Dirk Praetorius,
Marcus Page,
Thomas Führer

Sommersemester 2013
16.04.2013

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 5

Aufgabe 5.1. Schreiben Sie eine Funktion `bin2dec`, die eine im Binärformat gegebene natürliche Zahl $0 \leq z < 256$ in eine Dezimalzahl umrechnet. Die Binärzahl werde als Vektor der Koeffizienten $a_i \in \{0, 1\}$ für $i = 0, \ldots, 7$ dargestellt. Dann lässt sich der Wert der Binärzahl mittels $z = \sum_{i=0}^{7} a_i 2^i$ berechnen. Schreiben Sie ferner ein aufrufendes Hauptprogramm, in dem die Koeffizienten a_i eingelesen und der Wert von z als Dezimalzahl ausgegeben werden. Speichern Sie den Source-Code unter `bin2dec.c` in das Verzeichnis `serie05`.

Aufgabe 5.2. Schreiben Sie eine Funktion `minmaxmean`, die von einem gegebenem Vektor $x \in \mathbb{N}^n$ das Minimum, das Maximum und den Mittelwert $\frac{1}{n} \sum_{j=1}^{n} x_j$ berechnet und geeignet zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor $x \in \mathbb{N}^n$ einliest und Minimum, Maximum und Mittelwert ausgibt. Die Länge $n \in \mathbb{N}$ des Vektors soll eine Konstante im Hauptprogramm sein, die Funktion `minmaxmean` ist für beliebige Länge n programmieren. Speichern Sie den Source-Code unter `minmaxmean.c` in das Verzeichnis `serie05`.

Aufgabe 5.3. Schreiben Sie eine Funktion `energy`, die für einen gegebenen Vektor $x \in \mathbb{R}^n$ die Energie $e = \sum_{j=1}^{n} x_j^2$ zurückgibt. Schreiben Sie ferner ein aufrufendes Hauptprogramm, das den Vektor x einliest und die Energie ausgibt. Die Dimension $n \in \mathbb{N}$ soll eine Konstante im Hauptprogramm ein, die Funktion `energy` ist für beliebige Dimension zu programmieren. Speichern Sie den Source-Code unter `energy.c` in das Verzeichnis `serie05`.

Aufgabe 5.4. Genauso wie der Inhalt von Variablen elementaren Datentyps kann auch der Inhalt eines Pointers mittels `printf` ausgegeben werden. Man verwendet hier `%p` als Platzhalter für Adressen. Die Ausgabe dafür erfolgt systemabhängig meist in Hexadezimaldarstellung. Schreiben Sie eine Funktion `void charPointerAbstand(char* anfangsadresse, char* endadresse)`, welche folgende drei Werte tabelliert:

- Anfangsadresse
- Endadresse
- Abstand (Differenz) der beiden Adressen (Platzhalter im `printf` beachten!)

Da Arrays zusammenhängend im Speicher liegen, entspricht der Abstand zweier aufeinanderfolgender Elemente genau dem Speicherverbrauch des entsprechenden Datentyps. Testen Sie Ihre Funktion für einen `char`-Array `c[2]` mit den beiden Aufrufen:

```
charPointerAbstand(&c[0],&c[1]);
charPointerAbstand(c,c+1);
```

Schreiben Sie nun nach obiger Manier eine Funktion `void doublePointerAbstand(double* anfangsadresse, double* endadresse)`, testen diese mit einem `double`-Array und vergleichen die unterschiedlichen Ergebnisse. Was passiert, wenn `charPointerAbstand` mit Adressen von `double`-Variablen aufgerufen wird? Speichern Sie den Source-Code unter `datentypen.c` in das Verzeichnis `serie05`.
Bonus: Finden Sie heraus, wieviel Speicher die Typen `short`, `int` und `long` auf dem Übungsserver verbrauchen.

Aufgabe 5.5. 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 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 5.6. Schreiben Sie eine Bibliothek zur Verwendung dynamischer Vektoren zur Speicherung von `int`-Werten. Diese soll die Funktionalität zur Allokation, Reallokation, sowie Freigabe von Vektoren dieses Typs enthalten. Außerdem sollen Vektoren über die Tastatur eingelesen, sowie ausgegeben werden können. Orientieren Sie sich an Folien 123-124 der Vorlesung. Erklären Sie in diesem Zusammenhang auch den Sinn von Headerfiles.Speichern Sie den Source-Code unter `intVectorBib.c` und `intVectorBib.h` in das Verzeichnis `serie05`.

Aufgabe 5.7. Schreiben Sie eine Funktion `firstPrimes`, welche alle Primzahlen berechnet die kleiner als eine gegebene Schranke $n \in \mathbb{N}$ sind. Diese sollen dann als `int`-Vektor v zurückgegeben werden. Verwenden Sie hierzu die Bibliothek aus Aufgabe 5.6. Beachten Sie, dass als zweiter „Rückgabewert“ auch die Länge von v „zurückgegeben“ werden muss. Speichern Sie den Source-Code unter `firstPrimes.c` in das Verzeichnis `serie05`.

Aufgabe 5.8. Schreiben Sie eine Funktion `kgV(a,b)`, die das kleinste gemeinsame Vielfache zweier natürlicher Zahlen $a, b \in \mathbb{N}$ berechnet. Zur Lösung können Sie entweder die Primfaktoren beider Zahlen berechnen oder den Zusammenhang $a \cdot b = ggT(a, b) \cdot kgV(a, b)$ berücksichtigen. Speichern Sie den Source-Code unter `kgv.c` in das Verzeichnis `serie05`.