1. Praktische Übung in "Programmieren": Textanalyse

Ein von der Standardeingabe einzulesender Text soll mittels eines **C-Programms** zeichenweise analysiert werden.
Folgende Merkmale sollen ermittelt werden:

- Anzahl der Zeilen des Textes,
- Anzahl der Wörter des Textes,
- Verteilung der Wortlängen des Textes

Ein **Wort** ist jede Symbolfolge, vor der sich ein Blank, TAB (`'\t'`) oder NL (`'\n'`) oder ein beliebiges anderes ASCII-Steuerzeichen befindet und die mit einem Buchstaben oder Unterstrich (`'_'`) beginnt und deren Folgesymbole beliebige darstellbare Zeichen außer den Satzzeichen sind. Ein Satzzeichen ist eines der Zeichen `'.'` , `','` , `';'` , `'!'` , `'?'` und `':'` . Es beendet ein Wort, gehört aber selbst nicht mehr zu dem Wort.
Auch ein Blank oder ein beliebiges ASCII-Steuerzeichen beendet ein Wort.

Beispiel: Die Zeichenfolge **`Beispiel Zahlen12.`** `+nein%..da` **`d-och#+#;`**`die` **`_%Testen`**`!!en`
enthält gemäß dieser Definition `4` Wörter der Länge `8`.

- **Lösungsvorgaben :**
  - Die Analyse des durch "`getchar()`" einzulesenden Textes erfolgt zeichenweise (keine Ablage in Arrays !)
  - Deutsche Umlaute sind – der Einfachheit halber – keine Buchstaben sondern andere darstellbare Zeichen
  - Die Zeilen werden mit NL (`'\n'`) abgeschlossen, `<Ctrl-Z>` (→ EOF) beendet die Texteingabe und kann auch am Ende der letzten Zeile stehen. EOF unmittelbar nach einem Zeilenendezeichen (`'\n'`) wird nicht als neue Zeile gezählt.
  - Die Analyse des Textes wird mit Hilfe eines **Zustandsautomaten** (***state machine***) durchgeführt :
    Der zu analysierende Text befindet sich – hinsichtlich der enthaltenen Wörter – in jeder Lesephase in einem **Zustand**, der durch das jeweils nächste gelesene Zeichen geändert werden kann. Der jeweilige **neue Textzustand** hängt dabei von dem **alten Textzustand** und der **Kategorie** des **aktuell gelesenen Zeichens** ab.

    Zur Kennzeichnung des Textzustands und der Zeichenkategorie sind die nachfolgend angegebenen **symbolischen Konstanten** einzusetzen. Sie sind geeignet zu definieren.

    - für den **Textzustand** :

`VOR_WORT`	nächstes Zeichen kann Anfang eines Wortes sein (u. Ausgangszustand zu Programmbeginn),
`IM_WORT`	innerhalb eines Wortes,
`KEIN_WORT`	innerhalb einer Zeichenfolge, die kein Wort ist – d.h. das folgende Zeichen kann nicht Beginn eines neuen Wortes sein.

    - für die **Zeichenkategorie** :

`CTL`	Blank sowie beliebige ASCII-Steuerzeichen (einschließlich TAB (`'\t'`) und NL (`'\n'`))
`BU`	Groß-, Kleinbuchstaben und Unterstrich,
`SAZ`	Satzzeichen (`'.'` , `','` , `';'` , `'!'` , `'?'` und `':'`)
`SONST`	die verbleibenden darstellbaren ASCII-Zeichen.

