Konzepte des prozeduralen Programmierens

Stand September 2022

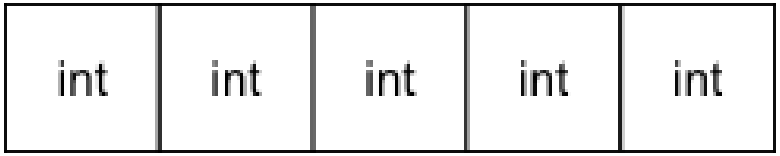
- Prof. Dr. Oliver S. Lazar / Christian Frank

6 Arrays (Felder)

Schleifen, Initialisierung, zwei- und mehrdimensionale Arrays, Zeigerarithmetik

Arrays (Felder)

- ein Array ist eine Datenstruktur, die aus mehreren gleichartigen Elementen zusammengesetzt ist
- jedes Element eines Arrays muss vom selben Datentyp sein



ein Array aus 5-int Elementen

- ein Elementtyp kann seinerseits wieder ein Array sein (also ein Array, dessen Elemente Arrays sind) → so entsteht ein mehrdimensionales Array
- die Elemente eine Arrays werden über Ihren Index angesprochen
  - ein Array wird mit einer bestimmten Länge definiert
  - der Index wird von 0 bis Länge-1 durchgezählt

Arrays (Felder)

- Für eine Software für eine Wetterstation sollen die Durchschnittstemperaturen des Tages berechnet werden
- Es soll jede Minute ein Wert gespeichert werden
  - 24 Stunden * 60 Minuten = 1.440 Variablen
  - > 1.000 Variablen zu deklarieren ist viel Arbeit!

```
float messwert1, messwert2, messwert3, ..., messwert1439, messwert1440;
```

- Einfacher geht es mit einem Array von Variablen

```
float messwerte[1440];
```

  - damit haben wir 1.440 float-Variablen auf einen Schlag angelegt

Arrays deklarieren und initialisieren

- Deklaration erfolgt analog zu den einfachen Variablen (Datentyp und Name)
- es folgen eckige Klammern, in denen die Größe (also die Anzahl der Elemente) angegeben wird
- der Zugriff auf die Elemente erfolgt per Index
  - Index ist eine Ganzzahl aus dem Bereich zwischen 0 und Arraygröße - 1

```
float messwerte[5];

messwerte[0] = 23.0;
messwerte[1] = 22.2;
messwerte[2] = 21.7;
messwerte[3] = 20.9;
messwerte[4] = 20.5;

printf("erster Wert (Index 0): %.2f\n", messwerte[0]);
printf("letzer Wert (Index 4): %.2f\n", messwerte[4]);
```

```
erster Wert (Index 0): 23.00
letzer Wert (Index 4): 20.50
```

Arrays und Schleifen

- Zählschleifen eignen sich hervorragend für den Umgang mit Arrays
  - die Zählvariable kann als Index für das Array verwendet werden
  - einfaches Schreiben und Lesen auch von sehr großen Arrays

```
int punkte[5], i;

// Werte setzen
for(i=0; i<5; i++) {
    punkte[i] = i+1;
}

// Werte auslesen
for(i=0; i<5; i++) {
    printf("(Index %d) Wert: %d\n", i, punkte[i]);
}
```

```
(Index 0) Wert: 1
(Index 1) Wert: 2
(Index 2) Wert: 3
(Index 3) Wert: 4
(Index 4) Wert: 5
```

Arrays und Schleifen

- Das Setzen der Werte kann man auch durch eine Benutzereingabe realisieren:

```
int punkte[5], i;

// Werte einlesen
for(i=0; i<5; i++) {
    printf("\nBitte geben Sie einen Wert ein (ganze Zahl): ");
    scanf("%d", &punkte[i]);
}

// Werte auslesen
for(i=0; i<5; i++) {
    printf("(Index %d) Wert: %d\n", i, punkte[i]);
}
```

```
Bitte geben Sie einen Wert ein (ganze Zahl): 3
Bitte geben Sie einen Wert ein (ganze Zahl): 9
Bitte geben Sie einen Wert ein (ganze Zahl): 7
Bitte geben Sie einen Wert ein (ganze Zahl): 2
Bitte geben Sie einen Wert ein (ganze Zahl): 4
```

```
(Index 0) Wert: 3
(Index 1) Wert: 9
(Index 2) Wert: 7
(Index 3) Wert: 2
(Index 4) Wert: 4
```

Initialisierung von Arrays

- Möchte man die Werte eines Feldes initialisieren, schreibt man die Werte einfach in geschweifte Klammern

```
int i, punkte[5] = { 1, 3, 5, 7, 9 };

// Werte ausgeben
for(i=0; i<5; i++) {
    printf("Wert Index %d: %d\n", i, punkte[i]);
}
```

```
Wert Index 0: 1
Wert Index 1: 3
Wert Index 2: 5
Wert Index 3: 7
Wert Index 4: 9
```

