Inf Programmieren 1 Übung: Exceptions

Klaus Kusche

Mehrere Messwert-Reihen einlesen, Mittelwerte berechnen

Wir wollen ein Programm schreiben, das *zeilenweise* Input verarbeitet.
Der Input kann *drei verschiedene Arten von Zeilen* enthalten:

- Komplett *leere Zeilen* (nur **`'\n'`**):

  Eine komplett leere Zeile *beendet* das Programm.

  Vor dem Programmende wird noch das Ergebnis
  der gerade laufenden Mittelwertberechnung ausgegeben
  (außer der Input hat noch gar keine **`'#'`**-Zeile enthalten).

- Zeilen, die mit **`'#'`** beginnen:

  Sie beginnen eine *neue Mittelwert-Berechnung* und enthalten nach dem **`'#'`**
  den *Beschriftungstext*, der bei der Ausgabe vor dem Mittelwert aufscheinen soll.

  Auch hier wird zuerst das Ergebnis der gerade laufenden Mittelwertberechnung
  ausgegeben, falls es sich nicht um die erste **`'#'`**-Zeile im Input handelt.

  Die erste Zeile im Input (bevor Zahlen-Zeilen kommen) muss eine **`'#'`**-Zeile sein.

- Alle anderen Zeilen

  … müssen genau *eine Gleitkomma-Zahl und sonst nichts*
  (außer ev. Zwischenräumen und Tabs) enthalten.

  Die Zahl wird zur laufenden Mittelwert-Berechnung dazugezählt.

Ausgegeben werden soll für jede Zeilengruppe der *Text* aus der **`'#'`**-Zeile
und der *Mittelwert* der darauffolgenden Zahlen-Zeilen.

Beispiel:

```
#Linz
     11
     9
     7
#Wien
#Graz
     -0.5
     +2.5
```

liefert

```
Linz: Mittelwert 9
Wien: Keine Messwerte
Graz: Mittelwert 1
```

In der Praxis würde die Eingabe wohl von einem File und nicht vom Terminal kommen
(und die Ausgabe würde in einen File geschrieben werden),
aber wir sparen uns die Mühe des File-I/O's und lesen von **`cin`** bzw. schreiben auf **`cout`**.

Wir wollen den Input genau *auf Fehler prüfen*. Folgende Fehler können auftreten:

- Die *erste* Zeile ist eine Zahlen-Zeile.
- Eine Zeile beginnt *nicht* mit **'#'** (und ist auch nicht leer), aber enthält *keine* Zahl.
- Eine Zeile enthält eine Zahl, aber davor oder danach noch andere Zeichen außer Zwischenraum und Tab.

Weiters wollen wir zur Übung den Programmablauf weitestgehend über Exceptions steuern:

- Wir verpacken das *Einlesen einer Zeile* in eine *Funktion* **getNum**.
- Die Funktion hat nur einen *Returnwert*: Die *eingelesene Zahl* (als **double**).
  Sie hat keine Ausgangs-Parameter für die anderen Fälle oder Fehler,
  und sie macht selbst auch keine Fehlerbehandlung oder -ausgabe.
- Dafür kann sie *zwei Exceptions* werfen:
  - Ein Objekt der selbstdefinierten Klasse **SyntaxError**
    (bei einem Syntaxfehler).
  - Einen C++-**string** (bei einer **'#'**-Zeile oder leeren Zeile: Die gelesene Zeile).
- Die Funktion hat einen **bool**-Parameter, der anzeigt, ob eine Zahlen-Zeile erlaubt ist oder zuerst einmal die allererste **'#'**-Zeile kommen muss.

Im Falle eines **SyntaxError** wird *nur die aktuelle Zeile ignoriert*, das Programm macht mit demselben Mittelwert (samt bisherigen Daten) und der nächsten Eingabezeile *weiter*.
Im Falle einer **'#'**-Zeile oder Leerzeile wird hingegen die *aktuelle Mittelwertberechnung* abgeschlossen und *ausgegeben*, und das Programm endet (bei einer Leerzeile)
oder initialisiert die nächste Mittelwertberechnung (bei **'#'**).

