

- Was ist ein String
- String Operationen
- Analysieren von Strings
- Übung

Strings and Text

String

```
#include <stdio.h>

void main()
{
    printf("This is a string.");
    printf("This is on\n two lines!");
    printf("For \" you write \\\".");
}

>>
This is a string.This is on
two lines!For " you write \".
```

Stringvariablen:

```
char saying[20];
char saying[] = "This is a string.";
char str[40] = "To be";
const char message[] = "The end of the world is nigh.";

printf("\nThe message is: %s", message);
```

String – endet mit \0

"This is a string."

T	h	i	s		i	s		a		s	t	r	i	n	g	.	\0
84	104	105	115	32	105	115	32	97	32	115	116	114	105	110	103	46	0

"This is on\ntwo lines."

T	h	i	s		i	s		o	n	\n	t	w	o		i	i	n	e	s	.	\0
84	104	105	115	32	105	115	32	111	110	10	116	119	111	32	108	105	110	101	115	46	0

"For \" you write \\\"."

F	o	r		"		w	r	i	t	e		\	"	.	\0
70	111	114	32	34	32	119	114	105	116	101	32	92	34	46	0

Array of Strings

```
char sayings[3][32] = {  
    "Manners maketh man.",  
    "Many hands make light work.",  
    "Too many cooks spoil the broth."  
};
```

```
char sayings[][32] = {  
    "Manners maketh man.",  
    "Many hands make light work.",  
    "Too many cooks spoil the broth."  
};
```

```
for(unsigned int i = 0 ; i < sizeof(sayings)/  
    sizeof(sayings[0]) ; ++i)  
    printf("%s\n", sayings[i]);
```

Stringoperationen - strlen

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

void main() {

    char str[][70] = {
        "Computers do what you tell them to do, not what you want them to do.",
        "When you put something in memory, remember where you put it.",
        "Never test for a condition you don't know what to do with.",
    };

    unsigned int strCount = sizeof(str)/sizeof(str[0]); // Number of strings

    for(unsigned int i = 0 ; i < strCount ; ++i) {
        printf("The string:\n \"%s\"\n contains %zu characters.\n",
            str[i], strlen(str[i]));
    }
}

>>
The string:
"Computers do what you tell them to do, not what you want them to do."
contains 68 characters.
The string:
"When you put something in memory, remember where you put it."
contains 60 characters.
The string:
"Never test for a condition you don't know what to do with."
contains 58 characters.
```

Stringoperationen – strcpy / strncpy

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

void main()
{
    char source[] = "Only the mediocre are always at their best.";
    char destination[50];
    strcpy(destination, source);
    printf("The copied string is\n%s", destination);

    char source2[] = "Only the mediocre are always at their best.";
    char destination2[50];
    strncpy(destination2, source2, 17);
    destination2[17] = '\0';
    printf("\n\nThe copied string is\n%s", destination2);
}

>>
The copied string is
Only the mediocre are always at their best.

The copied string is
Only the mediocre
```

Stringoperationen – strcat / strncat

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

void main()
{
    char str1[50] = "To be, or not to be, ";
    char str2[] = "that is the question.";
    strcat(str1, str2);
    printf("The combined strings:\n%s\n", str1);

    char str12[50] = "To be, or not to be, ";
    char str22[] = "that is the question.";
    strncat(str12, str22, 4);
    printf("\n\nThe combined strings:\n%s\n", str12);
}

>>
The combined strings:
To be, or not to be, that is the question.

The combined strings:
To be, or not to be, that
```

Stringoperationen – strcmp / strncmp

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

void main()
{
    char str1[] = "The quick brown fox";
    char str2[] = "The quick black fox";
    if(strcmp(str1, str2) > 0)
        printf("str1 is greater than str2.\n");

    if(strncmp(str1, str2, 10) <= 0)
        printf("\n%s\n%s", str1, str2);
    else
        printf("\n%s\n%s", str2, str1);
}

>>
str1 is greater than str2.

The quick brown fox
The quick black fox
```


Stringoperationen – strchr

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

void main()
{
    char str[] = "Peter piper picked a peck of pickled pepper."; // The string to be searched
    char ch = 'p'; // The character we are looking for
    char *pGot_char = str; // Pointer initialized to string start
    int count = 0; // Number of times found
    while(pGot_char = strchr(pGot_char, ch)) // As long as NULL is not returned...
    { // ...continue the loop.
        ++count; // Increment the count
        ++pGot_char; // Move to next character address
    }
    printf("The character '%c' was found %d times in the following string:\n\"%s\"\n",
        ch, count, str);

    // Search str1 for the occurrence of str2
    char str1[] = "This string contains the holy grail.";
    char str2[] = "the holy grail";

    if(strstr(str1, str2))
        printf("\n\"%s\" was found in \"%s\"\n", str2, str1);
}

>>
The character 'p' was found 8 times in the following string:
"Peter piper picked a peck of pickled pepper."

