LUTin C-ohjelmoinnin tyyliohje

1. Johdanto

Tämä on LUTin C-ohjelmointikurssien tyyliohje. Nämä ohjeet on tarkoitettu kurssin harjoitustyön valikko-pohjaisen ohjelman tyyliohjeeksi auttamaan toimivan ja ymmärrettävän C-ohjelman tekemisessä. Tätä ohjetta voi soveltaa muissa vastaavissa projekteissa, mutta lähtökohtaisesti kaikilla ohjelmointiin vakavasti suhtautuvilla organisaatioilla ja projekteilla on omiin tarpeisiin sovitetut tyyliohjeet, joita tulee noudattaa ja soveltaa organisaation sisäisten ohjeiden mukaisesti.

LUTin C-ohjelmointikursseilla seuraavia ohjeita on noudatettava kaikissa ohjelmissa:

1. Ohjelman on käännyttävä, mentävä automaattitarkastuksesta läpi ja toimittava tehtäväksiannon mukaisesti
2. Tiedostojen avaamisen ja muistin varaamisen yhteydessä on oltava virheenkäsittely
3. Kaikki ohjelman varaamat resurssit tulee vapauttaa ohjelman lopuksi
4. Globaalit muuttujat ovat kiellettyjä
5. goto-lauseen käyttö on kielletty
6. Kaikissa palautettavissa tiedostoissa on oltava alkukommentit
7. Kaikkien pää- ja aliohjelmien on päätyttävä return-käskyyn

Edellä olevien pakollisten ohjeiden noudattamatta jättäminen johtaa aina työn korjaamiseen tai hylkäämiseen tilanteesta riippuen. Muiden ohjeiden noudattaminen on kurssilla opetettavan ymmärrettävän ja ylläpidettävän ohjelmointityylin lähtökohta. Mikäli näistä ohjeista tulee poiketa, on se mainittu erikseen tehtäväksiannossa. Perustelut näille tyyliohjeille löytyy LUTin C-ohjelmointioppaasta ja tällä kurssilla hyvästä ohjelmointityylistä päättää kurssin vastuuopettaja.

2. Rakenne

2.1. Tiedostorakenne, yksi tiedosto

Tiedoston rakenne, kun ohjelma koostuu vain yhdestä tiedostosta:

1. Tiedoston alkukommentti
2. Kirjastojen otsikkotiedostojen sisällytys
3. Vakioiden määrittely
4. Uusien tietorakenteiden määrittely, erityisesti tietueet
5. Aliohjelmien esittelyt sisältäen muuttujien nimet
6. Pääohjelman koodi
7. Aliohjelmien koodi samassa järjestyksessä kuin esittelyssä

2.2. Tiedostorakenne, useista tiedostoista muodostuva ohjelma

Laajassa ohjelmassa on 1 pääohjelma ja paljon aliohjelmia. Aliohjelmat tulee jakaa tarpeen tulleen useisiin tiedostoihin alla olevien tyyliohjeiden mukaisesti:

Pääohjelman sisältävät tiedoston rakenne:

1. Tiedoston alkukommentti
2. Standardikirjastojen otsikkotiedostojen sisällytys
3. Omien kirjastojen otsikkotiedostojen sisällytys
4. Pääohjelman koodi

Muiden C-tiedostojen rakenne:

1. Tiedoston alkukommentti
2. Standardikirjastojen otsikkotiedostojen sisällytys
3. Omien kirjastojen otsikkotiedostojen sisällytys
4. Aliohjelmien koodi samassa järjestyksessä kuin esittelyissä

Otsikkotiedostojen sisällytykset tehdään aina lähdekooditiedostoissa, ei otsikkotiedostoissa.

