8 Internationalisierung

8.1 Übungsaufgabe

Verwenden Sie einen UTF8-Text, wstrings und Internationalisierung. Schreiben Sie ein C++ Programm, das eine Datei öffnet, das richtige Encoding setzt und Wörter aus dieser Datei liest.

Lesen Sie die Datei zeilenweise und extrahieren Sie der Reihe nach jedes Wort, das Sie mit Hilfe einer Funktion testen: `bool palindrom (wstring word)` gibt `true` zurück, falls `word` ein Palindrom ist.
Die Funktion `palindrom` sollte eine Funktion `reverse (wstring &word)` verwenden, die Sie selbst programmieren sollen und die die Buchstabenfolge eines Wortes umdreht.

```
bool palindrom(wstring word) {
  wstring r_word=word;
    reverse(r_word);
    if (r_word==word) return true;
}
```

Schreiben Sie eine Funktion `bool splice (wstring w1, wstring w2, wstring &erg)`, die `true` zurückliefert, wenn das Wort w2 in Wort w1 vorkommt und im Wort erg den Teil des Wortes speichert, der nicht in Wort w1 vorkommt. z.B. (bestehen, eh, XXX) –> XXX=besten

Schreiben Sie eine Funktion `split`, die eine Zeile von wstring in die einzelnen Wörter zerlegt und in einem Array von wstrings speichert. Der Rückgabewert der Funktion ist die Anzahl der gespeicherten Wörter.

Schreiben Sie eine Funktion `first`, die als Argumente ein Array von wstrings, die Anzahl der Wörter im Array und als drittes Argument das erste Wort des Arrays von wstrings bekommt. Falls das eingegebene Array leer war, wird als return Wert `false` zurückgeliefert, ansonsten `true`.

Schreiben Sie eine Funktion `bool last (wstring woerter[], int &anzahl_der_woerter)`, die aus einem Array von wstrings das erste Wort entfernt. Wenn das Array, mit dem die Funktion aufgerufen wird, leer ist, dann soll der return Wert `false` zurückgeliefert werden, ansonsten soll `true` zurückgegeben werden.
Im Parameter `anzahl_der_woerter` ist gespeichert, wie viele Elemente im Array enthalten

sind. Zur Entfernung des letzten Elements des Arrays reicht es, in das letzte Element einen String mit einem Space zu schreiben und die Anzahl der Wörter um eins zu vermindern.

Schreiben Sie eine rekursive Funktion `bool member (wstring w, ARRAY wstring w_arr)`, die `true` zurückgibt, wenn das Wort w in der Liste von Wörtern w_arr vorkommt. Verwenden Sie dazu die Funktionen `first` und `last` der letzten beiden Aufgaben.

Tipp:
Die Lösung geht rekursiv vor, indem jeweils das vorderste Element der Liste mit dem gesuchten Term verglichen wird. Abbruchbedingung: das vorderste Element ist mit dem gesuchten Term unifizierbar. Rekursiver Fall: Suche im Rest der Liste (d.h. ohne das vorderste Element) weiter.

8.2 Wiederholung und zusätzliche Informationen

8.2.1 Switch-Case

Eine Alternative zu if-Abfragen, falls es nur eine bestimmte Anzahl an möglichen Werten gibt, ist das Switch-Case-Statement. Hierbei können verschiedene Fälle der angegebenen Variable abgeprüft werden und entsprechende Statements je nach Fall ausgeführt werden. Ein Fall wird durch Angabe von `break;` beendet. Außerdem ist es sinnvoll einen `default`-Wert für nicht vorgesehene Eingaben abzufragen.

```
#include <iostream>
#include <locale>

int main() {
   int monate;
   std::wstring monateString;
   std::setlocale(LC_ALL, "de_DE.UTF-8");
   std::wcout << L"Zu welcher Zahl möchten Sie den Monatsnamen erhalten?" <<
   std::endl;
   std::wcin >> monate;

   switch (monate) {
      case 1:
         monateString = L"Januar";
         break;
      case 2:
         monateString = L"Februar";
         break;
      case 3:
         monateString = L"März";
         break;
      case 4:
         monateString = L"April";
```

```
            break;
        case 5:
            monateString = L"Mai";
            break;
        case 6:
            monateString = L"Juni";
            break;
        case 7:
            monateString = L"Juli";
            break;
        case 8:
            monateString = L"August";
            break;
        case 9:
            monateString = L"September";
            break;
        case 10:
            monateString = L"Oktober";
            break;
        case 11:
            monateString = L"November";
            break;
        case 12:
            monateString = L"Dezember";
            break;
        default:
            monateString = L"keine gültige Monatsangabe";
            break;
    }
    std::wcout << L"Ihre Zahl ist: " << monateString << std::endl;
}
```

