C H A M

Soll ein Datentyp in einen anderen konvertiert werden findet ein "Type-Casting" statt.

2.1 Implizite Datentypumwandlung

Werden in Ausdrücken Operanden mit unterschiedlichen Datentypen miteinander verknüpft, so erfolgt eine implizite (automatische) Datentypumwandlung.

2.1.1 Implizite >char< nach >int<

Der Datentyp >char< wird bei Berechnung oder Bewertung immer in den Datentyp >int< umgewandelt. Daraus kann man ableiten, dass >int< und >char< beliebig mischbar sind.

2.1.2. Implizites >float< nach >double<

Der Datentyp >float< wird ebenfalls bei Bewertung und Berechnung in den Datentyp >double< konvertiert. Somit erfolgen alle Berechnungen stets mit der selben Genauigkeit.

2.1.3 Datentypumwandlung bei arithmetischen Operanden

Bei arithmetischen Operationen erfolgt zunächst ein implizites Umwandeln von >char< nach >int< bzw. >float< nach >double< sämtlicher Operanden.

Treten anschließend noch Operanden mit unterschiedlichen Datentypen auf, wird in denjenigen Datentypen konvertiert, der in einer gewissen Rangfolge am weitesten oben steht.

Das Ergebnis ist ebenfalls von diesem Datentyp.

Rangfolge der impliziten Datentypkonvertierung:

char → short → int → long → longlong → float → double → long double

Ganzzahltypen — Gleitpunktzahlen

→ Die Rangfolge ist unabhängig vom Vorzeichen
→ Gleitpunktzahlen haben immer einen höheren Rang
→ Bei Umwandlung höherwertiger Datentypen in niederwertige Datentypen kann es zu Informationsverlust kommen.

→ Die Division zweier >int< Werte gibt nur den Ganzzahlanteil zurück!

→ Die implizite Datentypumwandlung funktioniert nicht bei Zuweisungsoperatoren (=) oder logischen Operatoren

→ Die implizite Datentypumwandlung funktioniert nicht bei Zuweisungsoperatoren (=) oder logischen Operatoren

↳ Folgen:

int = float
int = double
} Der Nachkommaanteil wird abgeschnitten

int = long
char = int
char = short
} Die höherwertigen Bits werden abgeschnitten

float = double → Der Wert wird entweder gerundet oder abgeschnitten

float = int
double = int
} Sollte keine Darstellung möglich sein, wird gerundet oder abgeschnitten

2.2 Explizite Datentypumwandlung

Da die implizite Datentypumwandlung nicht selten zu Fehler führt oder unerwünscht Ergebnisse erzielt kann durch die explizite Datentypumwandlung eine Konvertierung erzwungen werden.

Zur Anwendung kommt der >

Syntax: `(castType) Ausdruck;`

Dabei wird der Ausdruck unter Einbeziehung der impliziten Typecasting Regeln ausgewertet und dann gezielt in den >CastType< umgewandelt.

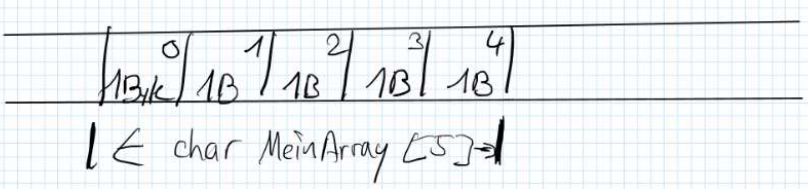
3.1 Deklarierung

Syntax: `Datentyp Name [Indexwert];`

Datentyp .. Datentyp aller Elemente
Name .. Name der Arrays
Indexwert .. Anzahl der Elemente, die im Speicher reserviert werden

Bsp.: `char MeinArray[5];`

Speicher



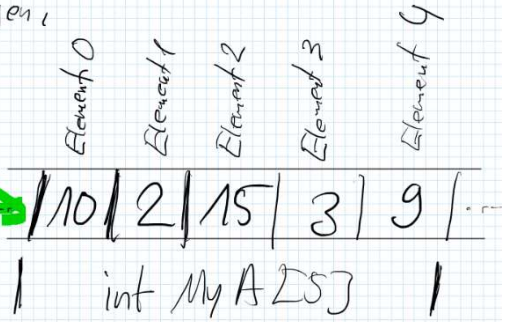
Im Speicher wird Speicherplatz für 5 Variablen vom Typ >char< reserviert.

Achtung: Die Größe eines Arrays muss zur Übersetzung bekannt sein.

5.2.1 Zugriff

Durch den Indizierungs Operator [ ] kann ein Zugriff auf den Datentyp eines Elementes erfolgen.

Beispiel:
```
int MyA[5];
MyA[0] = 10;
MyA[1] = 2;
MyA[2] = 15;
MyA[3] = 3;
MyA[4] = 9;
```

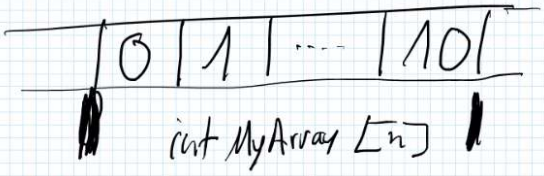


(Speicherbedarf von 20 Byte)

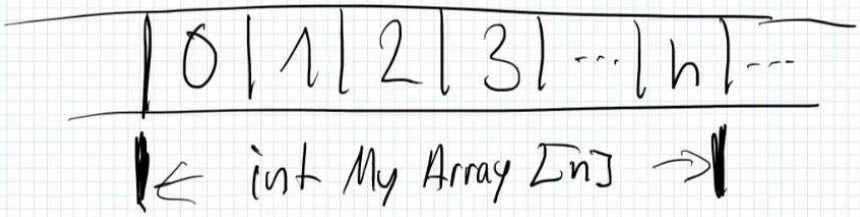
> ! Achtung: Die Zählweise des Indexwerts beginnt bei 0
> MyA[5] führt zu undefiniertem Verhalten.

1. einzelne Initialisierung

Beispiel
```
int MyArray[n];
MyArray[0] = 0;
MyArray[1] = 1;
MyArray[n] = 10;
```



Beispiel
```
int MyArray[] = {0,1,2,3,...,n};
```



Der Indexwert der Deklaration wird automatisch nach Anzahl der Werte definiert.

3. Initialisierung mit 0

⇒ for Schleife

```
int bigArray[1000];
for(int i=0; i<1000; i++) bigArray[i]=0;
```

⇒ Nullzuweisung.

```
int bigArray[1000] = {0};
```

⇒ Nullzuweisung mit Startwerten

```
int bigArray[1000]= {1,2,3,10};
```

1	2	3	10	0	0	...	0