Konstante Methoden

Konstante Methoden dürfen das Referenzelement nicht ändern. Man erkennt Sie am Schlüsselwort `const` **nach** den Funktionsklammern. Da das Referenzelement kein Funktionsparameter ist, gibt es auch keinen besseren Ort dafür. Man **sollte jede** Methode, die das Referenzelement nicht ändert, als konstante Methoden kennzeichnen: Konstante Methoden kann man nämlich auf konstante und nichtkonstante Referenzobjekte anwenden, nichtkonstante Methoden jedoch nur auf nichtkonstante Referenzelemente. 2 Methoden mit dem gleichen Namen, die sich nur im `const` unterscheiden, sind in C++ erlaubt und sind tatsächlich unterschiedliche Methoden: Für nichtkonstante Referenz-Elemente ruft C++ die nichtkonstante Methode auf, für konstante Referenzelemente die konstante Variante.

```
double abs();        // nichtkonstantes abs()
double abs() const; // konstantes abs()

Komplexe_Zahl z1{1,2};
z1.abs();            // ruft die nichtkonstante Version auf

const Komplexe_Zahl z2{1, 2};
z2.abs();            // ruft die konstante Version auf

void f(const Komplexe_Zahl& z)      //  lv-Referenz to const
{     z.abs();       // ruft die konstante Version auf
```

Manchmal will man auch für nichtkonstante Objekte die konstante Methode aufrufen (die umgekehrte Richtung ist immer verboten!). Seit C++20 ist das „ohne Gewaltanwendung“ mit `std::as_const` aus `<utility>` möglich:

vor C++20:

```
Komplexe_Zahl z1{1,2};
z1.abs();            // ruft die nichtkonstante Version auf
const Komplexe_Zahl& tmp = z;  // der Holzhammer
tmp.abs();           // ruft die konstante Version auf
```

C++20:

```
Komplexe_Zahl z1{1,2};
std::as_const(z1).abs(); // ruft die konstante Version auf
```