Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre og Objektorientert programmering Vår 2014

Øving 7

Frist: DD-MM-YYYY

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 (15 poeng)

a) Skriv et program som lar brukeren skrive inn ord (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.

Hint: `compare`-funksjonen til `string`-klassen kan brukes 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. 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, 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 (15 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 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 (a-z) fra ordene og konvertere dem til små bokstaver. For eksempel: Konverter "Don't" til "dont" før du lagrer/teller ordet.

Hint: Map er en standard datastruktur i C++ biblioteket som kan være nyttig i denne oppgaven. Den lar deg lage relasjoner mellom ulike variabler som for eksempel: "hei" -> 3

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 linjen 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
....
....
```

4 Dekode en kodet melding (40 poeng)

Noen av de første måtene som ble brukt for å skjule / gjøre meldinger uleselig kalles "Substitution cipher" (`http://en.wikipedia.org/wiki/Substitution_cipher`). Dette er et veldig enkelt konsept der en bokstav er byttet med en annen for å gjøre meldingen vanskelig å lese for uvedkommende. I denne oppgaven skal dere implementere et program som lar brukeren prøve å knekke denne krypteringsmetoden.

Et par viktige konsepter kan gjøre den vedlagte koden enkelere å forstå:

- `Plaintext` - dette er teksten **før** den er kryptert / koded, denne er fullt lesbar for alle (også uvedkommende)
- `Ciphertext` - dette er teksten `plaintext` når den har blitt kryptert / gjort vanskelig å lese.
- `Cipher` - kan bety både krypteringsmåte og koding brukt for å kryptere. I denne oppgaven referer `Cipher` til en relasjon mellom to alphabeter der bokstavene er byttet ut.

Teksten som brukeren skal forsøke å dekode er et utdrag fra filen som ligger ved oppgaven. For å gjøre denne uleselig for brukeren må dere implementere et "Substitution cipher". Det er tre ulike set av relasjoner som dere må jobbe med:

1. `cipher` - Den relasjonen / sammenhengen mellom bokstaver som er brukt for å kode / gjøre teksten vanskelig å lese.
2. `decodeCipher` - Den relasjonen mellom bokstaver som er brukt for å gjøre teksts som er vanskelig å lese lett å lese. Denne er motsatt av `cipher`, det vil si at dersom 'a' -> 'b' i `cipher` så vil 'b' -> 'a' i `decodeCipher`.
3. `guessCipher` - Dette er brukerens gjettning på et relasjon, når denne er lik `decodeCipher` har brukeren alltid vunnet.

Følgende klasse skal implementeres:

```
class Substitution{
    public:
        Substitution(char filename[]);
        void play();
    private:
        const int maxLength = 40;

        std::map<char,char> cipher;
        std::map<char,char> decodeCipher;
        std::map<char,char> guessCipher;
        std::string plaintext;
        std::string ciphertext;
        std::string rawText;

        char encodeChar(char a, std::map<char,char>&);

        std::string encodeString(std::string plainText, std::map<char,char>&);

        int countSpaces(std::string input);
        int checkForErrors(std::string input);
        std::string cleanString(std::string input);