Jokaiseen C-tiedostoon tarvittaessa liittyvän otsikkotiedoston eli .h-tiedoston rakenne:

1. Tiedoston alkukommentti
2. Otsikkotiedoston ehdollinen sisällytys
3. Vakioiden määrittely
4. Tietorakenteiden määrittely

5. Aliohjelmien esittelyt sisältäen muuttujien nimet

2.3. Tiedoston alkukommentti

Kaikki kurssin ohjelmointitehtävät ovat henkilökohtaisia ja siksi palautettaviin ohjelmatiedostoihin tulee laittaa alkukommentti. Kurssin kaikkiin palautettaviin tiedostoihin tulee laittaa kommenttina seuraavat tiedot:

```
/* pvm, tekijä, tiedostonimi, tehtävä */
```

Kurssin harjoitustyössä tulee olla alla oleva yksityiskohtaisempi kommentti. Kommentissa mainitut koodiin vaikuttaneet lähteet ja henkilöt auttaa kurssihenkilöstöä ymmärtämään koodin taustat ja välttämään turhia vilppiepäilyjä.

```
/*****************************************************************************/
/* CT60A2500 C-ohjelmoinnin perusteet
 * Tekijä:
 * Opiskelijanumero:
 * Päivämäärä:
 * Palauttamalla tämän työn arvioitavaksi vakuutan, että
 * 1) Olen itse kirjoittanut kaiken tässä tiedostossa olevan koodin
 * 2) En ole antanut tätä koodia kenenkään muun käyttöön
 *
 * Kurssin oppimateriaalien lisäksi työhön ovat vaikuttaneet seuraavat
 * lähteet ja henkilöt, ja se näkyy koodissa seuraavissa kohdissa:
 * - [esim. assistentti x: tiedoston lukeminen]
 * - ...
 */
/*****************************************************************************/
/* Tehtävä x, tiedoston nimi y */

/* eof */
```

2.4. Otsikkotiedoston ehdollinen sisällytys

Useista tiedostoista muodostuvan ohjelman otsikkotiedostoissa tulee olla ehdollinen sisällytys

1. Käytettävän vakion nimi on sama kuin tiedostonimi suuraakkosilla, esim. tiedosto.h - tiedoston vakion tulee olla TIEDOSTO_H
2. Ehdollinen sisällytys muodostuu seuraavista käskyistä: #ifndef - #define - #endif

2.5. Ohjelman rakenne

Pää- ja aliohjelmat muodostuvat lähtökohtaisesti seuraavista osista tilanteen mukaan:

1. Muuttujien määrittelyt ja alustukset
2. Tietojen kysyminen käyttäjältä
3. Toiminnallinen osuus, laskenta
4. Tulosten tulostaminen käyttäjälle
5. Lopetusrutiinit, muistin vapautus ja käyttäjän informointi ohjelman loppumisesta

2.6. Pääohjelman rakenne

Valikko-pohjainen ohjelma perustuu toistorakenteen sisällä olevaan valinta-rakenteeseen

1. Toistorakenteena käytetään do-while -rakennetta
2. Ohjelman normaali lopetus päättyy while-ehdon kautta.
3. Ohjelma voi päättyä aiemmin normaalin virheenkäsittelyn seurauksena exit-käskyllä
4. Ohjelman valikon käsittely tapahtuu omassa valikko-aliohjelmassa
5. Valintarakenteena käytetään monihaaraista if-rakennetta
6. Valintarakenteen valinnat käydään järjestyksessä alkaen valinnasta 1 valintaan N, jonka jälkeen on valinta 0 eli ohjelman lopetus
7. Jokaisen valinnan kohdalla tavoite on tehdä kaikki valintaan liittyvät toimenpiteet alusta loppuun asti, esim. tiedoston lukemisen yhteydessä kysytään luettavan tiedoston nimi, luetaan tiedosto ja kerrotaan käyttäjälle, että lukeminen onnistui
8. Mikäli valinnan suorittaminen edellyttää tiettyjä tietoja, esim. linkitettyä listaa, tarkistetaan tämän tiedon olemassaolo ennen siihen liittyvien operaatioiden suorittamista ja tarvittaessa kerrotaan käyttäjälle, ettei operaatioita voida suorittaa tiedon puuttumisen vuoksi. Käyttäjätiedotteissa on pyrittävä kertomaan selkeästi ongelma ja miten se voidaan ratkaista
9. Valintarakenteen viimeinen haara on else-haara, jossa käsitellään tuntemattomat valinnat
10. Valintarakenteen jälkeen viimeinen käsky ennen toistorakenteen lopetusehtoa on tyhjän rivin tulostus
11. Ohjelman kaikki lopetusrutiinit ovat toistorakenteen jälkeen ennen pääohjelman loppua

