**int (\*arr)[8]** – Pointer to an Int-Array

**int\* arr[8]** – Array of Int\*

**va\_start(ptr, länge)**: initialisiert varargs ptr

**va\_arg(ptr, größe)**: Liest Argument und erhoht va\_list ptr um 1.

arr[**outer**][**inner**]

arr[**1**][**2**] = \*(arr[**1**] + **2**) = \*(\*(arr + **1**) + **2**)) = (\*(arr + **1**))[**2**] *(Achtung Klammern!)*

Array-Initialisierung padded immer mit 0en!

sizeof (***pointer***) = systemarchitektur größe

sizeof (**array**) = anzahl\_elemente \* sizeof (elementtyp)

sizeof (**char**) = 1

sizeof (**short**) = 2

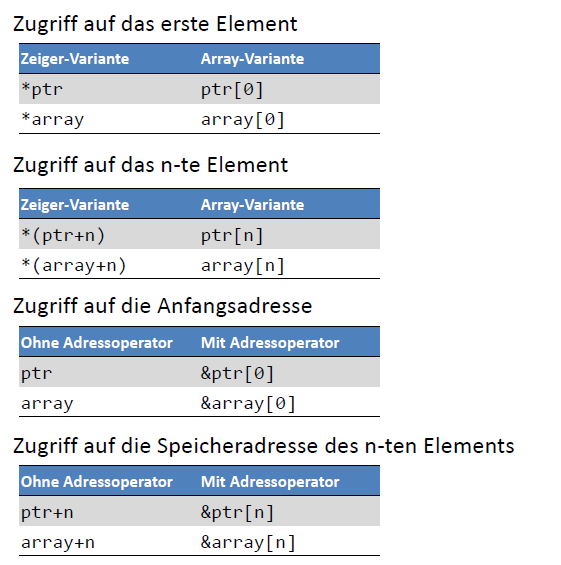
sizeof (**int**) = 4

sizeof (**long**) = 4 / 8

sizeof (**long long**) = 8

sizeof (**float**) = 4

sizeof (**double**) = 8



Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

**Strukturen**

Gesamtgröße ergibt sich durch Addition der einzelnen Komponenten. Sollte die Größe kleiner als 4 Bytes sein, dann soll man padden und auf 4 Bytes aufrunden.

*typedef*  ist ähnlich wie *data* in Haskell.

Zeiger auf Strukturen:

**struct** person \*p1; Zugriff: (\*p1).name oder p1 -> name

**Unions**

Unions werden bei kleinen Datentypen set **nicht** mit 0 gepadded!

**Speicherklassen**

Codesegment:

Maschinencode des Programms

Datensegment

Globale (externe) Variablen

Stack

Lokale Variablen

Rücksprungadresse

Parameter einer Funktion

Heap

Dynamische Variablen

**Speicherverwaltung**

malloc wird immer auf void gecastet

lieber die Antwort wo vor malloc gecasted wird!

**Makros**

**#**v → mit ““ einsetzen **##**v → as - is einsetzen

**\_\_LINE\_\_ \_\_FILE\_\_ \_\_func\_\_** *(Kein Makro)*

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung