Orientaatiotehtävä

**Tutustu** johdantolukuun ”A Brief Introduction to C++” kirjasta

*Andrist, B., Sehr, V., C++ High Performance - Second Edition. Packt, 2020.*

Johdantoluku (Chapter 1: A brief introduction to C++) on luettavissa ilmaiseksi [täältä](https://www.ebooks.com/en-fi/book/210193114/c-high-performance/bjorn-andrist/?_c=1) (”Click to preview”). Jos et ole kovin kokenut C++ -ohjelmoija, saattavat useat johdannon aiheet ja koodiesimerkit vaikuttaa heprealta. Älä kuitenkaan anna periksi, vaan lue koko teksti, ja pyri muodostamaan jonkinlainen yleiskuva käsiteltävistä asioista.

**Pyri siihen, että osaisit lopulta vastata jollakin tasolla seuraaviin kysymyksiin seuraavaan luentoon mennessä (kirjaa esim. muutama ranskalainen viiva joka kohdasta):**

1. Mitä ”abstraktioilla” tarkoitetaan ohjelmointikielten yhteydessä?

* Abstraktiot ohjelmointikielissä viittaavat monimutkaisten yksityiskohtien piilottamiseen käyttäjiltä tehdäkseen koodista helpommin luettavaa ja käytettävää.

1. Mitä tarkoitetaan käsitteellä ”zero-cost abstractions”? Millainen C++ on tässä suhteessaverrattuna muihin ohjelmointikieliin?

* Zero-cost abstractions" tarkoittaa, että abstraktioiden käyttö ohjelmoinnissa ei juurikaan lisää ylimääräistä suoritusaikaa tai hidasta ohjelmaa. C++ on pyrkinyt implementoimaan abstraktioita vähemmillä “suoritushinnoilla” kun suurin osa muista kielistä.

1. Mitä tarkoittaa ”zero-overhead”-periaate? Mitkä C++:n ominaisuudet EIVÄT noudata tätäperiaatetta?

* "Zero-overhead" -periaate C++:ssa tarkoittaa, ettet maksa ominaisuuksista, joita et käytä ja, että ominaisuudet joita käytät ovat jo optimoitu niin hyviksi kun mahdollista. Seuraavat eivät noudata pariaatetta:   
    
  - “(RTTI) Run-Time Type Information,” eli muuttujan tyypin tarkistus suoritusaikana.   
  - “Exceptions:” virheiden käsittely suoritusaikana. Try..catch.

1. Mitkä C++:n ominaisuudet edesauttavat toimintavarman (”robust”) koodin kirjoittamista?

* Strict ownership of resources, Const correctness, Value semantics, Type safety & Deterministic destruction of objects

1. Miten Java-ohjelman kääntäminen eroaa C++ -ohjelman kääntämisestä?

* Java kääntää koodin “bittikoodiksi ja siitä konekieleksi, kun taas C++ kääntää koodin suoraa konekileksi ennen suoritusta.

1. Miten dynaamisesti varatun muistin (olioiden) vapautus eroaa Javassa ja C++ :ssa?

* Java vapauttaa muistin automaattisesti, kun C++ taas vaatii manuaalisen muistin vapautuksen.

1. Millä tavoilla C++ tarjoaa enemmän vaihtoehtoja olioiden allokointiin kuin Java?

* C++ mahdollistaa molempien stackin ja heapin käytön sekä useamman objektin luomisen yhteen heap varaukseen. Kun Java taas sijoittaa kaikki objektit omiin heap varauksiinsa.

1. Miksi käytettävät oliot olisi hyvä sijoittaa lähekkäin muistissa?

* Jotta niiden tilan varaamiseen ja muistista hakemiseen ei menisi niin paljoa resursseja.

1. Mitä C++:n tarjoamat ”value semantics” ja ”reference semantics” tarkoittavat?

* C++:n "value semantics" tarkoittaa, että oliot kopioidaan siirrettäessä tai sijoitettaessa. "Reference semantics" puolestaan tarkoittaa, että oliosta siirretään tai sijoitetaan vain viite tai osoitin.

1. Yritä selvittää, mitä tekstissä esiintyneet std::shared\_ptr ja std::weak\_ptr ovat.

* Shared\_ptr mahdollistaa, että useampi osoitin voi omistaa saman objektin. Weak\_ptr ei omista objektia tai vaikuta sen elinkaareen, mutta pitää kumminkin referenssin kohteeseen tallessa.

1. Mitä ”const correctness” -konsepti tarkoittaa?

* "Const correctness" C++:ssa tarkoittaa muuttujien ja jäsenfunktioiden ilmoittamista const-avainsanalla osoittaakseen, etteivät ne voi muuttua.

1. Useassa koodiesimerkissä käytetään ”auto”-avainsanaa. Mitä se mahtaa tarkoittaa?

* Ikään kuin “yleistyyppi,” jota voi käyttää muiden tyyppien sijasta. Autoa käyttäessa muuttujan tyyppi määritellään kun sille alustetaan arvo.

1. Mitä tarkoittaa, että olioiden tuhoaminen on C++:ssa determinististä (”deterministic destruction”). Miksi se on hyvä asia?

* Tarkoittaa, että objektin elinkaari on täysin ohjelmoijan ohjattavissa, sen sijaan, että se tuhottaisiin esim automaattisesti. Hyviä puolia on esim, että kaikki objektin elinkaaren aikana tehdyn muutokset ovat luotettavasti peruutettavissa, tekee myös C++:sta ennalta arvattavamman.

1. Miten null-referenssien käsittely poikkeaa Javassa ja C++:ssa?

* C++:sassa referenssit eivät voi olla NULL.

1. Mitkä ovat tärkeimmät artikkelissa mainitut C++:n puutteet? Mitä mieltä itse olet?

* Suurimmat puutteet ovat kirjastojen vähäisä määrä ja niiden importtauksen vaikeus ja itse koodin pitkä käännös aika.