Strukturierte Datentypen und Klassen

Aufgabe 3.1 Führen Sie folgendes Programm in Ihrem Kopf aus. Welche Ausgabe erwarten Sie?

```
struct Obj {
  Obj()                          { std::cout << "default-construct "; }
  Obj(const Obj& o)              { std::cout << "copy-construct "; }
  Obj& operator=(const Obj& o) { std::cout << "assign "; return *this; }
  ~Obj()                         { std::cout << "destroy "; }
};

int main() {
  Obj o1;
  Obj* o2 = new Obj;
  { Obj o3 = o1; }
  Obj* o4 = new Obj(o1);
  o4 = &o1;
  delete o2;
  o1 = *o4;
}
```

Aufgabe 3.2 Die Klasse `std::unique_ptr<int>` aus der Standardbibliothek dient dazu, einen auf dem Heap allozierten `int` (oder auch ein Objekt eines beliebigen anderen Typs) zu verwalten und automatisch und zur rechten Zeit wieder freizugeben.

Erklären Sie in Ihren eigenen Worten, wie dieses gewünschte Verhalten der Klasse realisiert wird. Erklären Sie insbesondere, welche Member-Variablen die Klasse hat und wie die Konstruktoren und der Destruktor implementiert sind. Wie sieht es aus mit dem Kopierkonstruktor und dem Zuweisungsoperator? Sind die überhaupt sinnvoll implementierbar?

Aufgabe 3.3 Implementieren Sie eine Klasse `ScopeLogger`, die verwendet werden kann, um beim Betreten und beim Verlassen eines Scopes automatisch eine entsprechende Meldung auszugeben. Beispielsweise soll folgendes Programm links, in welchem der Anfang von jedem Scope mit der Definition eines entsprechenden `ScopeLogger`-Objekts markiert wurde, die rechts gezeigte Ausgabe produzieren.

```
class ScopeLogger {
  // Ihr Code kommt hier hin
};

void parity(int x) {
  ScopeLogger sl("parity");
  if(x % 2 == 0) {
    ScopeLogger sl("parity_if");
    cout << "Even number" << endl;
  }
  else {
    ScopeLogger sl("parity_else");
    cout << "Odd number" << endl;
  }
}
int main() {
  ScopeLogger sl("main");
  parity(1);
  parity(2);
}
```

```
. Enter scope main
.. Enter scope parity
... Enter scope parity_else
Odd number
... Leave scope parity_else
.. Leave scope parity
.. Enter scope parity
... Enter scope parity_if
Even number
... Leave scope parity_if
.. Leave scope parity
. Leave scope main
```

Versuchen Sie es zuerst ganz ohne die Punkte an den Zeilenanfängen. Für die Punkte werden sich statische Klassenvariablen und der Konstruktoraufruf `std::string(n, '.')` für eine Ganzzahl `n` als nützlich herausstellen.