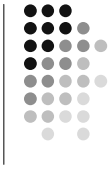
Imperatiivisten ohjelmien organisointiparadigmojen historia

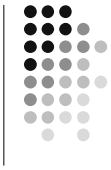
Timo Tapanainen
Helsingin yliopisto, tietojenkäsittelytieteen laitos
Tietojenkäsittelytieteen historia -seminaari, kevät 2007

Sisältö



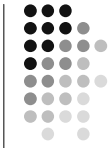
- Paradigma, ohjelmointiparadigma?
- Imperatiivinen paradigma
- Imperatiivisten ohjelmien organisointiparadigmojen historia
  - Rakenteeton ohjelmointi / lauseohjelmointi
  - Proseduraalinen ohjelmointi
  - Rakenteinen ohjelmointi
  - Modulaarinen ohjelmointi
  - Olio-ohjelmointi

Paradigma, ohjelmointiparadigma?



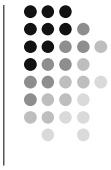
- Paradigma
  - Nykyinen merkitys Thomas Kuhnilta, “The Structure of Scientific Revolutions” (1962)
  - paradigma = ajattelun kaava, kysymättä hyväksytty periaate tai katsomus
  - Tiede edistyy paradigman vaihdoksina
- Ohjelmointiparadigma
  - Robert W Floyd, “The Paradigms of Programming” (1978)
  - Useita merkityksiä

Ohjelmointiparadigmalla voidaan tarkoittaa:



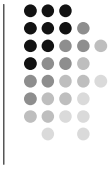
- Ajatusmalli laskennasta
  - Imperatiivinen, funktionaalinen, logiikkaohjelmointi …
- Ajatusmalli siitä, mistä ohjelma koostuu (rakenteen organisointimalli)
  - Olio-ohjelmointi, proseduraalinen ohjelmointi, …
- …

Sisältö



- Paradigma, ohjelmointiparadigma?
- Imperatiivinen paradigma
- Imperatiivisten ohjelmien organisointiparadigmojen historia
  - Rakenteeton ohjelmointi / lauseohjelmointi
  - Proseduraalinen ohjelmointi
  - Rakenteinen ohjelmointi
  - Modulaarinen ohjelmointi
  - Olio-ohjelmointi

Imperatiivinen paradigma



- Ensimmäiset ohjelmoitavat tietokoneet perustuivat von Neumannin konearkkitehtuuriin
- Imperatiivinen ohjelmointi = von Neumannin koneen ohjelmointia
- Laskennan mallina konkreettinen kone (vrt. funktionaalinen ohjelmointi)
- Menestynein laskennan malli
- Konekielet ovat ensimmäisiä imperatiivista ohjelmointia tukevia kieliä

Imperatiivisen ohjelmoinnin haasteet

- Tietokoneiden teho ja käyttäjäkunta laajenee
  - Prosessorien teho tuplaantuu 18 kuukauden välein (Mooren laki)
  - Keskuskoneet, minikoneet, pc, ...
- Tehokkaammilla koneilla voidaan ratkaista monimutkaisempia ongelmia
  - ⇒ Ohjelmien koko ja monimutkaisuus kasvaa
- Ennen: tietokoneet kyvyttömiä suorittamaan ihmisten antamia käskyjä
- Nyt: ihmiset kyvyttömiä käskyttämään konetta
- Tarvitaan menetelmiä monimutkaisuuden ja laajuuden hallintaan

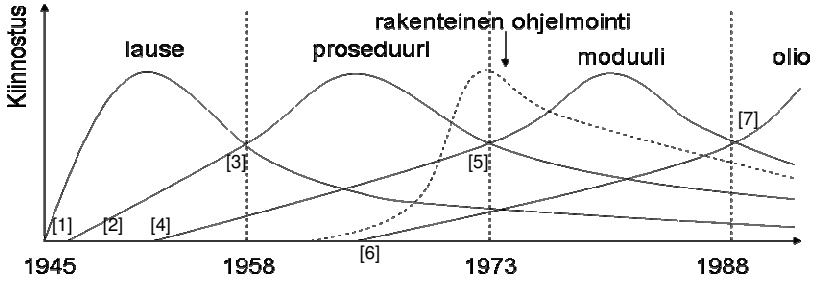
Sisältö

- Paradigma, ohjelmointiparadigma?
- Imperatiivinen paradigma
- Imperatiivisten ohjelmien organisointiparadigmojen historia
  - Rakenteeton ohjelmointi / lauseohjelmointi
  - Proseduraalinen ohjelmointi
  - Rakenteinen ohjelmointi
  - Modulaarinen ohjelmointi
  - Olio-ohjelmointi

