8100410 Olio-ohjelmoinnin jatkokurssi

Tentti 8.5.2003

Tentissä ei saa käyttää ylimääräistä kirjallista materiaalia, laskimia, tietokoneita tai muita lunttausvälineitä.

Muutama sana tenttivastauksen kirjoittamisesta:

1. Vastauksessa oleta vastaavasi sellaisen ihmisen esittämään kysymykseen, joka tuntee kohtalaisen hyvin ohjelmistotekniikan aihealuetta muutoin paitsi juuri tämän kysymyksen osalta.
2. Mieti etukäteen esim. ranskalaisilla viivoilla vastauksesi pääkohdat ja lajittele ne johdonmukaiseen järjestykseen — älä kirjoita yhteen pötköön kaikkea mieleen tulevaa.
3. Muista vastata kaikkiin tehtävän kysymyslauseisiin, sillä täysiä pisteitä ei voi saada jos kaikkiin kysyttyihin asioihin ei ole vastattu.
4. Järjen käyttö on sallittua, jopa toivottavaa...

Palauta kaikki *nimetyt* vastauspaperit *omiin pinoihinsa*!

..... Tehtävät 1. & 2. omalle paperilleen! **Nimi paperiin!** .....

1. Vastaa seuraaviin kysymyksiin lyhyillä virkkeillä esim. ranskalaisia viivoja käyttäen (n. 3-5 kysymystä kohti). Muista keskittyä kysymyksissä olio-ohjelmoinnin kannalta *oleellisimpiin* asioihin, älä tuhlaa tilaa vain tarpeettomiin pikkuyksityiskohtiin.

   a) Olioiden kopioiminen olio-ohjelmoinnissa yleensä ja C++:ssa. Mitä siihen liittyy, miksi sitä tarvitaan ja mitä siinä täytyy erityisesti ottaa huomioon. (3 p)

   b) Dynaaminen sitominen on tärkeä ominaisuus periytymisessä. C++:ssa sitä ei kuitenkaan käytetä oletusarvoisesti, vaan se täytyy erikseen ottaa käyttöön avainsanalla `virtual`. Mitä negatiivisia vaikutuksia virtuaalifunktioilla ja dynaamisella sitomisella voisi olla? (3 p)

2. Seuraavassa on joukko väittämiä olio-ohjelmoinnista ja C++:sta. Mitkä väittämät ovat oikein, mitkä väärin? Perustele mielestäsi vääristä väittämistä n. 3–6 rivillä, *miksi/miten* väittämä on väärin ja miten asia todellisuudessa on. (6 p)

   a) Luokkafunktiot (*static member functions*) ovat luokan vastuualueeseen liittyviä palveluita, joiden suorittaminen ei kuitenkaan kuulu minkään yksittäisen olion vastuulle.

   b) Suunnittelumallit (*design patterns*) ovat oliosuunnittelupohjia, jotka on tarkoitettu helpottamaan ohjelman suunnittelussa alkuunpääsemistä.

   c) C++:n toteutusmalleissa (*template*) tyyppiparametriksi käyvät mitkä tahansa tyypit, joilla ko. toteutusmallin koodi kääntyy.

   d) Matalakopioinnissa (*shallow copy*) periytetyn luokan oliosta kopioidaan vain sen omat jäsenmuuttujat, mutta kantaluokan jäsenmuuttujat jäävät kopioimatta.

   e) STL:n sisältämät algoritmit ovat yleiskäyttöisiä, koska ne ovat jäsenfunktioina omassa kantaluokassaan, josta omat näitä algoritmeja käyttävät tietorakenteet voi periyttää.

   f) Abstraktit kantaluokat (*abstract base classes*) ovat luokkia, joista ei enää voi periyttää aliluokkia.

KÄÄNNÄ!

..... Tehtävät 3. & 4. omalle paperilleen! **Nimi paperiin!** .....

3. Selitä (max. 5 riviä/kohta) seuraavat olio-ohjelmoinnin ja C++:n käsitteet ja mitä hyötyä niistä saadaan olio-ohjelmoinnissa. ***Älä*** selitä niistä pelkkää syntaksia tms. vaan kerro etupäässä, mitä ko. käsitteet *tarkoittavat*. (1 p/kohta)

   a) Luokkamuuttuja (*static data member*)

   b) Viipaloituminen (*slicing*)

   c) Virhehierarkia (*exception hierarchy*)

   d) Sopimussuunnittelu (*design by contract*)

   e) Rajapintaluokka (*interface class*)

   f) Funktiomalli (*function template*)

4. Seuraavassa listauksessa on esitetty yksinkertainen C++:n funktiomalli (*function template*), joka etsii ja palauttaa annetusta vektorista pienimmän alkion.

   a) Mitä ominaisuuksia (jäsenfunktioita, operaatioita yms.) se vaatii tyyppiparametriksi T annettavalta tyypiltä? (4 p)

   b) Kirjoita funktiomallista vastaavan toiminnallisuuden tarjoava "optimoidumpi" versio, jossa vaaditaan tyyppiparametrilta mahdollisimman vähän. Listaa myös tämän toteutuksen vaatimukset. (2 p)

```
#include <vector>

template <typename T>
T etsiPienin (const std:: vector<T>& taulukko)
{
  if (taulukko.empty())
  {
    // Virhekasittely jatetty tilan puutteen vuoksi pois,
    // eika sita tarvitse olla omassa koodissakaan. :)
  }

  T pienin;
  pienin = taulukko[0]; // Olisiko ensimmainen pienin?
  for (unsigned int i = 1; i < taulukko.size(); ++i)
  {
    if (taulukko[i] < pienin)
    {
      pienin = taulukko[i];
    }
  }

  return pienin;
}
```