Die rvalue-Referenz-Übergabe

```
void f(int&& x)          // x wird als rvalue-Referenz erstellt
     // oder void f(int &&x), void f(int && x)
{
     x = 5;
}
```

Diese Übergabemethode erkennt man an den 2 Ampersand. Eine rvalue-Referenz kann sich immer nur auf einen rvalue, das ist eine Konstante oder eine **temporäre** Objektinstanz (z.B. Rechenergebnis) beziehen, aber nie auf eine von uns selbst definierte Variable (= lvalue) oder einen Teil davon! Jedes Objekt (und jeder Teil davon) mit Namen ist immer ein lvalue.

Was darf man per rvalue-Referenz übergeben (am Beispiel: `int f(int&&);`)

- Konstante: `f(1);`
- unpassende umwandelbare Variable `char i; f(i);`
- Rechenergebnisse `f(i+3); f(i++);`

Was darf man nicht übergeben:

- Variable `int i; f(i);`
- nicht umwandelbare Objekte `string i; f(i); f("text");`

```
int x = 1; f(x);          // Fehler: x ist ein lvalue (nicht temporär!)

const int c = -1; f(c);   // Fehler: c ist ein lvalue (und obendrein konstant)
f(1);                     // korrekt: Konstante ist ein rvalue
double y = 1.5; f(y);     // korrekt: falscher wandelbarer Typ ist ein rvalue
f(x + 0);                 // korrekt: Rechenergebnis ist ein rvalue
f(abs(x));                // korrekt: dieses Funktionsresultat ist ein rvalue
int x = 1; f(std::move(x));
                          // korrekt: der lvalue x wird durch std::move()
                          // zu einem rvalue umdefiniert.
```

Die const rvalue-Referenz-Übergabe

Konstante rvalue-Referenzen sind laut Standard zwar erlaubt, machen aber keinen Sinn und werden daher nie verwendet!

Die `forwarding`-Referenz-Übergabe

C++ verwendet dieselbe Syntax **`&&`** auch für die **forwarding-Referenz**. Diese kann sich sowohl auf einen lvalue als auch auf einen rvalue beziehen. Forwarding Referenzen können nur in 2 Situationen vorkommen: in Funktions-Templates oder in Verbindung mit `auto`. Zusätze wie `const` oder `volatile` sind bei forwarding Referenzen nicht möglich, sondern erzeugen ausschließlich rvalue Referenzen! Manche Programmierer nennen eine forwarding Referenz auch universelle Referenz, da sie sich auf alles beziehen kann. Sie werden daher sehr häufig in der Standard-C++-Bibliothek eingesetzt!

```
for (auto&& i : …)                          // eine forwarding-Reference
for (const auto&& i : …)                    // eine const rvalue-Referenz

template<typename T> void f(T&& t)          // eine forwarding-Reference
template<typename T> void f(const T&& t)    // eine const rvalue-Referenz
```