Dirk Praetorius,
Marcus Page

Sommersemester 2012
über Ostern

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1*. (Dynamische Arrays) Modifizieren Sie Ihre Lösung von Aufgabe 4.3 dahingehend, dass für gegebenes $x \in [0,1]$ und gegebene Mantissenlänge M der (dynamischen) Vektor $a \in \{0,1\}^M$ der Ziffern zurückgegeben wird. Schreiben Sie (zur Kontrolle) eine Funktion, die den Vektor a und die Mantissenlüge M übernimmt und den Wert x zurückgibt.

Aufgabe 5.2*. (Kommandozeilenübergabe, Internetrecherche) Die Funktion main() hat in C formal die Signatur

```
int main (int argc, char *argv[])},
```

sodass man von der Konsole, bei Aufruf des Programms, Parameter an das C-Programm übergeben kann. Finden Sie im WWW heraus, wie das funktioniert. Schreiben Sie ein C-Programm, das alle Input-Parameter, die von der Konsole übergeben werden, am Bildschirm ausgibt.

Aufgabe 5.3*. (Kommandozeilenübergabe) Modifizieren Sie das Programm aus Aufgabe 5.2 so, dass Sie Ihrem C-Programm aus der Konsole heraus einen (dynamischen) double-Vektor übergeben können, d.h. der Aufruf

```
> programm 0 1.3 5 2.5 4
```

generiert im Programm einen dynamischen double-Vektor x der Länge 5 mit $x[0] = 0; x[1] = 1.3; x[2] = 5$ usw. Dieser Vektor soll dann mit der Funktion `printfvector` aus der Vorlesung ausgegeben werden. In den folgenden Aufgaben sollen Sie die so programmierte Schnittstelle verwenden (anstatt wie bisher scanf).

Aufgabe 5.4*. (Kommandozeilenübergabe, Schleifen) Schreiben Sie eine Funktion `maxnorm`, die die Maximumsnorm eines gegebenen Vektors berechnet und zurückgibt. Der Vektor soll dem Hauptprogramm über die Konsole übergeben werden. Das Hauptprogramm ruft die Funktion auf und gibt die Maximumsnorm aus.

Aufgabe 5.5. (Kommandozeilenübergabe, Polynome) Schreiben Sie ein Programm, das bei Aufruf in der Form

```
> programm a0 a1 a2 a3 , b0 b1 b2
```

mit reellen Zahlen aj, bk beispielsweise dynamisch Vektoren $a \in \mathbb{R}^4$ und $b \in \mathbb{R}^2$ generiert. Schreiben Sie eine `void`-Funktion, die jeden dieser Vektoren als Polynom interpretiert und in der Form $p(x) = \sum_{j=0}^{3} a_j x^j$ bzw. $q(y) = \sum_{k=0}^{2} a_k x^k$ ausgibt.

Aufgabe 5.6. (Polynome) Die Summe zweier Polynome p, q (nicht notwendigerweise gleichen Grades) ist wieder ein Polynom. Welchen Grad hat das Summenpolynom? Schreiben Sie eine Funktion `addPoly`,

die die Polynome p, q in Form ihrer Koeffizientenvektoren übernimmt und den Koeffizientenvektor von $p + q$ (dynamisch) generiert und zurückgibt. Zur Eingabe der Polynome und zur Ausgabe des Summenpolynoms modifizieren Sie Aufgabe 5.5 geeignet.

Aufgabe 5.7. (Polynome) Das Produkt zweier Polynome p, q (nicht notwendigerweise gleichen Grades) ist wieder ein Polynom. Welchen Grad hat das Produktpolynom? Wie sieht die Formel für die Koeffizienten c_k des Produktpolynoms $r(x) = \sum_{k=0}^{n} c_k x^k$ in Abhängigkeit der Koeffizienten von p und q aus. Überlegen Sie sich das zunächst an einem Beispiel, z.B. $p(x) = 1 + 2x + 3x^2$ und $q(x) = 2 + x + x^2 + 4x^3$. Schreiben Sie eine Funktion `multPoly`, die für gegebene Polynome p und q den Koeffizientenvektor des Produktpolynoms $p \cdot q$ berechnet. Modifizieren Sie auch hier Aufgabe 5.5 geeignet, um sich Ein- und Ausgabe zu erleichtern.

Aufgabe 5.8. (Strukturen) Warum ist die Verwendung von Strukturen sinnvoll? Wie könnte man Strukturen nutzen, um die Polynome in Aufgabe 5.5 – 5.7 geeignet zu speichern.

Aufgabe 5.9. (Newton-Verfahren zur Nullstellensuche, Bedingungsschleifen) 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) durch

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

Man realisiere das Newton-Verfahren in einer Funktion `newton`, 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. Die Funktion soll mit einer beliebigen reellwertigen funktion `double f(double x)` arbeiten, für die Funktionen `evalF` und `evalDiffF` zur Verfügung stehen. Schreiben Sie ein aufrufendes Hauptprogramm, in dem x_0 eingelesen und x_n für verschiedene vorprogrammierte Funktionen f (z.B. sin) ausgegeben wird.

Aufgabe 5.10. (Nullstellensuche bei Polynomen) Wenden Sie das *Newton-Verfahren* aus Aufgabe 5.9 an, um Nullstellen von Polynomen zu finden. Die Eingabe des Polynoms (beliebiger Länge) soll hierbei wie in Aufgabe 5.3 erfolgen. Verwenden Sie die Aufgaben 4.4 bzw. 4.5 für die Auswertung von $p(x)$ bzw. $p'(x)$.

Frohe Ostern!