6. Pääteohjaus 4

Tässä ohjauksessa harjoitellaan osiinjaetun tiedoston kääntämistä ja itse valmiiksi kirjoitettujen aliohjelmakirjastojen käyttöä.

6.1 Projekti

1. Mene omaan hakemistoosi KOVALEVYLLÄ ja luo sen alle hakemisto nimeltä `KERHO` ja vaihda itsesi sinne.
2. Kopioi KERHO-ohjelma omaan hakemistoosi:

   ```
   C:\OMAT\OHJELMOI\VESA\KERHO>copy n:\kurssit\cpp\kerho\etsilaj.5\*.*[RET]
   ```

3. Käynnistä Borland-C (***BC[RET]***).
4. Aloita uusi projekti (***[Alt-P]*** ja Open Project).
5. Anna projektin nimeksi ***KERHO***.
6. Avaa tiedosto `kerhomai.cpp` ja laita projektiin ne tiedostot, jotka tiedoston `kerhomai.cpp` kommenteissa pyydetäään laittamaan projektiin (lisäämistä varten paina ***[Ins]***):

   ```
   tulkitse esim CPP\ALI\MJONOT.C =>  N:\KURSSIT\CPP\ALI\MJONOT.C
   ```

7. Katso, että kohdan Options|Directories kohdassa Include Directories lukee (älä hävitä rivin alkua!):

   ```
   ....INCLUDE;N:\KURSSIT\CPP\ALI
   ```

   Jollei lue, niin lisää viimeinen hakemisto.
8. Kokeile ajaa ohjelmaa (valmiina on kerho KELMIT). Toimiiko kuten suunniteltu?

6.2 Breakpoint

Otetaan lyhyt debuggerin kertausharjoitus. Debuggerilla ohjelmaa pystyttiin ajamaan mm. askel kerrallaan. Oikeassa ohjelmassa kukaan ei jaksa kahlata koko ohjelmaa tällä tavoin läpi. Pitää keskittyä vain olennaisuuksiin.

Tämän vuoksi debuggerissa on mahdollisuus laittaa keskeytyskohtia, eli paikkoja joihin tultaessa pysähdytään, muuten ohjelmaa ajetaan normaalisti. Keskeytyskohtia voidaan laittaa myös ehdollisesti hyvinkin monipuolisesti. Kaikkein hienoimmissa systeemeissä keskeytys saadaan jonkin tietyn muuttujan saadessa tietyn arvon (tai yleensä muuttuessa) missä tahansa kohti ohjelmaa.

1. Epäilet, että metodi `cNaytto::kysy_tiedot` ei toimi. Toimimattomuus ilmenee (ilmenisi esimerkiksi, koska nyt toimii) jotenkin henkilön lisäämisessä. Epäilet erityisesti aliohjelmaa kysy_kentta (`NAYTTO.CPP`).
2. Avaa tiedosto `NAYTTO.CPP` (esimerkiksi näyttämällä tiedoston nimeä projekti-ikkunassa tai painamalla ***`[F3]`***).
3. Etsi aliohjelma `kysy_kentta` (***`[Ctrl-Q]F kysy_kentta[RET]`***). Mikäli ei löydy 1. yrittämällä, niin ***`[Ctrl-L]`*** etsii uudelleen.
4. Siirry aliohjelmassa riville jolla lukee

   ```
   if ( paluu < OLETUS ) return TOIM_LOPETA;
   ```

   ja laita riville keskeytyskohta (breakpoint, joko ***`[Ctrl-F8]`*** tai ***`[Alt-D]`*** ja Toggle breakpoints).
5. Lisää watch-ikkunaan aliohjelman kannalta mielenkiintoisia arvoja:

