

1. Dezember

Pointer : Ein Zeiger ist in C++ ein Objekt, das auf eine Speicheradresse zeigt. Die Adresse eines Objektes `a` können wir mit `&a` ausgeben (nicht verwechseln mit Referenzen). Wir initialisieren einen Pointer / Zeiger wie folgt: `Typ* name;`

Der `Typ` muss dabei dem Typen der Adresse entsprechen.

new : Wir können mit new einen Speicher "reservieren". Dies tun wir mit

`new Typ [n];`

Dabei reservieren wir `n` Speicherplätze, die wir mit `Typ` versehen können. Beispielsweise

`new int [3];`

reserviert drei Speicherplätze vom Typ `int`.



Wir können auch gleich Werte einsetzen mit

`new int [3] { 1, 2, 3 }`

Typ des gesamten Ausdrucks (`new...`) ist ein Pointer auf das erste Element. Wir können also mit

`int* p = new int [3] { 1, 2, 3 };`

einen Pointer `p` definieren, der auf das erste Element (1) zeigt.

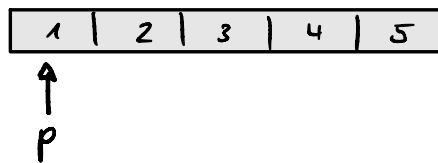
Dereferenz: Der Wert des Objekts an der Adresse auf welche ein Zeiger p zeigt, kann wie folgt ausgegeben werden

```
std::cout << *p;
```

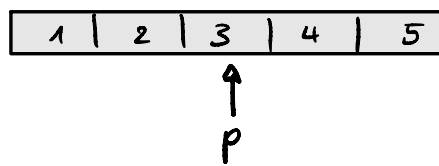
'*' wird in diesem Fall Dereferenz-Operator genannt.
(dies darf nicht mit dem Zeiger-* verwechselt werden)

nullptr: Wir können einen Zeiger auch ins nichts zeigen lassen mit $\text{int}^* p = \text{nullptr};$

Arithmetik: Sei $\text{int}^* p$ ein Pointer mit folgendem Speicher:



Mit $p+i$ ($i \in \mathbb{Z}$) können wir den Pointer nach links (bzw rechts) verschieben. Bspw $p = p+2;$



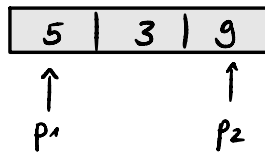
Somit $*(p+2)$ ergibt 3.

Falls zwei Pointer auf den selben Speicher zeigen, aber auf unterschiedliche Stellen, erhalten wir mit $p_1 - p_2$ genau wie weit die beiden voneinander entfernt liegen. Sei

```
int* p1 = new int[3] {5, 3, 9};
```

```
int* p2 = p1 + 2;
```

Dann sieht der Speicher wie folgt aus:



und somit $p_2 - p_1 = 2$ und $*(p_2 - (*p_1 - 4)) = 3$ (wieso?)

const: Es gibt nun die Möglichkeit, einen konstanten Pointer zu definieren. Dies geschieht mit

`int * const p;`

Wollen wir diesen Pointer auf eine konstante Zahl zeigen lassen, dann machen wir dies mit

`const int * const p;`

Bemerkung: `const int *` \Leftrightarrow `int const *`

Faustregel: Bezieht sich das `const` auf das Objekt, so steht es vor dem `*`. Bezieht es sich auf den Pointer steht es nach dem `*`.