Dirk Praetorius,
Michele Ruggeri

Wintersemester 2016/17
03.11.2016

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. Schreiben Sie eine *nicht-rekursive* Funktion `power`, die für gegebene reelle Zahlen $x > 1$ und $C > 0$ die kleinste Zahl $n \in \mathbb{N}$ berechnet mit $x^n > C$. Dabei soll die Funktion `log` nicht verwendet werden. Stellen weiters Sie mittels `assert` sicher, dass $x > 1$ und $C > 0$ gilt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x und C eingelesen werden und n ausgegeben wird. Speichern Sie den Source-Code unter `power.c` in das Verzeichnis `serie05`.

Aufgabe 5.2. Die Quotientenfolge $(a_{n+1}/a_n)_{n\in\mathbb{N}}$ zur Fibonacci-Folge $(a_n)_{n\in\mathbb{N}}$,

$$a_0 := 1, \quad a_1 := 1, \quad a_n := a_{n-1} + a_{n-2} \quad \text{für } n \geq 2,$$

konvergiert gegen den goldenen Schnitt $(1 + \sqrt{5})/2$. Insbesondere konvergiert die Differenz

$$b_n := \frac{a_{n+1}}{a_n} - \frac{a_n}{a_{n-1}}$$

gegen Null. Schreiben Sie eine *nicht-rekursive* Funktion `cauchy`, die zu gegebenem $k \in \mathbb{N}$ die kleinste Zahl $n \in \mathbb{N}$ mit $|b_n| \leq 1/k$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das die Zahl $k \in \mathbb{N}$ einliest und den zugehörigen Index $n \in \mathbb{N}$ ausgibt. Speichern Sie den Source-Code unter `goldenerSchnitt.c` in das Verzeichnis `serie05`.

Aufgabe 5.3. Für eine differenzierbare Funktion $f : [a, b] \to \mathbb{R}$ kann man die Ableitung $f'(x)$ in einem festen Punkt $x \in \mathbb{R}$ durch den einseitigen Differenzenquotienten

$$\Phi(h) := \frac{f(x + h) - f(x)}{h} \quad \text{für } h > 0$$

approximieren. Schreiben Sie eine Funktion `double diff(double x, double h0, double tau)`, die für $h_n := 2^{-n} h_0$ ($n \in \mathbb{N}$) die Folge der $\Phi(h_n)$ berechnet, bis gilt

$$|\Phi(h_n) - \Phi(h_{n+1})| \leq \begin{cases} \tau & \text{falls } |\Phi(h_n)| \leq \tau, \textit{oder} \\ \tau\,|\Phi(h_n)| & \text{anderenfalls.} \end{cases}$$

Die Funktion liefere in diesem Fall $\Phi(h_n)$ als Approximation von $f'(x)$ zurück. Stellen Sie mittels `assert` sicher, dass $\tau, h_0 > 0$ gilt. Die Funktion soll mit einer beliebigen reellwertigen Funktion `double f(double x)` arbeiten. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem x, h_0 und τ eingelesen werden und $\Phi(h_n)$ ausgegeben wird. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `diff.c` in das Verzeichnis `serie05`.

Aufgabe 5.4. Alternativ zum Bisektionsverfahren aus der Vorlesung kann eine Nullstelle von $f : [a, b] \to \mathbb{R}$ auch mit dem *Sekantenverfahren* berechnet werden. Dabei sind x_0 und x_1 gegebene Startwerte und man definiert induktiv x_{n+1} als Nullstelle der Geraden durch $(x_{n-1}, f(x_{n-1}))$ und $(x_n, f(x_n))$, d.h.

