7 Funktionen

7.1 Übungsaufgabe

Schreiben Sie ein C++ Programm, das eine Datei `idiotISO.txt`, zu finden unter: http://www.cis.uni-muenchen.de/kurse/max/C++/texte/idiotISO.txt, öffnet und zeilenweise einliest. Implementieren Sie die folgenden Funktionen (Aufgaben 1 bis 6), die pro Zeile aufgerufen werden sollen.

7.1.1 Aufgabe 1

Schreiben Sie eine Funktion `int count_spaces(...)`, die berechnet, wieviele Spaces in der jeweiligen Zeile zu finden sind. Geben Sie den Wert aus.

7.1.2 Aufgabe2

Schreiben Sie eine Funktion `turn_chars(line)`, die die Buchstaben in der Zeile umdreht. Nach dem Aufruf der Zeile soll die verdrehte Zeile im Hauptprogramm ausgegeben werden.

```
std::wstring line = L"Hallo Du";
turn_chars(line);  //line umdrehen
std::wcout << "turned line=" << line << std::endl;
```

7.1.3 Aufgabe 3

Schreiben Sie eine Funktion `std::wstring first_word = get_first_word(line)`, die das erste Wort einer Zeile als Rückgabewert liefert und gleichzeitig aus der Zeile entfernt.

```
std::wstring line = L"Hallo Du Gauner!";
std::wstring fword;

fword = first_wordl(line);
std::wcout << "first Word=" << fword << ", rest of line=" << line << std::endl;

// Ergibt: first Word=Hallo, rest of line= Du Gauner!
```

7.1.4 Aufgabe 4

Schreiben Sie eine Funktion `nter_char(line,n)`, die den i*n-ten(i=1,...) Buchstaben einer Zeile auf dem Terminal ausgibt.

```
std::wstring line = L"das ist die Zeile"
//                   12345678901234567
nter_char(line,2)
ergibt: a s i el
```

7.1.5 Aufgabe 5

Schreiben Sie eien Funktion `del_nter_char(line,n)`, die den i*n-ten (i=1,...) Buchstaben einer Zeile entfernt und daraus einen Wstring baut. Der Wstring soll zurückgegeben werden und im Hauptprogramm ausgegeben werden.

```
std::wstring line = L"das ist die Zeile"
//                  12345678901234567
std::wstring ret_str;

ret_str = del_nter_char(line,3)

std::wcout << "Cutted String = " << ret_str << std::endl:

// ergibt: Cutted String = da it ieZele
```

7.1.6 Aufgabe 6

Kreieren Sie in Ihrem C++ Programm eine UTF8-Datei `all_words.txt`, in der alle Wörter der Datei `idiotISO.txt` zeilenweise gespeichert werden sollen. Zum Kopieren der Wörter in die Datei schreiben Sie eine C++ Funktion `split_line(std::wstring line)`, die eine Zeile als Argument erhält und jedes Wort in die Ausgabedatei schreibt.

7.1.7 Aufgabe 7

Erzeugen Sie aus der Datei `all_words.txt` mit UNIX-Befehlen eine sortierte Frequenzliste. Was sind die häufigsten 5 Wörter?

7.2 Wiederholung und zusätzliche Informationen

7.2.1 Call by Value und Call by Reference

Argumente können einer Funktion auf verschiedenen Wegen übergeben werden:

- Call by Value

- Call by Reference

Bei Call by Value werden die Argumente kopiert, die Funktion arbeitet mit den kopierten Werten. Änderungen an den Daten während der Funktion haben keinen Einfluss auf die originalen Argumente, mit denen die Funktion aufgerufen wurde.

Bei Call by Reference werden die Argumente als Referenz übergeben, die Funktion arbeitet also mit den originalen Argumenten. Änderungen an den Argumenten während der Funktion sind damit Änderungen an den originalen Argumenten, mit denen die Funktion aufgerufen wird.

Das folgende Codebeispiel soll dieses Prinzip verdeutlichen. Nach dem `doubleByValue`-Aufruf, ist `meineZahl` immer noch 5. Nach dem `doubleByReference`-Aufruf, ist `meineZahl` jedoch auch außerhalb der Funktion 10. Die Funktion hat also auch `meineZahl` verändert.

```
#include <iostream>

void doubleByValue(int zahl);
void doubleByReference(int &zahl);

int main() {
   int meineZahl = 5;
   doubleByValue(meineZahl);
   std::cout<<"meine Zahl: " << meineZahl<<std::endl;

   doubleByReference(meineZahl);
   std::cout<<"meine Zahl: " << meineZahl<<std::endl;
}

//Aufruf mit call by value: der Wert wird übergeben
void doubleByValue(int zahl) {
   zahl = zahl * 2 ;
   std::cout<<"Verdoppelte Zahl: " << zahl << std::endl;
}

//Aufruf mit call by reference: der Zeiger auf die Variable wird übergeben
void doubleByReference(int &zahl) {
   zahl = zahl * 2 ;
   std::cout<<"Verdoppelte Zahl: " << zahl << std::endl;
}
```