- Dadurch lässt sich ein Feld auch einfach komplett mit Null-Werten initialisieren

```
int punkte[5] = { 0 };
```

Null-Initialisierung

- Ist die Anzahl der Werte bei der Initialisierung kleiner als die Feldgröße, werden die restlichen Werte auf Null gesetzt

```
int i, punkte[5] = { 1, 3, 5 };

// Werte ausgeben
for(i=0; i<5; i++) {
    printf("Wert Index %d: %d\n", i, punkte[i]);
}
```

```
Wert Index 0: 1
Wert Index 1: 3
Wert Index 2: 5
Wert Index 3: 0
Wert Index 4: 0
```

Feldgröße durch Initialisierung bestimmen

- Wird die Angabe der Feldgröße weggelassen, wird automatisch die Größe durch die Anzahl der Initialisierungswerte bestimmt

```
int i, punkte[] = { 1, 3, 5 };

// Werte ausgeben
for(i=0; i<3; i++) {
    printf("Wert Index %d: %d\n", i, punkte[i]);
}
```

```
Wert Index 0: 1
Wert Index 1: 3
Wert Index 2: 5
```

Aufgabe 06.01

- **Notendurchschnitt**

  Schreibe ein Programm, dass den Benutzer auffordert, zehn Schulnoten als Kommazahlen einzugeben. Diese Zahlen sollen in einem Array zwischengespeichert werden.

  Im Anschluss berechnet das Programm den Durchschnitt, welcher am Bildschirm ausgegeben werden soll.

Aufgabe 06.02 (Zusatzaufgabe)

Lege je ein Array mit 100 Elementen vom Typ int, long, float und double an. Lass dir jeweils die Größe eines Elements (in Byte) und des gesamten Arrays (in Byte) sowie die Anzahl der Elemente dieses Arrays (Array-Bytes/Element-Bytes) ausgeben.

⇒ Arbeiten Sie mit dem Operator `sizeof()`.

```
printf("int      : %d Bytes\n", sizeof(int));
```

```
int      : 4 Bytes
```

Aufgabe 06.03

- **Bubble-Sort**

  Bekannter Sortieralgorithmus

  „die größten Elemente steigen wie Luftblasen nach oben“

  1) Es gibt eine äußere Schleife, die so oft durchlaufen wird, wie die Liste Elemente hat
  2) Es gibt eine innere Schleife, in der die Elemente verglichen und evtl. getauscht werden
     - Schaue, ob das Element mit Index i größer ist, als das mit Index i+1
     - Wenn ja, dann Elemente tauschen und Index um 1 erhöhen u.s.w.
     - Hinweis: Nach jedem Durchlauf der inneren Schleife steht das größte Element ganz rechts, im nächsten Durchlauf muss daher nur noch eine um ein Element kürzere Liste sortiert werden

Aufgabenstellung:

- Erstelle ein PAP für den Algorithmus
- Schreibe den Algorithmus als Pseudocode
- Implementiere den Algorithmus in C:
  - Deklariere und initialisiere ein int-Array mit 20 Zahlen und implementiere den Bubble-Sort, der das Array sortiert
  - Gib zur Kontrolle abschließend das Array auf der Systemausgabe aus
  - Lade das Programm auf GitHub

Aufgabe 06.04

- **Sieb des Eratosthenes**

  Bekanntes Verfahren zur Bestimmung von Primzahlen

  Der Algorithmus ist z.B. hier: https://de.wikipedia.org/wiki/Sieb_des_Eratosthenes

  Für das Sieb bietet sich ein Array an, das in der Grösse des Startwerts mit 1 (true) initialisert wird

Aufgabenstellung:

- Erstelle ein PAP für den Algorithmus
- Schreibe den Algorithmus als Pseudocode
- Implementiere den Algorithmus in C
  - Vergleiche das Ergebnis mit Aufgabe 3.26
  - Lade das Programm auf GitHub

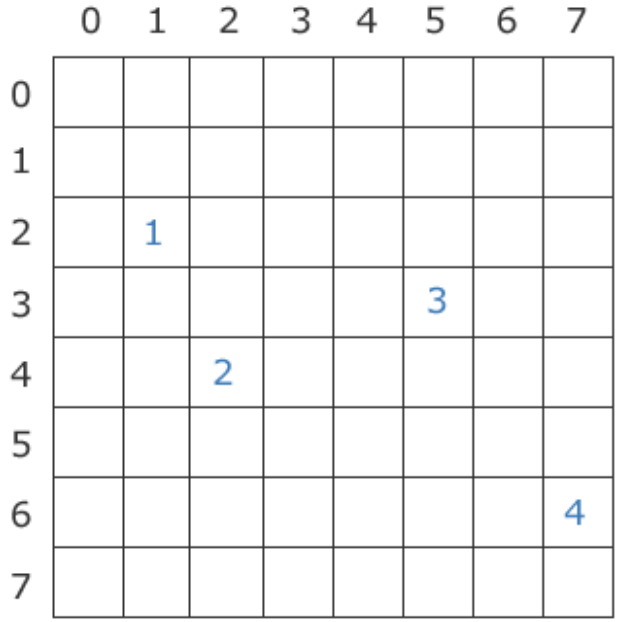
Zweidimensionale Arrays

- eindimensionales Array kann als einfache Liste betrachtet werden
- Zweidimensionales Array entspricht einer Matrix aus n Zeilen und m Spalten
  - z.B. ein Schachbrett mit 8 Zeilen und 8 Spalten

```
int brett[8][8];
```

