5 Isolatın und Stringverarbeitung

5.1 Übungsaufgabe

Verwenden Sie den folgenden Text: <http://www.zeno.org/Literatur/M/Dostoevskij,+F%C3%ABdor+Michajlovi%C4%8D/Romane/Der+Idiot>

Schreiben Sie ein C++ Programm, das statistische Angaben eines Textfiles (ISOLATIN) ermittelt. Intern soll das Programm mit Methoden der Klasse `string` arbeiten.

Das Programm hat folgende Aufgaben:

- Das Programm soll nach dem Namen des Textfiles fragen
- Das Programm soll die Datei zum Lesen öffnen
- Es soll die Anzahl der Zeilen ermittelt werden
- Es soll die Anzahl der Wörter ermittelt werden
- Es soll die Anzahl der Buchstaben ermittelt werden
- Es sollen bei allen Wörtern der erste und der letzte Buchstabe vertauscht werden. Die Wörter sollen einzeln ausgegeben werden.
- Es soll der Buchstabe „a“ aus allen Wörtern gelöscht werden. Alle Wörter, die länger als 5 Buchstaben sind, sollen auf der Konsole ausgegeben werden.

Tipps:

- Zum Lesen der Datei soll die I/O Methode `getline` verwendet werden.
- Zur Bestimmung der Wortgrenzen kann mit Hilfe der Methode `find()` der nächste Separator gesucht werden und so der Index von Anfang und Ende eines Wortes bestimmt werden.
- Wenn Sie den Anfangs- und Endindex eines Wortes gefunden haben, kopieren Sie das Wort in eine temporäre Stringvariable, die Sie dann modifizieren und ausgeben können.

5.2 Wiederholung und zusätzliche Informationen

5.2.1 Funktionen

Funktionen bestehen aus einem Funktionskopf und einem Funktionsrumpf. Im Funktionskopf stehen Funktionstyp, Funktionsname und falls vorhanden Funktionsargumente. Als Funktionstypen können alle Standard-Datentypen und Klassen verwendet werden. Eine Funktion, die nichts zurückgibt, hat den Funktionstyp `void`.

Um eine neue Funktion in ein C++ Programm einzufügen muss die Funktion an 3 Stellen im Programm eingefügt werden.

- Die Funktion muss über der Mainmethode deklariert werden
- Die Funktion muss in der Mainmethode aufgerufen werden
- Die Funktion muss implementiert werden

```
#include <iostream>

//Funktionen werden deklariert
int addiere(int zahl1, int zahl2);
void druckeSummeAus(int summe);

int main() {
    int zahl1 = 3;
    int zahl2 = 5;
    int summe = 0;

    //in der Mainmethode werden die Funktionen aufgerufen
    summe = addiere(zahl1, zahl2);
    druckeSummeAus(summe);
    return (0);
}

//Funktionen werden implementiert - hier Rückgabotyp int
int addiere(int zahl1, int zahl2) {
    return zahl1 + zahl2;
}

//Funktionen werden implementiert - hier Rückgabotyp void = kein Rückgabotyp
void druckeSummeAus(int summe) {
    std::cout << "Deine Summe lautet: " << summe << std::endl;
}
```

5.2.2 Kodierungseinstellungen

Damit das C++ Programm die korrekte Kodierung verwendet, müssen die Locale Einstellungen gesetzt werden.

- Präprozessordirektive `#include <locale>`
- Verwendung der Locale des Systems: `setlocale(LC_ALL, "");`
- Einstellung einer eigenen Locale: `std::locale myLocale("de_DE.utf-8");`
- Einfärben einer Datei auf eine bestimmte Locale: `text.imbue(myLocale);`

5.3 Lösungsvorschlag

```

/*Autor: Nicola Greth
 * Uebung 4
 * Programm: Erstellt Angaben ueber einen Text
 */
#include <locale>
#include <iostream>
#include <fstream>
#include <string>

//Hilfsfunktionen werden deklariert
std::wstring deleteBeginningSpaces(std::wstring line);
std::wstring deletePunct(std::wstring line);

int main() {
    //Zaehlvariablen werden deklariert und initialisiert
    int countLines = 0;
    int countWords = 0;
    int countLetters = 0;
    std::wstring line;

    //Achtung filename muss normaler String sein
    std::string filename;

    //Locale Einstellungen werden gesetzt
    setlocale(LC_ALL, "");
    std::locale utf8locale("de_DE.UTF-8");

    //Dateiname wird eingelesen und Datei wird geoeffnet
    std::wcout << L"Bitte geben Sie den Dateinamen ein:>>" << std::endl;

    //da filename normaler String ist muss cin verwendet werden
    getline(std::cin, filename);
    std::wifstream input(filename.c_str());

    //Datei wird auf die passende Kodierung eingefaebrt
    input.imbue(utf8locale);

    //falls die Datei nicht vorhanden ist
    if (!input){
        std::wcout << L"Das Inputfile konnte nicht geoeffnet werden."

