Cross-Reference-System

Aufgabe:

Es soll ein Programm geschrieben werden, das die Positionen aller Vorkommen eingegebener Worte in bestimmten Files anzeigt.

Die Namen aller zu durchsuchenden Files werden beim Aufruf des Programmes als Argumente angegeben. Die zu suchenden Worte werden am Terminal eingetippt (ein Wort pro Zeile).

Für jedes der angegebenen Worte sollen die Positionen aller Vorkommen anzeigt werden (Filename, Zeile und Spalte, aber *nicht* der Text der Zeile selbst, sinnvollerweise nach File gruppiert!) bzw. eine entsprechende Meldung, wenn ein Wort nicht in den Files enthalten ist.

Um nicht bei jedem zu suchenden Wort alle Files durchsuchen zu müssen, werden die Files beim Programmstart einmal gelesen, und es wird im Speicher ein Index aller vorkommenden Worte und ihrer Positionen erstellt.

In der ersten Programmvariante verwenden wir dafür eine Hashtable mit einer einfach verketteten Liste pro Hash-Index.

Hinweise:

- Wir definieren "Wort" wie einen Identifier in C: Beginnend mit einem Buchstaben oder `_`, bis vor das erste Zeichen, das nicht mehr Buchstabe, Ziffer oder `_` ist, mit Unterscheidung der Groß- und Kleinschreibung.
- Alle Daten im Index außer dem Hash-Array selbst werden dynamisch (mit `malloc`) angelegt.

  Achtung: Du solltest jedes Wort nur *einmal* allokieren, nicht einmal pro Vorkommen! Auch die Filenamen sollten *nicht* pro Position einmal kopiert werden!
- Jeder einzelne Wort-Eintrag in den Hash-Listen enthält wiederum einen Verweis auf eine weitere einfach verkettete Liste, nämlich die Liste der Positionen, an denen dieses Wort vorkommt. Ein Wort-Eintrag besteht also aus dem Wort-String, aus zwei Pointern auf den Anfang und das Ende der Liste seiner Vorkommen (weil wir hinten direkt anfügen können wollen), und aus dem Link zum nächsten Wort in der gleichen Hash-Liste.
- Ein Positions-Eintrag enthält den Filenamen, die Zeilen- und die Spalten-Nummer des Vorkommens, und einen Link zum nächsten Vorkommen desselben Wortes.
- Du solltest neue Einträge in den Wort-Listen *vorne* und in den Positions-Listen *hinten* anfügen: Bei den Wort-Listen steigert es die Such-Effizienz (weil dadurch tendentiell die gerade aktuellen Worte vorne und ältere weiter hinten stehen), bei den Positions-Listen erhält es die Reihenfolge der Vorkommen (sonst stehen sie in der Ausgabe "am Kopf").
- *Halte dich bei deinem Progamm an das von mir vorgegebene Programmgerüst (auf meinen Webseiten zum Download)!*

Zusatzaufgabe:

- Versuche, Statistiken über die Anzahl der Worte, die durchschnittliche und maximale Listenlänge, und die Verteilung der Listenlängen anzuzeigen!