Notizen AIK ProgTech 3: dynamic_cast

Klaus Kusche

Man steht oft vor folgender *Situation*:

- Man hat einen *Pointer* **Base *p** (oder ein Array von Pointern), der in Wirklichkeit auf ein Objekt der Basisklasse **Base** *oder einer beliebigen davon abgeleiteten Klasse* **Abgel1**, **Abgel2** usw. zeigen kann.
- Man möchte für das Objekt, auf das der Pointer zeigt, eine *Methode* **XMeth** aufrufen, die *nur in einer bestimmten abgeleiteten Klasse* **AbgelX** definiert ist (aber nicht in der Basisklasse) und nur auf Objekten dieser Klasse funktioniert.

Dadurch ergeben sich zwei *Probleme*:

- Zeigt **p** überhaupt auf ein **AbgelX**-Objekt, oder zeigt **p** auf ein Objekt einer anderen Klasse, das diese Methode **XMeth** gar nicht kennt?
- Wie kann ich **XMeth** für dieses Objekt aufrufen?
  Bei **p->XMeth()** beschwert sich der Compiler zu Recht,
  dass es in **Base** kein **XMeth** gibt.

In C++ gibt es *sechs verschiedene Arten von Typ-Umwandlungen* für verschiedene Zwecke, eine davon ist **dynamic_cast**.

dynamic_cast löst beide Probleme:

```
AbgelX *xp;
xp = dynamic_cast<AbgelX *>(p);
if (xp == NULL) cout << "*p ist gar kein AbgelX-Objekt" << endl;
else xp->XMeth();
```

dynamic_cast *verwandelt* einen Pointer einer Basisklasse in einen Pointer einer davon abgeleiteten Klasse und *prüft* dabei, ob der Pointer wirklich auf ein Objekt dieser abgeleiteten Klasse (oder einer davon weiter abgeleiteten Klasse) zeigt.
Zeigt der ursprüngliche Pointer auf ein Objekt einer *anderen* Klasse,
liefert **dynamic_cast** als Ergebnis einen **NULL**-*Pointer*.

Man kann mit **dynamic_cast** also einerseits prüfen, ob das Objekt zur "richtigen" Klasse gehört, und bekommt andererseits einen Pointer, mit dem man die Methode der abgeleiteten Klasse aufrufen kann.