Hochschule **Rosenheim**
University of Applied Sciences

Prozedurale Programmierung

Zeichenketten

Hochschule Rosenheim - University of Applied Sciences

WS 2018/19

Prof. Dr. F.J. Schmitt

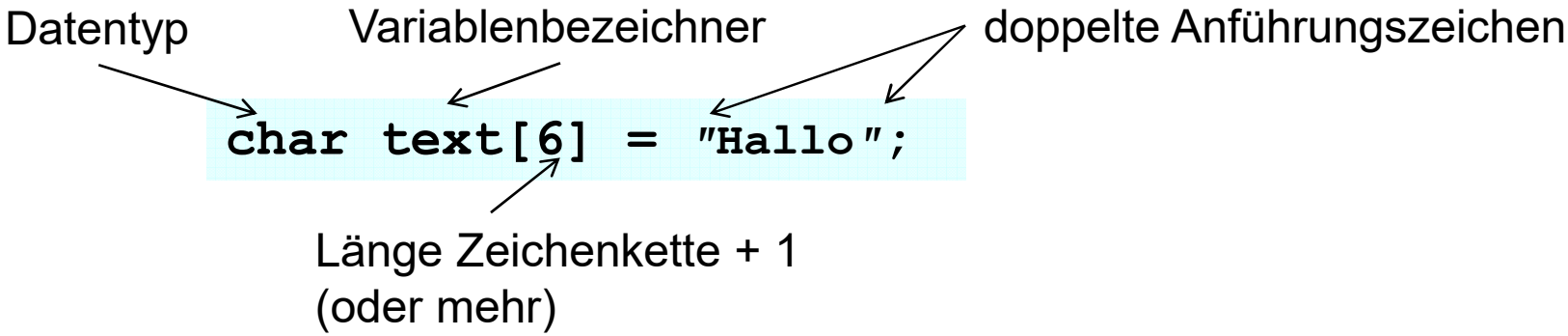
Überblick

- Einführung
- Verwendung von Zeichenketten
- Elementare Funktionen für Zeichenketten
- Felder von Zeigern auf Zeichenketten
- Argumente der Funktion `main`

Erinnerung: Zeichenketten

- sind in C kein elementarer Datentyp
- sondern ein Feld von Zeichen
  - Typ: char

- Definition mit Initialisierung



Achtung: Zuweisung mit = geht nur direkt bei der Initialisierung

Erinnerung: Zeichenketten

- folgendes geht **nicht**:

```
char text[6];
text = "Hallo";
```

- **richtig** wäre:

```
char text[6];
strcpy(text, "Hallo");
```

Achtung:
- es ist immens wichtig, dass die Längenangaben stimmen
- sonst: Programmabstürze / seltsames Verhalten
- generell ist strcpy eine unsichere Funktion (Buffer Overflow) genau wie viele andere Stringfunktionen

Literale von Zeichenketten

- Zeichenfolgen, die in doppelte Anführungszeichen gesetzt sind
- Beispiel:

```
"Hallo Welt!"
```

- Mehrzeilige Zeichenketten mit printf():

```
printf(" %s %s ","Dieser Text \n",
                 "geht über zwei Zeilen\n");
```

Speicherung

- Zeichenketten sind nullterminierte Folgen von Zeichen, die in Feldern von Zeichen abgespeichert sind
  - Interne Speicherung der Zeichenkette "Hallo"

'H'	'a'	'l'	'l'	'o'	'\0'

    - Zeichenkette wird mit Null-Byte (ASCII-Code 0 => Zahl 0) abgeschlossen
    - Zeichenkette, die in doppelten Anführungszeichen steht, wird automatisch mit Null-Byte ergänzt
    - Text "Hallo" ist nur 5 Zeichen lang – Zeichenkette benötigt aber 6 Zeichen Speicherplatz

Nullterminiertheit

- Konvention in C

'H'	'a'	'l'	'l'	'o'	'\0'

- dient sämtlichen Funktionen für Zeichenketten als **Markierung des Textendes**
- Null-Byte darf nicht weggelassen werden
- fehlt es kann es zu fatalen Programmabstürzen kommen
- Bsp: Ausgabe von Text (Ausgabe von Zeichen bis ein Null-Byte gelesen wird)

```
printf("Hallo Welt!");
```

Ausgabe: Hallo Welt!

```
printf("Hallo\0 Welt!");
```

Ausgabe: Hallo

Nullterminiertheit

- Vorteil Nullterminiertheit:
  - Beliebig lange Texte können abgespeichert werden
  - Text endet mit Null-Byte
- Nachteil Nullterminiertheit:
  - Textlänge steht im Vorhinein nicht fest
  - Bestimmung der Länge $\Rightarrow$ alle Zeichen bis zum Null-Byte müssen gelesen und gezählt werden
    (Null-Byte wird nicht mitgezählt!)

Verwendung von Zeichenketten (1)

- Definition
  - Feld von Zeichen anlegen (auf richtige Länge achten!)
  - Speichern des Textes "Hallo"

```
char text[6];
```

- Initialisierung

```
char text[6] = {'H', 'a', 'l','l', 'o', '\0'};
```

ist äquivalent zu

```
char text[6] = "Hallo";
```

Verwendung von Zeichenketten (2)

- Bei Initialisierung kann die Feldlänge weggelassen werden – sie wird dann vom Compiler ermittelt:

```
char text[] = "Hallo";
```

  - Kann problematisch sein ⇒ Vorsicht!

```
strcpy(text, "Hallo Welt!");
```

Verletzung der Feldgrenzen!

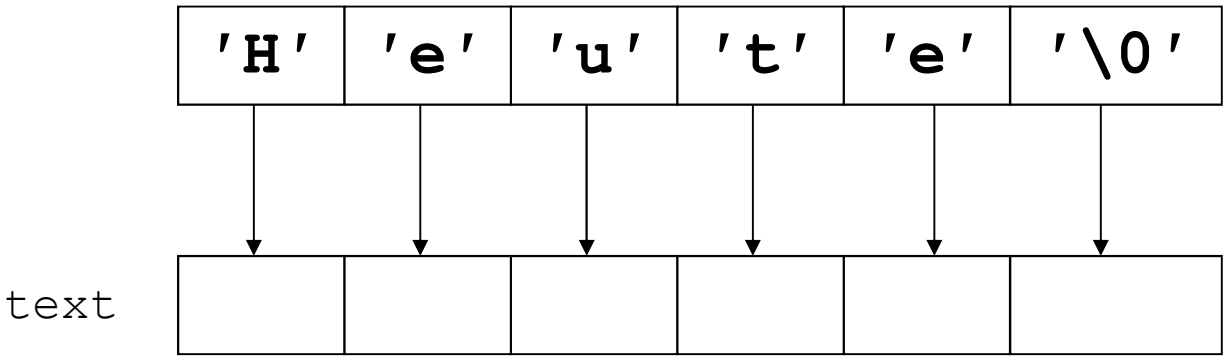
  - Bei konstanten Zeichenketten unproblematisch

```
const char text[] = "Hallo";
```

Verwendung von Zeichenketten (3)

- Zuweisungen nach der Initialisierung
  - Einfache **Zuweisung mit „=“ nicht möglich**
  - Funktion `strcpy` muss verwendet werden
    - Zeichenweise Übertragung der Zeichenkette in das Feld erfolgt

```
strcpy(text, "Heute");
```

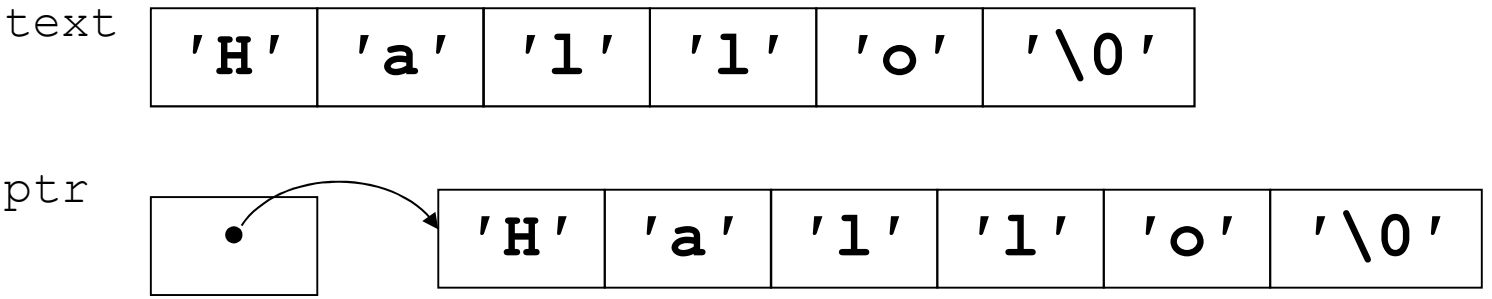


Verwendung von Zeichenketten (4)

- Unterschied zwischen Zeigern und Feldern:
  - Zeiger dienen zur Speicherung von Adressen – sie können nicht zur Speicherung von Daten verwendet werden
  - Felder dienen zur Speicherung von Daten

```
char text[] = "Hallo";
char *ptr   = "Hallo";
```

schlecht: Zeiger auf konstante Zeichenketten! → verwende const



Wird der Zeiger verändert, kann nicht mehr auf den Text zugegriffen werden!