"the holy grail" was found in "This string contains the holy grail."
```

Stringoperationen – strtok

?

Reading Strings – gets / fgets

?

Strings analysieren

```
#include <stdio.h>
#include <ctype.h>

// islower() Lowercase letter
// isupper() Uppercase letter
// isalpha() Uppercase or lowercase letter
// isalnum() Uppercase or lowercase letter or a digit
// iscntrl() Control character
// isprint() Any printing character including space
// isgraph() Any printing character except space
// isdigit() Decimal digit ('0' to '9')
// isxdigit() Hexadecimal digit ('0' to '9', 'A' to 'F', 'a' to 'f')
// isblank() Standard blank characters (space, '\t')
// isspace() Whitespace character (space, '\n', '\t', '\v', '\r', '\f')
// ispunct() Printing character for which isspace() and isalnum() return false

void main()
{
    const char message[] = "The quick brown fox. 0123?!";
    int nLetters = 0;
    int nDigits = 0;
    int nPunct = 0;
    size_t i = 0;

    while(message[i])
    {
        if(isalpha(message[i]))
            ++nLetters;
        else if(isdigit(message[i]))
            ++nDigits;
        else if(ispunct(message[i]))
            ++nPunct;
        ++i;
    }
    printf("\nDie Message enthaelt %d Letters, %d Zahlen, %d Satzzeichen", nLetters, nDigits, nPunct);
}

>>
Die Message enthaelt 16 Letters, 4 Zahlen, 3 Satzzeichen
```

Strings umwandeln – toupper / tolower

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

void main() {
    char text[100];
    char substring[40];
    size_t text_len = sizeof(text);
    size_t substring_len = sizeof(substring);

    printf("Geben sie einen Text ein (weniger als %d Buchstaben):\n", text_len);
    gets(text);
    printf("Geben sie einen Substring ein (wenige als %d Buchstaben):\n", substring_len);
    gets(substring);

    for(int i=0; (text[i] = (char)toupper(text[i])) != '\0'; i++);
    for(int i=0; (substring[i] = (char)toupper(substring[i])) != '\0'; i++);

    printf("Der Substring %s gefunden.\n", ((strstr(text, substring) == NULL) ? "wurde nicht" : "wurde"));
}

>>
Geben sie einen Text ein (weniger als 100 Buchstaben):
The quick brown fox.
Geben sie einen Substring ein (wenige als 40 Buchstaben):
brown
Der Substring wurde gefunden.
```

Strings umwandeln – atof... / strtod...

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

// atof()      Gibt einen Wert vom Typ double von einem String zueuck
// atoi()      Gibt einen Wert vom Typ int von einem String zurueck
// atol()      Gibt einen Wert vom Typ long von einem String zurueck
// atoll()     Gibt einen Wert vom Typ long long von einem String zurueck

// strtod()    Gibt einen oder mehrere Werte vom Typ double von einem String zurueck
// strtof()    Gibt einen oder mehrere Werte vom Typ float von einem String zurueck
// strtold()   Gibt einen oder mehrere Werte vom Typ long double von einem String zurueck
// strtoll()   Gibt einen oder mehrere Werte vom Typ long long von einem String zurueck
// strtol()    Gibt einen oder mehrere Werte vom Typ long von einem String zurueck
// strtoull()  Gibt einen oder mehrere Werte vom Typ unsigned long long von einem String zurueck

void main() {
    double value = 0;
    char str[] = "3.5 2.5 1.26";           // The string to be converted
    char *pstr = str;                     // Pointer to the string to be converted
    char *ptr = NULL;                     // Pointer to character position after conversion
    while(true)
    {
        value = strtod(pstr, &ptr); // Convert starting at pstr
        if(pstr == ptr)             // pstr stored if no conversion...
            break;                  // ...so we are done
        else
        {
            printf(" %f", value);    // Output the resultant value
            pstr = ptr;              // Store start for next conversion
        }
    }
}

>>
3.500000 2.500000 1.260000
```

Übung:

- Schreiben Sie ein Programm, das ein Text von der Tastatur einliest und die Häufigkeit aller Wörter ausgibt.
 - Die Gross- und Kleinschreibung soll nicht beachtet werden.
 - Der eingegeben Text kann innerhalb einer maximalen Länge, beliebig gross sein.