AIK Programmieren 1 Übung: "Händische" String-Verarbeitung

Klaus Kusche

1.) Wort erraten

Wir schreiben ein Programm, bei dem der Benutzer *buchstabenweise ein Wort erraten* muss (so wie früher im Fernsehen: "Ich möchte das '**e**' !").

Das Programm soll sich wie folgt verhalten:

- Als Erstes sucht sich das Programm *zufällig* aus einer *fixen Liste von Wörtern* das zu erratende Wort aus. Es ermittelt und speichert auch gleich dessen *Länge*.

  Erinnere dich: Was musst du tun, damit dein Programm bei jedem Aufruf ein *anderes* zufälliges Wort aus der Liste wählt?

  Verwende für die Wortliste ein *Array von Strings* (egal ob ein zweidimensionales **char**-Array oder ein Array von Zeigern auf **char** – beides funktioniert gleich, das zweite ist üblicher), das du gleich beim Anlegen *mit fixen Worten initialisierst*!

  Die *Anzahl* der Strings im Array, d.h. die Größe des Arrays (und damit den Bereich, in dem die Zufallszahl für die Auswahl des zu erratenen Wortes liegen muss), sollst du *berechnen* und *nicht* durch händisches Abzählen der Strings in der Initialisierungsliste ermitteln (welches Konstrukt verwendet man dafür?).
- Außer der Wortliste wirst du eine *String-Variable* für die *Eingabe des Benutzers* und eine zweite für das *angezeigte Wort* mit den '*' brauchen.

  Du darfst für beide eine fixe *Maximallänge* (Konstante definieren!) vorgeben.

  In dem String, den der Benutzer vom zu erratenden Wort angezeigt bekommt, wird am Anfang einmal ein String (was braucht der?) mit *lauter* '*' *gespeichert* (genau so viele, wie das zufällig ausgewählte, zu erratende Wort Buchstaben hat!).
- Dann passiert in einer *Schleife* immer wieder Folgendes:
  - Der Benutzer bekommt das angezeigt, was vom Wort schon bekannt ist (beim ersten Mal eben nur die '*').
  - Dann gibt er einen Text ein (nimm **scanf**, denn **fgets** hängt ein **'\n'** an). Das kann entweder ein *einzelner Buchstabe oder ein ganzes Wort* sein.
  - Hat er ein ganzes Wort (*mehr als 1 Zeichen* lang) eingegeben, so wird das eingegebene Wort mit dem zu erratenden Wort *verglichen*.
    - Ist es gleich, hat der Benutzer gewonnen.
    - Sonst wird "Falsch!" angezeigt, und der Benutzer muss den nächsten Versuch machen.
  - Hat er nur ein *einzelnes Zeichen* eingegeben, passiert Folgendes:
    - Das eingegebene Zeichen wird *mit jedem einzelnen Buchstaben* des zu erratenden Wortes verglichen (*Schleife*!).

An genau den Stellen, wo der eingegebene Buchstabe
im gesuchten Wort *vorkommt*, wird im angezeigten Wort
das '*' durch den richtigen Buchstaben *ersetzt*.

- Dann wird das angezeigte Wort mit dem gesuchten Wort *verglichen*.

  Falls es jetzt gleich ist (d.h. falls der Benutzer den letzten noch
  fehlenden Buchstaben erraten hat), hat der Benutzer auch gewonnen,
  sonst geht es normal mit dem nächsten Versuch weiter.

- Wenn der Benutzer das Wort erraten hat, soll angezeigt werden,
  wie viele Versuche der Benutzer gebraucht hat (jede Eingabe,
  egal ob ein Buchstabe oder ein ganzes Wort, egal ob richtig oder falsch,
  zählt als ein Versuch, also einfach in der Schleife mitzählen).

  Dann endet das Programm.

Hinweise:

- Für die *Länge* eines Strings und den *Vergleich* zweier Strings sollst du
  die entsprechenden *String-Funktionen* verwenden.

  Es ist sinnvoll, die Länge des zu erratenden Wortes nur einmal am Anfang
  auszurechnen und in einer Variable zu speichern.

- Zur Vereinfachung kümmern wir uns nicht darum, einen Großbuchstaben
  auch für den entsprechenden Kleinbuchstaben zu erkennen oder umgekehrt
  bzw. großschreibungsunabhängig zu vergleichen:
  Die Worte werden intern komplett klein geschrieben gespeichert,
  und es werden daher auch nur eingegebene Kleinbuchstaben als passend erkannt.

- Weiters verzichten wir zur Vereinfachung auf die Längenprüfung: Du darfst
  eine fixe Maximalgröße für das Eingabewort, die Worte in der Wortliste und das
  Ausgabewort vorgeben. Gibt der Benutzer mehr ein, darf das Programm abstürzen.

- *Achtung:* Auch wenn eine String-Variable nur ein einziges Zeichen enthält,
  ist sie trotzdem ein String und kein Einzelzeichen (**char**): Wenn man nur das eine
  Zeichen allein will, muss man eben das erste Zeichen des Strings auswählen!

2.) Text in Worte zerlegen

Gesucht ist ein Programm, das *zeilenweise* Text vom DOS-Fenster liest,
jede Zeile *in einzelne Worte zerlegt*, und die Worte 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 wieder 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. 82 Zeichen).
- 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 an 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* 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.