametrien arvojen kopiointi
- Milloin muistipaikat ovat olemassa ja mitä niissä on ohjelman suorituksen aikana.
- Onko väliä minkä nimisiä muuttujia moduulissa käytetään? Miksi?
- Onko väliä vaikka eri moduuleissa käyttää samannimisiä muuttujia

- Tehdään yhdessä tenttitehtävä (lopussa), jossa on 1+2+…+z. Miten algoritmi muunnetaan moduuliksi. Parametrit?
  Käyttö eli miten sitä kutsutaan?

- Tehtävän ratkaiseva algoritmi voidaan toteuttaa monella eri tavalla: katso esim. materiaalien moduuli alkuluku(n) ja Villen vastaava esimerkki sekä …

- moduuli, joka laskee summan 1+2+...+n **rekursiivisesti**.

  MODULE summa(n) RETURNS luku

  ENDMODULE

Mitä tapahtuu kutsuilla summa(0) ja summa(-1)?

TIETO- JA TALLENNUSRAKENTEET (viimeisen viikon asiat)

- maailma ei koostu pelkästään luvuista ja merkeistä, vaan asioita mallinnetaan paljon monimutkaisimmilla rakenteilla (jotka tietenkin loppujen lopuksi koostuvat vain merkeistä ja numeroista ja viimekädessähän nämä toteutetaan vain biteillä)
- **abstrakti tietotyyppi** (ADT) tarkoittaa mallia, jossa kuvataan siihen kuuluvat mallin kannalta oleelliset tiedot ja kaikki malliin kohdistettavat operaatiot ja niiden merkitykset (operaatiot määräävät tietorakenteen ominaisuudet ja merkityksen)
- ADT:n lista ja 2-haarainen puu (binääripuu) määrittely:
  - **lista** on joko
    - tyhjä tai
    - se on alkio, jota seuraa lista
  - **2-haarainen puu** on joko
    - tyhjä tai
    - se on alkio (solmu), jota seuraa kaksi 2-haaraista puuta

- ADT:t **pino** ja **jono** ovat (erityisiä) listoja, joille on määritelty vain lisäys- ja poisto-operaatio, jotka toimivat tietyllä tavalla (tällä mallinnetaan reaalimaailman tiettyä käyttäytymistä).
  - *pinon käyttäytyminen*: kun otetaan alkio pois, tulee sen olla se alkio, joka on sinne viimeksi lisätty.
  - *jonon käyttäytyminen*: kun otetaan alkio pois, tulee sen olla se alkio, joka on ollut jonossa kauimmin.
- **tulee erottaa ADT ja sen tallennusrakenne** (eli mikä on ohjelmointikielen väline ko. rakenteen toteuttamiseksi)
- lista voidaan toteuttaa esim. 1-ulotteisella taulukolla, joka on tallennusrakenne, joka on kaikissa imperatiivissa ohjelmointikielissä

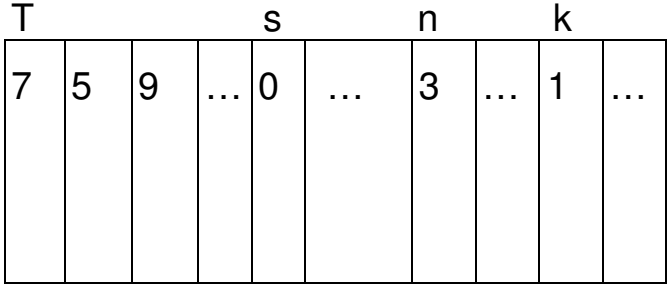
Taulukot

- taulukkoon voidaan viitata kokonaisuutena yhdellä muuttujalla (esim. v)
- taulukon alkiotyyppi (komponentin sisältö) riippuu käytettävästä ohjelmointikielestä, mutta yleensä niiden kaikkien tulee olla samaa tyyppiä (esim. kokonaislukuarvoisia)
- taulukon alkioihin viitataan indeksin avulla: esim. v[3], v[k] (indeksinä muuttuja) tai v[k+2]) (indeksinä lauseke)
- Miten alkiot saadaan taulukkoon? Alkiot tallennetaan taulukkoon yksitellen käyttäen esim. asetuslausetta (esim. v[1]:=3) tai kaikki kerralla vakioarvoina (esim. v:={23,56}; ei mahdollista kaikissa kielissä)
- **erota indeksi** (jolla siis kerrotaan vektorin se kohta, mihin halutaan viitata) **ja taulukon komponentti** (alkio, lokeron sisältö)
- taulukon komponenttien indeksointi (eli monesko 'lokero'): pseudokielessä 1,2,…, mutta Villessä ja Javassa 0,1,…
- taulukko on **staattinen** kiinteänmittainen rakenne: komponenttien lukumäärää tulee ilmoittaa luotaessa ja rakenne pysyy samankokoisena koko ohjelman suorituksen ajan
- vektori: 1-ulotteinen taulukko
  matriisi: 2-ulotteinen taulukko

- opiskeluopas s. 39: vektorin alkioiden summa:

MODULE summa(T,n) RETURNS vektorin T n:n ensimmäisen alkion summan

```
    s := 0
    k := 1
    WHILE k <= n DO
        s := s + T[ k ]
        k := k + 1
    ENDWHILE
    RETURN s
ENDMODULE
```



Tässä T:ssä on kolme alkiota: T[1]=7, T[2]=5, T[3]=9, joten kutsun summa(T,3) arvo on 21.

