Dr. habil. Bernd Schürmann
Dr. Annette Bieniusa
pinc-support@cs.uni-kl.de

TU Kaiserslautern
Fachbereich Informatik

Übungsblatt 10: Programmieren in C (WS 2020/21)

Abgabe: Montag, 25.01.21, 12:00

Texteditor

Diese Aufgabe bildet den Auftakt zur schrittweisen Entwicklung eines größeren Programms, das in den nächsten Übungsblättern fortgesetzt wird. Wir werden dazu einen einfachen konsolenbasierten Texteditor implementieren. Um Ihnen den Einstieg zu erleichtern, finden Sie eine Vorlage in OLAT. Bitte verwenden Sie in den Teilaufgaben geeignete Hilfsfunktionen (z. B. die Hilfsfunktionen zur Behandlung von Strings aus Aufgabe 3 in Aufgabe 4).

Aufgabe 1 *Einlesen der Benutzerkommandos (6 Punkte)*

Unser einfacher Editor soll über Kommandos der Standardeingabe gesteuert werden.[1] Schreiben Sie hierzu eine Funktion, die das nächste Eingabekommando liest und überprüft und die Kommandodaten als Ergebnis zurückliefert.

Folgende Kommandos sind für unseren Editor vorgesehen und müssen eingelesen und erkannt werden.

`d<row>`	Lösche Zeile `<row>`
`i<row>/<text>`	Füge eine Zeile `<row>` mit dem Text `<text>` ein
`p` bzw. `p<row>`	Drucke den gesamten Text bzw. die Zeile `<row>` auf dem Bildschirm aus
`q`	Editor beenden (quit)
`r`	Daten zurücksetzen (reset)

Implementieren Sie eine Funktion

```
void next_command (char *command_char, int *line_no, char text[]);
```

die eine Kommandozeile mittels `fgets` einliest und den Kommandobuchstaben, die Zeilenangabe sowie den Text **über die Parameter** zurückliefert. Ist in einem Kommando keine Zeilenangabe bzw. Text vorhanden, wird Null bzw. eine leere Zeichenkette zurückgegeben. Sie können davon ausgehen, dass nur korrekte Eingaben vorliegen, d. h. Sie müssen noch nicht überprüfen, ob die Kommandostruktur korrekt ist (das kommt auf dem nächsten Übungsblatt).

Auf OLAT im Order zu Übungsblatt 10 finden Sie als Vorlage für diese Teilaufgabe ein kleines Programm, mithilfe dessen Sie diese Funktion implementieren und testen können! Bitte geben Sie für diese Teilaufgabe nur die Datei `command.c` ab!

Beispiele für Kommandos: Die Kommandos sollen folgendermaßen aussehen (Reihenfolge ist für die Beispiele hier nicht wichtig):

```
d34
i1/Hallo Welt!
p
p1
q
r
```

[1] Ein Beispiel für einen solchen Editor ist der Unix-Editor vi.

Aufgabe 2 *Funktionlität des Texteditors (20 Punkte)*

Anlegen und Initialisieren der globalen Datenstruktur (2 Punkte)

Es soll ein globales zweidimensionales Array von Zeichen angelegt werden (`text_field`). Die Anzahl von Zeilen `LINE_COUNT` und Zeichen pro Zeile `LINE_LENGTH` sind als Präprozessor-Konstanten mit Wert 1000 bzw. 256 zu definieren. Weiterhin benötigen Sie eine Variable `max_line_no`, die die Anzahl beschriebener Zeilen speichert; diese hat initial den Wert 0. Die Verwendung von `max_line_no` erlaubt es uns, bei Ausgabe auf dem Bildschirm (Kommando `p`) den Text nur bis zur letzten beschriebenen Zeilen auszugeben.

Hilfsfunktionen für Strings (6 Punkte)

Implementieren Sie folgende Hilfsfunktionen zum Einfügen, Löschen und Kopieren von Zeilen:

`void copy_line (int to, int from);`	Kopiert den Text der Zeile `from` in die Zeile `to`.
`void add_line (int line_no);`	Fügt eine leere Zeile hinzu. Hierzu müssen zunächst alle Zeilen von `line_no` bis zum Textende (`max_line_no`) um jeweils eine Zeile nach unten kopiert werden. [2]
`void delete_line (int line_no);`	Hierzu müssen die Zeilen `line_no+1` bis zum Textende (`max_line_no`) um jeweils eine Zeile nach oben kopiert werden. Zusätzlich muss eine leere Zeile am Textende eingefügt werden.

Hinweis: Bei den beiden letzten Funktionen muss auch die globale Variable `max_line_no` aktualisiert werden.

Eingabe und Ausgabe (4 Punkte)

Implementieren Sie die folgenden zwei Ausgabefunktionen:

`void print_text (void);`	Gibt den gesamten Text auf dem Bildschirm aus.
`void print_line (int line_no);`	Gibt die Zeile `line_no` auf dem Bildschirm aus.

Bei der Ausgabe soll vor jede Zeile immer die Zeilennummer geschrieben werden (Zeilennummer mit 3 Positionen).

Beispiel:

```
  1  aaa
  2  bbb
  3  ccc
```

Editor-Funktionen (4 Punkte)

Implementieren Sie nun folgende Funktionen des Editors:

`void insert (int line_no, char text[]);`	Fügt den Text `text` in die Zeile `line_no` ein. Verwenden Sie hierzu die o.g. Hilfsfunktionen aus der vorherigen Aufgabe.
`void delete (int line_no);`	Löscht die Zeile `line_no`. Verwenden Sie hierzu die oben genannten Hilfsfunktionen aus dem Modul `sub_functions`.

Globale Steuerung (4 Punkte)

Schreiben Sie die Hauptfunktion `main`, die zunächst `text_field` initialisiert und dann die Benutzerinteraktion durchführt. Im Kern besteht die Mainfunktion aus eine Endlosschleife, die

jeweils ein Kommando über `next_command` einliest und abhängig vom zurück gegebenen Kommandobuchstaben die entsprechende Editorfunktion aus dem Modul `editor_functions` aufruft. Beim Kommando 'q' wird das Programm beendet. Die in `main.h` global deklarierten Variablen `text_field` und `max_line_no` sind im Modul `main` zu definieren und zu initialisieren.

Hinweise:

- Sie können die Aufgabe als Erweiterung von Aufgabe 1 implementieren.
- Dokumentieren Sie Ihr Programm ausführlich, d.h. der Zweck jeder globalen Variablen und jeder Funktion muss erläutert werden!
- Auf den nächsten Übungsblättern wird die Arbeit an dem Texteditor weitergeführt.
- Sie dürfen bei der Implementierung die String-Bibliothek `string.h` verwenden.
- Laden Sie bitte für diese Aufgabe **alle** Dateien hoch (nur .c und .h Dateien)!

Beispiel für eine Interaktionssequenz

Eingabe:

```
i1/Hello
i2/World!
i3/Programming
i4/in
i5/C
i6/is
i7/great!
p
i7/really
p
d7
p
q
```

Ausgabe:

```
1  Hello
2  World!
3  Programming
4  in
5  C
6  is
7  great!
1  Hello
2  World!
3  Programming
4  in
5  C
6  is
7  really
8  great!
1  Hello
2  World!
3  Programming
4  in
5  C
6  is
7  great!
```