Pääohjelman ja valinta-rakenteen standardifraasit ovat seuraavat:

1. "Tuntematon valinta, yritä uudestaan.\n"
2. "Kiitos ohjelman käytöstä.\n"

Valinta-rakenteessa tyypillisiä ohjeita ja tiedotteita käyttäjälle ovat mm. seuraavat:

1. "Anna luettavan tiedoston nimi: "
2. "Anna kirjoitettavan tiedoston nimi: "
3. "Ei analysoitavaa, lue tiedosto ennen analyysiä.\n"
4. "Ei kirjoitettavia tietoja, analysoi tiedot ennen tallennusta.\n"
5. "Tiedosto '%s' luettu.\n"
6. "Tiedosto '%s' kirjoitettu.\n"
7. "Tiedosto '%s' luettu ja tulostettu.”
8. "Tiedot luettu linkitettyyn listaan.”
9. "Muisti vapautettu.”

2.7. valikko-aliohjelma

Valikko-pohjaisen ohjelman valikko-aliohjelma tulee toteuttaa seuraavalla tavalla:

1. Aliohjelma ei saa parametrejä ja se palauttaa käyttäjän valinnan kokonaislukuna
2. Valikon jokainen rivi tulostetaan omalla printf-käskyllä, joka päättyy rivinvaihtoon
3. Käyttäjän valinta luetaan scanf-käskyllä ja sen jälkeen puskurista poistetaan rivinvaihtomerkki
4. Käyttäjän valinnat numeroidaan alkaen luvusta 1 ja viimeisenä on valinta Lopeta luvulla 0, numeron jälkeen on)-merkki ja välilyönti

valikko-aliohjelman standardifraasit ovat seuraavat:

1. "Valitse haluamasi toiminto:\n"
2. "Anna valintasi: "

2.8. Aliohjelmat

Aliohjelman tarve

Uusi aliohjelma tulee tehdä, jos sama/vastaava operaatio tehdään ohjelmassa useammin kuin kerran ja aliohjelma tarjoaa käyttäjälle systemaattisen ja samanlaisen toiminnon joka kerta.

Loogiset kokonaisuudet tulee sijoittaa omiin aliohjelmiin. Esimerkiksi tiedoston lukeminen, tiedoston kirjoittaminen, tietojen kysyminen, tulostaminen, listan luominen, listan läpikäynti, listan tyhjennys ja tietojen analyysi tulee toteuttaa omina aliohjelmiaan.

(Ali)ohjelman koon kasvaessa tulee siitä irrottaa sopivia osia aliohjelmiksi, jotta ohjelman koko pysyy kohtuullisena ja kokonaisuus selkiytyy.

Ohjelmoijan kannalta aliohjelman tulee helpottaa ohjelman ymmärrettävyyttä ja ylläpitoa, vaikka sen teko saattaa vaatia ylimääräistä työtä toteutusvaiheessa.

Paluuarvot

- Kaikki pää- ja aliohjelmat päättyvät return-käskyyn
- Mikäli mahdollista, kannattaa aliohjelmasta palauttaa hyödyllistä tietoa
    - Linkitetyn listan tyhjennys -aliohjelmasta kannattaa palauttaa NULL
    - Tiedoston luku -aliohjelmasta voi palauttaa luettujen rivien määrän
    - Aliohjelmasta voi palauttaa virhekoodin, jos tavoitteena on toipua virheistä

2.9. Nimeäminen

Ohjelmissa on erilaisia tunnuksia ja niiden nimeämiseen on erilaisia käytäntöjä. Kaikissa nimissä tulee olla johdonmukainen ja niissä tulee käyttää samaa logiikkaa

