Dynamische Speicherplatzverwaltung unter C++

- **C** : **malloc**, **calloc**, **free** : liefern nur **void *** - bzw. **char *** - Zeiger zurück; reservieren dynamischen Speicherplatz auf dem **"Heap"** ("Stack" ist schneller im Zugriff, jedoch ist der Stack stark limitiert, Größe kann eingestellt werden.**)**

  keine Konstruktor- bzw. Destruktorrufe

- C++ : **new**, **delete** sind Operatoren, reservieren dynamischen Speicherplatz auf dem **"Heap"**

  new ruft den **Konstruktor** für das Objekt

  delete ruft den **Destruktor** für das Objekt

  new liefert den korrekten **Typzeiger** zurück

  mit **new** lassen sich Objekte auch mit **variablen Feldgrenzen** während der Laufzeit instantiieren, **"späte Bindung"** von Methodenaufrufen während der Laufzeit möglich

  new liefert **0** zurück oder wirft eine **exception**, falls die Speicherplatzanforderung **nicht realisiert** wurde

Dynamische Speicherplatzverwaltung unter C++

mit **new** angeforderter Speicher darf nur mit **delete** freigegeben werden

mit **malloc**, **calloc** angeforderter Speicher darf nur mit **free** freigegeben werden

ohm *s1 = new ohm[30] ; ruft **30x** den **Defaultkonstruktor** von **class ohm** auf

ohm *s1 = new ohm[30](220.0, 10.0); Konstruktor mit Parametern ist hier nicht möglich

delete [] s1; s1 = 0; /* oder */ **s1 = nullptr**; ist die **korrekte Freigabe** des Vektors **s1**

delete s1; gibt nur das Objekt **s1[0]** frei ! ("Speicher-lack")

int *pi = new int(5); delete [] pi; pi = 0; bzw. **pi = nullptr;** erzeugt Laufzeitfehler !

int *pi = nullptr; delete pi; **delete** für Variablenwerte **0, nullptr** wird übergangen

```
int * p1 = new int[4], * p2 = new int(200);
delete [] p1, p2;                    // Fehler, nur eine Zeigervariable nach delete erlaubt !
delete [] p1; delete p2; p1=p2=0; // Korrekt, jeden Speicherbereich einzeln freigeben.
```