8.2.2 StringStreams

Ein Stringstream ist ein spezieller Stream für Strings. In einen Stringstream kann man eine Zeile oder einen ganzen Text eingeben, der Stringstream trennt die Eingabe am Space und gibt die getrennten Textteile wieder aus. Das folgende Codefragment zeigt die Verwendung von Stringstreams:

```
//wstringstream (stringstream für wstrings) wird erstellt und wordsOfLine genannt:
std::wstringstream wordsOfLine;

//line wird in den stringstream eingegeben:
wordsOfLine << line;

//solange der Stringstream noch Wörter enthält werden diese im wstring word gespeichert:
while (wordsOfLine >> word) {
    //hier kann man etwas mit den Wörtern machen, z.B. diese in einem Vector speichern
    myVector.push_back(word);
}
```

8.3 Lösungsvorschlag

8.3.1 Aufgabe 1

```
/*Autor: Nicola Greth
 * Uebung 7
 * Programm: verschiedene Funktionen
 */
#include <iostream>
#include <locale>
#include <fstream>
#include <sstream>
using namespace std;

//konstante maximale Anzahl von Arrayelementen,
//wird außerhalb der Hauptmethode deklariert, damit darauf auch in Funktionen
zugegriffen werden kann
static const int MAXDATA = 30;

//Funktionen werden deklariert
bool palindrom(wstring word);
void reverse(wstring &word);
bool splice(wstring w1, wstring w2, wstring &erg);
int split(wstring listSplit[], wstring line);
bool first(wstring listFirst[], wstring &firstWordFirst);
bool last(wstring listLast[], wstring &firstWordLast);
bool member(wstring word, wstring words[]);

//Hauptmethode
int main() {
   setlocale(LC_ALL, "de_DE.utf-8");

   //Dateiname wird eingelesen, in String konvertiert und Datei wird geöffnet
   wstring filename;
   wcout << L"Bitte geben Sie den Dateinamen ein: " << endl;
   getline(wcin, filename);

   //Filename wird in normalen String übersetzt
   string filenameString(filename.begin(), filename.end());

   //Datei wird geöffnet und Filename in cstring übersetzt
   wifstream datei(filenameString.c_str());

   //Datei wird auf utf8 eingefärbt
   datei.imbue(locale("de_DE.utf-8"));
   wstring line;

   //Die Zeilen der Datei werden durchlaufen
   while (getline(datei, line)) {
      wstring word;
      wstringstream stream;
```

```
      stream << line;

      //Die Wörter einer Zeile werden durchlaufen
      while (stream >> word) {

         //PalindromFunktion wird wortweise aufgerufen
         if (palindrom(word)) {
            wcout << L"Ihr Wort " << word << L" ist ein Palindrom"
            << endl;
         } else {
            wcout << L"Ihr Wort " << word << L" ist kein Palindrom"
            << endl;
         }
      }
   }
   datei.close();

   //Der Input Stream wird geleert
   wcin.clear();

   //Aufruf der Funktion splice
   wcout << endl << L"Jetzt kommt die Funktion splice!" << endl;
   wstring w1;

   wcout << L"Bitte geben Sie ein Wort ein, das Sie auf einen Substring
   testen möchten: " << endl;
   getline(wcin, w1);
   wcin.clear();

   wstring w2;
   wcout << L"Bitte geben Sie Ihren Substring ein: " << endl;
   getline(wcin, w2);
   wstring erg;

   bool spliced = splice(w1, w2, erg);

   if (spliced) {
      wcout << L"Ihr Wort " << w2 << L" kommt im Wort " << w1 << L" vor."
      << endl;
      wcout << L"Ihr Ergebniswort ist " << erg << endl;
   } else {
      wcout << L"Ihr Wort " << w2 << L" kommt nicht im Wort " << w1 << L"
      vor." << endl;
   }

   //Aufruf der Funktion split
   wcout << endl << L"Jetzt kommt die Funktion split!" << endl;
   wcin.clear();
   wstring lineSplit;
   wcout << L"Bitte geben Sie eine Zeile ein: " << endl;
   getline(wcin, lineSplit);
```

```
    wstring listSplit[MAXDATA];
    int sizeOfArray = split(listSplit, lineSplit);
    wcout << L"Die Anzahl der Woerter in Ihrer Zeile ist: " << sizeOfArray <<
    endl;

    //Aufruf der Funktion first
    wcout << endl << L"Jetzt kommt die Funktion first!" << endl;
    wstring firstWord;
    wstring listFirst[MAXDATA] = { L"das", L"ist", L"meine", L"Liste" };
    bool firstWordFound = first(listFirst, firstWord);

    if (firstWordFound) {
       wcout << L"Das erste Wort Ihrer Liste lautet: " << firstWord <<
       endl;
    } else {
       wcout << L"Ihre Liste ist leer" << endl;
    }

    //Aufruf der Funktion last
    wcout << endl << L"Jetzt kommt die Funktion last!" << endl;
    wstring first;
    wstring listLast[MAXDATA] = { L"Computerlinguistik", L"ist", L"ganz",
