Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre og Objektorientert programmering Vår 2014

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 (15 poeng)

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

```
ofstream outToFile(file, ios::out);

if (outToFile.fail()){
    cout << "There was an error opening the file." << endl;
    return;
}

string buffer;
while (cin >> buffer) {
    if (buffer.compare("quit") == 0){
        break;
    }else{
        outToFile << buffer << endl;
    }
}

outToFile.close();
```

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.

```
const unsigned int BUFFSIZE = 1024;
ifstream inFromFile(inFile, ios::in);
ofstream outToFile(outFile, ios::out);

if (inFromFile.fail() || outToFile.fail()) {
    cout << "There was an error opening the files." << endl;
    return;
}

char buffer[BUFFSIZE];
int counter = 1;
while (inFromFile.getline(buffer, BUFFSIZE)) {
    outToFile << counter++ << " " << buffer << endl;
}

outToFile.close();
inFromFile.close();
```

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.

```
int countAll(const int[], int size);

int main(){
    ifstream inFromFile("Part2.cpp", ios::in);
    if (inFromFile.fail()) {
        cout << "There was an error opening the file." << endl;
        return 0;
    }

    char c;
    int charCount[26] = {};
    while (!inFromFile.eof()) {
        c = tolower(inFromFile.get());
        if (c >= 'a' && c <= 'z') {
            charCount[c-'a']++;
        }
    }
    inFromFile.close();

    cout << endl << "Character statistics:" << endl;
    cout << "Total number of chararcters: ";
    cout << countAll(charCount, 26) << endl;

    for (int i = 0; i < 26; i++) {
        cout << (char)('a'+i) << ": " << charCount[i] << "\t";
        if (((i+1) % 4) == 0){
            cout << endl;
        }
    }
    cout << endl;

    return 0;

}

int countAll(const int t[], int size){
    int sum = 0;
    for (int i = 0; i < size; i ++){
        sum += t[i];
    }
    return sum;
}
```

3 Lese fra fil: ordstatistikk (30 poeng)

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

```
string cleanWord(string word){
    string temp = "";
    for (int i = 0; i < word.size(); i++){
        char x = tolower(word[i]);
        if ((x >= 'a') && (x <= 'z')){
            temp += x;
        }
    }
    return temp;
}
```

```
int main(){
    ifstream inFromFile("Part3.txt", ios::in);
    if (inFromFile.fail()){
        cout << "There was an error opening the file." << endl;
        return 0;
    }
    map<string, int> wordCount;

    int numberOfLines = 0;
    int numberOfWords = 0;
    int totalWordLength = 0;
    string line, word;
    string longestWord = "";
    while (getline(inFromFile,line)){
        stringstream ss(line);
        while (ss >> word){
            string cword = cleanWord(word);
            if (cword.size() > 0){
                if (wordCount.find(cword) != wordCount.end()){
                    wordCount[cword] += 1;
                }
                else{
                    wordCount[cword] = 1;
                }

                if (cword.size() > longestWord.size()){
                    longestWord = cword;
                }
                totalWordLength += cword.size();
                numberOfWords++;
            }
        }
        numberOfLines++;
    }

    cout << "Text statistics:" << endl;
    cout << "Longest word: " << longestWord << endl;
    cout << "Number of words: " << numberOfWords << endl;
    cout << "Number of lines: " << numberOfLines << endl;
    cout << "Average word length: ";
    cout << fixed << setprecision(2) << (double)totalWordLength / numberOfWords << endl;
    cout << "Average number of words per line: ";
    cout << fixed << setprecision(2) << (double)numberOfWords / numberOfLines << endl;
    cout << "Average number of characters per line: ";
    cout << fixed <<  setprecision(2) <<  (double)totalWordLength / numberOfLines << endl;

    for (map<string,int>::iterator it = wordCount.begin(); it != wordCount.end(); it++){
        cout << it->first << ": " << it->second << endl;
    }
    inFromFile.close();
    return 0;
}
```

