Programmieren C++: Exceptions, Stringstreams

Klaus Kusche

1.) Exceptions mit eigener Fehlerobjekt-Klasse

Auf meiner Homepage findest du eine C++-Variante unseres Programms "Abstand in Tagen zwischen zwei Datumsangaben". Mache dieses Programm benutzerfreundlicher:

- Derzeit wird bei einer falschen Eingabe die Funktion **`EingabeFehler`** aufgerufen, die eine Fehlermeldung ausgibt und das Programm mit **`exit(1)`** *beendet*:
  Nach einem Eingabefehler muss man das Programm neu starten und (falls der Fehler beim zweiten Datum passiert ist) auch das erste Datum nochmals eingeben.
- Ziel ist, den Benutzer bei einer falschen Eingabe *erneut zur Eingabe* nur des fehlerhaften Datums aufzufordern anstatt das Programm gleich zu beenden.

  Da die Fehlererkennung aber 1-3 Funktionen "weiter innen" als die Eingabe liegt, würde eine konventionelle Fehlerbehandlung (mit speziellem Returnwert o.ä.) einen größeren Umbau der Programmlogik erfordern.

  Gefordert ist aber, die normale *Programmlogik möglichst unverändert* zu lassen und für den Fehlerfall Exceptions zu verwenden, um von der Stelle, an der ein Fehler erkannt wird, *zurück in die Eingabe-Funktion* **`liesDatum`** zu kommen.

Im Detail sind folgende Schritte notwendig:

- Definiere eine **eigene Klasse** **`falschesDatum`** für die Objekte deiner Exceptions (sie braucht *nicht* von der C++-Standardklasse **`exception`** abgeleitet sein):
  - Die Klasse enthält zwei private Member-Variablen:
    Eine für den *Fehlermeldungstext* (**`const char *`**)
    und eine für den *falschen Wert* (den ungültigen Tag / Monat / ...: **`int`**).

    Beide werden durch einen *Konstruktor mit 2 Argumenten* gesetzt.
  - Weiters enthält die Klasse eine Methode **`print`**, die den Text und den falschen Wert in einer schönen Fehlermeldung auf **`cout`** *ausgibt* (also nicht gleich beim Erzeugen des Fehlers eine Meldung ausgeben!).
- Entferne die bisherige Fehler-Funktion **`EingabeFehler`** und *ersetze* alle Aufrufe der Funktion durch ein **`throw`** eines neuen Objektes der Klasse **`falschesDatum`**.

  Abgesehen davon sollen die bestehenden Datums-Funktionen *unverändert* bleiben!
- Baue die Funktion **`liesDatum`** so um, dass sie den Aufruf von **`TageBisDatum`** auf Exceptions prüft:
  - Auftretende Exceptions werden *gefangen*, zur *Ausgabe einer Fehlermeldung* wird *für das gefangene Fehlerobjekt* die **`print`**-Methode aufgerufen.
  - Die Ausgabe des Prompts, das Einlesen des Datums von **`cin`** sowie der Aufruf von **`TageBisDatum`** werden in einer *Schleife* so lange wiederholt, bis **`TageBisDatum`** *ohne Auftreten einer Exception* ein Ergebnis liefert (das erkennst du daran, dass ein Wert in **`tage`** *gespeichert* wird!).

2.) Exceptions mit Strings, Meldungs-Formatierung mit Stringstream

Die Lösung aus dem vorigen Punkt kann man in zweierlei Hinsicht verbessern:

- Unsere Fehlerklasse kann die Fehlermeldung derzeit nur mittels **`print`** auf **`cout`** ausgeben: Es ist im Exception-Handling *nicht möglich*, den *Fehlertext* z.B. stattdessen in einen Logfile zu schreiben oder sonstwie *als String zu verarbeiten*.[1]
- Die *eigene Fehlerklasse* ist für das bisschen Funktionalität ziemlich *viel Aufwand*, eigentlich wollen wir ja nur den Fehlertext vom Ort des Fehlers zum **`catch`** transportieren.

Entferne daher die eigene Fehlermeldungs-Klasse wieder
und baue die Lösung aus dem vorigen Punkt auf eine *andere Exception-Variante* um:

- Statt eines Objektes einer selbstdefinierten Fehlerklasse *werfen* wir einen simplen *C++-String* (also Typ **`string`** aus dem Header **`<string>`**, *nicht* **`char *`** C-String), der die *fertig formatierte Fehlermeldung* (incl. Wert und Hilfe-Text) enthält.
- Im **`catch`** könnten wir mit dem gefangenen String machen, was wir wollen. In unserem Fall geben wir ihn einfach auf **`cout`** aus.
- Unser "**`falschesDatum`**", das beim Auftreten eines Fehlers aufgerufen wird, bleibt *vom Aufruf her unverändert* (selber Name, selbe Argumente), ist aber nicht mehr der Konstruktur einer selbstdefinierten Klasse, sondern eine *globale Funktion*, die einen **`string`** als *Returnwert* liefert.

  Dieser String vom Typ **`string`** soll die fertig formatierte Fehlermeldung enthalten, mit eingesetztem Fehlertext, Fehlerwert und Hilfe-Text.

  Er entsteht, indem man einen *Stringstream anlegt*, die *Fehlermeldung* mit den beiden Werten mit **`<<`** *in den Stringstream ausgibt* (so wie sie bisher auf **`cout`** ausgegeben wurde), und den *internen String* (**`.str()`**) des Stringstream-Objektes *als Returnwert* zurückgibt.

  In diesem Fall ist es notwendig, wirklich das **`string`**-Objekt selbst (*als Kopie*) zurückzugeben (auch wenn der Compiler eine Warnung dafür liefert!) und *nicht* eine Referenz oder einen Pointer auf den String (das ginge ins Leere, weil Stringstream und String nach dem **`return`** weg sind).

---

1 Alle von der *vordefinierten Klasse* **`exception`** abgeleiteten Exception-Klassen erben zu diesem Zweck eine *virtuelle Methode* **`what`**, die eine *Beschreibung der Exception als String* zurückliefern soll. Jede Exception-Klasse muss **`what`** selbst geeignet implementieren, wobei intern praktisch immer *Stringstreams* zum Zusammenbasteln des Textes verwendet werden (so wie in unserem Beispiel).