Wörter zählen

Allgemeine Anmerkungen zur Aufgabenstellung:

- Beantworten Sie die Kontrollfragen in `TUWEL` bevor Sie mit der Übung beginnen.
- Legen Sie für `Teil A` der Angabe ein neues `Code::Blocks`-Projekt an und erweitern Sie dieses schrittweise für die darauffolgenden Punkte.
- Stellen Sie während der Übung laufend sicher, dass Ihr Programm kompilierbar ist und richtig funktioniert.
- Die Abgabe erfolgt durch Hochladen Ihres vollständigen `Code::Blocks`-Projektordners als zip-Datei (`Matrikel-Nr_UE4.zip`) in `TUWEL`. Anschließend melden Sie sich für das Abgabegespräch in `TUWEL` an. Dieses erfolgt über die Telekonferenz-App `Zoom` (www.zoom.us). Stellen Sie sicher, dass diese App rechtzeitig vor Beginn des Abgabegespräches auf Ihrem PC installiert ist und dass Sie Ihr aktuelles Übungsprogramm in `Code::Blocks` für die Abgabe geöffnet haben. Testen Sie bitte auch die korrekte Funktionalität Ihrer Videokamera, da Sie zu Beginn Ihres Abgabegespräches Ihren Studentenausweis herzeigen müssen und auch während des Gesprächs eine aktive Videoverbindung erforderlich ist.
- Beachten Sie, dass Upload und Terminauswahl für das Abgabegespräch erst möglich sind, wenn Sie 80% der Kontrollfragen richtig beantwortet haben. Bis **25.05.2021 23:59** müssen diese Fragen positiv beantwortet, Ihre Ausarbeitung hochgeladen und ein Termin für das Abgabegespräch ausgewählt werden.
- Inhaltliche Fragen stellen Sie bitte in der Fragestunde oder im zugehörigen Forum in `TUWEL`.
- Für eine positive Beurteilung Ihrer Abgabe **muss** diese kompilierbar sein, Ihr Programm fehlerfrei terminieren und die in der Angabe angeführten Funktionen mit den vorgegebenen Funktionsköpfen enthalten. Des Weiteren müssen die Funktionen hinreichend getestet sein. Sie müssen in der Lage sein, Ihr Programm beim Abgabegespräch zu erklären!

Folgende Hilfsmittel stehen Ihnen in `TUWEL` zur Verfügung:

- Alle bisherigen Referenzbeispiele sowie Vorlesungsfolien
- Interaktive Jupyter-Notebooks auf `prog1.iue.tuwien.ac.at`
- Das Buch zur LVA "Programmieren in C" (Robert Klima, Siegfried Selberherr)
- Kurzanleitung für die Entwicklungsumgebung `Code::Blocks`

Hinweis: Es wird erwartet, dass alle Abgaben zu den Übungen **eigenständig** erarbeitet werden und wir weisen Sie darauf hin, dass alle Abgaben in einem standardisierten Verfahren auf Plagiate überprüft werden. Bei Plagiatsvorfällen entfällt das Abgabegespräch und es werden keine Punkte für die entsprechende Abgabe vergeben.

Einleitung

In dieser Übung werden Sie ein Programm erstellen, welches einen Text einliest und ihn anschließend in Wörter und andere Zeichen zerlegt. Damit kann unter anderem die Wortanzahl in einem Text bestimmt werden oder die Häufigkeit von Wortlängen berechnet werden. Dabei erlernen Sie den Umgang mit Zeichenketten sowie deren Manipulation innerhalb der Programmiersprache C. Da Zeichenketten in C als Felder des Typs `char` realisiert werden, lernen Sie hier auch über Indizes oder Zeiger auf Feldelemente zuzugreifen um diese zu lesen oder zu verändern.

Zusätzlich zum Stoff aller vorangegangenen Übungen benötigen Sie für diese Übung folgende neue Themengebiete aus den Vorlesungen 6-8 (Kapitel 13-15):

- Eindimensionale Felder, Übergabe von Feldern, Sortieren und Suchen (Kapitel 13.1, Kapitel 13.5, Kapitel 13.6, 13.7)
- Zeiger (Kapitel 14)
- Zeichenketten (Kapitel 15)

Teil A

✘ Schreiben Sie eine Funktion

```
void my_getline(char input[], long len);
```

die einen Text von der Tastatur einliest und im Feld `input` abspeichert. Das Argument `len` gibt dabei die Länge des Feldes an. Falls die Eingabe länger als `input` ist, so sollen die übriggebliebenen Zeichen aus dem Eingabebuffer gelöscht werden. Vergessen Sie nicht, das Ende des Textes mit '\0' zu terminieren!

✘ Erstellen Sie die Funktion

```
long get_letters(char text[]);
```

die am Anfang des übergebenen Textes nach einem Wort sucht und seine Länge zurückgibt. Ein Wort darf dabei nur aus den Buchstaben a-z und A-Z bestehen. Überprüfen Sie dazu Schritt für Schritt die Zeichenkette `text`. Ist das erste Zeichen kein Buchstabe, so soll 0 zurückgegeben werden. Wird jedoch ein Buchstabe erkannt, so soll auch das zweite Zeichen kontrolliert werden und so weiter. Die Überprüfung wird beim ersten Zeichen, welches kein Buchstabe ist, abgebrochen und die Anzahl der bis dahin gefundenen Buchstaben wird zurückgegeben.

