Informatik IIa - Praktikum

Oliver Jack
Fachhochschule Jena
Fachbereich Elektrotechnik und Informationstechnik

Sommersemester 2010

Praktikum 7: Konstruktoren und Zuweisung in abgeleiteten Klassen

Kopierkonstruktor und Zuweisungsoperator in abgeleiteten Klassen

Kopierkonstruktor in abgeleiteten Klassen

Selbst definiert: Die Basisanteile der abgeleiteten Klassen werden bei einem selbst definierten Kopierkonstruktor nur dann vom Kopierkonstruktor der Basisklasse belegt, wenn dies explizit so programmiert wurde.

Automatisch erzeugt: Die Basisanteile der abgeleiteten Klassen werden von einem vom Compiler generierten Kopierkonstruktor durch den Kopierkonstruktor der Basisklasse belegt.

Zuweisungsoperator in abgeleiteten Klassen

Selbst definiert: Die Basisanteile der abgeleiteten Klassen werden bei einem selbst definierten Zuweisungsoperator nur dann vom Zuweisungsoperator der Basisklasse belegt, wenn dies explizit so programmiert wurde.

Automatisch erzeugt: Die Basisanteile der abgeleiteten Klassen werden von einem vom Compiler generierten Zuweisungsoperator durch den Zuweisungsoperator der Basisklasse belegt.

Muster für Kopierkonstruktor und Zuweisungsoperator abgeleiteter Klassen Wenn eine abgeleitete Klasse eigene Komponenten hat, dann müssen sie in einem definierten Copy-Konstruktor und Zuweisungsoperator kopiert werden.

```
class Basis { /*... irgend etwas ...*/ };
// Muster zur Kopie abgeleiteter Klassen
class Abgeleitet : public Basis {
private:
  X x; // eigener Anteil
public:
// Muster Copy-Konstruktor einer Ableitung -------
  Abgeleitet (const Abgeleitet & a)
  : Basis(a), // Basisanteil kopieren
    x(a.x) // eigene Komponente x kopieren
  { }
 };
```

```
// Muster Zuweisungs-Operator einer Ableitung -------
//
  Abgeleitet & operator= (const Abgeleitet & a) {
    if ( this != &a ) {
      this->Basis::operator=(a); // Basisanteil kopieren
      x = a.x; // eigene Komponente kopieren
    }
    return *this;
  }
};
```

Beachten Sie, wie hier mit dem Bereichsoperator (Basis::) auf eine verdeckte Methode der Basisklasse zugegriffen wird.

Aufgabe 1 (Zuweisungsoperator in abgeleiteten Klassen): Betrachten Sie das folgende Programm

```
#include <iostream>
using namespace std;
class Basis {
private:
  int b;
public:
  Basis() : b(0) {}
  Basis(int p_b): b(p_b) {}
  Basis & operator= (const Basis & p_basis) {
  if ( this != &p_basis )
     b = p_basis.b;
  return *this;
  }
  void schreib () {
     cout << "␣b␣=␣" << b << endl;
  }
};
```

```
class Ab : public Basis {
private:
  int a;
public:
  Ab() : a(0) {}
  Ab(int p_a) : Basis(p_a), a(p_a) {}
  Ab & operator= (const Ab & p_ab) {
    if ( this != &p_ab )
      a = p_ab.a;
    return *this;
  }
  void schreib () {
    Basis::schreib();
    cout << "␣a␣=␣" << a << endl;
  }
};
```

```
int main () {
  Ab ab1(7), ab2(13);
  ab1 = ab2;
  cout << "ab1:␣\n";
  ab1.schreib();
  cout << "ab2:␣\n";
  ab2.schreib();
```

```
  return (0);
}
```

a) Welche Ausgabe erzeugt das Programm?

b) Die Ausgabe ist nicht, wie für den Zuweisungsoperator beabsichtigt. Erklären Sie die Ausgabe.

c) Korrigieren Sie den Zuweisungsoperator der abgeleiteten Klasse so, dass die Ausgabe korrekt ist.

d) Wäre der automatisch vom Compiler erzeugte Zuweisungsoperator für die abgeleitete Klasse korrekt?

Konstruktoren und Destruktoren in abgeleiteten Klassen

Aufgabe 2 (Konstruktoren und Destruktoren in abgeleiteten Klassen):

- Implementieren Sie drei Klassen `Base`, `Derived1`, `Derived2`, wobei `Derived1` von `Base` und `Derived2` von `Derived1` abgeleitet sind.
- Implementieren Sie für jede der Klassen einen Konstruktor und einen Destruktor, der jeweils auf der Konsole ausgibt, dass er aufgerufen wurde.
- Implementieren Sie ein Hauptprogramm, in dem Objekte der Klassen `Derived1` und `Derived2` innerhalb einer zwei-mal durchlaufenen Schleife deklariert werden.
- Welche Ausgabe erzeugt Ihr Programm?
- Wieviele Konstruktor- und Destruktor-Aufrufe erfolgen für jede der Klassen?