    L"toll" };
    bool lastWord = last(listLast, first);

    if (lastWord) {
       wcout << L"Das erste Wort Ihrer Liste, " << first << L", wurde
       entfernt." << endl;
    } else {
       wcout << L"Ihre Liste ist leer" << endl;
    }

    //Aufruf der Funktion member
    wcout << endl << L"Jetzt kommt die Funktion member!" << endl;
    wcin.clear();
    wstring myWord;
    wstring listMember[MAXDATA] = { L"C++", L"macht", L"Spass" };
    wcout << L"Welches Wort moechten Sie suchen?" << endl;
    getline(wcin, myWord);
    bool isMember = member(myWord, listMember);

    if (isMember) {
       wcout << L"Ihr Wort wurde in der Liste gefunden!" << endl;
    } else {
       wcout << L"Ihr Wort wurde in der Liste nicht gefunden!" << endl;
    }
    return 0;
}

//Funktionen werden implementiert
//Funktion palindrom, die testet ob ein Wort ein Palindrom ist
```

```
bool palindrom(wstring word) {
   wstring original = word;
   reverse(word);

   //wenn das Wort gleich dem Wort rückwärts ist, ist es ein Palindrom
   if (original.compare(word) == 0) {
      return true;
   }
   return false;
}

//Funktion reverse, die ein Wort umdreht
void reverse(wstring &word) {
   wstring wordReverse;

   //alle Buchstaben werden umgedreht
   for (int i = word.size() - 1; i >= 0; i--) {
      wordReverse += word.at(i);
   }
   word.erase();
   word = wordReverse;
}

//Funktion, die nach den w2 in w1 sucht und den Reststring in erg speichert
bool splice(wstring w1, wstring w2, wstring &erg) {
   int found = w1.find(w2);

   //solange der Substring gefunden wird
   while (found != -1) {

      //lösche den Substring in w1
      erg = w1.erase(found, w2.size());
      w1 = erg;

      //berechne found neu
      found = w1.find(w2);

      //erst im Fall dass das Wort nicht mehr gefunden wird bricht die
      //Funktion ab
      if (found == -1) {
         return true;
      }
   }

   //falls das Wort nicht gefunden wurde
   return false;
}

//Funktion, die eine Zeile in ein Array aus Wörtern aufteilt
int split(wstring listSplit[], wstring line) {
   int countOfWords = 0;
```

```
    wstringstream stream;
    stream << line;
    wstring word;

    //solange es Wörter gibt, werden diese hinzugefügt
    while (stream >> word) {
        listSplit[countOfWords] = word;
        countOfWords++;
    }
    return countOfWords;
}

//Funktion first, die das erste Wort aus der Liste ausliest
bool first(wstring listFirst[], wstring &firstWordFirst) {
    bool firstFound = false;

    //durchläuft die komplette Liste bis das Wort gefunden ist
    for (size_t i = 0; i < MAXDATA and !firstFound; i++) {
        //sobald das erste Wort gefunden wurde, bricht die Schleife ab
        if (listFirst[i] != L"") {
            firstWordFirst = listFirst[i];
            firstFound = true;
        }
    }
    return firstFound;
}

//Funktion last, die das erste Wort aus der Liste entfernt
bool last(wstring listLast[], wstring &firstWordLast) {
    bool wordFound = false;

    for (size_t i = 0; i < MAXDATA and !wordFound; i++) {

        //das erste nicht-leere Wort wird gefunden
        if (listLast[i] != L"") {
            firstWordLast = listLast[i];

            //das erste Wort wird gelöscht
            listLast[i] = L"";

            //die Schleifenbedingung wird verletzt
            wordFound = true;
        }
    }
    return wordFound;
}

//Funktion member, die nach einem Wort in einer Liste sucht
bool member(wstring word, wstring words[]) {
    wstring firstWord;
    bool notEmpty = last(words, firstWord);
```

```
    //wenn die Liste nicht leer ist
    if (notEmpty) {

       //wenn das Wort gefunden wurde
       if (word.compare(firstWord) == 0) {
          return true;

       //wenn das Wort nicht gefunden wurde
       } else {
          return member(word, words);
       }
    } else {
       return false;
    }
}

/*
 * Zur Info: Man kann sämtliche Funktionen auch deutlich kürzer schreiben!
 * Als Beispiel die Funktion member:
 * bool member(wstring word, wstring word_list[]) {
 * wstring first_word;
 * return last(word_list, first_word)
 * ? (word.compare(first_word) == 0 ? true : member(word, word_list))
 * : false;}
 * In der Musterlösung schreibe ich Programme weiterhin ausformuliert, damit ihr
die einzelnen Schritte besser nachvollziehen könnt
 */
```