Beispiele:

Text	Rückgabewert
.0%test	0
.0%	0
Beispiel	8
bestehen. Ist	8

✘ Implementieren Sie die Funktion

```
long get_others(char text[]);
```

die *alle* Zeichen außer Buchstaben (daher auch Whitespaces) erkennt. Die Funktion soll analog zu `get_letters` die Anzahl der Treffer zurückgeben.

Beispiele:

Text	Rückgabewert
.0 %test	4
.0%	3
Beispiel	0
bestehen. Ist	0

✘ Schreiben Sie ein Hauptprogramm, welches einen Text einliest und darauf die Funktionen `get_letters` und `get_others` anwendet. Verwenden Sie $N = 100$ für die maximale Länge des Textes. Die Programmausgaben könnten in etwa so aussehen:

```
Texteingabe: Dies ist ein Beispieltext.
Buchstaben am Anfang:           4
----------------------------
Nochmal (j|n)? j

Texteingabe: 100% ist ein Ganzes.
Andere Zeichen am Anfang:       5
----------------------------
Nochmal (j|n)? n

------PROGRAMM-ENDE-------
```

Teil B

✘ Implementieren Sie die Funktion

```
void analyze_text(char text[]);
```

welche die Gesamtanzahl an Wörtern im Text bestimmt und auf der Konsole ausgibt. Wenden Sie dazu die Funktionen `get_letters` und `get_others` wiederholt auf den Text an, bis dieser zu Ende ist.

Weiters sollen die Häufigkeiten der auftretenden Wortlängen erfasst und ausgeben werden. Legen Sie dazu ein Feld mit 10 Elementen an, welches während des Bearbeitens des Textes die Anzahl von Wörtern mit 1,2,... Buchstaben speichert. Das letzte Feldelement speichert dabei, wie viele Wörter aus 10 *oder mehr* Buchstaben bestehen.

✘ Rufen Sie die Funktion `analyze_text` in Ihrem Hauptprogramm auf. Ein möglicher Programmablauf könnte so aussehen:

```
Texteingabe: Das ist ein ganz kurzer Beispieltext.
Buchstaben am Anfang:   3
Wortanzahl:             6
Statistik:
----------------------------
Laenge    |   Haeufigkeit
   1      |             0
   2      |             0
   3      |             3
   4      |             1
   5      |             0
   6      |             1
   7      |             0
   8      |             0
   9      |             0
>=10      |             1
----------------------------
Nochmal (j|n)? n

------PROGRAMM-ENDE-------
```

Teil C

✘ Implementieren Sie die beiden Funktionen

```
char* get_letters_ptr(char text[]);
char* get_others_ptr(char text[]);
```

welche analog zu `get_letters` und `get_others` arbeiten, allerdings nicht die ermittelte Länge zurückgeben sondern einen Zeiger auf das erste Zeichen nach dem ermittelten Wort bzw. Block zurückliefern.

Das bedeutet, `get_letters_ptr` gibt bspw. einen Zeiger auf das erste Zeichen in `text`, welches kein Buchstabe mehr ist, zurück. Wird das Ende von `text` erreicht, sollen beide Funktionen einen Nullzeiger zurückliefern.

✘ Schreiben Sie eine dazugehörige Funktion

```
void analyze_text_ptr(char text[]);
```

welche die neuen Funktionen `get_letters_ptr` und `get_others_ptr` analog zu Teil B verwendet.

Teil D

✘ Schreiben Sie eine Funktion

```
void trim_text(char text[]);
```

welche alle Zeichen, die keine Buchstaben sind, aus `text` entfernt und zwischen den einzelnen Wörtern nur ein einziges Leerzeichen übrig lässt. Achten Sie dabei darauf, dass keine Wörter gelöscht werden oder Speicherlücken in `text` entstehen. Nach Ausführung der Funktion `trim_text` soll der neue gekürzte Text nach dem letzten Wort nullterminiert werden, um eine Weiterverarbeitung zu ermöglichen.

Hinweis: Beachten Sie, dass die beim Löschen entstandene Lücke in einer for-Schleife Zeichen-für-Zeichen geschlossen werden muss. (Expertenfrage: Könnte man die Lücke durch einen geschickten Aufruf von `memmove` schließen? Warum sind `strcpy`/`strncpy`/`memcpy` in diesem Fall nicht geeignet?)

✘ Geben Sie den gekürzten Text im Hauptprogramm aus.

```
Texteingabe: 10, 11, 12, ... bis 99 sind definitiv kleiner
             als 100. Aber 1000 ist         noch viel groesser.
Andere Zeichen am Anfang:        16
Wortanzahl:                      10
Statistik:
----------------------------
Laenge     |   Haeufigkeit
1          |              0
2          |              0
3          |              3
4          |              4
5          |              0
6          |              0
7          |              1
8          |              1
9          |              1
>=10       |              0
----------------------------
Nur Woerter: bis sind definitiv kleiner als Aber ist noch viel
            groesser
Nochmal (j|n)? n

------PROGRAMM-ENDE-------
```