Übungen zur Vorlesung

"Objektorientiertes Programmieren im Wissenschaftlichen Rechnen"

Dr. O. Ippisch, Dr. C. Engwer

Abgabe am 04. 05. 2010 in der Vorlesung

ÜBUNG 1 VERKETTETE LISTE (CONST-CORRECTNESS)

In der letzten Übung haben sie eine verketteten Liste programmiert um das Zusammenspiel von Konstruktoren, Destruktoren und Zeigern zu üben.

Wir wollen nun diese Implementierung um das Konzept von `const` und eine saubere Kapselung erweitern.

1. Die Klasse `Node` hatte bisher nur `public` Member. Damit der `Node * next` Pointer nicht unkontrolliert geändert werden kann sollte er jedoch privat sein.
   - Ändern sie die Implementierung entsprechend ab, so dass `Node * next` privat ist.
   - Wie können sie jetzt der `List` Klasse zugriff auf den private `next` Pointer gewähren?
2. Erweitern Sie die Klasse `List` um das `const` Konzept.
   - Welche Methoden sollten `const` sein und welche nicht?
   - Welche Typen sollten die Parameter und Rückgabewerte sinnvollerweise haben.
3. Eine Funktion zur Ausgabe der Liste könnte folgendermaßen aussehen:

   ```
   void printList (const List & list)
   {
       for (const Node * n = list.first(); n != 0; n = list.next(n))
           std::cout << n->value << std::endl;
   }
   ```

   Testen sie ihre Implementierung mit dieser Funktion.
4. Schreiben sie eine freie Funktion

   ```
   void append (List & list1, const List & list2);
   ```

   die alle Einträge aus `list2` kopiert und an `list1` anhängt.

6 Punkte

ÜBUNG 2 VERKETTETE LISTE (MIN/MAX)

Die verkettete Liste soll Methoden erhalten, die das kleinste oder das größte Element finden:

```
const Node * findMin() const;
const Node * findMax() const;
```

1. Schreiben sie sich eine freie Funktion `void testListMinMax()`, mit der sie anschließend ihre Implementierung testen:

- was sind gute Testfälle?
- an welche Seiteneffekte muss man denken?
- ändern sie die Liste auch zwischendrin.

2. Implementieren sie die Methode `findMin` und `findMax`. Modifizieren sie keine anderen Methoden.

3. Ergänzen sie die Implementierung um *Caching*.

   - Die Suche nach dem größten oder kleinsten Eintrag ist $O(N)$, wenn N die Länge der Liste ist.
   - Wenn min/max mehrfach aufgerufen wird, soll nur einmal das entsprechende Element gesucht werden, der Wert soll bei der ersten Auswertung gecached werden.

6 Punkte

ÜBUNG 3 KNOBELAUFGABE

Probieren Sie es ruhig praktisch aus.

Sie haben folgendes Programm:

```
void foo ( const int ** );

int main()
{
    int ** v = new int[10];
    foo(v);

    return 0;
}
```

der Compiler wird ihnen nun einen Fehler melden, weil sie `int **` nach `const int **` umwandeln:

```
g++ test.cc -o test
test.cc: In function 'int main()':
test.cc:6: error: invalid conversion from
  'int**' to 'const int**'
test.cc:6: error:   initializing argument 1
  of 'void foo(const int**)'
```

- Eigentlich kann man doch immer von nicht-`const` nach `const` umwandeln...
- warum geht das hier nicht?

Tipp:

Warum folgendes Programm nicht kompiliert ist klar:

```
const int * bar ();

int main()
{
    int ** v = new int[10];
    v[0] = bar();

    return 0;
}
```

Was hat dieses Programm mit dem obigen gemein?

6 Punkte

ÜBUNG 4 CONSTNESS

Sie habe eine Liste von Funktionenprototypen, einige Variablen und Zuweisungen. Welche Konstruktionen sind nicht zulässig und warum?

```
int foo ( const int & );
int bar ( int & );

int main()
{
  int i = 0;
  int & j = i;
  static const f = i;
  int * const p = 0;
  p = &i;
  * p = f;
  const int & l = j;
  const int & k = f;
  foo ( j );
  bar ( l );
  foo ( k );
}
```

2 Punkte