

Programmiertechniken 1

Strings

Strings (Text)


Text wird in C als Zeichenkette (**String**) aus einzelnen Zeichen (**Character**) gehandhabt. Variablen, die Strings speichern können, sind vom Typ eines **Character-Arrays**.

Beispiel:

```
char name[] = "Thomas";
```



mehrere



Initialisierung

Die Klammer gibt an, dass eine unbestimmte Anzahl von char-Speicherplätzen unter dem Namen „name“ referenziert werden sollen. Die genaue Anzahl wird aus der Initialisierung bestimmt.

Zugriff auf Zeichen im String

Mittels des Klammer-Operators ([]) kann auf einzelne Zeichen im String zugegriffen werden.

Achtung: Zählung beginnt bei 0!

```
char name[] = "Thomas!";
```

```
char single = ' ';
```

```
single = name[1];
```

Single hat nun den Wert ,h‘.

```
name[6] = '?';
```

→ „Thomas?“

Aufgabe

- Definieren Sie eine Variable für einen String und initialisieren Sie diese. Geben Sie mittels einer Schleife die Zeichen einzeln aus.
- Was passiert, wenn der Zähler im negativen Bereich startet oder mehr Zeichen ausgeben will, als der String lang ist?

Null-Terminierung

Bei Strings muss die Länge bekannt sein oder der String muss **Null-Terminiert** '\0' sein.

```
char name[] = "Prof. Lehmann";
```

Konstanter String
automatisch Null-
Terminiert. → 14
Elemente!

Strings sollten immer Null-Termiert sein.

Ausgabe von Strings

Die Ausgabe von Strings mittels `printf()` wird durch das Formatierungssymbol `%s` signalisiert.

```
char name[] = "Prof. Lehmann";  
printf("Name: %s", name);
```

Achtung: `printf()` gibt den String bis zum Erreichen der Null-Terminierung aus!

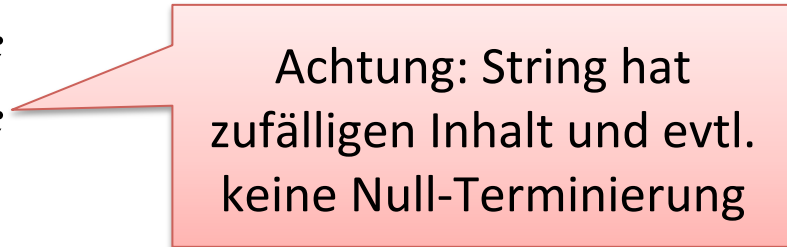
Aufgabe

Erstellen Sie einen String (auf dem Stack) und initialisieren Sie diesen. Geben Sie den String mit `printf()` aus. Schreiben Sie an eine beliebige Stelle im String den Wert Null. Geben Sie den String mit `printf()` aus. Was passiert bei der Ausführung?

String-Buffer

String-Variablen können auch mit einer Länge größer der Initialisierung angelegt werden. Hiermit können Speicherbereiche für die spätere Aufnahme von Strings (**String-Buffer**) reserviert werden.

```
char name[20] = "Thomas";  
char buffer[BUFFER_SIZE];
```



Achtung: String hat zufälligen Inhalt und evtl. keine Null-Terminierung

Hinweis: Die Größe des Buffers wird oft mehrmals im Programm benötigt. Somit sollte man diese als Konstante definieren.

Aufgabe

Legen Sie einen String und einen passenden Buffer an. Kopieren Sie den String in den Buffer und ersetzen Sie dabei alle Großbuchstaben durch Kleinbuchstaben (→ ASCII-Tabelle beachten).

Aufgabe Buffer-Overflow

Erstellen Sie zwei Strings (auf dem Stack) und initialisieren Sie diese. Der erste String sollte mindestens 10 Zeichen beinhalten. Der zweite String sollte genau drei Zeichen enthalten. Das Programm soll nun folgendes tun:

- An die erste Stelle des zweiten Strings wird ein ‚A‘ geschrieben. Der String wird ausgegeben.
- An die zweite Stelle des zweiten Strings wird ein ‚B‘ geschrieben. Der String wird ausgegeben.
- An die dritte Stelle des zweiten Strings wird ein ‚C‘ geschrieben. Der String wird ausgegeben.
- ...

Hören Sie bei ‚F‘ auf.

Was beobachten Sie bei der Ausführung?

Glossar

Begriff	Im Buch	Englisch
Klammer-Operator	Arrayelement-Operator	Index-Operator
String	String	String
Null-Terminierung	'\0'- terminiert	Zero-Terminated