- der erste Index steht für die Zeile, der zweite Index für die Spalte: **`brett[Zeile][Spalte]`**

```
brett[2][1] = 1;
brett[4][2] = 2;
brett[3][5] = 3;
brett[6][7] = 4;
```



Zweidimensionale Arrays

- Beispiel

```
int brett[8][8] = { 0 };

brett[2][1] = 1;
brett[4][2] = 2;
brett[3][5] = 3;
brett[6][7] = 4;

int i, j;

// Schleife fuer Zeilen
for(i=0; i<8; i++) {

    // Schleife fuer Spalten
    for(j=0; j<8; j++) {
        printf("%d ", brett[i][j]);
    }

    printf("\n");
}
```

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 3 0 0
0 0 2 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 4
0 0 0 0 0 0 0 0
```

Initialisierung

- Initialisierung erfolgt nach unserer Definition spaltenweise

```
int brett[8][8] = { 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4 };
```

```
1 2 3 4 5 6 7 8
1 2 3 4 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

- es werden also folgende Indexierungen getroffen:
  - Zeile 1: [0][0], [0][1], [0][2], [0][3], [0][4], [0][5], [0][6], [0][7]
  - Zeile 2: [1][0], [1][1], [1][2], [1][3]

Initialisierung

- Initialisierung geht auch so

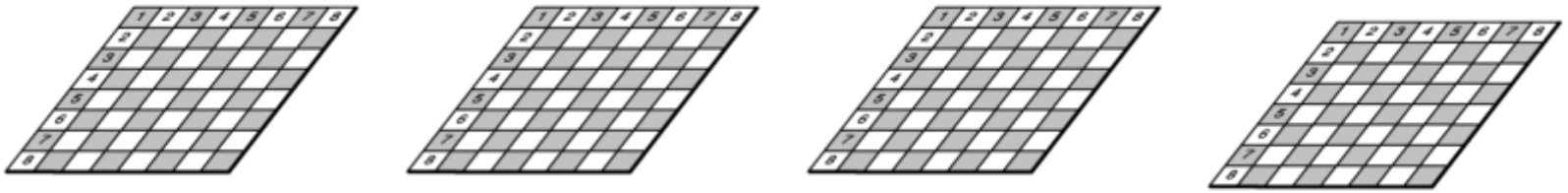
```
int myArray[4][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {0, 1, 2} };
```

- die geschweiften Klammern begrenzen die Dimensionen

```
1 2 3
4 5 6
7 8 9
0 1 2
```

Mehrdimensionale Arrays

- Es können beliebig viele Dimensionen eingesetzt werden, solange es für die Programmierung Sinn macht und genügend Speicher vorhanden ist



- z.B. drei Dimensionen, wenn man gleich mehrere Schachbretter verwalten möchte: Index 1 = Brett, Index 2= Zeile, Index 3 = Spalte
- der Umgang ist analog zu den zweidimensionalen Arrays

```
int brettArray[4][8][8];
```

Aufgabe 06.05

Lege ein zweidimensionales Array an, das fünf Wertepaare aufnehmen kann. Initialisiere das Feld mit Werten und gib die Wertepaare sowie für jedes Wertepaar das Produkt seiner Werte aus.

```
#include <stdio.h>


int main(){
  int paare[][2]={{1,2},{3,4},{5,6},{7,8},{9,10}};

  ***HIER ERGÄNZEN***

  return 0;
}
```

Zusatzaufgabe:
Klausur-WS-16_17_Nachschreib Aufgabe 13
Klausur SS15 Aufgabe 4

Aufgabe 06.06 (Zusatzaufgabe)

Welche Änderungen wären sinnvollerweise an der Lösung von Aufgabe 06.04 nötig, damit das Programm einfach auf Tripel, Quadrupel und n-Tupel geändert werden kann?

- Füge ggf. eine automatische Initialisierungssequenz ein oder initialisiere manuell
- Hinweise:
  - Arbeite mit verschachtelten Schleifen.
  - Nach Änderung der Feldgrenzen sollte das Programm weiterhin reibungslos funktionieren.

```
#define ZEILEN 5
//#define SPALTEN 3
#define SPALTEN 4

int main(void){

  // Tripel-Initialisierung
  //int array[ZEILEN][SPALTEN]={{1,2,3},{4,5,6},{7,8,9},{10,11,12},{13,14,15}};

  // Quadrupel-Initialisierung
  int array[ZEILEN][SPALTEN]={{1,2,3,4},{5,6,7,8},{9,10,11,12},{12,13,14,15},{16,17,18,19}};

 …
```