Ohjelman rakennetta organisoivat paradigmat

- Yrityksiä hallita ohjelmien monimutkaisuutta ja laajuutta
- Organisointiparadigmat ohjaavat ohjelman toiminnallisuuden ja tiedon organisointia
- Paradigmat perustuvat yleensä yhden abstraktion varaan: lause > proseduuri > moduuli > luokka
- Paradigmojen kehitys on kulkenut:
  - Koneen toiminnan ohjauksesta enemmän ongelma-alueen kuvaamiseen (deklaratiivinen)
  - Ohjelmointiin suuressa mittakaavassa

Organisointiparadigmojen historia



1. Konekielet
2. Fortran
3. Funktiot Algoliin
4. Ensimmäiset assemblerit ja linkittäjät
5. C-kieli
6. Simula
7. C++

Lähde: Raccoon, L. B. 1997. Fifty years of progress in software engineering. SIGSOFT Softw. Eng. Notes 22, 1 (Jan. 1997), 88-104

Rakenteeton ohjelmointi

- ~ 1945 – 1958
- Rakenteeton ohjelma koostuu:
  - yhdestä toiminnallisuuden sisältävästä pääohjelmasta
  - julkisesta tiedosta
- Ohjelmointi koneläheisillä abstraktioilla esim. goto-lauseet (helppo toteuttaa kääntäjiä)
- Rakenteettoman ohjelmoinnin ongelmia:
  - Saman toiminnallisuuden käyttö edellytti kopiointia
  - “matala” rakenne ei sovellu laajojen ohjelmien toteuttamiseen

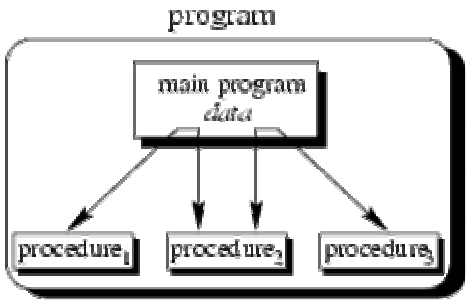
Rakenteettoman ohjelman rakenne

Lähde: http://www.zib.de/Visual/people/mueller/Course/Tutorial/node3.html

Proseduraalinen ohjelmointi

- ~ 1958 – 1972
- Korjaa rakenteettomien ohjelmien ongelmia
  - Samaa toiminnallisuutta voidaan käyttää ilman kopioimista
  - Proseduurikutsut korvaavat hypyt toimintoihin => ohjelman toimintaa helpompi seurata
- Ensimmäisiä proseduraalisia kieliä olivat Fortran ja Algol
- Proseduraalisen ohjelman toiminnallisuus jaetaan proseduureihin, jotka käsittevät julkista tai parametrina välitettyä tietoa
- Toiminnallisuuden asteittainen osittaminen proseduureiksi

Proseduraalisen ohjelman rakenne



Proseduraalisen ohjelmoinnin ongelmat

- Suurissa ohjelmissa paljon pieniä proseduureja
- Proseduurien välinen yhteistyö globaalin tiedon avulla
  - Vaikea jäljittää mitkä proseduurit käyttävät mitäkin globaalia tietoa
  - Tiedon suojaus heikkoa

⇒ Laajojen proseduraalisten ohjelmien ylläpito hankalaa

- Heikko tuki työn osittamiselle

Goto-lauseet

- Imperatiivisissa ohjelmissa keskeisessä roolissa on ohjelman kontrollin ohjaus
- Kontrollia voidaan ohjata erilaisilla rakenteilla:
  - Erittäin monikäyttöisillä hyppy- ja Goto-lauseilla
  - Rajoitetummilla rakenteilla: for, if – else, while …
- Hyppylauseet ainoa tapa ohjata kontrollia konekielissä
- Fortran jäljitteli hyppylauseita Goto-lauseella (muut korkean tason kielet seurasivat perässä)
- Joissakin ohjelmointikielissä Goto-lause oli ainoa tapa hoitaa:
  - Ennenaikaiset poistumiset funktioista
  - Poikkeuskäsittely
  - Silmukan keskeytys

Goto-lauseiden ongelma

- Goto-lauseiden holtiton käyttö johti vaikeasti ymmärrettävään “spagettikoodiin”

```
GOTO 10 ! jump forward
23 CONTINUE
       i = i - 1
       IF (i .eq. 0) GOTO 99
10 PRINT*, "Line 10"
69 j = j - 1 ! loop
       ...
       IF (j .ne. 0) GOTO 69
       GOTO 23 ! jump back
099 CONTINUE
```

