Statische Variablen in Funktionen

Variablen in Funktionen, die mit dem Schlüsselwort *static* deklariert wurden, sind statische Variablen. Initialisiert werden diese Variablen beim ersten Aufruf und ihr Wert geht beim erneuten Aufruf der Funktion, in der die Varibale definiert wurde, nicht verloren. Das selbe könnte man auch mit globalen Variablen erreichen. Der Unterschied ist jedoch, dass statische Variablen nur in dem Block sichtbar sind, in dem sie auch definiert wurden.

Beispiel:

```
int globalVar = 1;

void foo() {
  int localVar = 1;
  static int statVar = 1;

  cout << „lokal: “ << localVar;
  cout << „ / global: “ << globalVar;
  cout << „ / statisch: “ << statVar;

  localVar++;
  globalVar++;
  statVar++;
}

int main() {
  foo();
  foo();
  foo();

  // localVar = 6;    // Compiler-Fehler weil Variable nicht sichtbar
  globalVar = 6;
  // statVar = 6;     // Compiler-Fehler weil Variable nicht sichtbar

  foo();
  foo();
}
```

Ausgabe:

```
lokal: 1 / global: 1 / statisch: 1
lokal: 1 / global: 2 / statisch: 2
lokal: 1 / global: 3 / statisch: 3
lokal: 1 / global: 6 / statisch: 4
lokal: 1 / global: 7 / statisch: 5
```

Statische Variablen/Funktionen in Klassen

Statische Variablen in Klassen haben mit solchen in Funktionen wenig gemeinsam. Variablen in Klassen die mit dem Schlüsselwort *static* gekennzeichnet werden, existieren pro Klasse genau einmal. Im Gegensatz dazu werden normale Variablen in Klassen für jedes neue Objekt der Klasse erzeugt. Zum Einsatz kommen statische Variablen z.B. dann, wenn man die Instanzen einer Klasse zählen möchte. Aufpassen muss man jedoch bei der Initialisierung einer statischen Variable. Diese dürfen nämlich nicht im Konstruktor initialisiert werden, da sie sonst jedes Mal beim Anlegen einer neuen Instanz zurückgesetzt werden würden. Stattdessen muss man sie ähnlich zu globalen Variablen initialisieren. Mit dem Unterschied, dass man den Klassennamen gefolgt von zwei Doppelpunkten dem Variablennamen voranstellt.

Beispiel:

```
class Foo {
  public:
    Foo() { cnt++; }       // Zähler der bei jedem Aufruf des
                           // Konstruktors inkrementiert wird,
    ~Foo() { cnt--; }      // und im Destruktor wieder dekrementiert wird

    static int cnt;
  ...
}

int Foo::cnt = 0;          // Initialisierung
```

Statischen Funktionen gehören wie statische Variablen nicht zu einem Objekt, sondern zur jeweiligen Klasse. Dadurch können solche Funktionen auch nicht auf die Datenelemente eines Objektes zugreifen. Stattdessen kann man sie als Schnittstelle für statische Variablen benutzen. So kann z.B. der zuvor deklarierte Instanz-Zähler privat deklariert werden, damit dieser nicht von außen verändert werden kann. Damit man trotzdem Zugriff auf den Zähler hat verwendet man eine statische Funktion. Aufgerufen werden solche Funktionern wieder mit dem Voranstellen des Klassennamens mit anschließendem Doppelpunkt.

Beispiel:

```
class Foo {
  public:
    Foo() { cnt++; }
    ~Foo() { cnt--; };
    static int getInstances() { return cnt; }
  private:
    static int cnt;
  ...
}

int Foo::cnt = 0;

int main() {
```

```
  ...
  cout << Foo::getInstances() << endl;
  ...
}
```