Übungen zur Vorlesung
„Grundlagen der Informatik 1“
Studiengang E-Technik

WS 2001/02

AG Betriebssysteme, FB 3
Jan Bredereke

ausgegeben: 30.11.2001
Abgabe und Besprechung:
in der Übung am **6.12.2001**

Serie 4

Zeiger (Pointer) in C

Diese Aufgabenserie enthält viele Aufgaben, damit viele Teilnehmer vorrechnen können. Falls ihr nicht alle Aufgaben lösen wollt, dann laßt einige der Aufgaben 1 bis 5 fort, aber nicht 6. Aufgabe 6 ist besonders interessant, weil sie ein gutes Beispiel dafür ist, wie man Zeiger auf Arrays einsetzen kann.

Aufgabe 1: Eine selbstgemachte `strcpy()`-Funktion

Es gibt in der C-Bibliothek eine Funktion `strcpy()`, mit der man einen String kopieren kann. Sie ist etwa folgendermaßen deklariert:

```
char *strcpy(char *dest, char *src);
```

Die Funktion kopiert die Zeichenkette, auf die der Zeiger `src` zeigt, inklusive des Endezeichens „`\0`“ an die Stelle, auf die `dest` zeigt. Die Zeichenketten dürfen sich nicht überlappen und `dest` muß groß genug sein. Die Funktion `strcpy()` gibt einen Zeiger auf `dest` zurück.

Programmiert eine Funktion `mystrcpy()`, die das gleiche tut.

Aufgabe 2: Eine selbstgemachte `strcat()`-Funktion

In der C-Bibliothek gibt es eine ähnliche Funktion `strcat()`. Die Funktion `strcat()` hängt die Zeichenkette `src` an die Zeichenkette `dest` an, wobei das Stringendezeichen „`\0`“ überschrieben wird und ein neues „`\0`“ am Ende der gesamten Zeichenkette angehängt wird. Die Zeichenketten können sich nicht überlappen und `dest` muß Platz genug für die gesamte Zeichenkette haben. Die Funktion `strcat()` liefert einen Zeiger auf die gesamte Zeichenkette `dest` zurück.

Programmiert eine Funktion `mystrcat()`, die das gleiche tut.

Aufgabe 3: Eine selbstgemachte `strchr()`-Funktion

Es gibt in der C-Bibliothek eine Funktion `strchr()`, mit der man einen Buchstaben in einem String suchen kann. Sie ist etwa folgendermaßen deklariert:

```
char *strchr(char *s, int c);
```

Die Funktion `strchr()` gibt den Zeiger auf das erste Vorkommen des Zeichens `c` in der Zeichenkette `s` zurück, ansonsten `NULL`.

Die Konstante `NULL` könnt ihr benutzen, wenn ich vorher schreibt:

```
#include <string.h>
```

Programmiert eine Funktion `mystrchr()`, die das gleiche tut.

Aufgabe 4: Eine selbstgemachte `strrchr()`-Funktion

In der C-Bibliothek gibt es eine ähnliche Funktion `strrchr()`. Die Funktion `strrchr()` gibt den Zeiger auf das *letzte* Vorkommen des Zeichens `c` in der Zeichenkette `s` zurück, ansonsten `NULL`.

Programmiert eine Funktion `mystrrchr()`, die das gleiche tut.

Aufgabe 5: Eine selbstgemachte `strstr()`-Funktion

Es gibt in der C-Bibliothek eine Funktion `strstr()`, mit der man einen Teilstring in einem String suchen kann. Sie ist etwa folgendermaßen deklariert:

```
char *strstr(char *haystack, char *needle);
```

Die Funktion `strstr()` findet die erste Position der Zeichenfolge `needle` im String **haystack**. Abschließende „\0“-Zeichen werden nicht miteinander verglichen. Die Funktion `strstr()` gibt einen Zeiger auf den Anfang der gefundenen Zeichenkette in `haystack` zurück oder `NULL`, wenn die Zeichenkette nicht gefunden wurde.

Programmiert eine Funktion `mystrstr()`, die das gleiche tut.

Aufgabe 6: Einlesen und Vergleichen von Werten

Schreibt ein Programm, das vom Benutzer nacheinander zehn Integer-Zahlen abfragt. Nachdem es das getan hat, soll das Programm erneut zehn Zahlen abfragen. Danach soll es ausgeben, welche Zahlen sich gegenüber der vorigen Runde geändert haben. Wenn also beim ersten Mal (7, 9, -3, 5, 5, 42, 0, 1, -100, 2) eingegeben wurden, und beim zweiten Mal (7, 11, -3, 5, 5, -1, 0, 1, -100, 2), dann soll ausgegeben werden, daß sich an der 2. und an der 6. Stelle etwas geändert hat.

Damit das Programm leichter zu schreiben ist, sollen auch nach der ersten Runde bereits die „Unterschiede“ ausgegeben werden, und zwar im Vergleich zu einer Serie von zehnmal 0. Nach der Ausgabe der Unterschiede soll das Programm fragen, ob noch eine Runde durchgeführt werden soll. Je nach Antwort des Benutzers sollen beliebig viele Runden möglich sein. Wenn die Antwort „j“ ist, gilt das als „Ja“, alles andere als „Nein“.

Verwendet zum Einlesen der Zahlen und des Antwortbuchstabens die Funktion `scanf()`, die in der Vorlesung eingeführt wurde.

Löst diese Aufgabe so, daß ihr zwei Arrays von je zehn Integern anlegt, ein Eingabe-Array und ein Vergleichs-Array. Das Vergleichs-Array belegt ihr entsprechend mit Nullen vor. Nach Ende einer Runde tauschen die beiden Arrays ihre Rollen. Löst dieses Vertauschen auf folgende Weise: Legt die Eingaben nicht direkt im Eingabe-Array ab und vergleicht nicht direkt mit dem Vergleichs-Array, sondern verwendet jeweils eine Pointer-Variable, die auf eines dieser Arrays zeigt. Dann könnt ihr nach Ende einer Runde einfach die Inhalte der beiden Pointer-Variablen vertauschen (mit Hilfe einer dritten Pointer-Variablen und entsprechenden Zuweisungen), so daß das Array mit den alten Eingaben zum Vergleichs-Array wird, während das bisherige Vergleichs-Array zum neuen Eingabe-Array wird.