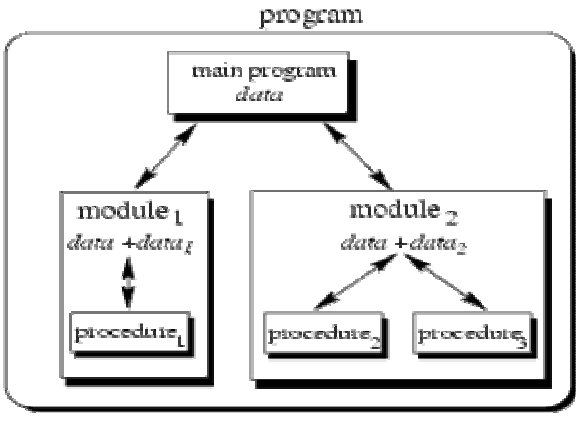
Rakenteinen ohjelmointi

- 70-luvun vaihde
- Tavoitteena helposti ymmärrettävät ja ylläpidettävät ohjelmat
- Ohjelmointi ilman goto-lauseita
- Käytetään rajoitetumpia, mutta helpommin ymmärrettäviä rakenteita: if – else, do – while, for …
- Rakenteilla yksi sisäänmeno ja yksi ulostulo
- Ohjelman kontrollivuota voidaan seurata lukemalla ohjelmatekstiä ylhäältä alas
- Edsger Dijkstralla merkittävä rooli
  - 1968, “Go To Statement Considered Harmful”
    - Piti goto-lauseita haitallisena ohjelman ymmärtämisen kannalta
    - Osoitti että goto-lauseita ei tarvittu, sillä kontrolli voidaan hoitaa pelkästään peräkkäisyyden, toiston ja valinnan mahdollistavilla rakenteilla
  - 1969, ”Notes on Structured Programming”

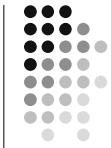
Modulaarinen ohjelmointi

- ~ 1972 – 1988
- Korjaa monia proseduraalisen ohjelmoinnin ongelmia
- Tieto ja sitä käsittelevät proseduurit samaan rakenteeseen = moduuli.
- Moduuli peittää sisältämänsä tiedon esitystavan
- Moduulit erillään käännettäviä rajapintojen asiosta
  => Työ voidaan osittaa moduulitasolla
- Suunnittelun lähtökohtana on tieto
- Moduulaarisia kieliä: C, Clu, Modula-2, Ada 83, ML …
- Moduulityyppi / moduuli manageri

Modulaarisen ohjelman rakenne

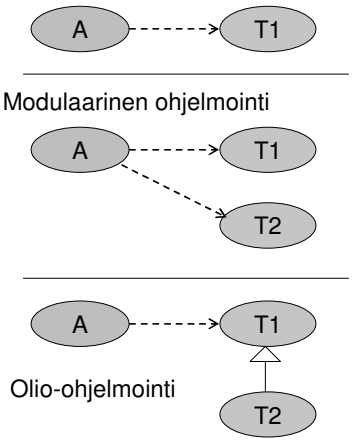


Modulaarisen ohjelmoinnin ongelmat

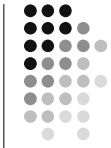


- Algoritmit eivät ole avoimia laajennoksille

```
toimintoX() {
    switch (type)
        case T1 …
        case T2 …
}
```

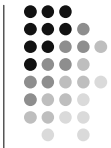


Olio-ohjelmointi



- Nousi esiin C++:an myötä 80-luvun loppupuolella
- Simula 60- ja Smalltalk 70-luvulla
- Korjaa modulaarisen ohjelmoinnin ongelmia
- Uusi rakenneyksikkö: luokka
- Laajentaa modulaarista ohjelmointia perinnällä ja monimuotoisuudella
- Tiedon ositus monella tasolla: globaali, moduuli (luokkataso), olio ja proseduuri
- Ongelmana läpileikkaavat näkökulmat (aspektiohjelmointi)

Yhteenveto



- Laskentakyvyn kasvu on johtanut monimutkaisempiin ja laajempiin ohjelmiin
- Ohjelmien monimutkaisuutta on hallittu erilaisilla organisointiparadigmoilla
- Organisointiparadigmat ohjaavat ohjelman toiminnallisuuden ja tiedon organisointia

Paradigma	**Rakenneyksikkö**
Rakenteeton	-
Proseduraalinen	Proseduuri
Rakenteinen	Kontrollirakenteet
Modulaarinen	Moduuli
Olio-ohjelmointi	Luokka