4 Dekode en kodet melding (40 poeng)

a) Implementer funksjonen `cleanString(string)`.

```
string Substitution::cleanString(string input){
    string temp = "";
    for (int i = 0; i < input.size(); i++){
        char x = tolower(input[i]);
        if (((x >= 'a') && (x <= 'z')) || x == ' '){
            temp += x;
        }
    }
    return temp;
}
```

b) Implementer konstruktoren `Substitution()`.

```
Substitution::Substitution(char filename[]){
    ifstream inFromFile(filename, ios::in);
    if (inFromFile.fail()){
        cout << "There was an error opening the file." << endl;
    }
    rawText = "";
    string line;
    while (getline(inFromFile,line)){
        rawText += cleanString(line);
    }
    inFromFile.close();
    srand(time(0));
}
```

c) Implementer funksjonen `selectRandomText()`.

```
void Substitution::selectRandomText(){
    int textLength = rawText.length();
    int sindex = rand()%(textLength-maxLength);
    sindex = rawText.find(" ",sindex);
    int t = min(textLength-sindex,maxLength);
    int eindex = rawText.substr(sindex,t).find_last_of(" ");
    plaintext = rawText.substr(sindex,eindex);
}
```

d) Implementer funksjonen `generateCipher()`.

```
void Substitution::generateCipher(){
    char t[] = "abcdefghijklmnopqrstuvwxyz";
    char t2[] = "zyxwvutsrqponmlkjihgfedcba";
    for (int i = 0; i < 26; i++){
        cipher[t[i]] = t2[i];
        decodeCipher[t2[i]] = t[i];
        guessCipher[t[i]] = t[i];
    }
}
```

e) Implementer funksjonen `encodeChar(char, map)`.

```
char Substitution::encodeChar(char a, map<char,char>& cipher){
    a = tolower(a);
    if (cipher.find(a) != cipher.end())
        return cipher[a];
    else
        return a;
}
```

f) Implementer funksjonen `encodeString(string, map)`.

```
string Substitution::encodeString(string a, map<char,char>& cipher){
    string r = "";
    for (int i = 0; i < a.length();i++)
        r += encodeChar(a[i],cipher);
    return r;
}
```

g) Implementer funksjonen `countSpaces(string)`.

```
int Substitution::countSpaces(std::string input){
    int count = 0;
    for (int index = 0; index < input.length(); index++)
        if (input.at(index) == ' ')
            count++;
    return count;
}
```

h) Implementer funksjonen `checkForErrors(string)`.

```
int Substitution::checkForErrors(string decoded){
    int errors = 0;
    for (int index = 0; index < decoded.length(); index++)
        if (plaintext[index] != decoded[index])
            errors++;
    return errors;
}
```

i) Implementer funksjonen `askUser()`.

```
void Substitution::askUser(){
    cout << "Please enter a character to replace:";

    char c = 0;
    while (c < 'a' || c > 'z')
        cin >> c;

    cout << "What should be replace it?";

    char d = 0;
    while (d < 'a' || d > 'z')
        cin >> d;

    char t = guessCipher[c];
    guessCipher[c] = d;
}
```

j) Implementer funksjonen `printErrors(string)`.

```
int Substitution::printErrors(string userPlaintext){
    int spaces = countSpaces(userPlaintext);
    int errors = checkForErrors(userPlaintext);
    int textLength = userPlaintext.length() - spaces;
    cout << "Number of errors: " << errors;
    cout << " (" << 100.0*(1.0 - (double)(errors)/textLength);
    cout << "% correct)" << endl;
    return errors;
}
```

k) Implementer funksjonen `parseDecryptedText(string)`.

```
string Substitution::parseDecryptedText(string userplaintext){
    string output = "";
    for (int index = 0; index < userplaintext.size(); index++){
        if (userplaintext[index] == plaintext[index])
            output += toupper(userplaintext[index]);
        else
            output += ciphertext[index];//"_";//userplaintext[index];
    }
    return output;
}
```

l) Implementer funksjonen `printUserInformation(string)`.

```
void Substitution::printUserInformation(string userplaintext){
    cout << "Ciphertext: \t\t";
    cout << ciphertext << endl;
    cout << "Decoded ciphertext: \t";
    cout << parseDecryptedText(userplaintext);
    cout << endl << endl;
}
```

m) Implementer funksjonen `play()`.

```
void Substitution::play(){
    selectRandomText();
    generateCipher();
    ciphertext = encodeString(plaintext,cipher);
    cout << "Plaintext: \t\t" << plaintext << endl;

    string userPlaintext  = encodeString(ciphertext,guessCipher);
    printUserInformation(userPlaintext);
    int errors = printErrors(userPlaintext);

    while(errors > 0){
        askUser();
        userPlaintext = encodeString(ciphertext, guessCipher);
        printUserInformation(userPlaintext);
        errors = printErrors(userPlaintext);
    }
    cout << "You broke the code! Congratulations!" << endl;
}
```