

# Einführung in die Programmierung

## **Grundbegriffe**

Prof. Dr. Peter Jüttner

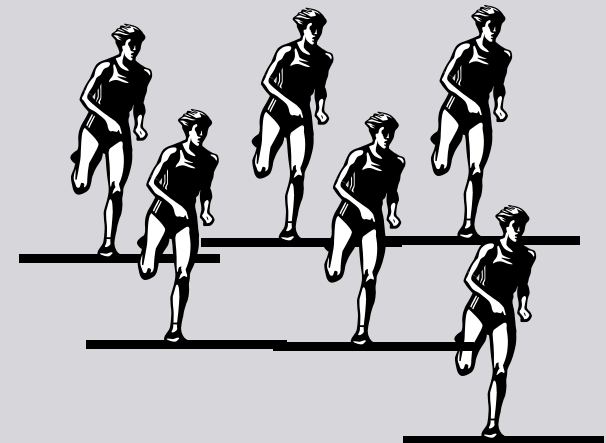
# Inhalt

- **Grundbegriffe**
  - Grundstruktur eines C Programms
  - Variable
  - Konstante
  - Typen
  - Ein-/Ausgabe
  - Operatoren
  - Kontrollstrukturen
  - **Vektoren (Arrays, Felder)**

# Vektoren

## Motivation

- bisher nur einzelne (singuläre) Variable betrachtet
- manchmal müssen viele gleichartige / sehr ähnliche Variable betrachtet werden
- ➔ viele Variable individuell vereinbaren  
➔ “mühsam“
- ➔ viele Variable unter einem Namen zusammenfassen



# Vektoren

## Beispiel

- Zahlen in einem (mathematischen) Vektor
- Buchstaben in einer Zeichenkette
- Mitarbeiter in einer Firma



# Vektoren

## Beispiel

- Einlesen eines Wortes aus 10 Buchstaben

```
char b1, b2, b3, b4, b5, b6, b7, b8, b8, b10;
```

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Aneinanderreihung einer endlichen Anzahl von Variablen gleichen Typs
- unter einem gemeinsamen Namen
- einzelne Variable individuell ansprechbar über eine Nummer (Index)
- Ablage in einem gemeinsamen Speicherbereich

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Definition: `<typ> variablenname [<ganzzahliger Ausdruck>]`
- Beispiele:
  - `float vektor[5];`
  - `char name[20];`
  - `int zahlenfeld[10];`
- Zugriff (global): `variablenname` (z.B. als Parameter)
- Zugriff (individuell): `variablenname[<ganzzahliger Ausdruck>];`

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Beispiele (globaler Zugriff):
  - `float vektor[5];`  
...  
`int groesse = sizeof(vektor);`
  - `char text[20] = "Hallo Welt";`  
...  
`printf(text);`
  - `int zahlfeld[5];`  
...  
`... &zahlfeld ... ;`



# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Beispiele (individueller Zugriff):
  - `float vektor[5];`  
`int i = 3`  
`...`  
`vektor[i] = 7.5;`  
`vektor[i+1] = 3.5;`  
`vektor[0] = vektor[1];`
  - `char text[20] = "Hallo Welt";`  
`text[0] = 'h';`  
`printf(text); /* erzeugt Ausgabe "hallo Welt" */`

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- generelle Regeln (in C oder C++)
    - das erste Element im Vektor hat immer den Index 0
    - das letzte Element im Vektor hat immer den Index Anzahl Vektorelemente - 1
    - Anzahl der Elemente compilerspezifisch begrenzt
    - ein falscher Index wird beim Lesen oder Schreiben zur Laufzeit nicht überprüft
    - ein falscher Index beim Schreiben überschreibt eine Speicherstelle
- ➔ die richtige Indizierung ist in der Verantwortung des Programmierers!

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Besonderheiten
  - keine Zuweisung von Arrays an Arrays

```
int feld1[3] = { 1, 2, 3 };  
int feld2[3];
```

```
feld2 = feld1; ← nicht möglich!!
```

```
/* stattdessen : */
```

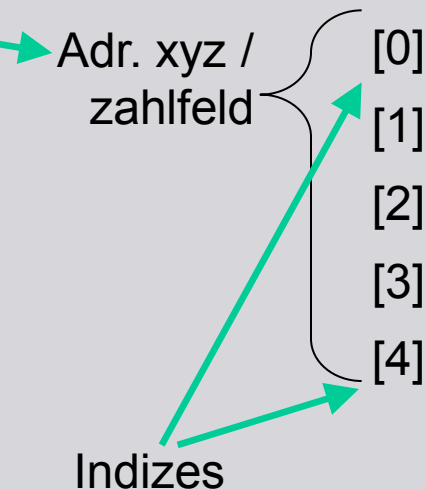
```
feld2[1] = feld1[1];  
feld2[2] = feld1[2];  
feld2[3] = feld1[3];
```

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Abbildung im Speicher
  - `int zahlfeld[5];`

Speicher an der Adresse xyz wird unter dem Namen `zahlfeld` reserviert, Größe des reservierten Bereichs entspricht 5 mal dem Typ `int`.



# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Initialisierung ggf. mit automatischer Ermittlung der Elementanzahl
  - `int zahl[] = {3, 5, 7, 1};`  
→ Anzahl Elemente 4
  - `char zeichenkette[] = "Hallo Welt";`  
→ Anzahl Elemente 11 inkl. `'\0'`
  - `char zeichenkette[] = {'H', 'a', 'l', 'l', 'o', ' ', 'W', 'e', 'l', 't'};`  
→ Anzahl Elemente 10

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Initialisierung ggf. mit expliziter Angabe der Elementanzahl
  - `int zahl[4] = {3, 5, 7, 1};`
  - `char zeichenkette[11] = "Hallo Welt";`  
entspricht  
`char zeichenkette[11] = {'H', 'a', 'l', 'l', 'o', ' ', 'W', 'e', 'l', 't', '\0'}`

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Manipulation von Zeichenketten
  - Headerfile <string.h> includieren  
`#include <string.h>`
  - ...  
`char string[20];`  
`scanf("%s", string);`  
...
  - liest bis zu einem Leerzeichen oder bis zum Zeilenende
  - fügt '\0' (Stringendezeichen) als letztes Zeichen ein.

# Vektoren

## Zeichenketten

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

int main(void)
{ char string[20];
  scanf("%s",string);
  for (int i = 0; i<strlen(string); i++)
    printf("%d:%c\n",i , string[i]);
  printf("%d\n",string[strlen(string)]);

  system("PAUSE");
  return 0;
}
```



# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Manipulation von Zeichenketten
  - Headerfile <string.h> includieren  
`#include <string.h>`
  - ...  
`char string[20];`  
`gets(string);`  
...
  - liest bis zum Zeilenende
  - fügt '\0' (Stringendezeichen) als letztes Zeichen ein.

# Vektoren

## Zeichenketten

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

int main(void)
{ char string[20];
  gets(string);
  for (int i = 0; i<strlen(string); i++)
    printf("%d:%c\n",i , string[i]);
  printf("%d\n",string[strlen(string)]);

  system("PAUSE");
  return 0;
}
```

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Manipulation von Zeichenketten mittels Bibliotheksfunktionen
  - `scanf("%s", ...)` Einlesen Zeichenkette
  - `printf("%s", ...)` Ausgeben von Zeichenketten
  - `gets(...)` Einlesen Zeichenkette
  - `puts(...)` Ausgeben von Zeichenketten
  - `... = getchar()` Einlesen Buchstabe
- `char* strchr(str, zeichen)`
  - Suchen nach zeichen in str
  - Ergebnis NULL, falls zeichen nicht vorkommt, sonst Adresse des ersten Vorkommens von zeichen in str

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Manipulation von Zeichenketten mittels Bibliotheksfunktionen
  - strcpy (zielstr,quellstr)
    - Kopieren von quellstr in zielstr
    - Länge von zielstr muss passen!
    - Endezeichen wird mitkopiert
  - strcat(str1,str2)
    - str2 an str1 anhängen
    - Länge von str1 muss passen!
    - Endezeichen wird mitkopiert

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Manipulation von Zeichenketten mittels Bibliotheksfunktionen
  - `int strcmp(str1, str2)`
    - Vergleich str1 mit str2 (alphabetisch)
    - Ergebnis 0 bei Gleichheit
    - Ergebnis  $< 0$ , falls  $str1 < str2$
    - Ergebnis  $> 0$ , falls  $str1 > str2$
  - `int strcspn(str1, str2)`
    - Länge des Teilstrings in str1 bis zum ersten Zeichen, das in str2 vorkommt
    - Groß-/Klein wird unterschieden

# Vektoren

## Zeichenketten

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

int main(void)
{ char str1[20] = "Hallo Welt";
  char str2[10] = "ABo";

  printf("%d\n",strcspn(str1,str2));

