Übungsblatt 9

(Decade II, Septidi de Frimaire de l'Annee 218 de la Revolution)

Aufgabe 1 (2 Punkte) In diesem Blatt sollen Sie eine kleine Bibliothek zum Finden von Reimwörtern schreiben. Hierzu steht in die Bibliotek `Words.h` bereits zu großen Teilen implementiert zur Verfügung. Machen Sie sich mit der Datenstruktur `WordList` vertraut und implementieren Sie die noch nicht umgesetzten Funktionen.

a) Betrachten Sie, wie die Datei `Words.h` für das Tool `doxygen` (zu finden auf `www.doxygen.org/`) dokumentiert ist. Die generierte Dokumentation findet sich auf: `http://panitz.name/c/blatt9/doc/html`. Dokumentieren Sie die Datei `StringUtil.h` entsprechend und lassen sich die html-Dokumentation generieren.

b) Ergänzen Sie die Implementierung von StringUtil aus dem letzten Übungsblatt um die Funktion `isUmlaut`. Beachten Sie, dass Sie den C-Quelltext in Ihrem Editor im Encoding latin-1 (ISO-8859-1) abspeichern.

c) `boolean contains(string* words,nat length,string w);`
Die Funktion soll in der Reihung `words` von Strings schauen, ob sie das Wort `w` enthält. Die Reihung enthält `length` Elemente.

d) `void sortWith(WordList* this,boolean le(string,string));`
Diese Prozedur soll die in der Wörterliste gespeicherten Wörter neu sortieren. Die übergebene Funktion `le` soll benutzt werden, um zu bestimmen, ob das erste Argument kleiner oder gleich dem zweiten ist. Benutzen Sie zum Sortieren den in der Vorlesung gezeigten Bubble-Sort Algorithmus. Testen Sie die Implementierung mit dem Programm `PrintRhymingDictionary.c`. Als Eingabedateien können Sie hierzu gut Texte aus dem Projekt Gutenberg verwenden: `http://www.gutenberg.org/browse/languages/de`.

e) `string findRhyme(WordList* this,string w);`
Die Funktion soll in der Wörterliste ein Reimwort für das zweite Argument finden. Die Funktion geht davon aus, dass die Wortliste bereits so sortiert wurde, dass reimende Wörter hintereinander stehen. Testen Sie Ihre Implementierung mit dem Programm `GetRhyme.c`