	Konstruktoren in der Klassendeklaration
①	`class demo {`
②	`  private:`
③	`    Typ attr1;`
④	`    Typ attr2;`
⑤	`      :`
⑥	`  public:`
⑦	`    demo (){attr1=0;  attr2=0;}`
⑧	`    demo (Typ1 var1, Typ2 var2);`
⑨	`    demo (const demo &obj);`
⑩	`    ~ demo();`
⑪	`      :`
⑫	`};`
⑬	`demo::demo (Typ1 var1, Typ2 var2){`
⑭	`     attr1=var1 ; attr2=var2; };`
⑮	`demo::demo (const demo &obj) { ... }`
⑯	`demo::~demo () { ... }`
	demo.cpp

Damit eine ordnungsgemäße Arbeit mit Objekten einer Klasse (z.B. Erzeugung und Zerstörung von Objekten) möglich wird, müssen mindestens drei spezielle Elementfunktionen immer in einer Klasse definiert sein:

- **mindestens** ein Konstruktor zur Erzeugung (Instanzierung) eines Objektes (hier in ⑦ als Default-Konstruktor und ⑧ als überladener (Parametrisierter Konstruktor)).
- **genau** ein Destruktor zur Zerstörung (Freigabe) eines Objektes (Zeile ⑩ und ⑯).
- **genau** einen Kopier-Konstruktor zur Erzeugung eines leeren Objektes und Zuweisung des Inhalts (Werte) eines konstanten Objektes (Zeile ⑨ und ⑮).

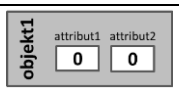
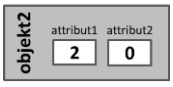
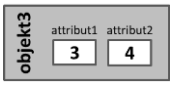
Wenn diese speziellen Klassen-Member nicht explizit implementiert wurden, erstellt das System diese bei Bedarf automatisch.
Alle *-Struktoren müssen immer den Namen der Klasse tragen und haben keinen Rückgabetyp (auch nicht **void**).

Wie im obigen Beispiel gezeigt, können Konstruktoren auch (wie Methoden oder Operatoren) überladen werden. Deklaration und Definition kann in der Klasse (inline) gemeinsam oder getrennt (extern) erfolgen.

Konstruktoren werden nicht explizit aufgerufen, sondern indirekt bei der Objekterzeugung bzw. beim Kopieren oder Löschen von Objekten. Allerdings können Konstruktor Aufrufe weitergeleitet (delegiert) werden.

	Konstruktoren weiterleiten (delegieren)
①	`class demo {`
②	`  private:`
③	`    int attr1;  int attr2;`
④	`  public:`
⑤	`    demo () : demo( 0 ){}`
⑥	`    demo (int var) : demo (var, 0 ){}`
⑦	`    demo ( int  var1,  int  var2);`
⑧	`};`
⑨	`demo::demo ( int  var1,  int  var2){`
⑩	`   attr1=var1; attr2=var2; };`
	demo.cpp

Für das Beispiel mit den Konstruktoren zeigt die Tabelle die verschiedenen Möglichkeiten der Objekterzeugung.

	Konstruktor Aufruf	Reihenfolge	erzeugte Objekte
❶	`demo objekt1;` Achtung **nicht** `demo objekt1();`	⑤ -> ⑥ -> ⑦ `demo->demo(0)->demo(0,0)`	
❷	`demo objekt2(2);`	⑥ -> ⑦ `demo(2)->demo(2,0)`	
❸	`demo objekt3(3,4)`	⑦ `demo(3,4)`	

Dabei sind folgende Regeln zu beachten:

- jeder Konstruktor wird höchsten einmal ausgeführt
- Konstruktor-Aufrufe dürfen keinen Zyklus bilden
- jeder Konstruktor darf höchstens an einen Konstruktor delegieren