  system("PAUSE");
  return 0;
}
```

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Manipulation von Zeichenketten mittels Bibliotheksfunktionen
  - `int strlen(str)`
    - Länge von str ohne(!) Stringendezeichen
  - `strncat(str1, str2, anzahl)`
    - wie `strcat(...)` , nur dass nur anzahl Zeichen angehängt werden.
  - `strncpy(str1, str2, anzahl)`
    - wie `strcpy(...)` , nur dass nur anzahl Zeichen kopiert werden.

# Vektoren

## C Sprachelement Vektor (Array, Feld)

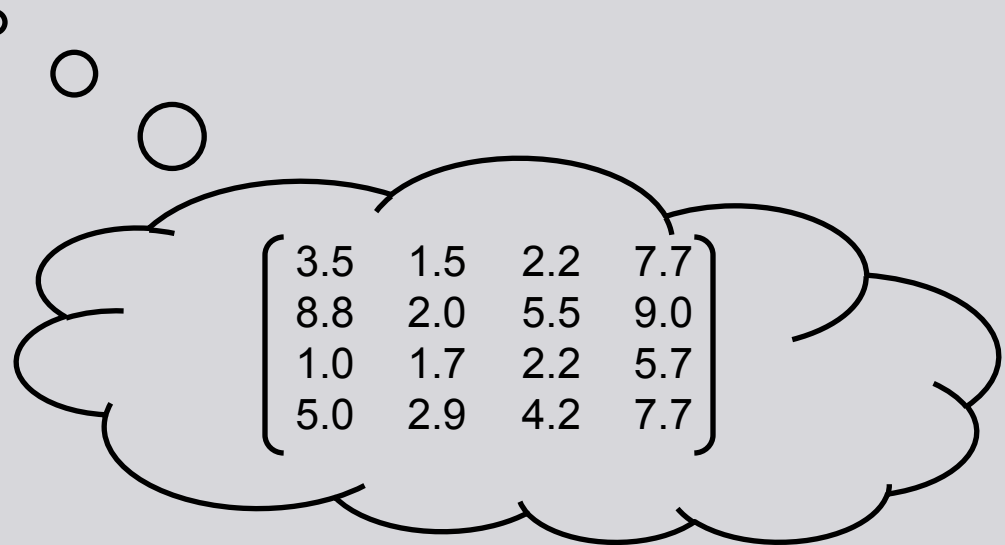
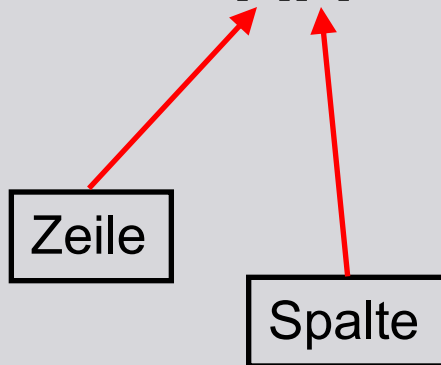
- Manipulation von Zeichenketten mittels Bibliotheksfunktionen
  - `char strstr(str1, str2)`
    - Feststellen, ob `str2` in `str1` als Teilstring vorkommt
    - Ergebnis ist 0, falls `str2` kein Teilstring von `str1`



# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Mehrdimensionale Vektoren
  - `float matrix[4][4] = {{3.5, 1.5, 2.2, 7.7},`  
`{8.8, 2.0, 5.5, 9.0},`  
`{1.0, 1.7, 2.2, 5.7},`  
`{5.0, 2.9, 4.2, 7.7}};`



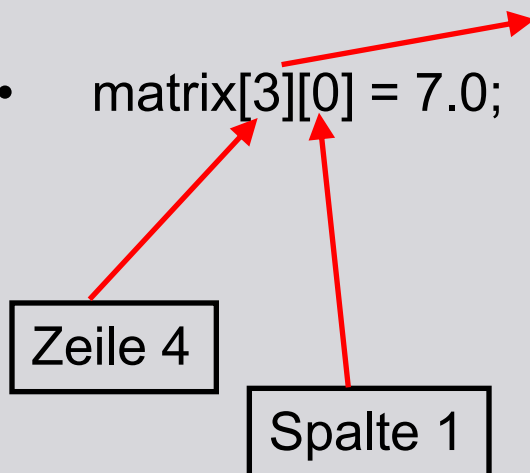
A thought bubble containing a 4x4 matrix of floating-point numbers, representing the data stored in the 'matrix' array.

3.5	1.5	2.2	7.7
8.8	2.0	5.5	9.0
1.0	1.7	2.2	5.7
5.0	2.9	4.2	7.7

# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Mehrdimensionale Vektoren, Zugriff auf Elemente
  - `float matrix[4][4] = {{3.5, 1.5, 2.2, 7.7},  
                          {8.8, 2.0, 5.5, 9.0},  
                          {1.0, 1.7, 2.2, 5.7},  
                          {5.0, 2.9, 4.2, 7.7}};`
  - `matrix[3][0] = 7.0;`



# Vektoren

## C Sprachelement Vektor (Array, Feld)

- Mehrdimensionale Vektoren, Zugriff auf Elemente, Beispiel:
  - Multiplikation 3x3 Matrix mit 3-elementigem Vektor

```
#include <stdio.h>
main()
{ double vek[3]   = { 2.0, 4.7, 3.0}, erg[3];
  double matrix[3][3] = { { 11.0, 7.6, 1.7},
                          { 2.0, 0.2, 0.9},
                          { 0.0, 2.0, 1.1}  };

  int zeile, spalte;
  for (zeile=0; zeile<3; zeile++)
  { erg[zeile]=0.0;
    for (spalte=0; spalte<3; spalte++)
      erg[zeile] += vek[spalte]*matrix[zeile][spalte];
  }
  for(zeile=0; zeile<3; zeile++)
    printf("%f\n", erg[zeile]);
}
```

# Vektoren



Hochschule  
Deggendorf

**Zum Schluss dieses Abschnitts ...**

**Noch Fragen ??**