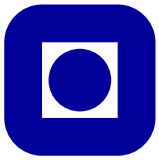

Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre- og objektorientert programmering Vår 2017

Øving 8

Frist: 2017-03-10

Mål for denne øvinga:

- Lese fra og skrive til filer
- Assosiative tabeller (map)
- Strømmer (streams)

Generelle krav:

- Bruk de eksakte navn og spesifikasjoner som er gitt i oppgava.
- Det er valgfritt om du vil bruke en IDE (Visual Studio, XCode), men koden må være enkel å lese, kompilere og kjøre.
- Skriv all nødvendig kode for å demonstrere programmet ditt.

Anbefalt lesestoff:

- Kapittel 12 og deler av 19, Absolute C++ (Walter Savitch)

1 Lese fra og skrive til fil (20%)

a) Skriv en funksjon som lar brukeren skrive inn ord (bruk `cin`), og lagrer hvert ord på en separat linje i en tekstfil.

Lagre hvert ord i en `string` før du skriver det til filen og la et spesielt ord, som «quit», avslutte programmet.

b) Skriv en funksjon som leser fra en tekstfil, og lager en ny fil (med et annet navn) med den samme teksten, men med linjenumre.

Sørg for at programmet ditt sjekker for vanlige feil som at filen ikke eksisterer.

Hint: For å lese en hel linje av gangen, bruk `getline()`-funksjonen. For å teste om du har kommet til slutten av filen, kan du bruke det læreboken omtaler som «the macho way to test for end of file». Dette finner du på side 558 i 5. utgave og side 536 i 6. utgave av læreboken.

Nyttig å vite: Filstier

Merk: Filstier er en kilde til forvirring når man leser fra og skriver til filer. Programmene du lager vil typisk lagre filene i prosjektets *arbeidsmappe* (working directory) hvis du kun bruker filnavnet (og ikke inkluderer stien). Den letteste måten å finne ut hvor filene du lager blir lagret, er å lage en fil og lete etter den i prosjektmappene.

Tekstfiler du bruker i Visual Studio-prosjekter kan legges til i prosjektet (eller opprettes) som elementer i mappen «Resource Files». Filer legges til på lignende måter i XCode. Hvis du ikke bruker en IDE, kan du lage filer i samme mappen som programmet ditt kjører fra.

2 Lese fra fil: tegnstatistikk (20%)

I denne deloppgava skal du lese fra en tekstfil og lage statistikk over bokstavene. For å teste programmet ditt trenger du en tekstfil som inneholder en passende mengde vanlig tekst (minst noen få linjer). Bruk hvilken som helst tekst du vil, eller lag en ny tekstfil med programmet du skrev i del 1.

a) Skriv en funksjon som leser en tekstfil og viser statistikk over bokstavene i filen på skjermen.

Programmet skal telle antall bokstaver i filen og hvor mange ganger hver bokstav forekommer. Du kan begrense antall forskjellige bokstaver du teller til normale engelske bokstaver (a–z) og du kan også forenkle oppgaven ved å konvertere alle bokstavene til små bokstaver med `tolower(char)`.

Hint: Du kan løse denne oppgaven ved å bruke en tabell (array) eller en `std::vector` med lengde lik antall forskjellige bokstaver. Hvis du for eksempel tar inn bokstaven `'e'`, kan du inkrementere elementet i tabellen på posisjon `'e'-'a'` (`characterCount['e'-'a']`). Pass på at du ikke går utenfor grensene til tabellen.

Eksempel:

```
Character statistics:
Total number of characters: 555
a: 38   b: 1    c: 45   d: 13
e: 46   f: 26   g: 4    h: 15
i: 64   j: 0    k: 0    l: 33
m: 14   n: 48   o: 40   p: 8
q: 0    r: 36   s: 29   t: 60
u: 24   v: 3    w: 3    x: 1
y: 0    z: 4
```

3 Map: Emnekatalog (30%)

Emner (også kjent som «fag») på NTNU har alltid en emnekode og et emnenavn, for eksempel `TDT4102` og `Prosedyre- og objekt-orientert programmering`. Vi ønsker i denne oppgaven å lage en klasse som kan inneholde en slags kobling mellom disse to verdiene slik at vi kan holde styr på alle de forskjellige emnene her på NTNU.

