Mittuniversitetet
MID SWEDEN UNIVERSITY

DT019G Objektbaserad programmering i C++

Projekt:
Ett sorteringsverktyg

Martin Kjellqvist*

sort.tex 1055 2013-05-16 06:03:50Z martin

Innehåll

1 Introduktion

Att skriva ett program som är användbart är en mer tidskrävande processs än att skriva ett program som löser enn uppgift. I detta projekt kommer du att bekanta dig med ett bibliotek som underlättar för dig att skriva ett program som följer konventioner.

Detta projekt är utformat så att det ska ta runt 3 dagar att genomföra.

Man författar med fördel detta projekt på ett *nix eller Mac system, biblioteken som används är anpassade för dessa. På windowssystem måste kommandoradsargumenten tolkas för hand".

2 Syfte

Syftet är att fördjupa dina kunskaper inom och utveckla din vana för programmering. Du kommer att få fördjupade kunskaper om hur sorteringsmetoderna som presenterats under kursens gång kan tillämpas i riktiga scenarion.

3 Teori

Välj något av följande biliotek för att underlätta hanteringen av kommanoradsargumenten.

- `getopt` `http://www.gnu.org/software/libc/manual/html_node/Getopt.html`
- `argp` `http://www.gnu.org/software/libc/manual/html_node/Argp.html`

Getopt är klassikern, argp är något mer hjälpsam att implementera vissa speciella konstruktioner såsom `--help` och `--version`.

Biblioteken är C-biliotek. Dokumentation och exempel ges av respektive länk.

Om du skriver en egen sorteringsmetod eller använder `std::sort` är inte av vikt, men det kan vara värt att notera att det kan vara något enklare att göra sin egna sorteringsfunktion mer flexibel. För `std::sort` krävs ett komparatorobjekt som vi inte behandlat i kursen.

4 Uppgift

Projektet är uppdelat i olika betygssteg. För betyget D krävs att du genomför grunduppgiften perfekt utan anmärkningar enligt granskningsprotokollet. Vid maximalt två anmärkningar ges betyget E, vid fler anmärkningar betyget F.

För betygen C-A krävs de extrauppgifter som ges efter grunduppgiften, för ett betyg krävs att samtliga föregående extrauppgifter även är genomförda. (Även dessa ska vara utan anmärkningar, vid anmärkning sänks betyget fortfarande till E.)

4.1 Grunduppgift

Skapa ett program som är avsett att anropas från kommandoraden. Programmet ska sortera det indata det ges tillgängligt. Indatat ges som radbaserad text. Indatat ska betraktas som kolumnbaserat. Sortering sker på första kolumnen (i normalfallet, se nedan för switchar som kan ändra detta).

Exempel `input.txt`

```
15 Femton XV
12 Tolv   XII
122 Hundratjugotvå CXXII
```

Här ska sorteringen utföras med avseende på de numeriska värdena i första kolumnen.

Hur sorteringen sker beror på vilka argument som ges till programmet. Argument som måste implementeras är

- `--numeric | -n`. Behandla sorterinsdatat som numeriska värden. Även decimaltal ska hanteras.
- `--reverse | -r`. Skriv ut resultattet i omvänd ordning.
- `(--output | -o)file`. Skriv resultatet till filen `file` istället för standard ut.
- `--version | -v`. Skriver ut versionsnummer.
- `--help | -h`. Skriver ut tillgängliga switchar.

Indata ges via standard in. Man kan alltså sortera ovanstående `input.txt` genom följande kommandon:

```
$./sort <input.txt
12 Tolv   XII
122 Hundratjugotvå CXXII
15 Femton XV
$./sort -n <input.txt
12 Tolv   XII
15 Femton XV
122 Hundratjugotvå CXXII
```

Programmet ska inte under några omständigheter krasha, såvida inte systemresurserna räcker till. Dvs. minnet tar slut, filsystemet har slut på plats eller liknande.

4.2 Extrauppgift C

Ytterligare argument accepteras av programmet.

- `--keyonly | -k`. Ge bara det sorterade värdet per rad.
- `(--column | -c)N`. Sortera med avseende på kolumn `N`.
- `(--input | -i)file`. Använd filen `file` som indata.

Felhanteringen ska göras enligt följande:

- Om kolumn `N` inte finns tas raden inte med i utdatat.
- Om `file` inte finns skrivs ett felmeddelande till `cerr`.
- Om `--input | -i` inte angivits och ingen indata är pipat till programmet, ge ett felmeddelande till `cerr`. Det tidigare beteendet ska vara att läsa från `cin`. Se `http://linux.die.net/man/3/isatty`.

Om ett fel inträffat ska main returnera `EXIT_FAILURE` annars `EXIT_SUCCESS`.

Körexempel:

```
$./sort
No input.
$./sort -c 2 --keyonly <input.txt
Femton
Hundratjugotvå
Tolv
$./sort -c 2 --numeric <input.txt
$
```

God struktur innebär att argument till programmet kapslas in i en `struct` eller en `class`. Programmet ska vara uppdelat i lämpliga header- och implementationsfiler med relaterade klasser/funktioner. Sorteringsrelaterade göromål hanteras av en separat klass. Klasser och funktioner ska inte vara specialiserade i onödan.

Koden ska vara välkommenterad med beskrivande kommentarer som gör koden lättläst.

4.3 Extrauppgift A

Skapa en *man*sida för programmet. *Man*sidan ska innehålla en utförlig beskrivning av programmet, argumentalternativen samt körexempel. Se `http://www.schweikhardt.net/man_page_howto.html`.

Extra fokus på struktur och välskriven kod. Endast `main` ligger som fri funktion.

5 Examination

Ditt färdiga projekt ska först granskas av en annan student på kursen med hjälp av det granskningsprotokoll som återfinns på lärplattformen. Efter att du åtgärdat alla påpekanden som uppkommit under granskningen ska du tillsammans med granskaren redovisa ditt projekt och granskningen för en av kursens lärare vid något av de inbokade redovisningstillfällena. När du redovisat och fått godkänt laddar du upp din källkod med tillhörande byggskript[1] till inlämningslådan i lärplattformen.

[1] Notera att det är obligatoriskt att ha ett byggskript för programmet.