Vakiot

- Esikääntäjällä (#define) ja enum:lla määritellyt vakiot kirjoitetaan kaikki kirjaimet suurakkosilla, esim. ”#define LKM 10”
- Vakioita tulee käyttää mm. taulukoiden kokojen määrittelyyn

Muuttujat

- Muuttujien nimet kannattaa määritellä käyttäen unkarilaista notaatiota, jolloin muuttujan tietotyyppi näkyy sen nimestä. Tällä kurssilla keskitytään yleisimpiin perustietotyyppeihin eikä harvinaisempia tietotyyppejä huomioida
- Käytettävät tietotyypit ovat seuraavat
    - i – int, kokonaisluku
    - l – long, pitkä kokonaisluku
    - f – float, liukuluku
    - d – double, kaksoistarkkuuden liukuluku
    - c – char, merkki
    - a – array, taulukko ottamatta kantaa taulukon tietoalkioiden tyyppiin, joka voi näkyä muuttujan nimessä, esim. aNimi, aNumerot
    - p – pointer, osoitin ottamatta kantaa osoitettavaan tietotyyppiin, esim. pAlku. Lisäksi geneerinen liukuri-osoitin on tyypillisesti nimeltään ptr
- Muuttujien nimet kannattaa muodostaa systemaattisesti niiden kuvaavan tiedon perusteella, esim. iLukumaara, aNimiEtu, aNimiSuku, dPaino, dPainoNetto, dPainoMax

Tietueet

- Tietue tulee nimetä pienaakkosilla, esim. struct henkilo
- Tietueesta tulee määritellä uusi tietotyyppi, joka nimi on sama kuin tietueella, mutta kirjoitettuna suurakkosilla, esim. typedef struct henkilo HENKILO;

Tiedostot

- Pääohjelman lähdekooditiedoston ja suoritettavan tiedoston nimien tulee olla samat. Tällä kurssilla viikkotehtävät kannattaa nimetä luennon ja tehtävän perusteella, esim. L1T1.c, L7T2.c, tai esim. HTMinimi.c
- Aliohjelmia sisältävien tiedostojen otsikko- ja ohjelmatiedostojen tulee olla saman nimisiä siten, että vain tiedostojen tarkenteet eroavat, esim. tiedosto.c ja tiedosto.h
- Tentissä tiedostot tulee nimetä tenttipaperin ohjeiden mukaisesti

Aliohjelmat

- Aliohjelmien nimet tulee muodostaa tarvittaessa useista sanoista ja nimien tulee kertoa, mitä aliohjelmissa tapahtuu
- Yhdestä sanasta muodostuvat aliohjelmanimet tulee kirjoittaa kaikki kirjaimet pienellä ja jos sanoja on useita, seuraavien sanojen ensimmäiset kirjaimet kirjoitetaan suuraakkosilla, esim. tiedostoLue, tiedostoKirjoita, listaMuodosta, listaLisaa, listaTyhjenna, kysyNimi

Yksikirjaimiset ja mitäänsanomattomat tunnukset/nimet ovat kiellettyjä

- Kiellettyjä tunnuksia ovat mitään sanomattomat tunnukset, esim. kirjaimet a, b, c ja a1, a2, a3 jne.
- C-ohjelmien vakiintuneet muuttujat esim. i, j ja k ovat hyväksyttäviä
- Epävarmoissa tapauksissa kannattaa käyttää kuvaavia nimiä yllä olevan mukaisesti

2.10. Näkyvyys

Tunnusten näkyvyyteen liittyen tulee muistaa seuraavat asiat:

1. Globaalit muuttujat ovat kiellettyjä
2. Muuttujat määritellään lokaaleina ohjelmissa/koodilohkoissa
3. Vakiot, tietueet ja aliohjelmat määritellään globaaleina

2.11. goto-käskyn käyttö kielletty

Ohjelmissa ei saa käyttää goto-käskyä.

2.12. Rekursio

Rekursiota ei tule käyttää ellei tehtäväksiannossa ole niin erikseen kehotettu.

