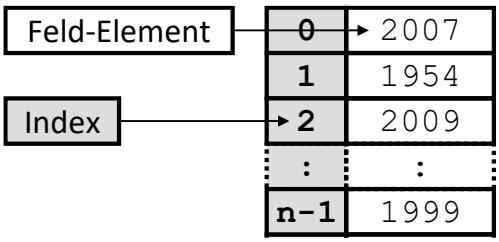
Arrays in C++

1. Was sind Arrays?

Durch Arrays (Felder) ist es möglich, mehrere Daten des gleichen Typs in einer Struktur zu speichern. Über den Index (numerische Angabe) besteht ein wahlfreier Zugriff auf die einzelnen Daten (Array-Elemente oder Feld-Elemente).




```
①  type FeldName[n] ;

②  FeldName[0] = Wert0 ;
③  FeldName[1] = Wert1 ;
④                :
⑤  FeldName[n-1] = Wertn-1 ;
```

Deklarieren und Initialisieren

2. Syntax für die Deklaration und Initialisierung

In C sind Arrays statisch (d.h. die Größe des Arrays muss zum Zeitpunkt der Übersetzung bekannt sein). In Schritt ① wird ein Feld mit n Elementen (Index von 0 bis n-1) deklariert. Durch ② bis ⑤ kann jedes Feld-Element individuell initialisiert werden.

Eine andere Syntax-Variante ist die Deklarierung und Initialisierung in einer Anweisung (auch Kurzform genannt). Dabei kann die Größenangabe entfallen, da sie durch die Anzahl der zugewiesenen Elemente (hier n- Elemente) bestimmt ist. Die Größe eines Feldes kann nachträglich nicht mehr geändert werden.

```
①  type FeldName[] = {Wert0 , Wert1 , ... , Wertn-1};
```

Kurzform - Deklarieren und Initialisieren

3. Typischer Fehler

In C werden Array-Über- oder -Unterläufe nicht überprüft. Im nebenstehenden Beispiel wird ein Array A (Zeile ②) mit vier Elementen deklariert und initialisiert. Das letzte Element hat demzufolge den Index 3. Durch die fehlerhafte Schleifenbedingung (Zeile ③) L<=4 wird hier das Feldelement A[4] angesprochen. **Dies führt nicht zu einem Compiler-Fehler**, sondern für A[4] wird ein (nicht bestimmbarer) Wert angezeigt, der in diesem Speicherbereich liegt und eventuell von einem anderen Programmteil reserviert und genutzt wird. Das kann zu ungewollten Effekten (z.B.: bei einer Wertzuweisung A[4]=200;) und schwer aufzuspürenden Fehlern führen. Abhilfe schafft hier (Zeile ③) L<4. Besser ist es, die Anzahl der Array-Elemente programmtechnisch zu bestimmen.

```
①  int L;
②  int A[] = {2,4,6,8} ;
③  for ( L=0 ; L <= 4 ; L++ ) {
④     cout << A[L] << endl ;
⑤     }
```

Array-Fehler

4. Anzahl der Array-Elemente bestimmen

Da **`sizeof()`** die Größe des reservierten Speichers für das gesamte Array angibt, berechnet der Quotient: **Feldgröße / Elementgröße** die Anzahl der Feldelemente.

```
①  int Anz =sizeof(A_Name)/sizeof(A_Typ);
```

Elementeanzahl