C++: Operatoren

Carsten Gips (HSBI)

Überladen von Operatoren *in* Klassen

```
MyString a, b("hallo");
a = b;       // ???
```

Überladen von Operatoren *in* Klassen

```
MyString a, b("hallo");
a = b;       // ???
```

```
a.operator=(b);
```

Überladen von Operatoren *in* Klassen

```
MyString a, b("hallo");
a = b;      // ???
```

```
a.operator=(b);
```

```
class MyString {
    MyString &operator=(const MyString &s) {
        if (this != &s) {
            // mach was :-)
        }
        return *this;
    }
};
```

Überladen von Operatoren *außerhalb* von Klassen

```
MyString a("hallo");
cout << a << endl;
```

Überladen von Operatoren *außerhalb* von Klassen

```
MyString a("hallo");
cout << a << endl;
```

```
class MyString {
    ostream &operator<<(ostream &o) { return o << str; }
};
```

Überladen von Operatoren *außerhalb* von Klassen

```
MyString a("hallo");
cout << a << endl;
```

```
class MyString {
    ostream &operator<<(ostream &o) { return o << str; }
};
```

- Erinnerung: `cout << a` entspricht `cout.operator<<(a)`
  - Operator kann nicht in `MyString` überladen werden!
  - Klasse `ostream` müsste erweitert werden
    => Geht aber nicht, da System-weite Klasse!

=> Lösung: Operator **außerhalb** der Klasse überladen => 2 Parameter

Überladen von Operatoren *außerhalb* von Klassen (cnt.)

```
ostream &operator<<(ostream &out, const MyString &s) {
    return out << s.str;
}
```

- Nachteil: Benötigt Zugriff auf Klassen-Interna
  - entweder umständlich über Getter-Funktionen
  - oder als `friend` der Klasse `MyString` deklarieren

Anmerkung: Rückgabe der Referenz auf den Stream erlaubt die typische Verkettung:
`cout << s1 << s2 << endl;`

Meine Freunde dürfen in mein Wohnzimmer

```
void test();

class TestDummy {
    int ganzTolleMethode();
};


class Dummy {
    private:
        int *value;

    friend class TestDummy;
    friend int TestDummy::ganzTolleMethode();
    friend void test();
};
```

(Fast) alle Operatoren lassen sich überladen

- **Ausnahmen**:
  1. `.`
  2. `::`
  3. `?:`
  4. `sizeof`

- Anmerkungen:
  - Beim Überladen muss die Arität erhalten bleiben
  - Nur *existierende* Operatoren lassen sich überladen
    => Es lassen sich keine neuen Operatoren erschaffen

Vgl. Tabelle 9.1 (S. 318) im Breymann (2011)

Implizite Typkonvertierungen bei Aufruf

```
MyString s;
s != "123";      // ???
"123" != s;      // ???
```

- Operatoren **in** Klasse überladen: Typ der linken Seite muss **exakt** passen
- Operatoren **außerhalb** überladen: Konvertierung auf *beiden* Seiten möglich

Anmerkung zu “++” und “--” Operatoren: Präfix und Postfix

- Präfix: `o1 = ++o2;`
  - Objekt soll **vor Auswertung** inkrementiert werden
  - Signatur: `Typ &operator++()`
- Postfix: `o1 = o2++;`
  - Objekt soll erst **nach Auswertung** inkrementiert werden
  - Signatur: `Typ operator++(int)`

Weitere Anmerkungen

- Operatoren werden **nicht** vom System zusammengesetzt
- Operatoren lassen sich in C++ verketten:

```
Dummy a(0); Dummy b(1); Dummy c(2);
a = b = c;  // a.operator=(b.operator=(c));
```

- Übertreiben Sie nicht!

```
Firma f;
Person p;
f += p;  // ??!
```

Wrap-Up

- Überladen von Operatoren (innerhalb und außerhalb einer Klasse)
  - Innerhalb: 1 Parameter (Objekt auf der rechten Seite)
  - Außerhalb: 2 Parameter
- Zugriff auf Attribute: `friend` einer Klasse
- Implementierung von Post- und Präfix-Operatoren

LICENSE