

# Arrays in C

# Lernziele

- Sie können einen Array korrekt definieren und initialisieren.
- Sie können die einzelnen Elemente eines Arrays bearbeiten.
- Sie können zweidimensionale Arrays definieren und anwenden.

# Was ist ein Array?

## Fallbeispiel:

Stelle dir vor, du sollst die Software für eine Wetterstation schreiben. Das Programm soll die gemessenen Temperaturwerte zwischenspeichern und die Durchschnitts-Temperatur des Tages berechnen.

Die Werte werden

1. Jede Stunde
  2. Jede Minute
- gemessen und gespeichert.

Welche Variablen müssen für das Programm definiert werden?

## Lösungen mit Variablen:

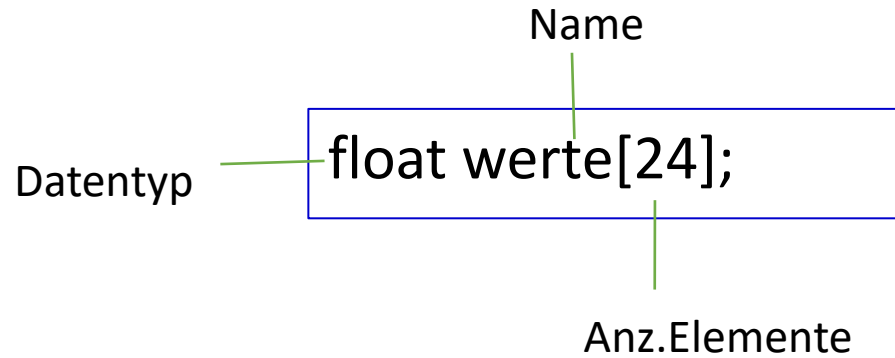
1. `float wert1, wert2, wert3, ..., wert24 ;`
2. `float wert1, wert2, wert3, ..., wert1439, wert1440;`

## Lösungen mit Arrays:

1. `float werte[24];`
2. `float werte[1440];`

# Was ist ein Array?

Ein Array besteht aus einer konstanten Anzahl gleichartiger Datenelemente, die unter einem Namen zusammenhängend angeordnet sind.



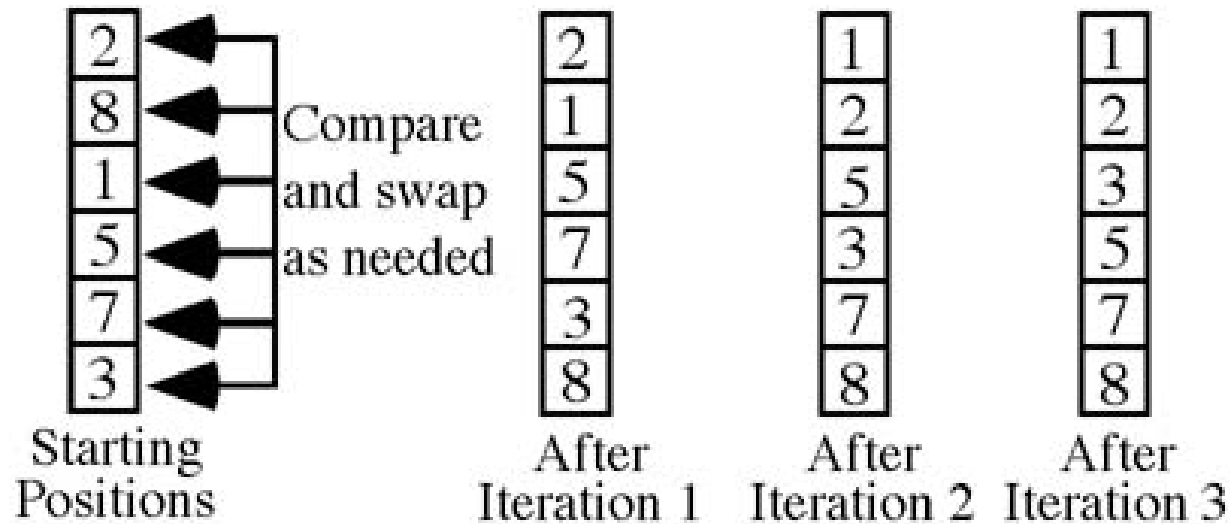
Hinweise:

- Alle Elemente haben den gleichen Datentyp.
- Die Anzahl von Elementen ist fest und muss zur Kompilierzeit bekannt sein.

