Programmieren C: Strings, "by Reference"-Parameter: Funktion um die Wörter im Input zu suchen

Klaus Kusche

Gesucht ist ein Programm, das *zeilenweise* Text vom DOS-Fenster liest
und die Worte in jeder gelesenen Zeile einzeln zeilenweise ausgibt.
Ein Wort ist dabei eine Folge von *Buchstaben und Ziffern*
(welche Funktion aus **`ctype.h`** verwendest du für diese Prüfung sinnvollerweise?).
Satz- und Sonderzeichen werden *nicht* ausgegeben.

Für die *Zerlegung in Worte* verwenden wir eine selbstgeschriebene *Funktion*:

- Sie wird mit *zwei Argumenten* aufgerufen:
  Dem *String*, in dem das nächste Wort gesucht werden soll, und der *Position*,
  ab der es gesucht werden soll. Die Position soll so übergeben werden,
  dass sie von der Funktion *geändert* werden kann.
- Die Funktion hat einen *Returnwert*: Die Position, an der das nächste Wort *beginnt*
  (die Position des ersten Buchstabens des Wortes).
- Der Startpositions-Parameter, der Returnwert und die Schleifenvariablen
  sind vorläufig **`int`**'s, den Umbau auf Pointer machen wir später.
- Weiters soll die Funktion *im Startpositions-Parameter*
  die erste Position *hinter* dem gefundenen Wort *speichern*
  (also die Position des ersten Zeichens, das *nicht mehr* zum Wort gehört).
  Das ist nämlich die Position, ab der der nächste Aufruf weitersucht.
- Intern besteht die Funktion aus *zwei aufeinanderfolgenden Schleifen*:
  - Die erste Schleife sucht ab der Startposition den *Beginn des Wortes*,
    d.h. den *ersten Buchstaben oder die erste Ziffer*. Die so gefundene Position
    wird am Ende der Funktion als *Returnwert* zurückgegeben.

    Trifft die Schleife auf die Ende-Markierung, *ohne* einen Wortanfang
    gefunden zu haben, wird als Returnwert **`-1`** zurückgeliefert und
    als neue Startposition die *Position der Ende-Markierung* gespeichert.
  - Die zweite Schleife beginnt ein Zeichen *hinter* dem oben gefundenen
    Wortanfang und sucht das erste Zeichen, das *nicht* Buchstabe oder Ziffer ist.
    Das ist die Position, die für die nächste Suche *im Startpositions-Parameter gespeichert* wird.

Das *Hauptprogramm* sieht wie folgt aus:

- Es deklariert einen String **`zeile`**, in dem eine Eingabe-Zeile Platz hat
  (z.B. 4096 Zeichen, Konstante deklarieren!).
- Es liest *in einer Schleife immer wieder eine Zeile* vom Terminal in diesen String.
  Das geht mit **`fgets(zeile, sizeof(zeile), stdin)`** .

  Wenn **`fgets`** als Ergebnis **`NULL`** liefert (bei Dateiende), soll die Schleife enden
  (im DOS kannst du mit **`Ctrl/Z`** das Dateiende eingeben, in Linux mit **`Ctrl/D`**).

- In dieser Schleife läuft eine *zweite Schleife*, die die *einzelnen Worte* der Zeile sucht und ausgibt:
    - Sie ruft zuerst einmal die oben beschriebene *Funktion* auf
      (beim *ersten* Aufruf für eine frisch gelesene Zeile mit Startposition 0,
      sonst mit der vom vorigen Aufruf gespeicherten Startposition).
    - Liefert die Funktion **`-1`**, so endet die Wort-Schleife:
      Die Zeilen-Schleife liest dann die nächste Zeile.
    - Gib sonst das gefundene Wort mit einer *dritten Schleife* *zeichenweise* aus
      (von der gefundenen Position bis ein Zeichen *vor der nächsten* Startposition).

      Zeichenweise Ausgabe am Terminal macht man mit **`putchar(…)`** .
      Auf das **`'\n'`** nach jedem Wort nicht vergessen!

Umbau auf Pointer

- Die Funktion bekommt statt zwei *nur mehr einen Parameter*:
  Er zeigt auf die Stelle, ab der die Funktion nach einem Wort suchen soll.