Elementare Funktionen für Zeichenketten

Funktion	Beschreibung
`strcat(s,t)`	hängt t an s an
`strncat(s,t,n)`	hängt die ersten n Zeichen von t an s an
`strcmp(s,t)`	vergleicht s und t zeichenweise und liefert negative Zahl (s<t), 0 (s==t) oder positive Zahl (s>t)
`strncmp(s,t,n)`	wie strcmp, vergleicht jedoch nur die ersten n Zeichen
`strcpy(s,t)`	kopiert t nach s
`strncpy(s,t,n)`	wie strcpy, kopiert jedoch nur die ersten n Zeichen
`strlen(s)`	liefert die Länge von s ohne Null-Byte

`string.h` muss inkludiert werden

Ausgabefunktionen

- Ausgabefunktionen für Zeichenketten:

Funktion	Beschreibung
`printf(f, ...)`	Gibt eine formatierte Zeichenkette aus
`snprintf(s,n,f, ...)`	wie `printf`, schreibt den Text allerdings in das Feld `s`, wobei nicht mehr als `n` Zeichen geschrieben werden

  - `f` symbolisiert Zeichenkette, wobei mit Platzhaltern ein Format angegeben wird (wie gehabt, %s für Strings)
  - `stdio.h` muss inkludiert werden

Aufgabe

- Schreiben Sie ein C-Programm, in dem Sie drei Variablen für Zeichenketten definieren.
- Initialisieren Sie Variable 1 mit Ihrem Vornamen und Variable 2 mit Ihrem Nachnamen.
- Fügen Sie anschließend Vornamen und Nachnamen mit einem Leerzeichen dazwischen zusammen und speichern Sie das Ergebnis in Variable 3.
- Geben Sie anschließend den Inhalt von Variable 3 mit Angabe der Anzahl von Zeichen aus.

Umwandlung Zeichenkette → Zahl

- Typumwandlung: Funktion `atol(s)` wandelt die Zeichenkette s in ein `long` um und liefert dieses zurück
  - Header-Datei `stdlib.h` muss inkludiert werden

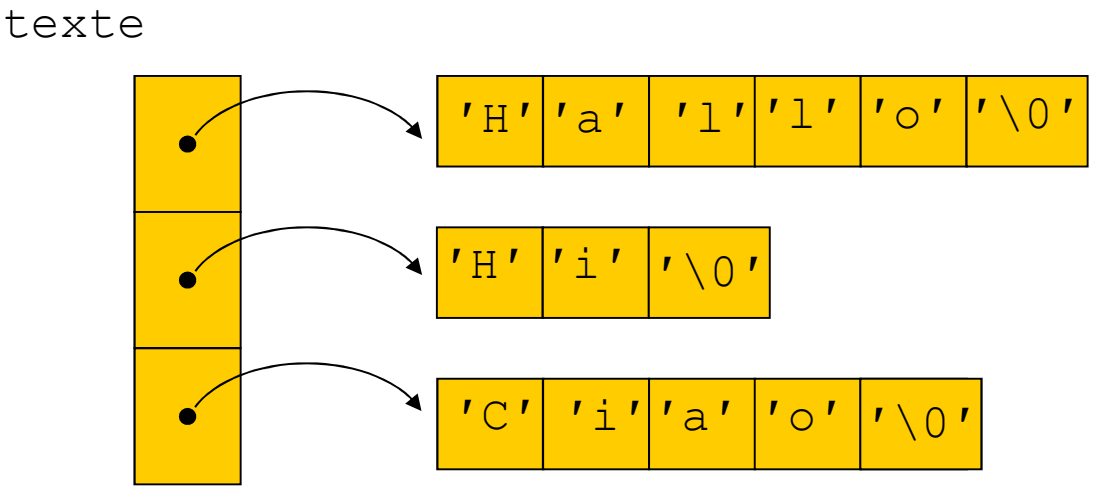
```
#include <stdlib.h>

int main()
{
  long i;
  char text[3] = "42";

  i = atol(text);
  //...
  return 0;
}
```