# Anwendungsbeispiele

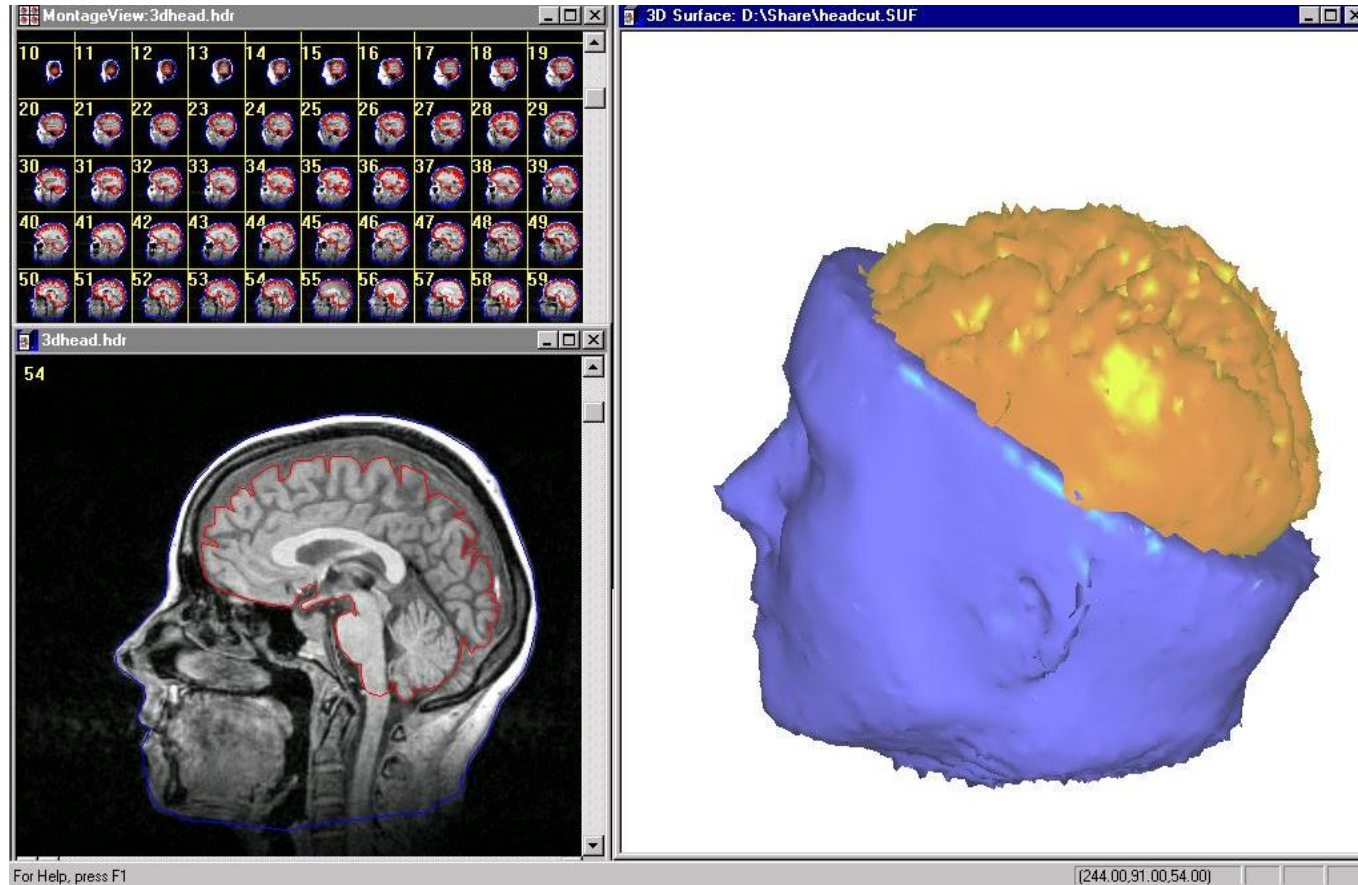
## Beispiel 1 Sortierung

Es ist eine Reihe von Zahlen gegeben. Sie müssen ein C-Programm schreiben um die Zahlen aufsteigend oder absteigend zu sortieren.



# Anwendungsbeispiele

## Beispiel 2 Digitale Bildverarbeitung

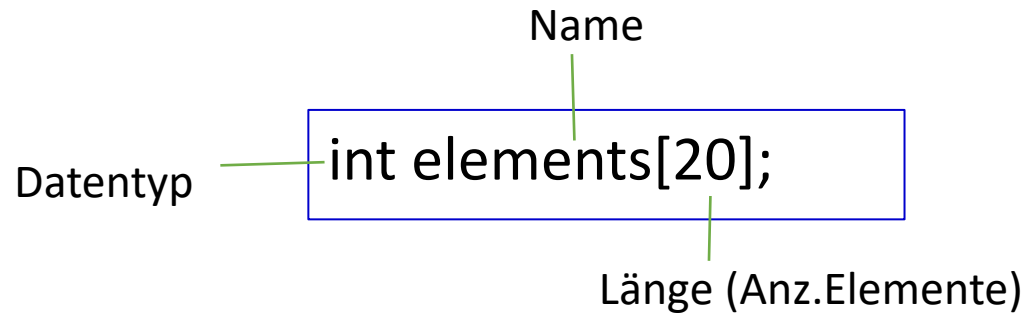


$i \times j$

Row 1	$a_{1,1}$	$a_{1,2}$	...	$a_{1,j}$
	$a_{2,1}$	$a_{2,2}$	...	$a_{2,j}$
	$\vdots$	$\vdots$		
Row i	$a_{i,1}$	$a_{i,2}$	...	$a_{i,j}$
	Column 1			Column j