$$x_{n+1} := x_n - f(x_n)\,\frac{x_{n-1} - x_n}{f(x_{n-1}) - f(x_n)}$$

Schreiben Sie eine Funktion `sekante(x0,x1,tau)` die die Folge der Iterierten berechnet, bis entweder

$$|f(x_n) - f(x_{n-1})| \leq \tau$$

oder

$$|f(x_n)| \leq \tau \quad \text{und} \quad |x_n - x_{n-1}| \leq \begin{cases} \tau & \text{für } |x_n| \leq \tau, \\ \tau|x_n| & \text{sonst} \end{cases}$$

gilt. Es werde dann x_n als Approximation einer Nullstelle z_0 von f zurückgegeben. Im ersten Fall gebe man zusätzlich eine Warnung aus, dass das numerische Ergebnis vermutlich falsch ist. Stellen Sie mittels `assert` sicher, dass $\tau > 0$ gilt. Die Funktion soll mit einer beliebigen reellwertigen Funktion `double f(double x)` arbeiten. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 und x_1 eingelesen werden und x_n ausgegeben wird. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `sekante.c` in das Verzeichnis `serie05`.

Aufgabe 5.5. Eine Variante zur Berechnung einer Nullstelle einer Funktion $f : [a, b] \to \mathbb{R}$ ist das *Newton-Verfahren*. Ausgehend von einem Startwert x_0 definiert man induktiv eine Folge $(x_n)_{n\in\mathbb{N}}$ durch

$$x_{k+1} = x_k - f(x_k)/f'(x_k).$$

Man realisiere das Newton-Verfahren in einer Funktion `double newton(double x0, double tau)`, wobei die Iteration abgebrochen wird, falls entweder

$$|f'(x_n)| \leq \tau$$

oder

$$|f(x_n)| \leq \tau \quad \text{und} \quad |x_n - x_{n-1}| \leq \begin{cases} \tau & \text{für } |x_n| \leq \tau, \\ \tau|x_n| & \text{sonst} \end{cases}$$

gilt. Im ersten Fall gebe man zusätzlich eine Warnung aus, dass das numerische Ergebnis vermutlich falsch ist. Stellen Sie mittels `assert` sicher, dass $\tau > 0$ gilt. Die Funktion soll mit einer beliebigen reellwertigen Funktion `double f(double x)` und Ableitung `double fstrich(double x)` arbeiten. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 eingelesen und x_n ausgegeben wird. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `newton.c` in das Verzeichnis `serie05`.

Aufgabe 5.6. Das Newton-Verfahren aus Aufgabe 5.5 benötigt neben der Funktion `f` auch eine Funktion `fstrich`, die die Ableitung f' der Funktion f auswertet. Alternativ kann man $f'(x_k)$ durch den Differenzenquotienten $\Phi_h(x_k)$ aus Aufgabe 5.3 ersetzen. Realisieren Sie dieses Vorgehen indem Sie eine Funktion `double newton2(double x0, double h0, double tau)` schreiben, die zur Approximation der Ableitung $f'(x_k)$ das Ergebnis von `diff(xk,h0,tau)` verwendet. Stellen Sie mittels `assert` sicher, dass $\tau, h_0 > 0$ gilt. Wie und mit welchen Beispielen können Sie Ihren Code auf Korrektheit testen? Speichern Sie den Source-Code unter `newton2.c` in das Verzeichnis `serie05`.

Aufgabe 5.7. Was ist der Unterschied und der Zusammenhang zwischen einer Variable und einem Pointer? Was könnten Vor- und Nachteile dieser Konstrukte sein?
Schreiben Sie eine Funktion `swap`, welche die Werte zweier Variablen `x` und `y` vertauscht. Warum funktioniert das folgende Vorgehen nicht?

```
void swap(double x, double y)
{
   double tmp;
   tmp = x;
   x = y;
   y = tmp;
}
```

Speichern Sie den Source-Code unter `swap.c` in das Verzeichnis `serie05`.

Aufgabe 5.8. Wo liegen die Fehler im folgenden Programm?

```
#include <stdio.h>

void square(double* x)
{
  double* y;
```

```
  x=(*y)*(*x);
}

int main(){
  double x=2.1;
  square(&x);
  printf("x^2=%f\n",x);
  return 0;
}
```

Verändern Sie *nur* die Funktion `square`, so dass der Output des Codes den Erwartungen entspricht.