8.3.2 Aufgabe 1 mit Vektoren

```
/*Autor: Nicola Greth
 * Uebung 7
 * Programm: verschiedene Funktionen mit Vektor
 */
#include <iostream>
#include <locale>
#include <fstream>
#include <sstream>
#include <vector>
using namespace std;

//Funktionen werden deklariert
int split(vector<wstring> &listSplit, wstring line);
bool first(vector<wstring> listFirst, wstring &firstWordFirst);
bool last(vector<wstring> &listLast, wstring &firstWordLast);
bool member(wstring word, vector<wstring> words);

int main() {
   setlocale(LC_ALL, "de_DE.utf-8");

   //Aufruf der Funktion split
```

```
    wcin.clear();
    wstring lineSplit;
    wcout << L"Bitte geben Sie eine Zeile ein: " << endl;
    getline(wcin, lineSplit);
    vector<wstring> listSplit;
    int sizeOfArray = split(listSplit, lineSplit);
    wcout << L"Die Anzahl der Woerter in Ihrer Zeile ist: " << sizeOfArray << endl;

    //Aufruf der Funktion first
    wstring firstWord;
    bool firstWordFound = first(listSplit, firstWord);

    if (firstWordFound) {
       wcout << L"Das erste Wort Ihrer Liste lautet: " << firstWord << endl;
    } else {
       wcout << L"Ihre Liste ist leer" << endl;
    }

    //Aufruf der Funktion last
    wstring first;
    bool lastWord = last(listSplit, first);

    if (lastWord) {
       wcout << L"Das erste Wort Ihrer Liste, " << first << L", wurde entfernt." << e
    } else {
       wcout << L"Ihre Liste ist leer" << endl;
    }

    //Aufruf der Funktion member
    wcin.clear();
    wstring myWord;
    wcout << L"Welches Wort moechten Sie suchen?" << endl;
    getline(wcin, myWord);
    bool isMember = member(myWord, listSplit);

    if (isMember) {
       wcout << L"Ihr Wort wurde in der Liste gefunden!" << endl;
    } else {
       wcout << L"Ihr Wort wurde in der Liste nicht gefunden!" << endl;
    }
    return 0;
}

//Funktionen werden implementiert
//Funktion, die eine Zeile in ein Array aus Wörtern aufteilt
int split(vector<wstring> &listSplit, wstring line) {
    int countOfWords = 0;
    wstringstream stream;
    stream << line;
    wstring word;
```

```
    //solange es Wörter gibt werden diese hinzugefügt
    while (stream >> word) {
        listSplit.push_back(word);
        countOfWords++;
    }
    return countOfWords;
}

//Funktion first, die das erste Wort aus der Liste ausliest
bool first(vector<wstring> listFirst, wstring &firstWordFirst) {
    if (not listFirst.empty()) {
        firstWordFirst = listFirst.front();
        return true;
    }
    return false;
}

//Funktion last, die das erste Wort aus der Liste entfernt
bool last(vector<wstring> &listLast, wstring &firstWordLast) {
    bool word_found = false;

    if (not listLast.empty()) {
        if (first(listLast, firstWordLast)) {
            listLast.erase(listLast.begin());
            word_found = true;
        }
    }
    return word_found;
}

//Funktion member, die nach einem Wort in einer Liste sucht
bool member(wstring word, vector<wstring> words) {
    wstring firstWord;
    bool notEmpty = last(words, firstWord);

    //wenn die Liste nicht leer ist
    if (notEmpty) {

        //wenn das Wort gefunden wurde
        if (word.compare(firstWord) == 0) {
            return true;

        //wenn das Wort nicht gefunden wurde
        } else {
            return member(word, words);
        }

    } else {
        return false;
    }
}
```