- Opiskeluoppaan s. 38 tehtävä: Mikä on lausekkeen summa(T,1)+summa(T,2)+summa(T,3) arvo?
- Mitä edellä pitää muuttaa, jos halutaankin palauttaa parametrina annetun taulukon alkioiden keskiarvo

Javalla vektorin alkioiden summa (ihan vaan esimerkkinä):

```
public static double summa (double[] t)
{
    double s = 0.0;
    int k = 0;
    while (k < t.length)
    {
        s = s + t[ k ];
        k = k + 1;    // tai: k++;
    }
    return s;
}
```

- Tehdään yhdessä: Kirjoita moduuli, joka palauttaa tiedon siitä kuinka monta kappaletta parametrina annetun taulukon T alkioista on suurempia kuin parametrina annettu luku a.

Kirjoitetaan vielä ns. pääohjelma eli *main*-moduuli, josta ohjelman suoritus alkaa. Moduulissa määritellään kaksi vektoria ja kutsutaan em. metodia.

- **Vaihtolajittelu**: opintomoniste s. 61, harjoitustehtävä.

```
MODULE vaihtolajittelu(T) RETURNS vektorin T lajiteltuna
    n := T.length
    FOR i := 1, 2, ..., n-1 DO
        FOR k := i+1, ..., n DO
            IF T[ i ] > T[ k ] THEN
            (* vaihda T[i]:n ja T[k]:n sisällöt keskenään *)
              apu := T[i]
              T[i]  := T[k]
               T[k] := apu
             ENDIF
        ENDFOR
    ENDFOR
    RETURN T
ENDMODULE
```

i=1

66	55
55	66
44	44
33	33
	k=2 k=3 k=4

i=2

k=3 k=4

i=3

k=4

Yo. taulukot kuvaavat tilannetta aina mahdollisen vaihdon jälkeen eli tilannetta kun IF-lause on suoritettu.

Seuraavassa on vanhoja tenttitehtäviä.

Yksi oli jo tämän materiaalin alussa.

1. a) Olkoon muuttujassa z jokin positiivinen kokonaisluku. Kirjoita WHILE- (tai REPEAT-) lausetta käyttävä algoritmi (ei MODULE), jonka suorituksen jälkeen muuttujassa k on lausekkeen 1+2+3+…+z arvo. Algoritmin suoritus ei saa muuttaa muuttujan z arvoa. Mitä algoritmisi suoritus saa aikaan, jos z olisikin 0 tai negatiivinen?

b) Miten muutat a)-kohdan algoritmin funktionaaliseksi moduuliksi (MODULE), joka palauttaa lausekkeen 1+2+3+…+z arvon.

c) Kirjoita lause, joka asettaa muuttujaan y arvon

35 + (1+2+3+4+5+6+7+8)/8.

Lauseessa tulee olla b)-kohdan moduulin kutsu.

Ratkaisu:

2. a) Anna määritelmä abstraktille tietotyypille *lista* ja esitä listalle suoritettavat tyypilliset operaatiot. Minkälaisia rakenteita ovat *pino* ja *jono*?

b) Oletetaan, että lukuja sisältävä lista toteutetaan *vektorina*. Kirjoita moduuli, joka palauttaa parametrina annetun listan T alkioiden suurimman alkion arvon.

4. Oletetaan, että käytössä on funktio lue(), joka lukee syötejonon (syötevirran) ensimmäisen luvun ja palauttaa sen arvon (samalla kyseinen luku häviää syötejonosta). Kirjoita algoritmi, joka lukee syötejonosta sata lukua ja laskee niiden keskiarvon. Anna ratkaisu käyttäen WHILE-lausetta.

5. Selitä lyhyesti käsitteet: muuttuja, aritmeettinen lauseke, looginen eli totuusarvoinen lauseke ja lause. Anna myös esimerkit näistä. Mikä rooli on varatuilla sanoilla ENDIF ja ENDWHILE ja millä muulla tavalla ohjelmointikielissä tämä voidaan myös ilmaista. Vastauksen tulee mahtua yhdelle sivulle.

Huom. Yllä ei ole mallikysymystä Tietokoneen toiminnasta ja rakenteesta ym. (katso jakson TTP I kurssisivut).

Menestystä tenttiin!