7.3 Lösungsvorschlag

7.3.1 Aufgaben 1 bis 6

```
/*Autor: Nicola Greth
 * Uebung 6
```

```
 * Programm: verschiedene Funktionen
 */
#include <iostream>
#include <locale>
#include <fstream>
#include <sstream>
using namespace std;

//Funktionen werden deklariert
int count_spaces(wstring line);
void turn_chars(wstring &line);
wstring get_first_word(wstring &line);
void nter_char(wstring line, int n);
wstring del_nter_char(wstring line, int n);
void split_line(wstring line);

int main() {
   setlocale(LC_ALL, "de_DE.utf-8");
   wstring line;
   wifstream datei("idiotISO.txt");
   datei.imbue(locale("de_DE.utf-8"));

   while (getline(datei, line)) {
      wcout << L"Original : " << line << endl;
      wcout << endl;

      //Funktionen werden aufgerufen
      //count_spaces wird aufgerufen
      int spaces = count_spaces(line);
      wcout << L"In dieser Zeile gibt es: " << spaces << " Spaces."
      << endl;
      wcout << endl;

      //turn_chars wird aufgerufen
      wstring lineCopy = line;
      turn_chars(lineCopy);
      wcout << L"turned line=" << lineCopy << endl;
      wcout << endl;

      //get_first_word wird aufgerufen
      lineCopy = line;
      wstring firstWord = get_first_word(lineCopy);
      wcout << L"first Word=" << firstWord << ", rest of line="
      << lineCopy << endl;
      wcout << endl;

      //nter_char wird aufgerufen
      lineCopy = line;
      nter_char(line, 2);
      wcout << endl;
```

```
        //del_nter_char wird aufgerufen
        lineCopy = line;
        wstring ret_str = del_nter_char(line, 3);
        wcout << "Cutted String = " << ret_str << endl;
        wcout << endl;

        //split_lines wird aufgerufen
        lineCopy = line;
        split_line(lineCopy);
    }
    datei.close();
    return 0;
}

//Funktionen werden implementiert
//Funktion count_spaces: gibt zurück wieviele Spaces in der Zeile sind
int count_spaces(wstring line) {
    int count = 0;

    for (int i = 0; i < line.size(); i++) {
        if (iswspace(line.at(i))) {
            count = count + 1;
        }
    }
    return count;
}

//Funktion turn_chars: Aufruf by reference, dreht die Chars der Zeile um
void turn_chars(wstring &line) {
    wstring lineReverse;

    for (int i = line.size() - 1; i >= 0; i--) {
        lineReverse += line.at(i);
    }

    line.erase();
    line = lineReverse;
}

//Funktion get_first_word: Gibt erstes Wort der Zeile zurück
//und löscht es aus der Zeile per Referenz
wstring get_first_word(wstring &line) {
    wstring firstWord;
    wstringstream stream;
    stream << line;
    stream >> firstWord;
    line.erase(0, firstWord.size());
    return firstWord;
}

//Funktion nter_char: gibt jeweils n_ten Char aus
```

```
void nter_char(wstring line, int n) {
   wcout << "nter char=";

   for (int i = n - 1; i < line.size(); i = i + n) {
      wcout << line.at(i);
   }
   wcout << endl;
}

//Funktion del_nter_char: löscht jeweils n_ten Char
wstring del_nter_char(wstring line, int n) {
   for (int i = n - 1; i < line.size(); i = i + n) {
      line.erase(i, 1);
   }
   return line;
}

//Funktion split_line: schreibt Wörter zeilenweise in eine Datei
void split_line(wstring line) {
   wofstream output("all_words.txt", ios::app);
   wstringstream stream;
   stream << line;
   wstring word;

   while (stream >> word) {
      output << word << endl;
   }
   output.close();
}
```

7.3.2 Aufgabe 7

```
sort -f < all_words.txt| uniq -ic | sort -nr | less
```

- sort -f: ignoriert Groß/ Kleinschreibung für die Sortierung
- uniq -ic: zählt Tokenvorkommen von groß- und kleingeschriebenen Wörtern zusammen
- sort -nr: sortiert rückwärts numerisch