Freiwillige Offline-Aufgabe 08-05 (INF & WI): Wortliste eines Texts, mit Häufigkeiten, mittels C-Strings

(geübte C++ Konstrukte: C-Strings, Pointer, Arrays, Schleifen, struct)

Schreiben Sie ein C++ Programm, welches einen Text in Form von mehreren Eingabezeilen einliest. Das Programm soll die in dem Text auftretenden Worte in der Reihenfolge ihres Auftretens mitsamt der Häufigkeit ihres Vorkommens im Text ausgeben.

Einzelne Eingabezeilen und Worte sollen als C-Strings gespeichert werden. Die `cstring` Bibliothek mit den darin enthaltenen Funktionen darf im Rahmen dieser Aufgabe aber ***nicht*** verwendet werden, sondern entsprechende Funktionen müssen selbst programmiert werden. (Hinweis: C-Strings als `char*` sind auch ohne die `cstring` Bibliothek verwendbar).

Die „Sammlung aller nicht-leeren Eingabezeilen" sowie die „Sammlung der Worthäufigkeiten" sollen als statische Arrays gespeichert werden. Es kommen maximal `const unsigned int eingabezeilen_max = 5;` Eingabezeilen und maximal `const unsigned int wort_max = 1000;` Worte vor. Jede Eingabezeile enthält maximal `const unsigned int max_line_length = 80;` Zeichen (inklusive des Nullterminators).

Eine leere Eingabezeile beende die Eingabe. Eingabezeilen können auch Leerzeichen enthalten. Leerzeichen, Punkt, Komma sowie Anfang und Ende jeder Eingabezeile fungieren als Wortbegrenzungen. Im Eingabetext sollen auch keine anderen Satzzeichen als Punkt und Komma (und Leerzeichen) vorkommen, ihr Programm kann davon ausgehen, dass der Benutzer nur korrekte Eingaben macht.

Die Worthäufigkeit jedes Worts soll jeweils in einer Datenstruktur …

```
struct w_haeufigkeit {
    char* wort;
    unsigned int haeufigkeit;
};
```

… gespeichert werden (und davon ein Array mit `wort_max` Einträgen).

Es sollen für das Ermitteln der Worthäufigkeiten folgende Funktionen definiert werden (diese Funktionen werden von Jenkins ggfs. auch einzeln getestet, sie müssen also vorhanden sein und wie vorgeschrieben definiert werden):

```
unsigned int my_strlen(const char* const str);
```

… berechnet die Länge eines C-Strings, indem sie die Zeichen bis zum ersten Nullterminator zählt (d.h. ein C-String `"abc"` hat die Länge `3`),

```
int my_strcmp(const char* str1, const char* str2);
```

… vergleicht zwei C-Strings und gibt den Wert `0` zurück, wenn diese zeichenweise identisch sind, ansonsten einen Wert ungleich `0`,

```
char* naechstes_wort(const char* const zeile,
                     unsigned int& pos);
```

… suche ausgehend von der Position `pos` in der Zeile das nächste Wort und gebe es kopiert in einen eigenen Speicherbereich auf dem Heap (für dieses Wort) zurück. `pos` wird dann auf die erste Position (in der Zeile) hinter dem Wort geändert. Sollte es ausgehend von `pos` kein nächstes Wort in der Zeile geben, so werde der `nullptr` als Wert zurückgegeben.

```
void zaehle_wort(char* wort,
                 struct w_haeufigkeit w_haeufigkeiten[],
                 unsigned int& w_count);
```

… suche das Wort `wort` in der Wortsammlung `w_haeufigkeiten[]`, die aktuell bis zur Position `w_count-1` belegt sei. Kommt das Wort schon vor, so werde seine Häufigkeit um `1` erhöht. Kommt das Wort bisher noch nicht vor, so werde es hinzugefügt (der Zeiger auf den Heap-Speicher des Wortes werde in die Datenstruktur umgehängt, d.h. die Datenstruktur wird zum Besitzer des Wortes) und seine Häufigkeit auf `1` gesetzt.

Die Arrays für den Eingabetext und die gesammelten Worthäufigkeiten seien lokale Variablen des Hauptprogramms.

Testläufe (Benutzereingaben sind unterstrichen):

```
Eingabezeile = ? Dies ist die erste Zeile des Texts.
Eingabezeile = ? Und dies, ist, noch          eine Zeile.
Eingabezeile = ? .Und noch eine weitere Zeile, im Text.
Eingabezeile = ?
Dies: 1
ist: 2
die: 1
```

```
erste: 1
Zeile: 3
des: 1
Texts: 1
Und: 2
dies: 1
noch: 2
eine: 2
weitere: 1
im: 1
Text: 1
Drücken Sie eine beliebige Taste . . .
```