Programmieren C: “Händische” String-Verarbeitung: Wort erraten

Klaus Kusche

Wir schreiben ein Programm, bei dem der Benutzer *buchstabenweise ein Wort erraten* muss (so wie früher im Fernsehen: “Ich möchte das '**e**' !”).
Das Programm erwartet *keine* Angaben auf der Befehlszeile.

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. (was musst du tun, damit dein Programm bei jedem Aufruf ein *anderes* zufälliges Wort aus der Liste wählt?)
  Es ermittelt und speichert auch gleich die *Länge* des gewählten Wortes.

  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), solltest du *berechnen* und *nicht* durch händisches Abzählen der Strings in der Initialisierungsliste ermitteln (welches Konstrukt verwendet man dafür?).
- Außer der Wortliste wirst du *zwei String-Variablen* für die *Eingabe des Benutzers* und 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, werden am Anfang einmal *lauter* '*' *gespeichert*, und zwar genau so viele, wie das zufällig ausgewählte, zu erratende Wort Buchstaben hat.
  Was musst Du *nach* der *-Schleife mit diesem String noch tun?
- 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 '*' in der richtigen Länge).
  - Dann *gibt er etwas 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* (Schleife *verlassen*).
    - 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*.

- Danach 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 gewonnen (Schleife *verlassen*), sonst geht es normal mit dem nächsten Versuch weiter.

  Alternativ kannst du auch *prüfen*, ob das angezeigte Wort noch einen '*' *enthält* (wenn nicht, sind alle Buchstaben erraten).

- Nachdem der Benutzer das Wort erraten hat (also nach der großen Schleife), 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 vordefinierten *String-Funktionen* verwenden.

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

- Im ersten Versuch kümmern wir uns *nicht* darum, einen Großbuchstaben auch für den entsprechenden Kleinbuchstaben zu erkennen oder umgekehrt bzw. beim Vergleichen die Groß- und Kleinschreibung zu ignorieren:
  Die Worte in der Initialisierung der Liste werden *komplett klein geschrieben*, und es werden daher auch *nur eingegebene Kleinbuchstaben* als passend erkannt.

  Als ***Zusatzaufgabe*** kannst Du das Programm so umbauen, dass es *Groß- und Kleinschreibung ignoriert*:

  - Wenn ein *ganzes Wort* eingegeben wird:
    Es gibt eine *vordefinierte Funktion*, die *Großschreibungs-unabhängig vergleicht*.
    Achtung: Lies deren Doku genau, die Funktion braucht einen *eigenen Header*!
  - Bei der Eingabe eines *einzelnen Zeichens*:
    Verwandle zum Vergleich *beide Zeichen in Kleinbuchstaben*!

- 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 ein zu langes Wort 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!