
Zeichenketten (Strings) in C

Strings werden in C als Arrays vom Typ `char` verwaltet. In C sind alle Strings nullterminiert, d.h. das letzte Zeichen in einem String ist immer die Ordinalzahl 0. Um die folgenden Stringfunktionen anzuwenden, muss die Datei **String.h** in das Programm eingebunden werden: `#include <string.h>`

Definition eines Strings

Strings werden als Arrays vom Typ `char` definiert. Hier einige Beispiele für solche Definitionen:

Normale Array-Definition mit 20 Elementen:

```
char MyStri[20];
```

Länge wird automatisch durch C bestimmt und eine Zeichenkette zugeordnet:

```
char Satz[]="Hallo Welt";
```

Hier zeigt Tag[2] auf Mi:

```
char *Tag[]={ "Mo", "Di", "Mi", "Do" };
```

Bemerkung: Der Arrayname (String) ist in C immer ein Zeiger. Die meisten String-Funktionen erwarten auch Zeiger als Argumente. Für Zeiger werden zwei Operatoren verwendet:

& Liefert die Adresse (Zeiger) eines Objekts
* Liefert den Inhalt eines Objekts

String Funktionen in Standard C

```
char *strcat(char *s1, char *s2)
```

hängt den String s2 an den String s1 an. s1 wird zurückgegeben.

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

sucht nach dem ersten Auftreten des Zeichens c im String s (einschliesslich des abschliessenden Null-Zeichens). Die Funktion gibt die Adresse der ersten gefundenen Übereinstimmung zurück, oder aber Null, wenn keine Übereinstimmung festgestellt wurde.

```
int strcmp(char *s1, char *s2)
```

vergleicht zwei Strings miteinander. Bei Übereinstimmung wird Null zurückgegeben. Ein Wert grösser Null wird zurückgegeben, wenn das erste abweichende Zeichen in s1 grösser als das korrespondierende Zeichen in s2 ist. Im umgekehrten Fall wird entsprechend ein Wert kleiner Null zurückgegeben.

```
char *strcpy(char *s1, char *s2)
```

kopiert den String s2 nach s1. s1 wird zurückgegeben.

```
size_t strcspn(char *s1, char *s2)
```

sucht nach dem ersten Zeichen in s1, das mit einem beliebigen Zeichen in s2 übereinstimmt. Die abschliessenden Null-Zeichen werden als Bestandteil der Strings erachtet. Es wird der Offset zum Beginn von s1 zurückgeliefert, an dem die Übereinstimmung gefunden wurde.

```
size_t strlen(char *s)
```

liefert die Länge von s.

```
char *strncat(char *s1, char *s2, size_t n)
```

Hängt bis zu n Zeichen aus s2 (die abschliessende Null von s2 nicht mitgezählt) sowie ein terminierendes Null-Zeichen an s1 an. s1 wird zurückgegeben.

```
int strncmp(char *s1, char *s2, size_t n)
```

verhält sich wie strcmp(), es werden jedoch maximal n Zeichen verglichen.

```
char *strncpy(char *s1, char *s2, size_t n)
```

verhält sich wie strcpy(), es werden jedoch maximal n Zeichen kopiert.

```
char *strpbrk(char *s1, char *s2)
```

verhält sich wie strcspn() mit dem Unterschied, dass ein Zeiger auf das übereinstimmende Zeichen nur dann zurückgegeben wird, wenn es sich nicht um die abschliessende Null handelt. Ansonsten wird ein Null-Pointer zurückgegeben.

```
char *strrchr(char *s, int c)
```

sucht nach dem letzten Auftreten von c im String s und gibt einen Zeiger auf diese Stelle zurück. Wird keine Übereinstimmung gefunden, wird ein Null-Pointer zurückgegeben.

```
size_t strspn(char *s1, char *s2)
```

sucht nach dem ersten Zeichen in s1, das nicht mit einem beliebigen Zeichen aus s2 übereinstimmt. Die abschliessende Null von s2 wird als Teil von s2 angesehen. Es wird der Offset zum Beginn von s1 zurückgegeben, bei dem die Bedingung zutrifft.

```
char *strstr(char *s1, char *s2)
```

sucht nach dem ersten Auftreten der Zeichenkette s2 innerhalb der Zeichenkette s1. Zurückgeliefert wird die Anfangsadresse der gefundenen Übereinstimmung, andernfalls Null.