Kapitel 14

Variablen, typedef, enum

Gültigkeitsbereiche von Variablen

- Jede Variable hat einen Gültigkeitsbereich.
- Innerhalb dieses Gültigkeitsbereichs kann auf die Variable zugegriffen werden.
- Es werden folgende Arten von Variablen unterschieden:
  - globale Variablen
  - funktionslokale Variablen
  - blocklokale Variablen

Globale Variablen

- Globale Variablen sind an einer beliebigen Stelle innerhalb einer Datei, aber außerhalb eines Blocks {} definiert.
- Sie sind ab der Stelle in der Datei gültig, an der sie definiert sind.
- Globale Variablen werden automatisch beim Programmstart mit 0 initialisiert.
- Ihr Gültigkeitsbereich endet am Dateiende.

Funktionslokale Variablen

- Funktionslokale Variablen sind an einer beliebigen Stelle innerhalb einer Funktion definiert.
- Ihr Gültigkeitsbereich ist auf die Funktion beschränkt, in der sie definiert sind.
- Sie werden nicht automatisch initialisiert. Ihre Namen können in unterschiedlichen Funktionen gleich sein.
- Ist der Name einer funktionslokalen Variable gleich dem einer globalen Variable, so überdeckt die funktionslokale Variable die globale Variable.

Blocklokale Variablen

- Blocklokale Variablen sind innerhalb eines Blocks innerhalb des Funktionsblocks definiert.
- Sie haben die gleichen Eigenschaften, wie funktionslokale Variablen, allerdings auf ihren Block beschränkt.
- Blocklokale Variablen überdecken gleichnamige globale und funktionslokale Variablen.

Beispiel

```
int g;                          // globale Variable
void main()
{
        int f = 0;              // funktionslokale Variable
        {
                int b = 0;      // blocklokale Variable
        }
}
```

Gültigkeitsbereichsoperator ::

- Werden globale Variablen durch funktionslokale oder blocklokale Variablen verdeckt, kann mit dem Gültigkeitsbereichsoperator vor dem entsprechenden Namen der Variablen trotzdem auf die globalen Variablen zugegriffen werden.
- Dieser Zugriff funktioniert aber nur auf globalen Variablen.

Gültigkeitsbereichsoperator ::

```
int g=1;
void main()
{
        int f = 2;
        {
                int b=3, g=4;
                cout << setw(2) << ::g << setw(2) << g << endl;
        }
}
```

Ausgabe: `1  4`

Umbennung von Datentypen - *typedef*

- In C++ kann man Datentypen einen neuen Namen geben:

```
typedef unsigned long int ULONG;

ULONG i;
```

Aufzählungen - *enum*

- Eine Aufzählung ist ein selbst definierter ganzzahliger Datentyp.
- Die Definition einer Aufzählung erfolgt über das Schlüsselwort *enum*.
- Es werden mögliche Werte und Namen für diese Werte festgelegt.
- Die in der Liste angegebenen Namen bezeichnen jetzt ganzzahlige Konstanten.
- Die erste Konstante hat den Wert 0, die zweite den Wert 1, etc.

Aufzählungen - *enum*

```
#include <iostream>
using namespace std;
enum Automarke {Audi, BMW, Ford, Mercedes, Opel, Porsche};
void main()
{
    Automarke marke;
    marke = Porsche;
    cout << marke << endl;
}
```

Ausgabe: 5