Reihenentwicklungen, Polynome und Schleifensteuerung

1. Wie viele andere analytische Funktionen auch, läßt sich die Exponentialfunktion im Prinzip durch eine Reihenentwicklung

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

darstellen. (Dies ist jedoch *nicht* die Art und Weise, wie in Programmiersprachen "eingebaute Funktionen" üblicherweise implementiert werden!) Schreibe ein kurzes Programm, das ein Argument x und einen Index n einliest, die Exponentialfunktion durch Summation aller Terme der Reihenentwicklung bis einschließlich $k = n$ approximiert und mit dem "exakten" Wert der eingebauten Funktion `exp(x)` vergleicht.
Hinweis: Da sowohl x^k als auch $k!$ sehr schnell mit k wachsen können, empfiehlt es sich, nicht für jeden Summanden Zähler und Nenner separat zu berechnen und dann durcheinander zu dividieren, sondern jeweils den k-ten Term durch Multiplikation mit x/k aus dem $(k-1)$-ten zu bilden.

2. Wird eine analytische Funktion durch eine endliche Reihenentwicklung approximiert, so bedeutet dies, daß sie eigentlich durch ein Polynom (von eventuell sehr hohem Grad) dargestellt wird, z.B. für den Logarithmus

$$\ln(1+x) \approx \sum_{k=1}^{n} (-1)^{(k+1)} \frac{x^k}{k} = x - \frac{x^2}{2} + \frac{x^3}{3} + \ldots + (-1)^{(n+1)} \frac{x^n}{n}$$

Da Polynome oft numerisch ungünstige Eigenschaften haben, ist es besser, sie nicht wie oben angeschrieben, sondern nach dem *Hornerschema* "von innen nach außen" auszuwerten, d.h. in diesem Fall

$$\ln(1+x) \approx x \left[1 - x \left[\frac{1}{2} - x \left[\frac{1}{3} - x \left[\ldots - x \left[\frac{1}{n-1} - \frac{x}{n}\right] \ldots\right]\right]\right]\right]$$

wobei man mit dem Term in den innersten Klammern beginnt und sich nach außen durcharbeitet. Das hat den Vorteil, daß in jedem Schritt x nur linear vorkommt und nicht hohe Potenzen von x direkt miteinander kombiniert werden müssen.

Schreibe ein kurzes Programm, das nach Eingabe von x und n das obige Approximationspolynom sowohl nach dem naiven als auch nach dem Hornerschema berechnet und mit dem exakten Wert `log(1.0+x)` der eingebauten Funktion vergleicht. Um die Unterschiede besser sichtbar zu machen, sollten die Rechnungen in "einfacher Präzision" (Typ `real` in `F` bzw. `float` in `C`) durchgeführt werden.

3. Durch die endliche Rechengenauigkeit bei Floating Point (im Gegensatz zu ganzen) Zahlen kann es schon bei den einfachsten Operationen zu unerwarteten Ergebnissen kommen: Untersuche, wie oft die folgende Schleife in `F`

```
real::x
...
do x=1.0,2.0,0.1
  ...
end do
```

bzw. in `C`

```
float x;
...
for(x=1.0;x<=2.0;x=x+0.1) {
  ...
}
```

tatsächlich ausgeführt wird. Hängt das Verhalten von Sprache und Compiler ab?

Dies ist einer der Gründe, warum die Verwendung von nicht ganzzahligen Schleifenzählern, obwohl in `F9x` geduldet, in `F` verpönt ist.