Felder von Zeigern auf Zeichenketten (1)

- Häufiges Anwendungsgebiet: Verwaltung von mehreren Zeichenketten über ein Feld von Zeigern
- Beispiel:



Felder von Zeigern auf Zeichenketten (2)

```
#include <stdio.h>
int main()
{
  char *texte[] =
   { "Hallo",
     "Hi",
     "Ciao"
   };
  int Texte_Len = sizeof(texte) / sizeof(char *);

  int i;

  for (i = 0; i < Texte_Len; i++)
    printf("%s\n", texte[i]);

  //...
}
```

Argumente der Funktion `main` (1)

- Es ist möglich bei Programmstart Argumente oder Optionen anzugeben, die den Programmablauf beeinflussen
  - diese werden als Zeichenketten gespeichert und an die Funktion `main` übergeben
  - Bisher weggelassen
  - Vollständiger Funktionskopf der `main`-Funktion:

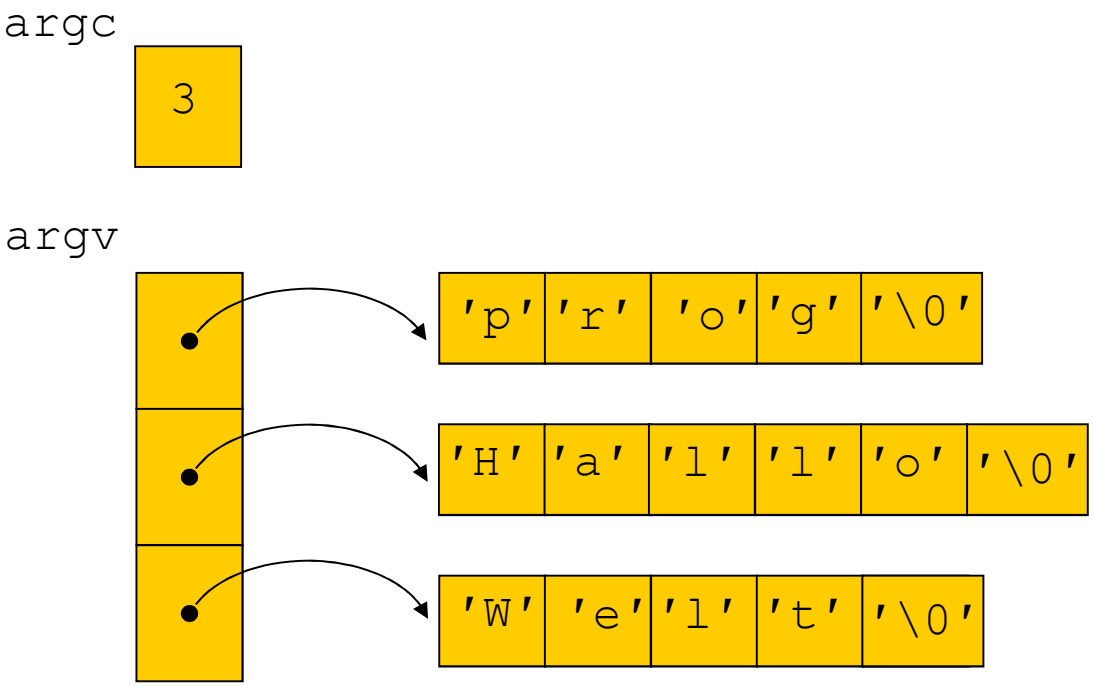
```
int main(int argc, char *argv[]);
```

    - `argv` (engl. argument value): Feld von Zeigern auf Zeichenketten – erste Zeichenkette = Programmname
    - `argc` (engl. argument counter): Länge des Feldes `argv`

Argumente der Funktion `main` (2)

Beispiel:

- Programm heißt `prog` und wird in der Eingabeaufforderung mit `prog Hallo Welt` aufgerufen



Argumente der Funktion `main` (3)

Beispiel:

- Ausgabe des Programmnamens und aller Argumente

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i;

  // Ausgabe des Programmnamens und aller Argumente
  for (i = 0; i < argc; i++)
    printf("%s\n", argv[i]);

  return 0;
}
```

Aufgabe

- Schreiben Sie ein C-Programm, in dem der Text ″Hallo Welt!″ in einem Feld abgelegt wird. Überschreiben Sie ein Zeichen des Feldes mit dem Null-Byte und geben Sie die Zeichenkette wieder aus.

Zusammenfassung

- Zeichenketten als Felder von char
- Nullterminierung
- Argumente von main()