11.Templates

Templates (oder auch Schablonen) sind ein nützliches Hilfsmittel bei der Erstellung allgemeiner Funktionen und Klassen. Sie erlauben eine Funktion oder Klasse ziemlich unabhängig von erwarteten Typen zu deklarieren und zu definieren und anschließend für spezielle Typen anzuwenden.

Ein Funktionstemplate wird auf sehr einfache Weise deklariert:

Beispiel 1. eine Template-Funktion

```
template <class T>
inline T quad(T x) { return x*x; }
```

Das Schlüsselwort template eröffnet eine Teplate-Argumentliste, die aus Bezeichnern besteht, die in der folgenden Funktionsdeklaration oder -definition als Alias für beliebige Klassen gelten wird. Der Name ist dabei unerheblich. Es ist aber üblich kurze Bezeichnungen zu verwenden und mit T zu beginnen. Die folgenden Funktionsdeklaration oder -definition (oder Klassendeklaration, Methodendefinition s. unten) kann diesen Typen als bekannt benutzen. Die Gültigkeit des Alias beschränkt sich ausschließlich auf den folgenden Block.

Operationen auf diesen Typ werden prinzipiell als vorhanden angenommen. Das wird bei der konkreten Anforderung später überprüft. Die Nutzung dieser Funktion erfolgt auf die denkbar einfachste Weise: man ruft sie einfach mit dem gewünschten Typ auf.

Beispiel 2. Aufrufe von einer template-Funktion mit verschiedenen Typen als Parameter.

```
y = quad(x);  // = 9 ok.
n = quad(m);  // = 16 ok.
b = quad(a);  //
y = quad(x++); // = 9 ok.
```

Der Compiler erzeugt für jeden Typ, der die Funktion anfordert eine eigene Kopie. Letzlich passiert also nichts anderers, als wenn man die Funktion für jeden Typ extra geschrieben hätte. Nur der Compiler macht das selbst, produziert keine weiteren Rechtschreibfehler und erzeugt nur die wirklich benötigten Funktionen.