2.13. Sisennykset

Ohjelmarivit on sisennettävä loogisesti samalla tyylillä koko ohjelmassa.

3. Perusoperaatiot

3.1. Tietojen kysyminen käyttäjältä

Tiedot tulee kysyä käyttäjältä seuraavilla tavoilla:

1. Numerot luetaan scanf'lla
2. Merkit luetaan scanf'lla, puskuriin jäävä rivinvaihtomerkki tulee poistaa
3. Sanat luetaan scanf'lla

4. Lauseet luetaan fgets'llä, merkkijonoon sisältyvä rivinvaihtomerkki tulee poistaa

3.2. Tiedoston lukeminen

Tekstitiedoston luku tehdään omassa aliohjelmassaan

1. Aliohjelma saa parametrinä osoittimen luettavan tiedoston nimeen
2. Avaa tiedosto ja tarkista sen onnistuminen normaalilla virheenkäsittelyllä
3. Tiedoston lukemisen jälkeen tiedosto on suljettava samassa aliohjelmassa
4. Tiedoston lukemiseen käytetään tyypillisesti fgets-funktioita. Merkkijonojen pilkkominen alkioihin voidaan tehdä esim. strtok-funktiolla ja merkkijonoja voi muuttaa numeroiksi esim. atoi ja atof-funktioilla

3.3. Tiedoston kirjoittaminen

Tekstitiedoston kirjoittaminen tehdään omassa aliohjelmassaan

1. Aliohjelma saa parametrinä osoittimen kirjoitettavan tiedoston nimeen
2. Avaa tiedosto ja tarkista sen onnistuminen normaalilla virheenkäsittelyllä
3. Kirjoitettava tiedosto avataan kirjoitustilassa, joka hävittää aiemman saman nimisen tiedoston, jos sellainen on
4. Tiedoston kirjoittamisen jälkeen tiedosto on suljettava samassa aliohjelmassa
5. Tiedoston kirjoittamiseen käytetään tyypillisesti fprintf-funktioita

Mikäli samat tiedot kirjoitetaan tiedostoon ja tulostetaan näytölle, tulee varsinainen tulostaminen tehdä yhdessä aliohjelmassa. Tässä tapauksessa tulostuksen tekevä aliohjelma saa parametrina joko osoittimen tulostettavaan tiedostoon sen avaamisen jälkeen, tai stdout-standarditulostevirran osoittimen.

3.4. Linkitetyn listan luominen

Luotaessa linkitetty lista tiedoston riveillä olevasta datasta

1. Tiedosto luetaan omassa aliohjelmassa ja sen tiedoista muodostetaan lukemisen yhteydessä linkitetty lista. Aliohjelma saa parametreina luettavan tiedoston nimen merkkijonona ja osoittimen listan alkuun
2. Mikäli lista on jo olemassa, tyhjennä lista ja vapauta varattu muisti
3. Lue tiedostosta yksi rivi tietoa kerrallaan ja lisää tiedot listaan omassa aliohjelmassaan. Aliohjelma saa parametreina osoittimen listan ensimmäiseen alkioon ja lisättävän merkkijonon osoitteen
4. Yhdellä rivillä olevat tiedot laitetaan aina yhden tietueen tiedoksi siten, että CSV-tiedoston jokaisesta tietoalkiosta tulee oma jäsenmuuttuja
5. Linkitetyssä listassa jokainen tietue sisältää tieto-jäsenmuuttujien lisäksi seuraavaan listan alkioon osoittava osoittimen, jonka nimi on pSeuraava
6. Varaa tietueen tarvitsema muisti malloc'lla ja normaalilla virheenkäsittelyllä
7. Kaikki riveillä olevat tietoalkiot muutetaan niiden luonnollisiksi tietotyypeiksi. Näin ollen kokonaisluvut muutetaan kokonaislukutyypeiksi (int, long, …), desimaaliluvut liukuluvuiksi (float, double, …), nimet merkkijonoiksi, päivämäärät aika-tiedoksi (struct tm, time_t, …) jne.
8. Listaan lisäys -aliohjelmasta palautetaan osoitin listan ensimmäiseen alkioon tai NULL-osoitin virheen merkkinä
9. Kun kaikki tiedoston rivit on lisätty linkitettyyn listaan, palataan tiedoston luku -aliohjelmasta ja palautetaan listan ensimmäisen alkion osoite

Mikäli linkitetty lista luodaan käyttäjän antamista tiedoista, toimitaan edellä olevan ohjeen mukaisesti paitsi, että tietojen tiedostosta lukemisen sijaan ne kysytään käyttäjältä.

