Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre og Objektorientert programmering Vår 2013

Øving 7

Frist: 22.3.2013

Mål for denne øvingen:

- strømmer (streams)
- lese fra filer
- skrive til filer

Generelle krav:

- bruk de eksakte navn og spesifikasjoner som er gitt i oppgaven.
- 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, Absolute C++ (Walter Savitch)
- It's Learning-notater

1 Lese fra og skrive til fil (20 poeng)

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

Hint: Lagre hvert ord i en *string* før du skriver det til filen og la et spesielt ord, som "quit", avslutte programmet. Bruk *compare*-funksjonen til *string*-klassen for å teste om avslutningsordet er funnet (funksjonen vil returnere 0 hvis strengene er like).

b) Skriv et program som leser fra en tekstfil, og lager en ny fil (med et annet navn) med den samme teksten, men med linjenummerere. Implementer med den koden som trengs for å teste 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, på side 524, omtaler som "the macho way."

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 poeng)

I denne deloppgaven 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 et program 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 oppgave ved å konvertere alle bokstavene til små bokstaver med *tolower*(char).

Hint: Du kan løse denne oppgave ved å bruke en tabell (array) med lenge lik antall forskjellige bokstaver. For eksempel, hvis du 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 Lese fra fil: ordstatistikk (30 poeng)

a) Skriv et program som lager statistikk over ordene i en tekstfil. Du kan selv velge hvilken statistikk du vil føre, men du må minst telle antall ord i filen, antall ganger hvert ord forekommer og finne det lengste ordet. Skriv resultatet av statistikken ut på skjermen.

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
....
....
```

Hint: Du kan hente ut ett og ett ord fra filen ved å bruke »-operatoren og en string-variabel. Det kan være en god idé å lage en klasse eller *struct* med medlemsvariabler for ord og antall. Du bør fjerne alle tegn og ikke-standard bokstaver (a-z) fra ordene og konvertere dem til små bokstaver. For eksempel: Konverter "Don't" til "dont" før du lagrer ordet i vektoren. Husk å sjekke om det ordet fremdeles inneholder bokstaver før du lagrer det i vektoren.

4 Hangman (30 poeng)

Hangman er et spill som vanligvis spilles med papir og blyant. En spiller skal tenke på et ord og de andre skal prøve å gjette dette ordet ved å foreslå bokstaver. Ordet som skal gjettes blir representert av en rekke streker, som gir antall bokstaver i ordet. Hvis en foreslått bokstav ikke finnes i ordet, tegnes det et element i hangman-tegningen. Spillet er ferdig når enten tegningen, som forestiller en hengt man, er ferdigtegnet eller når alle bokstavene i ordet er funnet. I denne oppgaven skal du ikke tegne en tegning, men kun angi antall gjenstående forsøk med et heltall.

Begynn med å skrive et program som leser ord fra en tekstfil og lagrer hvert ord i en vektor (du skal lagre alle ordene i samme vektor). Tekstfilen skal inneholde et ord på hver linje.

Eksempel på innhold i filen hangman.txt:

```
swordfish
programming
blue
freebird
```

Utvid programmet ditt slik at spilleren kan gjette på bokstaver i et tilfeldig valgt ord fra vektoren. La spilleren få tilbakemelding om hvor mange bokstaver ordet har og hvilke bokstaver som gjenstår å gjette. Gjenta dette til spillet er ferdig. For å gjøre programmet enklere kan du bruke kun små bokstaver.

Når spilleren har gjettet alle bokstavene skal programmet gratulere brukeren. Hvis spilleren har brukt mer en et gitt antall forsøk (velg selv et fornuftig tall) skal spilleren få beskjed om at han at tapt. Hvordan du velger å implementere spillet er opp til deg, men du skal lage et spillbart program.

Hint: Du kan bruke medlemsfunksjonen *find(char c)* til string-klassen for å sjekke om en streng inneholder et tegn:

```
if (myString.find(currentLetter) != string::npos){
        // currentLetter finnes i strengen
}
```

string::npos representerer en indeks som ikke kan eksistere i en streng. Rent numerisk er det den største mulige verdien til en *unsigned integer*, eller bare -1. *find(char c)* returnerer denne verdien hvis c ikke finnes i strengen.

- *Valgfritt:* Ikke la spilleren gjette samme bokstav to ganger, og vis bokstavene han allerede har gjettet.
- *Valgfritt:* Skriv ut en liste (på skjermen) etter hvert forsøk med alle bokstavene spilleren har gjettet, men som ikke finnes i ordet.
- *Valgfritt:* Endre programmet ditt, slik at spilleren selv kan legge til ord (bruk kode fra del 1.) La også spilleren bestemme lengden på ordet som programmet plukker ut.

Eksempel:

```
Welcome to Hangman!
A random word has been chosen, try to guess it!
Current word: _______
Tries: 5
> e
Current word: __ee____
Tries: 5
> i
Current word: __ee_i__
> x
Current word: __ee_i__
Tries: 4
(... etc ...)
>b
Current word: freebird
YOU WIN!
```