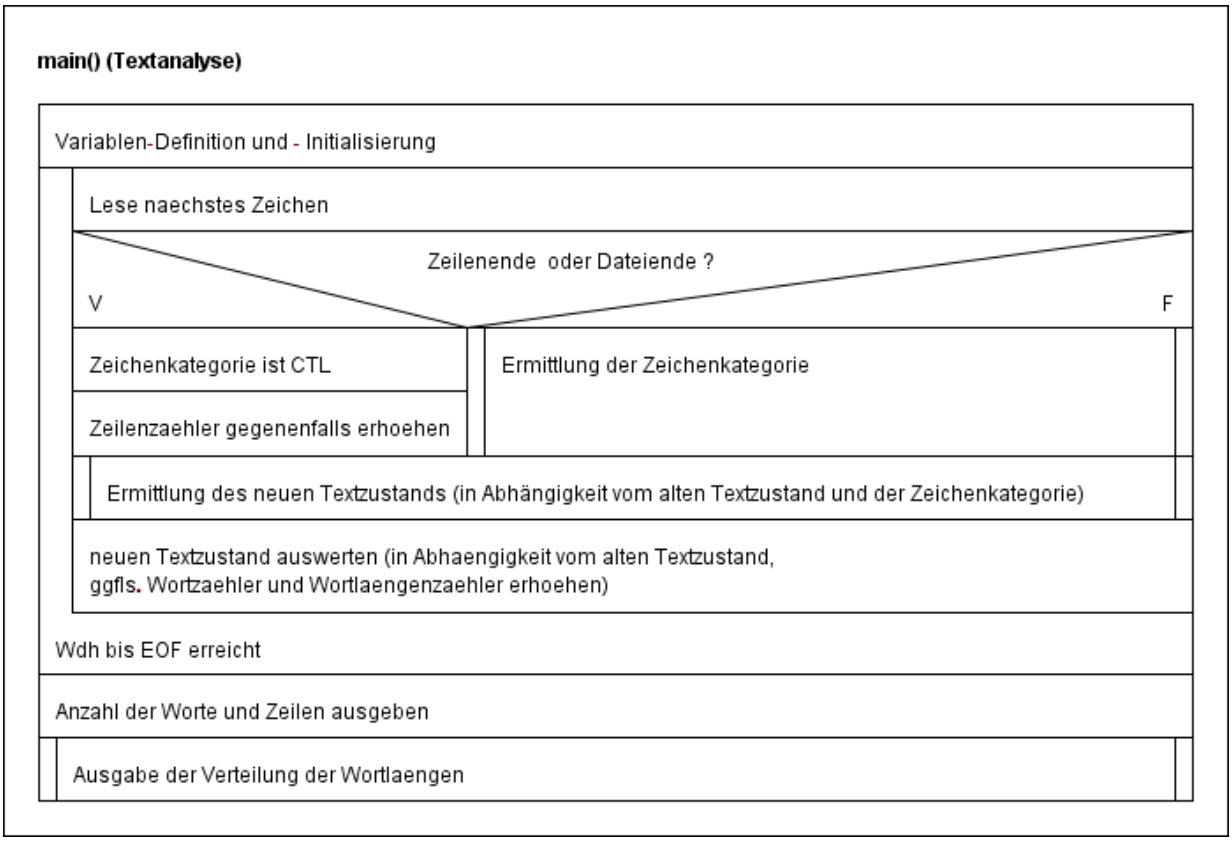
  - Die verschiedenen **Wortlängen** sind in den Elementen eines zu definierenden **`int-Arrays`** abzulegen.
    Die Größe dieses Arrays ist durch die symbolische Konstante `MAXLAENGE` festgelegt. Defaultmässig soll `MAXLAENGE` den Wert `18` besitzen.
    Der Array-Index bestimmt die Wortlänge, wobei der Index 0 für die Wortlänge 1, der Index 1 für die Wortlänge 2 usw steht. Das jeweilige Element selbst enthält die Anzahl der Worte mit der entsprechenden Länge. Dabei sind alle Worte mit einer Wortlänge `>= MAXLAENGE` im letzten Array-Element zusammenzufassen.
  - **Globale Variable** sind in dem zu realisierenden C-Programm **nicht zulässig**.
  - Das zu realisierende C-Programm ist **modular** zu strukturieren, d.h. sinnvoll gewählte **Teilfunktionalitäten** sind in **eigene Funktionen** auszulagern.

weiter Seite 2

Hierbei handelt es sich um die folgenden Funktionen :

- ▻ **`int zeichenKat(int c)`** **Ermittlung** der **Kategorie des Zeichens `c`.**
- ▻ **`int stateMachine(int old, int cat)`** **Ermittlung** des **neuen Textzustands** in Abhängigkeit vom alten Zustand **`old`** und der Zeichenkategorie **`cat`**
- ▻ **`void ausWL(int wla[], int len)`** **Ausgabe** der **Wortlängenverteilung** in die Standardausgabe. Die Wortlängenverteilung ist im Array `wla` der Größe `len` abgelegt

◈ Die Funktionalität der **`main()`–**Funktion wird dabei durch das folgende Struktogramm beschrieben :



- **Aufgaben :**

a) Beschreiben Sie die Übergänge zwischen den Textzuständen in Abhängigkeit von der Zeichenkategorie durch ein **Zustandsfolgediagramm**.

b) Beschreiben Sie den von der Funktion **`stateMachine()`** zu realisierenden Algorithmus durch ein **Struktogramm**

c) Formulieren Sie eine **C-Headerdatei `textanaly.h`,** die die **Definitionen** aller o.a. **symbolischen Konstanten** sowie die **Deklarationen** der drei o.a. **Funktionen** enthält.

d) **Definieren** Sie die drei Funktionen `zeichenKat()`, `stateMachine()` und `ausWL()` in einer eigenen C-Quelldatei (vorgeschlagener Name : `tafunc.c`)

e) **Formulieren** Sie die C-Quelldatei **`textanaly_m.c`** mit der **`main()`–**Funktion des Programms

Erzeugen Sie aus den drei o.a. Dateien ein **lauffähiges Programm** und führen Sie es vor !
Lesen Sie dabei Text aus den zur Verfügung gestellten **Testdateien** mittels **Eingabeumleitung** ein (Programmaufruf aus der Kommandozeile).