3.5. Varattujen resurssien vapauttaminen

C-ohjelmissa varattavat resurssit kuten muisti ja tiedostokahvat on vapautettava ennen ohjelman päättymistä

1. Dynaamisesti varattu muisti pitää vapauttaa free-käskyllä ja asettaa osoitin NULL:ksi
2. Tiedostokahvat tulee vapauttaa sulkemalla avatut tiedostot samassa aliohjelmassa missä ne avattiin

3.6. Tietojen analysointi

Tietojen analyysi tehdään tyypillisesti linkitetyssä listassa oleville tiedoille

1. Analyysi-aliohjelma saa parametrina osoitteen listan ensimmäiseen alkioon sekä osoitteen tulostietorakenteeseen. Tulostietorakenne vaihtelee tilanteen mukaan ollen tyypillisesti yksi tietue tai linkitetty lista
2. Mikäli tulostietorakenne on dynaaminen ja jo olemassa, vapauta se
3. Tee analyysi ja sijoita tulokset tulostietorakenteeseen sekä varaa tarvittaessa muistia normaalilla virheenkäsittelyllä
4. Etsittäessä datasta tiettyjä arvoja, esim. minimi tai maksimi, tulee etsintä aloittaa datasetin ensimmäisen alkion arvoilla
5. Mikäli datasetissä on useita etsittäviä arvoja, esim. minimi tai maksimi, käytetään datasetin ensimmäistä arvoa, jos datasettiin liittyy merkityksellinen aikaleima
6. Mikäli datasetistä on etsittävä esim. ensimmäiseen tai viimeiseen alkioon liittyvää tietoa kuten päivämäärä, on tieto etsittävä datasta
7. Palauta tulostietorakenteen osoite kutsuvaan ohjelmaan

3.7. Virheenkäsittely

Virheenkäsittely pitää toteuttaa aina tiedostonkäsittelyn yhteydessä tiedostoa avattaessa ja muistia varattaessa

1. Mikäli operaatio ei onnistu, kerrotaan käyttäjälle ongelmasta perror-käskyllä ja virheilmoituksella
2. Ohjelman suoritus lopetetaan exit(0)-käskyllä

Tyypillisimmät virheilmoitukset ovat seuraavat:

1. "Muistinvaraus epäonnistui, lopetetaan"
2. "Tiedoston avaaminen epäonnistui, lopetetaan"

perror’a käytettäessä virheilmoitukseen ei laiteta rivinvaihtoa, sillä sen perään tulee perror’n standardi virheilmoitus. Ilmoitettaessa virheestä printf-käskyllä, tulee virheilmoituksen päättyä rivinvaihtomerkkiin.

3.8. Ohjelman kääntäminen ja Makefile

Useiden tiedostojen kääntämiseen tulee käyttää make:a ja Makefile:ä.

1. Jokainen lähdekooditiedosto on käännettävä objektitiedostoksi omalla käskyllä. Objektitiedostojen nimien tulee olla samat kuin lähdekooditiedostojen
2. Suoritettava koodi tehdään omalla käskyllä käännetyistä objektitiedostoista
3. Käännettäessä on käytettävä seuraavia optiota: -Wall, -pedantic, -std=c99
4. Makefile:ssä on näyttävä tiedostojen riippuvuudet
5. Makefile:ssä tulee olla alkukommentti sisältäen vähintään tekijän ja päivämäärän
6. Harjoitustyön suoritettavan tiedoston tulee olla nimeltään HT