Dr. Olaf Ippisch / Adrian Ngo
IWR, Universität Heidelberg

Übung 1 *Kondition der Standardoperationen*

Berechne die relativen Konditionszahlen (Verstärkungsfaktoren) der Standardoperationen *Multiplizieren, Dividieren* und *Wurzel ziehen*, also der Abbildungen:

$$F(x,y) = x \cdot y, \qquad F(x,y) = \frac{x}{y} \qquad F(x) = \sqrt{x}.$$

(3 Punkte)

Übung 2 *Kondition quadratischer Gleichungen*
Das analytische Lösen der quadratischen Gleichung

$$x^2 - 2px + 1 = 0$$

(z.B. mittels quadratischer Ergänzung) führt auf zwei Lösungen $x_1(p)$ und $x_2(p)$, die natürlich vom Parameter p abhängen. Berechne die relativen Konditionszahlen (Verstärkungsfaktoren) der Funktion

$$F(p) := \begin{pmatrix} x_1(p) \\ x_2(p) \end{pmatrix},$$

welche eine Lösungsvorschrift der quadratischen Gleichung darstellt. Für welche Werte von p ist die Lösung überhaupt in der Menge der reellen Zahlen definiert?

Betrachte nun das alternativ parametrisierte Problem:

$$x^2 - \frac{t^2+1}{t}x + 1 = 0 \qquad\qquad t \in [1, \infty).$$

Berechne auch für diesen Fall die relativen Konditionszahlen der zugehörigen Lösungsvorschrift $\tilde{F}(t)$ und vergleiche die relativen Konditionszahlen mit denen der ursprünglichen Formulierung. Für welche Werte von p bzw. t wird das jeweilige Problem schlecht konditioniert?

(6 Punkte)

Übung 3 *Landau Symbole*
Schreibe die folgenden Ausdrücke in der Form $f(h) = \mathcal{O}(h^m)$ (für $h \to 0, h > 0$) mit einem möglichst großen $m \in \mathbb{N}$.

$$\begin{aligned} f(h) &= (ph^2 + h)^2 - p^2h^4 \qquad p \in \mathbb{N} \\ f(h) &= -\frac{h}{\ln(h)} \\ f(h) &= e^h - 1 - h; \\ f(h) &= 1 - \frac{\sin(h)}{h}; \end{aligned}$$

und skizziere jeweils $f(h)$ zusammen mit dem jeweiligen $c \cdot h^m$ auf dem Intervall $(0, 1)$.
(Hinweis: Für einen der Ausdrücke ist die Form $f(h) = o(h^m)$ vorzuziehen!) **(4 Punkte)**

Übung 4 *Kommandozeilenparameter und templatisierte Funktion (Praktische Übung)*

Das C++-Programm `precision` aus dem letzten Übungsblatt soll nun um folgende Features erweitert werden:
Die Benutzereingabe soll nun komplett über die Kommandozeile

```
./precision <Datentyp> <Zahl>
```

möglich sein, d.h. der Benutzer des Programms `precision` gibt für `<Datentyp>` entweder `double` oder `float` ein und für `<Zahl>` eben eine beliebige Zahl z.

Es soll wieder (näherungsweise) ermittelt werden, für welche Fließkommazahl x_0 gerade noch die Bedingung

$$z < z + x_0$$

erfüllt ist. Der Algorithmus zur Ausgabe des Ergebnisses kann von der letzten Lösung übernommen werden.

Hinweise:

- Lagere die Implementierung des Algorithmus in eine separate C++-Funktion aus. Um die Funktion nicht doppelt (also für float und für double) schreiben zu müssen, bietet es sich hier an, ein Template-Parameter (z.B. `REAL`) für den Datentyp zu verwenden, also:

  ```
  template<typename REAL>
  void precision_loop( REAL x )
  { ... }
  ```

- In der Zusammenfassung "Grundlegende Anweisungen in C++" (siehe Homepage der Vorlesung) ist ein Abschnitt über Kommandozeilenparameter aufgeführt.
  - Benutze die Hauptfunktion

    ```
    int main( int argc, char **argv )
    { ... }
    ```

  - Überprüfe, ob die Anzahl der Kommandozeilenargumente (`argc`) mindestens 3 ist.
  - Wenn nicht, gebe die Meldung aus

    ```
    Gebrauch: ./precision <double|float> <Zahl>
    Beispiel: ./precision float 1.0
    ```

    und verlasse das Programm mit `exit(0);`
  - Ansonsten lese den ersten Parameter als String ein:

    ```
    std::string s1 = argv[1];
    ```

  - Mit der Anweisung

    ```
    if( s1 == "float" )
    { ... }
    ```

    kann man überprüfen, ob der erste Parameter `float` heißt oder nicht.
  - Der zweite eingelesene Parameter soll als Zahl weiterverarbeitet werden. Deshalb ist eine Umwandlung nach float (bzw. double) notwendig, hier ein Beispiel für float:

    ```
    float x;
    x = atof( argv[2] );
    ```

  - Beachte ausserdem: Die Einbindung folgender Header-Dateien sind notwendig:

```
#include <iostream>   // Bildschirmausgabe
#include <iomanip>    // Formatierte Ausgabe
#include <string>     // String (nicht zwingend, da schon in iostream vorha
#include <cstdlib>    // wegen atof() und exit()
```

Teste das Programm mit den folgenden Parametern:

```
./precision float 1e-3
./precision float 1.0
./precision float 100.0

./precision double 1e-5
./precision double 1.0
./precision double 1e+5
```

(5 Punkte)