Dette ønsker vi å gjøre gjennom bruk av en template fra STL (Standard Template Library): `std::map`. Dette er en datastruktur i standardbiblioteket til C++, og implementerer det som på norsk kalles for en assosiativ tabell eller innholdsadressert tabell. Den lar deg lage relasjoner mellom ulike variabel-verdier, for eksempel *"hei"*→3 slik at om du slår opp på `myMap["hei"]` får du 3 i retur.

Du kan lese om `std::map` på side 910-917 i 5. utgave av Absolute C++ og på side 892-899 i 6. utgave av boken.

a) Deklarer klassen `CourseCatalog`. Denne klassen skal inneholde emnekoder og emnenavn i et `std::map<std::string,std::string>`. Du skal implementere de følgende funksjonene:

- `std::ostream& operator<<(std::ostream&, const CourseCatalog&)` – skriv ut alle emnekoder med tilhørende emnenavn.
- `CourseCatalog::addCourse` – legg til et kurs med emnekode og emnenavn.
- `CourseCatalog::removeCourse` – fjern et kurs gitt emnekoden.
- `CourseCatalog::getCourse` – finn emnenavnet til et kurs med en gitt emnekode.

Ingen av disse funksjonene skal bruke `cin`, men ta inn argumenter.

b) Implementer klassen.

c) Test klassen.
Lag en funksjon som legger til emnene `TDT4110 Informasjonsteknologi grunnkurs`, `TDT4102 Prosedyre- og objektorientert programmering` og `TMA4100 Matematikk 1`. Skriv så ut en oversikt over emnene.

d) Oppdatering av verdier i et `map`.
Emnet TDT4102 blir som oftest bare kalt for «C++» blant studentene. Legg til en linje som oppdaterer emnenavnet til TDT4102 (vha. `addCourse`)i testfunksjonen du lagde i forrige oppgave (uten å fjerne kurset først). Hva skjer?

Det finnes to metoder for å legge til verdier i et `std::map`: `operator[]` og medlemsfunksjonen `insert`. Eksperimenter med de ulike metodene i `addCourse`. Endrer oppførselen til funksjonen seg?

e) (Valgfritt) Lagre dataene.
Et problem med vår implementasjon er at alt vi lagrer vil bli slettet hver gang vi skrur av programmet og vi må legge det til igjen ved oppstart. Lag derfor en funksjon som laster inn informasjonen fra en tekstfil og en funksjon som lagrer informasjonen til en tekstfil.

Hint 1: Dersom du har overlagret `operator<<` for å skrive ut `CourseCatalog` kan du bruke denne til å skrive til et objekt av typen `ofstream` på samme måte som du ville gjort det med `cout`.

Hint 2: Måten du lagrer dataen i filen er opp til deg selv. En mulighet er å lagre det som `TDTXXXX Emnenavn` for så å lese inn hele linjen og bruke substr til å klippe informasjonen fra hverandre.

4 Lese fra fil: ordstatistikk (30%)

a) Skriv en funksjon som lager statistikk over ordene i en tekstfil.

Du kan selv velge hvilken statistikk du vil føre, men du må minst ha med:

- Antall ord i filen
- Antall ganger hvert ord forekommer
- Hvilket ord som er lengst
- Antall linjer i filen

Skriv resultatet av statistikken ut på skjermen. Du bør fjerne alle tegn og ikke-standard bokstaver (æ, ø, å, ö, osv.) fra ordene og konvertere dem til små bokstaver. For eksempel: Konverter «Don't» til «dont» før du lagrer/teller ordet.

Hint: Map kan være nyttig i denne oppgaven også

Hint: I C++ kan du velge om du vil hente en linje, et ord eller en bokstav fra filen av gangen. Dersom du velger en linje kan et enkelt ord i en gitt linje hentes ut på følgende måte:

```
#include <sstream>

stringstream ss(linje);
string ord;
ss >> ord;
```

Eksempel:

```
Text statistics:
Longest word: alphabet
Number of words: 49
Number of lines: 5
Average word length: 7
this: 1
is: 1
a: 1
very: 1
exercise: 1
....
....
```