Hinweise:

- Abgesehen von **getNum** habe ich im ganzen Programm generell
  nur die C++-Klasse **string** statt den C-Strings **char *** verwendet
  (damit geht es in diesem Fall etwas leichter).
- **SyntaxError** ist eine *selbstdefinierte Klasse* nur für das **throw**:
  - Da **what()** etwas schwierig wäre (man bräuchte Stringstreams dafür),
    leiten wir sie *nicht* von der Standard-Klasse **exception** ab.
  - Dafür definieren wir einen **operator<<** für Objekte dieser Klasse
    als globale Funktion (**friend** verwenden!):
    Die im Objekt gespeicherten Daten werden ausgegeben.
  - Die Klasse enthält *3 Member* (Fehlermeldung, Text der fehlerhaften Zeile als C++-String, Zeilennummer), einen Konstruktor mit eben diesen Argumenten, und sonst nichts.
- **getNum** ist etwas trickreich:
  - Es zählt intern bei jedem Aufruf die *Zeilennummer* mit.
    (welche Art von Variable brauchst du dafür?)
  - Es liest eine Zeile Input mit der vordefinierten C++-Funktion **getline**
    in ein **string**-Objekt *input*: **getline(cin,** *input***);**

getline entfernt automatisch das **'\n'** am Zeilenende.

Am Fileende liefert **getline** in *input* einen leeren String.
Da eine leere Eingabezeile ohnehin das Programm beendet,
ist dieser Fall also gleich miterledigt.

- Mit *input***.c_str()** erhält man einen **char** *-Pointer auf den internen C-String eines **string**-Objektes. Den brauchen wir in der Folge mehrmals.
- Ist der eingelesene String *leer* oder beginnt er mit **'#'**, werfen wir ihn.
- Hätte laut Parameter eine **'#'**-Zeile kommen müssen, werfen wir einen **SyntaxError**.
- Mit der Standard-C-Funktion **strtod(***beg***,** *end***)** (siehe **man**-Page!) versuchen wir, den Input in einen **double** zu verwandeln.

  Achtung: *end* ist ein **char** *-Ausgangsparameter, also ein **char** **!
- Konvertiert **strtod** nichts (d.h. ist nach dem Aufruf *beg* == *end*), werfen wir einen **SyntaxError**.
- War die Konvertierung erfolgreich, müssen wir noch prüfen, ob *alle Zeichen* von *end* bis zum Zeilenende **isspace** sind, sonst werfen wir ebenfalls einen **SyntaxError**.
- Schließlich returnieren wir den von **strtod** gelieferten Wert.

- Die eigentliche Mittelwert-Berechnung erfolgt im **Hauptprogramm**:
  - Verwende zwei Variablen für die *Summe* und die *Anzahl* der Werte in der aktuellen Messwert-Reihe.
  - Merke dir im Hauptprogramm auch den Text der letzten **'#'**-Zeile. Solange dieser Text *nicht gesetzt* wurde
    - muss im Input zuerst einmal eine **'#'**-Zeile und keine Zahl kommen (Parameter im **getNum**-Aufruf!),
    - und wird bei einer gefangenen **'#'**-Zeile oder Leerzeile *kein* alter Mittelwert ausgegeben!
  - **main** besteht aus einer Endlos-Schleife, die ein **try** mit 2 **catch** enthält.
  - Im **try** wird **getNum** aufgerufen und zur Summe dazugezählt sowie die Anzahl der Messwerte erhöht.

    Achtung: Wenn in der aktuellen Zeile eine Exception auftritt, darf die Anzahl *nicht* erhöht werden, da ja auch zur Summe nichts dazugezählt wird!
  - Das **catch** für **SyntaxError** gibt das Fehlerobjekt aus und macht normal weiter.
  - Das **catch** für einen String gibt zuerst einmal den bisherigen *Mittelwert* aus, außer es gibt noch gar keine Messreihe oder sie hat 0 Messwerte (*name***.substr(1)** liefert den String *name* ohne das erste Zeichen **'#'**).

    Dann wird entweder das Programm beendet (bei leerem String) oder der neue Messungs-Name gespeichert sowie Summe und Anzahl auf 0 gesetzt.