C Programmierkurs

7. Stunde: dynamischer Speicher, Arrays

Johannes Hayeß & Mirko Seibt

6. Dezember 2018

Technische Universität Dresden

Arrays

Fixed Lenght Arrays - FLAs

Statische Arrays:

```
int array[42] = {};
```

Datenfeld fester Länge

Speicherbedarf: 42 * 4 Byte = 168 Byte

Fixed Lenght Arrays - FLAs

Statische Arrays:

```
int array[42] = {};
```

Datenfeld fester Länge
Speicherbedarf: 42 * 4 Byte = 168 Byte

Speicherzugriff:

```
array[13] = 1337;
```

Variable Lenght Arrays - VLAs

Arrays ohne Längenangabe:

```
int array[] = { 1, 3, 3, 7 };
```

Datenfeld mit beliebig vielen Feldern der Länge 4 Byte
Länge ergibt sich aus der Eingabe - hier: 4

Variable Lenght Arrays - VLAs

Arrays ohne Längenangabe:

```
int array[] = { 1, 3, 3, 7 };
```

Datenfeld mit beliebig vielen Feldern der Länge 4 Byte
Länge ergibt sich aus der Eingabe - hier: 4

Speicherzugriff wie gewohnt:

```
printf("%d", array[2]);
```

Dynamische Arrays

Dynamische Arrays:

```
int *array = calloc(42, sizeof *array);
```

Datenfeld mit 42 Feldern der Länge 4 Byte

Dynamische Arrays

Dynamische Arrays:

```
int *array = calloc(42, sizeof *array);
```

Datenfeld mit 42 Feldern der Länge 4 Byte

Speicherzugriff wie gewohnt:

```
array[1] = 3;
```

⇒ dynamisch zugewiesener Speicher kann als Array verstanden werden.
⇒ demnach kann jedes Array auch als Pointer verstanden werden.

Kleines Beispiel

```c
/* FLA */
int array[4] = { 1, 3, 3, 7 };
/* Pointer auf das erste Element */
int *array_ptr = &array[0];
/* Iteration mit Hilfe von Pointern */
for (int *ptr = array_ptr;
    ptr < (array_ptr + sizeof array / sizeof array[0]);
    ++ptr) {
    printf("%d", *ptr);
}
```

NULL-Pointer

NULL-Pointer

Zum Bsp. fehlgeschlagene Zuweisungen erzeugen Pointer die auf **nichts** zeigen. Sogenannte Nullpointer.

```c
int *pointer; // weist auf keine Adresse
int *ptr = (void*)0; // Nullpointer
```

Nullpointer sind i.d.R. schlecht, sie machen unser Programm kaputt!
Allerdings als Rückgabewert von Funktionen sind sie unvermeidbar.

```c
if (ptr == NULL) {
    /* Fehlerbehandlung */
}
```

Pfeil-Operator

Pfeiloperator

Operator: `->`

Zeigt auf einen Pointer und dereferenziert ihn.

Pfeiloperator

Operator: `->`

Zeigt auf einen Pointer und dereferenziert ihn.

Beispiel:

```
// Zugriff auf die x Koordinate
(*((*(g.e[i])).from)).x
```

Pfeiloperator

Operator: `->`

Zeigt auf einen Pointer und dereferenziert ihn.

Beispiel:

```
// Zugriff auf die x Koordinate
(*((*(g.e[i])).from)).x
```

```
// Auch Zugriff auf die x Koordinate
g.e[i]->from->x0
```

Header-Dateien

Headerdateien

Dateiendung: ***.h**
Standardbibliotheken hinzufügen:

```
#include <stdlib.h>
```

Eigene Headerdateien hinzufügen:

```
#include "my_lib.h"
```

Headerdateien enthalten **ausschliesslich**
Deklarationsprototypen von Funktionen, typedef structs und globale Konstanten.
Implementiert werden Funktionen in einer seperaten *.c Datei mit dem **selben** Namen wie der Header.