Kotitehtävien tekeminen ja arvostelu

Tarkoitus

Ohjelmointia voi oppia vain itse tekemällä, kokeilemalla ja pohtimalla, mitä ohjelman suorituksen aikana tapahtuu.

Ongelmanratkaisu. Kotitehtäviä tehdessä opetellaan, kuinka ongelma pilkotaan pienempiin osaongelmiin, joiden ratkaisemiseksi kehitetään algoritmi. Vaatimusten mukaisesti varsinainen ohjelma toteutetaan C-kielellä.
Kotitehtävät tutustuttaa myös testaukseen yksinkertaisimmilla tasolla. Tällä varmistutaan, että ohjelma ratkaisee määritetyn ongelman ja toimii kaikissa tilanteissa oikein. Lisäksi ohjelmistokehitykseen kuuluu riittävän dokumentaation tuottaminen.

Vaatimukset

Kotitehtävässä on osoitettava keskeisten Ohjelmointi I -kurssin asioiden hallitseminen. Ylimääräisistä rajoituksista on syytä neuvotella työn ohjaajan kanssa ennen työn palauttamista. Tehtävässä saa olla lisäominaisuuksia, jos ne liittyvät selkeästi alkuperäiseen ongelmaan. Myös ne vaikuttavat arvosteluun joko nostavasti tai laskevasti (jos huonosti toteutettuja).

Ohjelmointikieli. Ohjelmointi I -kurssilla käytettävä ohjelmointikieli on C. Ratkaisussa saa hyödyntää vain vakio C-kirjastoja, eli esimerkiksi erilaisten työkalujen omia laajennuksia ei saa käyttää. Muuten työn voi tehdä missä tahansa ympäristössä.

Rakenne. Ohjelmakoodin tulee olla rakenteeltaan selkeä. Käytännössä se tarkoittaa esimerkiksi funktioiden käyttämistä siten, että ongelma on ositettu perusteltuihin osiin. Ohjelma koostuu pääfunktiosta (main) ja yhdestä tai useammasta alifunktiosta, joilla on tarkasti määritelty tehtävä. Pääfunktio kannattaa pitää mahdollisimman lyhyenä.

Tarkistukset. Ohjelman on tarkistettava syötteet. Näitä ovat esimerkiksi käyttäjän antamat arvot näppäimistöltä tai komentoriviltä sekä tiedostojen sisällöt. Ohjelma ei saa kaatua sellaiseen virhetilanteeseen, jossa käyttäjä antaa vaikkapa kokonaisluvun sijaan kirjaimia.

Kommentit. Jokainen funktio on dokumentoitava käyttäen kommentteja. Ennen kutakin funktiota on oltava kommenttilohko, jossa kerrotaan funktion tarkoitus ja kuvataan yleisellä tasolla sen toimintaperiaate. Tarkoitus ei kuitenkaan ole toistaa luonnollisella kielellä ohjelman koodia rivi riviltä. Lisäksi kommenteissa on lueteltava kaikki funktion parametrit ja paluuarvo (return) sekä kuvattava niiden merkitys.

Funktion dokumentoiva kommentti voi näyttää esimerkiksi tältä:

```
/* kysyKokonaisluku - kysyy käyttäjältä kokonaisluvun
 * Tulostaa ruudulle parametrina annettavan kysymyksen
 * ja lukee sen jälkeen näppäimistöltä int -tyyppisen
 * syötteen.
 * Paluuarvona on tämä luettu luku.
 *
 * Parametrit:
 * - kysymys (char *): teksti, joka tulostetaan ruudulle
 *
 * Paluuarvo (int): luku, jonka käyttäjä syöttää
 * vastauksena
 *
 */

int kysyKokonaisluku(char * kysymys) {
   int luku;
   // ...
   return luku;
}
```

Kommentoinnin on oltava selkeä. Itse funktion sisään ei tarvitse kirjoittaa kommentteja. Esimerkiksi muuttujien nimien tulee olla itsessään kuvaavia, jotta erillisiä kommentteja ei tarvita. Ohjelman toiminnan ymmärtämisessä rakenne erityisesti funktiojaon ja em. funktioiden dokumentoinnin suhteen tulee olla niin selkeää, että se vähentää kommenttien tarvetta.

Erityisesti lopettavien aaltosulkeiden kommentointia (esimerkiksi `} // for`) on syytä välttää, koska se johtaa helposti harhaan. Kyseistä menettelyä käytetään luentoesimerkeissä vain helpottamaan opettelua, mutta se ei kuulu hyviin ohjelmointikäytäntöihin.

Ulkoasu. Lähdekoodin tulee olla ulkoasultaan siistiä. Lähtökohtana on, että ohjelma on sisennetty asiallisesti ja jokaisella rivillä on vain yksi ohjelmalause (tai ei yhtään). Kannattaa miettiä, minkälaista lähdekoodia itse haluaisi lukea.

Lähdekoodin muotoilussa suositellaan, että jokaisen aaltosulun aloittamaa lohkoa sisennetään yhdellä tasolla, vakiot kaikki isoilla kirjaimilla ja loput (funktiot ja muuttujat) pienillä kirjaimilla. Alaviivaa (_) käytetään vain vakioiden yhteydessä; muuten keskellä nimeä sanat erotetaan isolla alkukirjaimilla.

Selostus

Selostus palautetaan tekstidokumenttina tai -tiedostona nimeltä `lueminut.pdf`, `lueminut.rtf` tai `lueminut.txt`. Sallitut tiedostomuodot ovat siis PDF (Portable Document Format / Adobe Acrobat), RTF (Rich Text Format) ja puhdas teksti (.txt-päätteellä). Testiajot tulee olla tallennettu pelkkää tekstiä sisältävään tiedostoon `testit.txt` ilman muotoiluja.

Selostuksessa tulee olla seuraavat luvut:
1. Ongelma. Lyhyt kuvaus ohjelman tarkoituksesta ja toiminnasta omin sanoin. Ideana on kertoa, miten tehtävän on itse ymmärtänyt.

2. Ratkaisu. Toisessa luvussa kerrotaan ratkaisun mahdollinen teoreettinen tausta ja annetaan ohjelman rakenteen yleiskuvaus. Tarkoituksena on selittää, kuinka alkuperäinen ongelma on jaettu pienempiin osaongelmiin. Kuvauksessa annetaan yleisen tason algoritmi sekä mahdolliset osaongelmien algoritmit. Lisäksi esitetään ohjelmassa käytettyjen funktioiden sekä tiedostojen nimet ja käyttötarkoitukset. Nämä tiedot on järkevintä koota esimerkiksi taulukkoon. Funktioiden tarkemmat kuvaukset löytyvät **vaatimusten** mukaan lähdekoodin kommenteista.
3. Käyttöohjeet. Luvun tarkoituksena on kuvata keskiverrolle tietokoneenkäyttäjälle, miten ohjelma käynnistetään ja miten sitä käytetään. Erityistä huomiota tulee kiinnittää ohjelman syöttötietojen muotoon ja tulostuksien tulkintaan. Lisäksi tulee kirjata, mitä rajoituksia ohjelman käytössä mahdollisesti on (esimerkiksi syötettävien sanojen pituudelle tai numeroiden vaihteluvälille annetaan usein rajoituksia).

4. Kokemukset. Lopuksi kuvataan lyhyesti kotitehtävien tekemisessä ilmenneitä vaikeuksia ja kokemuksia.

4. Testaus. Testauksesta laaditaan lyhyt suunnitelma, josta tähän lukuun kootaan, minkälaisia testiajoja ohjelmalle on tehty. Testien on katettava sekä ohjelman normaali toiminta että poikkeustilanteet, joten niitä on oltava useita. Testiaineisto tulee laatia sellaiseksi, että se koettelee ohjelman toimintaa erityisesti virhealttiissa kohdissa, esimerkiksi ohjelman syöttötietojen ollessa virheellisiä. Lisäksi jokaisen ohjelmalauseen on tultava suoritetuksi. Esitystapana voi olla esimerkiksi taulukko, jonka sarakkeina on erilaiset syöttötiedot ja odotetut tulosteet.
Testit on myös ajettava. Testien aikana syntyneet ohjelman tulostukset on tallennettava tiedostoon nimeltä `testit.txt`, ja se on toimitettava palautuksen yhteydessä. Itse selostukseen niitä ei kopioida.

