Die `for` Schleife

Syntax:

```
for (Init; Bedingung; Update)
        Anweisung;      // oder mit {} geklammerter Block
```

Diese `for`-Schleife lässt sich zu der folgenden äquivalenten `while`-Schleife umschreiben:

```
{
    Init;
    while (Bedingung) {
        Anweisung;
        Update;
    }
}
```

`Init`: Anweisung, **die immer ausgeführt wird** (auch wenn der eigentliche Rumpf gar nicht durchlaufen wird). Sie wird vor dem erstmaligen Check der Bedingung ausgeführt. Hier definiert man oft Zählvariablen oder Summenvariablen und/oder setzt sie auf einen Anfangswert. Wird hier eine Variable definiert, so ist diese **nur INNERHALB der ganzen `for`-Schleife bekannt (**also auch in der *`Bedingung`* und in *`Update`*). Wird eine Variable innerhalb des Schleifenrumpfs definiert, **ist sie nur dort bekannt** (kann also nicht in *`Bedingung`* oder *`Update`* *verwendet*!).

`Bedingung`: Was bei `while` in den runden Klammern steht. Die Bedingung wird hier **OHNE** die Klammern geschrieben. Eine fehlende Bedingung wird als **immer wahr** (1 bzw. `true`) interpretiert.

`Update`: Wird meist zum Erhöhen der Zählvariablen verwendet.

Beispiel: Countdown von 10:

```
for (int i{10}; i >= 0; --i)  // besser nicht: unsigned i = 10;
    cout << i << '\n';
```

Da die `for`-Schleife sehr kompakte Konstruktionen erlaubt, wird sie am häufigsten von allen Schleifentypen eingesetzt. Falls man kein *`Init`* und/oder kein *`Update`* benötigt, muss man trotzdem die Strichpunkte machen.

```
for (int i{10}; i >= 0;) // das Dekrementieren wurde
    cout << i-- << '\n';       // hierher verlagert
```

```
for (;;)                          // „ewige“ Schleife
```

Beispiel: Einlesen von Zahlen in einen `vector<int>`:

```
vector<int> my_ints;              // definiere hier den Container
for (int x; cin >> x;)            // solange x gelesen wurde
     my_ints.push_back(x);        // an den Container anhängen
```

Warnungsbeispiele:

1)
```
for (int i{10}; i >= 0; --i)
     cout << i << '\n';
cout << "Die Schleife endete bei " << i;     // Fehler!!
```

Die Variable `int i` ist nur **innerhalb der gesamten Schleife definiert**!

2)
```
int i{10};
for (; i >= 0; --i)
     cout << i << '\n';
cout << "Die Schleife endete bei " << i;     // ok!!
```

Die Variable `int i` ist vor der Schleife definiert und existiert ab dort **innerhalb und auch nach** der Schleife!

3)
```
int i{10};
for (int i{10}; i >= 0; --i)
     cout << i << '\n';
cout << "Die Schleife endete bei " << i;     // ??????
```

Vorsicht: Hier handelt es sich um **2 verschiedene** Instanzen `int i`:

a) Das `i` vor der Schleife ist innerhalb der Schleife nicht verfügbar, da es dort vom anderen `i` verdeckt ist. Nach der Schleife ist es wieder sichtbar und hat seinen alten Wert 10.

b) Das in der Schleife definierte `i` hört am Schleifenende auf zu existieren.

4)
```
for (; !finished; {  // Fehler: finished ist hier undefiniert
     bool finished{false};
     ...
```