# Definition

- Syntax



- Länge: eine Zahl oder eine symbolische Konstante

```
#define arrayMaxSize 10  
int number[arrayMaxSize];
```

- C verwendet einen nullbasierten Index:

Das erste Element hat den Index 0. Der Index läuft von 0 bis *Länge - 1*.

```
int elements[4];  
elements[0] = 1;    /*das erste Element */  
elements[3] = 4;    /*das letzte Element */
```

# Initialisierung

Arrays lassen sich bei der Definition auch initialisieren:

1. `double result[5]={1.0, 2.0, 3.0, 4.0, 5.0};` //Explizit Angabe der Länge des Arrays
2. `double result[]={1.0, 2.0, 3.0, 4.0, 5.0};` //Die Länge ist die Anzahl Elemente
3. `double result[5]={1.0, 2.0, 3.0};` // Die Restlichen werden auf 0 initialisiert.
4. `int myArray[100] = {0};` //Alle Elemente werden auf 0 gesetzt
5. `const float currency[3] = {1.4, 1.5, 1.6};` //Array mit Schreibschutz



# Zugriff auf die Elemente

Jedes Element lässt sich via Index zugreifen:

```
int elements[3];  
// Array mit Werten initialisieren  
elements[0] = 1234;  
elements[1] = 3456;  
elements [2] = 7890;  
  
// Inhalt der Array-Element ausgeben  
printf(" elements[0] = %d\n", elements[0]);  
printf(" elements[1] = %d\n", elements[1]);  
printf(" elements[2] = %d\n", elements[2]);
```

Wichtig: C-Compiler überprüft die Arraygrenze nicht.

# Array und Funktion

- Arrays können an Funktionen übergeben werden.
- Arrays können nicht von Funktionen zurückgeliefert werden

```
/* Die Groessenangabe ist für die erste Dimension optional, und wird auch nicht überprüft.*/  
int Summe(int a[])  
{  
    int summe = 0;  
    for (int i = 0; i < 10; i++) {  
        summe += a[i];  
    }  
    return summe;  
}
```

```
/* Man übergibt besser auch die effektive Grösse des Arrays als weiteren Parameter.*/  
void Modify(int a[], int size)  
{  
    if (size >= 4)  
        a[3] = 17; /*Originalarray von Aufrufer wird verändert*/  
}
```

FunkArraysApp

# Zweidimensionales Array

## ■ Definition:

Bei der Definition ist pro Dimension ein Klammerpaar erforderlich.

```
int tabelle[3][4] = { {12, 23, 34, 50},  
                     {45, 56, 67, 51} };
```

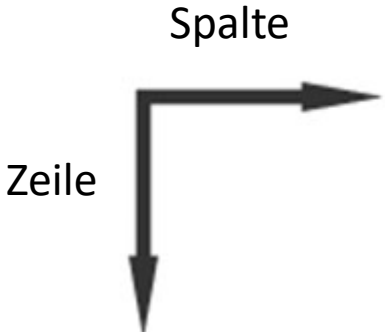
1. Index entspricht Zeilen (3 Zeilen)

2. Index entspricht Spalten (4 Spalten)

oder auch:

```
int tabelle[][4] = { 12, 23, 34, 50, 45, 56, 67, 51 };
```

Bei dieser Schreibweise wird die erste Dimension anhand der angegebenen Anzahl in der zweiten Dimension und der vorhandenen Initialisierungselemente berechnet.



	0	1	2	3
0	12	23	34	50
1	45	56	67	51
2		1		

## ■ Zugriff:

```
tabelle[2][1] = 1; //Schreib 1 in das Element in der Zeile 2 und Spalte 1
```

# Zweidimensionales Array

```
#define DIM 7

int brett[DIM][DIM] = { 0 };    //Alle Elemente werden mit 0 initialisiert.

printf("Dieses Programm zeichnet die Diagonale auf einem Brett...\n");
for (int i = 0; i < DIM; i++) {           //Zeile
    for (int j = 0; j < DIM; j++) {       //Spalte
        if (i + j == DIM - 1) {
            brett[i][j] = 1;
        }
    }
}
```

	Spalte						
	0	1	2	3	4	5	6
0							1
1						1	
2					1		
3				1			
4			1				
5		1					
6	1						
Zeile							

# Übung 2 : Zweidimensionales Array (15')

Modifizieren Sie das Beispielprogramm Brett.cpp, damit es folgende Figuren zeichnet:

1. Ein diagonales Kreuz

1	0	0	0	0	0	1
0	1	0	0	0	1	0
0	0	1	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
1	0	0	0	0	0	1

2. Fahne des Kantons Zürich (Blau:1 Weiss:0)

1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0
1	1	1	1	1	1	1