```

```
<< std::endl;
return (-1);
}

//solange es noch Zeilen in der Datei gibt
while(getline(input, line)) {
    countLines++;

    //Punktuationszeichen werden entfernt, muss nur einmal pro Zeile
    //aufgerufen werden (damit Buchstaben keine Punktuationszeichen zaehlt)
    line = deletePunct(line);

    //die Zeile wird von Spaces am Beginn bereinigt
    line = deleteBeginningSpaces(line);
    int sizeOfLine = line.size();

    //solange die Zeile nicht leer ist
    while (sizeOfLine > 0) {

        //das naechste Space wird gefunden
        int nextSeparator = line.find(L" ");
        std::wstring original;

        //wenn es ein Space gibt
        if (nextSeparator != -1){

            //dann ist das Wort der String von 0 bis zum Space
            original = line.substr(0, nextSeparator);

            //das Wort wird aus der Zeile geloescht
            line.erase(0, nextSeparator+1);

            //wenn es kein Space gibt
        } else {

            //dann gibt es nur noch ein Wort (der Rest der Zeile)
            original = line;
            line.erase();
        }

        //die Zeile wird fuer den naechsten Durchlauf bereinigt
        line = deleteBeginningSpaces(line);
        sizeOfLine = line.size();

        //Anzahl der Woerter wird erhoeht
        countWords = countWords + 1;

        //Anzahl der Buchstaben ist die Laenge des Wortes
        countLetters = countLetters + original.size();
        std::wstring firstLetter, lastLetter, changedWord;
```

```

//Erster und letzter Buchstabe werden gespeichert
firstLetter = original.at(0);
lastLetter = original.at(original.size()-1);
int sizeOfWord = original.size();

//Erster und letzter Buchstabe werden vertauscht
changedWord = original;
changedWord.replace(0,1, lastLetter);
changedWord.replace(sizeOfWord-1,original.npos, firstLetter);
std::wstring wordWithoutA = changedWord;

//a wird solange gesucht bis kein a mehr im Wort vorhanden ist
int positionOfA = wordWithoutA.find(L"a");

while (positionOfA != -1){
    //wenn es ein a gibt wird es geloescht
    wordWithoutA.erase(positionOfA,1);
    positionOfA = wordWithoutA.find(L"a");
}

//Woerter werden ausgegeben
sizeOfWord = wordWithoutA.size();

if (sizeOfWord > 5 and wordWithoutA != changedWord){
    std::wcout << L"Originales Wort: " << original << std::endl;
    std::wcout << L"Wort mit vertauschten Buchstaben: "
    << changedWord << std::endl;
    std::wcout << L"Wort ohne den Buchstaben a: "
    << wordWithoutA << std::endl;
    std::wcout << std::endl;
}
}
}
input.close();

//Ergebnisse werden ausgegeben
std::wcout << L"Im Text gibt es " << std::endl;
std::wcout << countLines << L" Zeilen, " << std::endl;
std::wcout << countWords << L" Wörter und " << std::endl;
std::wcout << countLetters << L" Buchstaben." << std::endl;
return 0;
}

//Hilfsfunktion, die Spaces am Beginn der Zeile loescht
std::wstring deleteBeginningSpaces(std::wstring line){
    int sizeOfLine = line.size();

    //solange die Zeile nicht leer ist und es ein Space an
    //der ersten Position gibt
    while(sizeOfLine!=0 && iswspace(line.at(0))){

```

```
        //dann wird das Space geloescht
        line.erase(0,1);
        sizeOfLine = line.size();
    }
    return line;
}

//Hilfsfunktion, die die Punctuationzeichen der Zeile durch Spaces ersetzt
std::wstring deletePunct(std::wstring line){
    int sizeOfLine = line.size();

    //Alle Punctuationzeichen werden durch Spaces ersetzt
    for (int i = 0; i < sizeOfLine; i++) {
        if (ispunct(line.at(i))) {
            line.at(i) = ' ';
        }
    }
    return line;
}
```