Kehittää ohjelmointitehtävien ratkaisemisessa tarvittavia metakognitioita!

eli ...

Hyvä kaava sanoo enemmän kuin,
. . . tuhat riviä koodia!
. . . sata riviä tekstiä!
. . . kymmenen diagrammia!

Sopimusohjelmointi

Ohjelmat koostuvat *vekottimista* (moduleista, olioista ym), joilla on kullakin oma rajapintansa. Vekottimet toimivat yhteistyössä käyttämällä toisten vekottimien rajapinnan nappuloita (= aliohjelmia).

Rajapinta on *sopimus* vekottimen käyttäjän ja toteuttajan välillä.

Sopimuksessa kuvataan aliohjelmien nimi, parametrien määrä ja tyypit sekä paluuarvon tyyppi. Tämä kaikki voidaan kuvata tavallisella ohjelmointikielellä, ja näiltä osin rajapinnan oikea käyttö voidaan mekaanisesti varmistaa (esim. käännösaikaisilla tai ajonaikaisilla tarkistuksilla).

Soipimuksessa kuvataan myös, *mitä aliohjelma tekee*. Normaalisti tätä ei voida mekaanisesti varmistaa, mutta sopimusohjelmointi tuo siihen keinoja.

Sopimusohjelmointikielistä tunnetuin on Eiffel (Bertrand Meyer, 1980-luku)

Tilapredikaatti on ehtolauseke, jonka totuusarvo riippuu ohjelman tilasta (muuttujien arvot).

Jälkiehdot

Aliohjelman toiminta voidaan kuvata tilapredikaatilla, joka on tosi täsmälleen silloin, kun ohjelman tila on sellainen, minkälaiseksi aliohjelman pitää se jättää. Tätä tilapredikaattia sanotaan aliohjelman *jälkiehdoksi*.

Esimerkiksi aliohjelman

```
int jaakahdella(const int n);
```

jälkiehto on (jos paluuarvoa merkitään `rv`:llä)
`2 * rv = n`.

Jälkiehto kertoo

aliohjelman käyttäjälle, mitä hän voi olettaa ohjelman tilasta aliohjelman suorituksen päätyttyä, sekä

aliohjelman kirjoittajalle, mihin tilaan aliohjelman tulee ohjelma jättää päättyessään.

Esiehdot

Aliohjelma ei voi toimia oikein kaikissa tilanteissa. Joko aliohjelman toteuttajan tulee tarkistaa nämä kielletyt lähtötilanteet aliohjelmassa tai hänen pitää kieltää aliohjelman kutsumisen niissä tilanteissa.

Aliohjelman *esiehto* on tilapredikaatti, joka on tosi jos ja vain jos aliohjelmaa saa kutsua kyseisessä tilanteessa. Se kertoo

aliohjelman kirjoittajalle, mitä hän voi olettaa ohjelman tilasta (mukaanlukien parametrit) aliohjelman suorituksen alkaessa, sekä

aliohjelman käyttäjälle, minkälaisessa tilanteessa aliohjelmaa on lupa kutsua.

Jos esiehto ei pidä aliohjelman käynnistyessä paikkaansa, ei aliohjelmalla ole mitään velvollisuutta mihinkään suuntaan.

Edellä mainitulle `jaakahdella`-aliohjelmalle voitaisiin asettaa esiehdoksi vaikkapa `n % 2 == 0` (eli `n` on jaollinen kahdella).

Invariantit

Tilainvariantti (*state invariant*) on sellainen tilapredikaatti, joka on tosi ohjelman alusta ohjelman loppuun.

Se on ohjelman tietorakenteiden eheydestä kertova väittämä: jos se jossain tilanteessa muuttuu epätodeksi, se kertoo, että ohjelmassa on jotain mätää.

Periaatteessa tilainvariantti on näkymätön osa jokaisen aliohjelman esiehtoa ja jälkiehtoa (aliohjelma voi olettaa sen pätevän alussa, ja sen on huolehdittava, että se pätee lopussa).

Olio-ohjelmoinnissa käytetään muunnelmaa tilainvariantista. Jokaisella oliolla on *luokkainvariantti* (*class invariant*), joka määrittää olion attribuuttien sallitut arvot. Luokkainvariantti on näkymätön osa jokaisen muodostimen jälkiehtoa sekä jokaisen metodin esi- ja jälkiehtoa.

