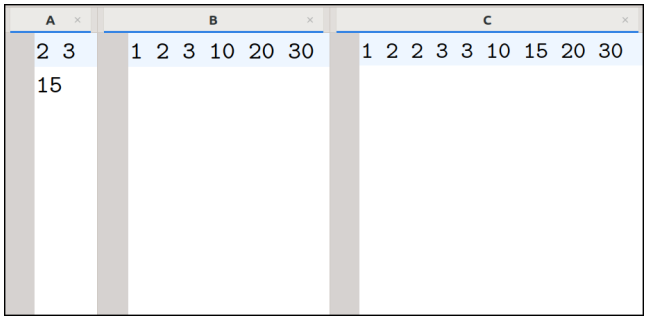
Mittuniversitetet
MID SWEDEN UNIVERSITY



~ Laboration – *Sortering* ~

Introduktion till programmering

— 2021-11-25 v1 —

Revisioner

2015-09-14 Martin Kjellqvist

2021-11-25 Jan-Erik Jonsson

Läs igenom hela instruktionen innan du börjar!

Introduktion

Sortering är en väldigt vanlig uppgift för datorprogram. Det finns många metoder för detta. Ofta introduceras metoder som *bubble sort* och *selection sort*, de är enkla att förstå men är inte särskilt effektiva.

I den här laborationen ska vi undersöka en operation som heter *merge* som används i metoden *merge sort*.

Dock inleds laborationen med att implementera en operation som undersöker om en fil med heltal är sorterad eller inte.

Syfte

Du kommer att skapa och anropa egna funktioner. I dessa ska du implementera algoritmer.

Du kommer även att dela upp källkoden i flera filer.

Läsanvisningar

- Du ska ha färdigställt den tidigare labben innan du fortsätter med denna.
- Du ska vara klar med samtliga moment fram till och med funktioner.
- Du ska även ha gått igenom hur man delar upp källkoden med hjälp av headerfiler.

Om main( )

I de tidigare laborationerna ha du kunnat skriva hela programmet i en enda funktion, *main*. Då programmen blir större blir det allt svårare att hålla reda på vad olika delar av programmet gör, därför delar man då upp programmen i fler funktioner. Dessutom kan man lägga funktionerna i egna filer. Det blir då enkelt att återanvända funktionerna i nya program.

Allt detta kommer du att göra i denna laboration.

Genomförande

Laborationen är uppdelad i 3 delar för att strukturera ditt arbete. Det är lämpligt att skapa en ny mapp för varje del.

- I del 1 skapar du en funktion i en .cpp-fil med en tillhörande headerfil (med filändelsen .h).
    - Du skapar även en separat .cpp-fil med en main-funktion där du testar din funktion.
        - Skapa egna textfiler *med få värden* så att du kan *kontrollera för hand* att funktionen fungerar.
    - Detta program kommer alltså att vara uppdelat i 3 filer.
        - Skapa ett byggskript som kompilerar filerna till ett körbart program.
- Del 2 är upplagd på samma sätt som del 1, men för en annan funktion.
- I del 3 sätter du ihop allt till ett nytt program där du demonstrerar dina funktioner. Har du gjort rätt så bör du kunna kopiera dina filer med funktionerna från del 1 och 2 hit.

*Se det bifogade exemplet i **headerfiler_exempel/** om att använda headerfiler m.m.*

Del 1

En textfil innehåller heltal.

- Skapa en sorteringsstest-funktion som avgör om heltalen i filen är sorterade i ordning eller inte. Se **Algoritm 1** på nästa sida. Försök att komma på en metod på egen hand innan du tittar på algoritmen.
- Komplettera ditt program med en main-funktion.
  - I main-funktionen anropar du din skapade sorteringsstest-funktion för att testa att den fungerar med olika filer, både sorterade och osorterade.
  - När din funktion fungerar som den ska, testa om de bifogade filerna **A**, **A1** och **B** är sorterade.

Tips

Börja med att skapa ett program med en main-funktion och din sorteringsstest-funktion. Funktion kan till att börja med vara tom eller bara innehålla någon enkel utskrift. I main-funktionen anropar du din sorteringsstest-funktion. Testkör och utveckla vidare.

Del 2

Två textfiler med sorterade heltal.

- Skapa en merge-funktion som lägger ihop talen i de två filerna till en ny fil, på ett sådant sätt att den nya filen innehåller samtliga tal i sorterad ordning. Se **Algoritm 2** på nästa sida. Försök att implementera den på egen hand innan du tittar på algoritmen.
  - **Observera:** *Talen ska inte behöva sorteras efter sammanslagningen.*
- Komplettera med en *main-funktion* där du testar din funktion med olika filer med sorterade heltal.
  - Kontrollera att den nya filen *är sorterad*, och *innehåller alla värden* från in-filerna.
  - När det fungerar, testa med två av filerna **A**, **A1** eller **B**. (Lämpligtvis de två som är sorterade.)

Del 3

Sätt ihop allt till ett program som demonstrerar att dina funktioner fungerar. Det är detta program du ska redovisa och lämna in.

Du ska använda filerna **A**, **A1** och **B** i ditt program.

Dina funktioner ska ligga i var sin källkodsfil (.cpp). Varje källkodsfil ska i sin tur ha var sin headerfil (.h). Det blir alltså 2 filer per funktion.

Till sist skapar du en cpp-fil med en main-funktion, den måste inte ha en headerfil.

Detta bör förslagsvis ingå i main:

- Testa om de ingående filerna är sorterade.
- Om de inte är det, skriv ett felmeddelande.
- Om de är sorterade, genomför en merge.
- Kolla om den sammanslagna filen är sorterad.

Allt detta görs förstås genom att anropa de funktioner du skapat tidigare.

Algoritmer

Algoritm 1: Avgör om en fil med värden är sorterad

```
indata: Ett filnamn
utdata: Sant eller falskt beroende på om filen är sorterad.

    a = readValue(A)
    medan notEndOf File(A)
        b = readValue(A)
        om a > b
            returnera false

        a = b

    returnera true
```

Algoritm 2: Merge av två filer med *sorterade värden*

```
indata: Filnamn på två infiler och en utfil.
utdata: Filen C med samtliga värden från A och B i sorterad ordning.

    a = readValue(A)
    b = readValue(B)

    medan notEndOf File(A) And notEndOf File(B)
        om a < b
            Skriv a till C
            a = readValue(A)
        annars
            Skriv b till C
            b = readValue(B)

    medan notEndOf File(A)
        Skriv a till C
        a = readValue(A)

    medan notEndOf File(B)
        Skriv b till C
        b = readValue(B)
```

Observera!

- Det är själva ihopslagningsalgoritmen (merge) som ska hämta värden ur infilerna på ett sådant sätt att elementen blir sorterade. *Ditt program ska i övrigt inte innehålla någon sortering*.
- Läs inte in all data först. Du ska aldrig behöva spara mer än ett fåtal värden åt gången i programmet.

Examination

Du redovisar din lösning muntligt för din labbhandledare. När du fått godkänt lämnar du in *källkoden* och *byggskriptet* i lärplattformens inlämningslåda för denna uppgift.