Palauttaminen ja tarkastus

Palauttaminen. Kotitehtävä toimitetaan Moodle -järjestelmän kautta. Niin kauan kuin työlle on palautusaikaa, suoritusta voi korjata toimittamalla uusi versio tehtävästä.

Plagiointi. Plagiointi on toisen työn esittämistä omana. Se on *ehdottomasti kiellettyä*. Myös toisen henkilön ideoiden ja ajatusten esittäminen omina on epäeettistä toimintaa ja sitä voidaan pitää plagiointina. Opiskelusuoritusten (esimerkiksi toisten harjoitustöiden ja muiden kurssisuoritusten, kuten kuvien, tekstien ja multimediatöiden) esittäminen omana ilman lähdeviittauksia on plagiointia. Yleensä myöskään omaa työtään ei saa käyttää uudelleen jollakin toisella kurssilla. Plagiointiin syyllistyminen johtaa vähintään kyseisen kurssisuorituksen hylkäämiseen, mutta rangaistuksena voi olla jopa määräaikainen erottaminen.

Arvostelu (tehtävät 1- 8 yhteensä)

40 pistettä: ammattitaitoinen, erinomainen suoritus. Opiskelijan suoritus ylittää selvästi harjoitustyön vaatimukset ja tavoitteet varsinkin laadullisesti mutta mahdollisesti myös määrällisesti. Näin opiskelija osoittaa syvällisesti ymmärtävänsä ongelman ja sen ohjelmallisen ratkaisemisen periaatteet sekä pystyy soveltamaan poikkeuksellisen lahjakkaasti oppimaansa. Työn looginen rakenne on johdonmukainen ja vaadittua tasoa selkeämpi valittujen suunnitteluratkaisujen ansiosta. Seloste on moitteeton, selkeä sekä huolellisesti viimeistely, ja siinä on tuotu

erinomaisesti esiin ongelmakenttä ja sen ratkaisuperiaatteet. Opiskelija on pystynyt itsenäiseen ammattitaitoa osoittavaan työskentelyyn.

30: odotukset ylittävä, esimerkillinen suoritus. Opiskelijan suoritus täyttää erinomaisesti kaikki harjoitustyön vaatimukset erityisesti laadullisesti. Lisäksi opiskelija osoittaa syvällisesti ymmärtäneensä ongelman ja sen ohjelmallisen ratkaisemisen periaatteet. Työn looginen rakenne on esimerkillisen selkeä ja johdonmukainen. Ratkaisua voisi käyttää mallina opetuksessa. Seloste on moitteeton ja kuvaa ongelmakentän ja sen ratkaisuperiaatteet hyvin. Opiskelija on pystynyt itsenäiseen työskentelyyn.

20: odotuksia vastaava, hyvä suoritus. Opiskelijan suoritus täyttää kaikki harjoitustyön vaatimukset. Opiskelija ymmärtää ongelman ja sen ohjelmallisen ratkaisemisen periaatteet. Ratkaisu on laadukas keskeisimmillä osa-alueilla. Työn looginen rakenne on johdonmukainen. Seloste kuvaa ongelmakentän ja sen ratkaisuperiaatteet vaatimusten mukaisesti. Opiskelija pystynyt pääsääntöisesti itsenäiseen työskentelyyn.
10: tyydyttävä suoritus. Opiskelijan suoritus täyttää työn perusvaatimukset, mutta joillakin osa-alueilla on pieniä puutteita. Opiskelija ymmärtää periaatteessa ongelman ja sen ohjelmallisen ratkaisemisen periaatteet, mutta niiden soveltamisessa on vaikeuksia. Työ ei ole rakenteeltaan looginen, minkä vuoksi valittuja ratkaisuja voi olla vaikea ymmärtää. Selosteessa on puutteita kieli- tai ulkoasussa tai ongelmakentän ja sen ratkaisun kuvauksessa.

5 pistettä: kehittymistä edellyttävä suoritus. Opiskelijan suoritus ei täytä kaikkia vaatimuksia ja työssä on puutteita. Työn tekijällä on suuria vaikeuksia ongelman ja sen ratkaisun hahmottamisessa. Hän on kuitenkin selvästi osoittanut aktiivista otetta työn kohteena olevien asioiden harjoittelussa, ja huomattavaa kehittymistä on jo tapahtunut.. Tämä on tärkeää esimerkiksi jatkokursseilla menestymisen kannalta.