Inariantit: Esimerkki

C++-luokalla

```
class Paivays {
  int pv, kk, v;
  ...
};
```

on seuraava luokkainvariantti:

```
  1 <= kk && kk <= 12 &&
  impl(kk == 1 || kk == 3 || kk = 5 || kk == 7
       || kk == 8 || kk == 10 || kk == 12,
       pv <= 31) &&
  impl(kk == 4 || kk == 6 || kk == 9 || kk == 11,
       pv <= 30) &&
  impl(kk == 2, pv <= 29) &&
  impl(kk == 2 && !(v % 4 == 0 && (v % 100 != 0
       || v % 400 == 0)), pv <= 28) &&
  1 <= pv
```

(missä `impl` on aliohjelma, joka palauttaa toden, jos ja vain jos sen ensimmäinen argumentti implikoi toisen argumentin).

Joukko:

Kokoelma samaa tyyppiä olevia elementtejä, joita kutsutaan joukon *alkioiksi* (member/element).

Esimerkkejä:

PienetAlkuluvut $\widehat{=} \{2, 3, 5, 7, 11, 13, 17, 19\}$

ParillisetLuonluvut $\widehat{=} \{0, 2, 4, 6, 8, 10, \ldots\}$

$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$

Perusominaisuudet:

1. alkioiden esiintymiselle ei vaadita järjestystä eli $\{a, b\} = \{b, a\}$,
2. saman alkion esiintymien lukumäärällä ei merkitystä eli $\{a, b, a, a\} = \{b, a\}$.

Laukku (bag):

Kokoelma samaa tyyppiä olevia elementtejä, joiden lukumäärällä on väliä. Esimerkiksi $(c, a, b) = (a, b, c)$, mutta $(a, b, b, c) \neq (a, b, c)$.

Eli

$$(a, b, b, c) \simeq \{a \mapsto 1, b \mapsto 2, c \mapsto 1\}$$

Operaatioesimerkki:

$$\mathit{count}\ (a, b, b, c)\ a = 1$$

$$\mathit{count}\ (a, b, b, c)\ b = 2$$

$$\mathit{count}\ (\mathit{kissa}, \mathit{koira}, \mathit{hevonen}, \mathit{koira}, \mathit{lammas})\ \mathit{hevonen} = 1$$

$$\mathit{count}\ (\mathit{kissa}, \mathit{koira}, \mathit{hevonen}, \mathit{koira}, \mathit{lammas})\ \mathit{aasi} = 0$$

Osalaukku: kaikki laukun $a : \mathrm{bag}\ X$ "matkatavarat" löytyvät myös laukusta $b : \mathrm{bag}\ X$.

Laukkujen yhdistelmä:

$$(a, b, b, c) \sqcup (b, c, d) = (a, b, b, b, c, c, d)$$
$$\simeq \{a \mapsto 1, b \mapsto 3, c \mapsto 2, d \mapsto 1\}$$

ja laukkujen leikkaus:

$$(a, b, b, c) \sqcap (b, c, d) = (b, c) \simeq \{b \mapsto 1, c \mapsto 1\}$$

Sekvenssi (seq):

Kokoelma samaa tyyppiä olevia elementtejä, joiden sekä järjestyksellä että lukumäärällä on väliä. Esimerkiksi $(c, a, b) \neq (a, b, c)$.

Eli

$$\langle a, b, c, \ldots \rangle \simeq \{1 \mapsto a, 2 \mapsto b, 3 \mapsto c, \ldots\}.$$

Operaatioesimerkki: $s : \mathrm{seq}\ X$:

$\langle a, b, c\rangle[2] = b$,

$\langle \mathit{kissa}, \mathit{koira}, \mathit{hevonen}, \mathit{lammas}\rangle[4] = \mathit{lammas}$,

$\langle \{a, b\}, \{c\}, \{d, e\}\rangle[2] = \{c\}$,

Perusoperaatioita:

Operaatio	**Esimerkki**	**Merkitys**
$\#$	$\#\langle a, b, c\rangle = 3$	sekvenssin pituus
$\frown$	$\langle a, b\rangle \frown \langle c\rangle = \langle a, b, c\rangle$	kahden sekvenssin yhdistäminen
head	$\mathit{head}\langle a, b, c\rangle = a$	sekvenssin ensimmäinen alkio
tail	$\mathit{tail}\langle a, b, c\rangle = \langle b, c\rangle$	sekvenssin häntä eli sekvenssi, josta poistettu ensimmäinen alkio
last	$\mathit{last}\langle a, b, c\rangle = c$	sekvenssin viimeinen alkio
front	$\mathit{front}\langle a, b, c\rangle = \langle a, b\rangle$	sekvenssin alku eli sekvenssi, josta poistettu viimeinen alkio