Grundlagen der C++-Programmierung

Assignment due Tuesday May 01 (23:59)

Assignment 4 - ScopeTimer

Implementiere eine Klasse `ScopeTimer`, die die Zeit zwischen Aufruf des Konstruktors und des Destruktors misst. Nutze diese sinnvoll, um die Laufzeit der Funktionen in `main.cpp` zu messen.

Der Konstruktor bekommt den Namen des `ScopeTimer`s als C-String (`const char*`). Er ermittelt die aktuelle Zeit und speichert sie in der Klasseninstanz. Außerdem gibt er eine Meldung nach `std::cout` aus, dass der Timer gestartet wurde.

Der Destruktor misst erneut die Zeit, ermittelt die Distanz zur Startzeit, und gibt das Ergebnis in Millisekunden zusammen mit dem Namen des Timers nach `std::cout` aus.

Verändere `main.cpp` so, dass `ScopeTimer` genutzt wird, um die Laufzeit des gesamten Programms, sowie jeder der drei Funktionen zu messen.

Welche Klassen-Invarianten hat `ScopeTimer`? Sorgt Deine Klasse dafür, dass diese Invarianten immer erhalten bleiben?

Hinweise

Du kannst die Funktion `std::clock()` aus dem Header `<ctime>` der Standardbibliothek verwenden, um Zeit zu messen. `std::clock()` liefert die (CPU-)Zeit, die ein Prozess seit dem Start benötigt hat in Vielfachen von `CLOCKS_PER_SEC`. Der Rückgabetyp `std::clock_t` ist ganzzahlig.

Zusatzaufgabe

Sorge dafür, dass Ausgaben von verschachtelten `ScopeTimer`s eingerückt werden. Die Ausgabe von drei verschachtelten `ScopeTimer`s könnte etwa so aussehen:

```
 Starting Timer öuter$$tarting Timer middle$$tarting Timer ïnnerTimer
ïnnertook 52 ms Timer middletook 153 ms Timer öutertook 221 ms
```

Optional

Schau Dir die folgenden Regeln aus den C++ Core Guidelines an. Diese Guidelines erwähnen teilweise Sprachfeatures und Begriffe, die Du wahrscheinlich noch nicht kennst. Lass dich davon nicht stören und lese einfach nur das, was Du verstehst.

- C.40: Define a constructor if a class has an invariant
- C.41: A constructor should create a fully initialized object
- C.30: Define a destructor if a class needs an explicit action at object destruction

- C.31: All resources acquired by a class must be released by the class's destructor
- R.5: Prefer scoped objects, don't heap-allocate unnecessarily
- F.5: If a function is very small and time-critical, declare it inline