  Die Funktion *ändert* diesen Parameter: Nach dem Aufruf soll er
  auf *das erste Zeichen hinter dem gefundenen Wort* zeigen.

  Achtung: Wie muss die Deklaration des Parameters aussehen,
  wenn es ein *Pointer* ist, der *geändert* (d.h. *"by Reference"* übergeben) werden soll?
- Als *Returnwert* wird ein *Pointer* auf das erste Zeichen des gefundenen Wortes zurückgegeben (bzw. **`NULL`**, wenn kein Wort gefunden wurde).
- Auch die beiden *Schleifen* in der Funktion arbeiten mit Pointern
  statt mit Index-Werten.
- Aus den beiden *Positions-Variablen im Hauptprogramm*
  (Wort-Anfang und nächste Such-Position) werden Pointer
  (der *Anfangswert* für die Such-Position ist dann eben nicht Index 0,
  sondern ein *Pointer auf den Anfang* der gelesenen Zeile).
- Die *Schleife zur zeichenweisen Ausgabe* soll ebenfalls
  mit einem Pointer über die Buchstaben des Wortes wandern,
  der Pointer läuft von der gefundenen Position (Wort-Anfang)
  bis eins vor der nächsten Such-Position.

Explizite Trennzeichen

Die Festlegung auf *"Buchstabe oder Ziffer"* macht unsere Funktion unnötig spezifisch.
Man kann die Funktion *universeller* machen, indem man ihr entweder einen String mit *allen in einem Wort erlaubten Zeichen* übergibt oder einen String mit *allen Trennzeichen*.

Wir entscheiden uns für die zweite Variante: Unsere Wort-Such-Funktion bekommt einen *weiteren Parameter*, und zwar einen *String*, der *alle Zeichen* enthält, die *Worte trennen* (also *nicht* zu einem Wort gehören).

In den Schleifen der Funktion wird das aktuelle Zeichen statt auf Buchstabe oder Ziffer mit einer *vordefinierten Stringfunktion* (welcher?) geprüft,
ob es in diesem Trennzeichen-String *vorkommt oder nicht*.

Achtung:
Die *"Buchstabe oder Ziffer"*-Prüffunktion durfte man *auch mit der Ende-Markierung* aufrufen, sie lieferte *"nein"* für die Ende-Markierung.
Die neue Prüffunktion solltest Du *nicht* mit der Endemarkierung aufrufen (sie würde einen *Treffer* melden, aber das ist nicht jedem Programmierer bekannt). Du solltest daher deine Logik etwas umbauen und *vor jedem Aufruf* der Prüffunktion jedesmal *explizit auf die Ende-Markierung* prüfen.

String-Terminierung

Dass unser Hauptprogramm das Wort dann nochmals
mit einer Schleife Zeichen für Zeichen durchgehen und ausgeben muss,
bis die *Schleife bei der Ende-Position* ist, ist etwas unpraktisch.

Wenn wir unseren eingelesenen String *beim Durchgehen "zerstören" dürfen*,
ist eine *brutalere Variante* weit verbreitet: Unsere Wort-Finde-Funktion
***ersetzt das erste Zeichen hinter dem Wort durch eine Ende-Markierung*.**

Damit zeigt unser Returnwert-Pointer auf einen *String*, der *genau das nächste Wort enthält*. Dieser String kann ganz normal mit **`puts`** oder **`printf`** ausgegeben oder mit Stringfunktionen weiterverarbeitet werden.

Den *Pointer-Parameter zum Weitersuchen* für den nächsten Aufruf muss die Funktion dann natürlich *nicht* auf diese eingefügte Ende-Markierung zeigen lassen (außer es handelt sich um das "echte", schon vorher vorhandene String-Ende!), sondern auf *das nächste Zeichen dahinter*.

Bau die Funktion entsprechend um (falls Du deine Parameter usw. bisher brav **`const`** deklariert hast, wirst du einige **`const`** entfernen müssen),
und *ändere die Ausgabe* des Wortes im Hauptprogramm auf ein simples **`puts`**.