   ```
   paluu
   toim
   apu
   jono
   viesti
   ```

6. Aja ohjelma ***`[Ctrl-F9]`*** siten, että yrität lisätä jotakin tietoa. Kun ohjelma pysähtyy, aja askel kerrallaan joko suorittaen tai ylittäen aliohjelmat. (Huom! Koska `mjonot.c` tiedostoa ei kannata debuta, pitää sinne kuuluvat kutsut ajaa ***`[F8]`***-nappulalla!). Seuraa muuttujien arvon käyttäytymistä. Kun olet “steppaillut” ulos epäilemästäsi aliohjelmasta paina taas ***`[Ctrl-F9]`***. Jatka tätä kunnes “Olet saanut vian selville”. Sitten voit poistaa keskeytyskohdan vastaavasti kuin sen asetitkin. Huom! Keskeytyskohtia voi olla useitakin.
7. Kokeile joitakin muita kohtia vastaavasti (mielellään sellaisia, joista itse olet kiinnostunut, esimerkiksi metodia `lue_tiedostosta` tiedostossa `kerho.cpp`).
8. Näin voit aina menetellä, mikäli ohjelmastasi löytyy toimimattomia (tai epäilyttäviä) kohtia jonkin pitkän rämeikön takaa. Muista sitten joskus lopuksi myös poistella keskeytyskohdat!

6.3 lue_jono_oletus

Kokeillaan `lue_jono_oletus` aliohjelman toimintaa.

1. Sulje kaikki ikkunat.
2. Aloita uusi projekti nimellä `lue`.
3. Lisää projekti-ikkunaan vielä kirjoittamaton tiedosto `lue.c`.
4. Lisää myös `n:\kurssit\cpp\ali\mjonot.c` (voit myös kopioida aliohjelmat omaan hakemistoosi ja laittaa vain `mjonot.c`).
5. Kirjoita seuraava pääohjelma tiedostoon `lue.c`:

   ```
   #include <stdio.h>
   #include "mjonot.h"
   int main(void)
   {
     char nimi[40]="Minä";
     int paluu;
     paluu = lue_jono_oletus("Anna nimesi",17,33,nimi,nimi,sizeof(nimi));
     if (paluu == OLETUS) printf("Et muuttanut nimeä!\n");
     printf("Nimesi oli %s\n",nimi);
   ```

```
  return 0;
}
```

6. Jos ohjelma ei käänny, niin tarkista että kohdan Options|Directories kohdassa Include Directories lukee:

   ```
   ....INCLUDE;N:\KURSSIT\CPP\ALI
   ```

   Jollei lue, niin lisää viimeinen hakemisto.
7. Kokeile käyttää ohjelmaa siten, että kirjoitat joskus uuden nimen ja joskus vastaat pelkän ***[RET]***.
8. Muuta ohjelmaa siten, että `nimi` muutetaan isoiksi kirjaimiksi ennen tulostamista ja kokeile.
9. Muuta ohjelmaa siten, että `nimi` muutetaan alku-isoiksi kirjaimiksi ennen tulostamista ja kokeile.
10. Muuta ohjelmaa siten, että kaikki turhat välilyönnit poistetaan ja kokeile.
11. Kokeile mitä merkitystä on parametrien 17 ja 33 muuttamisella.
12. Kokeile korvata ensimmäinen kutsussa oleva `nimi`-parametri vakiolla `"Kalle"`.
13. Poistu käyttöjärjestelmään ja kopioi hakemiston `n:\kurssit\cpp\ali` sisältö omaan hakemistoosi.
14. Käynnistä jälleen Borland-C ja luo nyt uusi projekti nimelle `lue2`.
15. Lisää projektiin tiedostot `lue.c` ja `mjonot.c` (ei tarvitse nyt polkua).
16. Käännä ohjelma. Siis aliohjelmakirjastot voi joko kopioida omaan hakemistoon tai käyttää tietystä hakemistosta. Levyaseman nimeä ei kuitenkaan mielellään saa laittaa polkuihin!

6.4 string-luokka

1. Tee uusi projekti `luecpp`, lisää siihen `lue.cpp`:

   ```
   #include <iostream.h>
   #include "mjonotpp.h"
   int main(void)
   {
     string nimi("Minä");
     int paluu;
     paluu = lue_jono_oletus("Anna nimesi",17,33,nimi);
     if (paluu == OLETUS) cout << "Et muuttanut nimeä!\n";
     cout << "Nimesi oli " << nimi << endl;
     return 0;
   }
   ```

2. Lisää vielä projektiin `mjonot.c` ja `mjonotpp.cpp` (sekä `string.cpp` jos käytät jotakin vanhempaa kääntäjää < BC 4.5?)
3. Muuten tee samat tehtävät kuin edellisessä kohdassa.

6.5 pvm

1. Mene käyttöjärjestelmään ja kopioi hakemiston `n:\kurssit\cpp\demot` sisältö omaan hakemistoosi.