        void generateCipher();
        void selectRandomText();
        void askUser();
        int printErrors(std::string userPlaintext);
        void printUserInformation(std::string userPlaintext);
        std::string parseDecryptedText(std::string userPlaintext);
};
```

a) **Implementer funksjonen `cleanString(string)`.** Denne funksjonen fjerner alle symboler fra en tekststreng slik at det kun gjenstår bokstaver.

b) **Implementer konstruktoren `Substitution()`.** Denne leser en tekst fra en fil med navnet `"filename"` og lagrer denne teksten i sin helhet i tekst strengen `rawText`. Pass på at teksten ikke innholder annet en små bokstaver og mellomrom.

c) **Implementer funksjonen `selectRandomText()`.** Denne funksjonen velger en tilfeldig utdrag av teksten som tidligere ble lest fra fil og lagrer denne som `plaintext`. Tekst utdraget skal ikke starte eller slutte i midten av et ord og skal ikke være lengre enn `maxLength` (men kan være kortere).

d) **Implementer funksjonen `generateCipher()`.** Denne funksjonen setter opp `map` relasjonene `cipher`, `decodeCipher` og `guessCipher`. `cipher` skal genereres tilfeldig, tilsvarende for `guessCipher`. Se oppgaveteksten for en mer detaljert beskrivelse av hva de forskjellige skal være / gjøre.

e) **Implementer funksjonen `encodeChar(char, map)`.** Denne funksjonen tar inn et tegn, hvis tegnet er en bokstav skal den konverteres til en kodet bokstav ved hjelp av `map` variablen som følger med.

f) **Implementer funksjonen `encodeString(string, map)`.** Tilsvarende som `encodeChar`(char, map), men denne kan jobbe på en hel `string` av gangen.

g) **Implementer funksjonen `countSpaces(string)`.** Denne funksjonen skal telle antall mellomrom i en tekst streng.

h) **Implementer funksjonen `checkForErrors(string)`.** Denne funksjonen sjekker en tekststreng opp mot `plaintext` og returnerer antall feil.

i) **Implementer funksjonen `askUser()`.** Denne funksjonen spør brukeren etter en bokstav å endre i `guessCipher` og setter denne til en ny bokstav også gitt av brukeren.

j) **Implementer funksjonen `printErrors(string)`.** Denne funksjon tar inn brukerens forsøkt dekoded tekststreng og returnere antall feil i forhold til `plaintext`. Samtidig skal den skrive ut antall feil til skjerm og hvor mange prosent riktig dette forsøket var.

k) **Implementer funksjonen `parseDecryptedText(string)`.** Dette er en hjelpefunksjon som skal gjøre det lettere for brukeren å se hva som er riktig i forsøket på å dekode teksten. Den tar inn brukerens forsøkte dekodede tekst og returnere en tekststreng som er generert på følgende måte, for hver bokstav:

   1. Hvis bokstaven er riktig (dvs. den stemmer med bokstaven i `plaintext` på samme posisjon) skal bokstaven fra `plaintext` legges til tekststrengen.
   2. I alle andre tilfeller skal bokstaven fra `ciphertext` legges til tekststrengen.

l) **Implementer funksjonen `printUserInformation(string)`.** Denne funksjonen skal skrive ut cipherteksten og brukerens forsøk på å dekode teksten (som gitt av `parseDecryptedText`(string)).

m) Implementer funksjonen `play()`. Funksjonen play skal sette sammen alle øvrige funksjoner til et program der brukeren kan forsøke å dekode en tilfeldig tekst ved å endre på `guessCipher`. Følgende er et eksempel på utskrift fra et slikt program:

```
Plaintext:   in the spring dont forget to remove
Ciphertext:   rm gsv hkirmt wlmg ulitvg gl ivnlev
Decoded ciphertext:   rm gsv hkirmt wlmg ulitvg gl ivnlev

Number of errors: 29 (0% correct)
Please enter a character to replace:r
What should be replace it?i
Ciphertext:   rm gsv hkirmt wlmg ulitvg gl ivnlev
Decoded ciphertext:   Im gsv hkiImt wlmg ulitvg gl ivnlev

Number of errors: 27 (6.89655% correct)
Please enter a character to replace:m
What should be replace it?n
Ciphertext:   rm gsv hkirmt wlmg ulitvg gl ivnlev
Decoded ciphertext:   IN gsv hkiINt wlNg ulitvg gl ivnlev

Number of errors: 24 (17.2414% correct)
Please enter a character to replace:g
What should be replace it?t
Ciphertext:   rm gsv hkirmt wlmg ulitvg gl ivnlev
Decoded ciphertext:   IN Tsv hkiINt wlNT ulitvT Tl ivnlev

Number of errors: 20 (31.0345% correct)
Please enter a character to replace:s
What should be replace it?h
Ciphertext:   rm gsv hkirmt wlmg ulitvg gl ivnlev
Decoded ciphertext:   IN THv hkiINt wlNT ulitvT Tl ivnlev

Number of errors: 19 (34.4828% correct)
Please enter a character to replace:v
What should be replace it?e
Ciphertext:   rm gsv hkirmt wlmg ulitvg gl ivnlev
Decoded ciphertext:   IN THE hkiINt wlNT ulitET Tl iEnleE

Number of errors: 15 (48.2759% correct)
Please enter a character to replace:
```