Ohjelmointiparadigmat

T-106.3100

Ohjelmointiparadigma

- Kuvaa ohjelmien keskeisiä toteutus-periaatteita
  - Minkälaisista komponenteista ohjelma rakentuu?
  - Miten ohjelman kontrollin eteneminen määritellään?
- Eri ohjelmointikielet liittyvät tyypillisesti yhteen paradigmaan, mutta yleensä niillä voidaan jäljitellä muita paradigmoja

Tärkeimmät paradigmat

- Olio-ohjelmointi
- Proseduraalinen (lausekielinen) ohjelmointi
- Funktionaalinen ohjelmointi
- Logiikkaohjelmointi
- Rinnakkaisohjelmointi

Olio-ohjelmointi / yleistä

- Perusidea jo Simula 67:ssä ja Smalltalkissa
  - Tiedon ja siihen liittyvien operaatioiden esittäminen luokkien ja olioiden avulla.
- Läpimurto tapahtui 80-luvun lopulla C++:n myötä
- Tärkeimmät kielet C++ ja Java
- Tullut erittäin yleiseksi, koska luokkien ja olioiden avulla voidaan helposti mallittaa monien reaalimaailman ongelmien käsitteistöä.
- Laajat luokkakirjastot helpottavat uusien sovellusten rakentamista.

Olio-ohjelmointi / periaate

- Ohjelman rakenne jäsennetään datasta käsin
  - Minkälaista tietoa käsitellään ja mitä toimintaa kuhunkin tietoon liittyy?
  - Käsiteltävä tieto jäsennetään luokkina.
  - Ajon aikana tieto talletetaan luokista luotuihin olioihin ja mahdollisesti itse luokkiin.
  - Kuhunkin luokkaan liittyy metodeja, jotka määrittelevät, miten luokassa / luokan olioissa esitettyä tietoa käsitellään.
  - Perintä mahdollistaa analogisten rakenteiden yhteisten/eriävien asioiden luontevan kuvauksen.

Olio-ohjelmointi / kontrolli

- Kontrolli etenee siten, että luokkien metodit kutsuvat toisiaan.
- Pääohjelma ei kontrolloi kokonaisuutta, vaan sitä tarvitaan lähinnä käynnistymiseen.
- Ns. hyppylauseet eivät ole mielekkäitä.

Olio-ohjelmointi / arviointia

Etuja

- Perintä, luokkakirjastot ja tiedon kapselointi helpottavat ohjelmien rakentamista ja ylläpitoa.
- Tietosisällön mallittaminen usein luontevaa.

Ongelmia

- Turhan raskas koneisto sovelluksiin, joissa tietosisältö on rakenteeltaan varsin yksinkertaista
  - Usein teknistieteelliset laskentasovellukset sekä systeemiohjelmat kuvataan luontevammin proseduraalisten kielten avulla
- Erityisesti Javassa olio-ominaisuuksien systemaattinen käyttö tuo ohjelman suoritukseen raskautta.

Proseduraalinen ohjelmointi / yleistä

- Ohjelman rakenne jäsennetään toiminnasta käsin.
  - Keskeiset abstraktiot määrittelevät toimintoja (funktiot/proseduurit)
- Varhaisin paradigma, jota käytettiin jo konekielisen ohjelmoinnin tasolla
- Keskeisiä kieliä C, Fortran, Ada, (Pascal)
- Hyvin tärkeä teknismatemaattisissa sovelluksissa ja systeemiohjelmoinnissa. Näissä käytössä laajoja aliohjelmakirjastoja.

Proseduraalinen ohjelmointi / periaate

- Suunnittelussa keskeisenä asiana, miten ohjelma jaetaan toiminnallisiin osiin.
  - Top-down-suunnittelussa ohjelman kokonaistoiminta jaetaan osiin, joita edelleen tarkennetaan pienempinä osina (asteittain tarkentamisen periaate)
  - Bottom-up-suunnittelussa pohditaan aluksi sitä, mitä toiminnallisia abstraktioita tarvitaan ohjelman toteuttamisessa.
  - Käytännössä molempia tarvitaan.

Proseduraalinen ohjelmointi / kontrolli

- Ohjelmaa suoritetaan lause kerrallaan, apuna ehto- ja toistolauseet.
- Toiminnalliset abstraktiot kuvataan funktioina, joille voidaan välittää parametreja.
- Pääohjelma jäsentää ohjelman kokonaisuuden eräänlaisena sisällysluettelona.
- Pääohjelma kutsuu funktioita/proseduureja toteuttamaan yksityiskohtia. Nämä vastaavasti kutsuvat muita funktioita/proseduureja.
- Suuret ohjelmat jaetaan eri tiedostoihin, esim. moduuleihin.

Proseduraalinen ohjelmointi / tietojen esittäminen

- Luonteeltaan staattinen tieto esitetään yksinkertaisissa ja taulukkomuuttujissa
- Dynaaminen tieto esitetään tietorakenteina (listat, puut, ...), joissa alkioista toiseen viitataan *osoitinmuuttujien* avulla.
- Abstraktit tietotyypit tärkeitä
  - Samoja periaatteita kuin olio-ohjelmoinnissa (tiedon kapselointi, moduulien rajapinnat)
- Ei perintää, joskin sitäkin voidaan simuloida.

Funktionaalinen ohjelmointi / yleistä

- Ohjelman rakenne jäsennetään toiminnasta käsin.
  - Puhtaassa funktionaalisessa ohjelmoinnissa **kaikki** toiminnot tehdään funktiokutsujen avulla, ei siis suoriteta peräkkäisiä lauseita.
- Keskeisin kieli LISP (v. 1960) ja sen eri murteet (mm. Common Lisp, Scheme)

Funktionaalinen ohjelmointi / periaate

- Kaikki toiminnalliset asiat ilmaistaan funktioiden ja niiden argumenttien avulla.
- Rekursio erittäin tärkeää
  - Toisto tehdään yleensä rekursion avulla
- Tietorakenteet esitetään taulukkoina tai listoina, joista viitataan toisiin listoihin.
- Useissa kielissä mukana myös proseduraalisia elementtejä. (esim. Scheme)

Funktionaalinen ohjelmointi / muuta

- Toteutukset usein tulkkeja, jolloin ohjelmakehitys hyvin vuorovaikutteista.
- Kääntäjiä tarvitaan tehokkuussyistä.
- Monissa kielissä laaja valmis funktiovalikoima.
- Etuina yksinkertainen syntaksi (LISP) ja suuri ilmaisuvoima.

Paradigman valinnasta

- Ohjelmoijan ammattitaitoon kuuluu kyky arvioida, minkä tyyppinen kieli / paradigma kuhunkin sovellukseen parhaiten soveltuu.
- Ei ole olemassa ratkaisua, joka toimii kaikkialla.