Abteilung für Angewandte Mathematik
Prof. Dr. S. Bartels
M.Sc. C. Palus

25.06.2018

Übung zur Vorlesung

Einführung in die Programmierung

SoSe 2018 – Blatt 10

Abgabe: Briefkästen RZ/E-Mail bis Montag, den 02.07.2018 um 16:00 Uhr

Aufgabe 1 (4+4+2 Punkte)**.** Zur übersichtlichen Darstellung von Bitfolgen bietet sich das *Hexadezimalsystem* an. Genau wie das Dual- und Dezimalsystem ist es ein Stellenwertsystem, in dem anstatt der Basis 2 bzw. 10 die Basis 16, welche selbst eine Zweierpotenz ist, verwendet wird. Die bekannten Ziffern $0, \ldots, 9$ werden dazu ergänzt um die Ziffern A, B, C, D, E und F, welche den dezimalen Werten 10, 11, 12, 13, 14 und 15 entsprechen. Um hexadezimale Zahlen zu kennzeichnen und von Dezimalzahlen zu unterscheiden, wird häufig der Index „hex“ oder das Präfix „0x“ verwendet, wobei Letzteres vor allem in der Programmierung und technischen Informatik verbreitet ist.

(i) Wandeln Sie die folgenden Hexadezimalzahlen in Bitfolgen um:

$$\mathrm{AD4B}_{\mathrm{hex}}, \quad \mathrm{0xC328}, \quad \mathrm{0x2013}, \quad \mathrm{AFFE}_{\mathrm{hex}}.$$

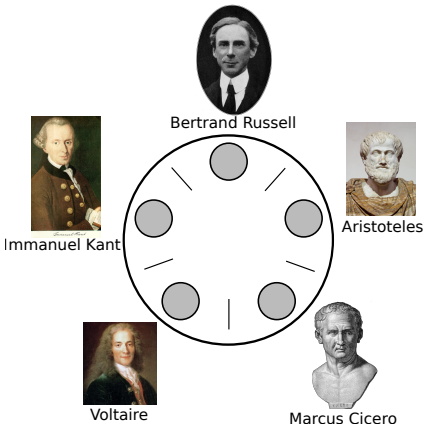
(ii) Wandeln Sie die folgenden Bitfolgen in Hexadezimalzahlen um:

$$0111100100100100, \quad 1011110000111111, \quad 1101000100000000, \quad 0000000011110000.$$

(iii) Führen Sie die folgende Addition schriftlich im Hexadezimalsystem durch und wandeln Sie das Ergebnis in eine Dezimalzahl um.

$$\mathrm{BB2A} + 5123$$

Aufgabe 2 (2+6+2 Punkte)**.** Fünf Philosophen sitzen um einen runden Tisch herum. Jeder von ihnen hat vor sich eine Schüssel Nudeln und benötigt genau zwei Essstäbchen, um davon zu essen. Jedoch gibt es am Tisch nur insgesamt 5 Essstäbchen, die jeweils zwischen den Tellern liegen (siehe Abbildung).



Die Philosophen und Essstäbchen stellen in diesem Fallbeispiel Prozesse bzw. geteilte Ressourcen dar. Jeder Philosoph denkt so lange still nach, bis er hungrig wird. Ist er hungrig, so beginnt er zu essen, sobald es ihm möglich ist, und isst so lange, bis er satt ist. Anschließend denkt er wieder so lange still nach, bis er erneut hungrig wird. Jeder der Philosophen kann nur die beiden Stäbchen rechts und links seines Tellers erreichen und keiner von ihnen weiß, wann einer der anderen hungrig wird oder mit dem Essen fertig ist. Naiv könnten die Philosophen nach folgenden Regeln vorgehen: Wenn ein Philosoph hungrig wird, so wartet er, bis sein rechtes Essstäbchen verfügbar ist und nimmt es in die Hand. Anschließend wartet er, bis sein linkes Essstäbchen verfügbar ist, nimmt es in die Hand und beginnt zu essen. Wenn er genug gegessen hat, so legt er zuerst das rechte und dann das linke Stäbchen zurück an seinen Platz.

Dieses naive Vorgehen kann jedoch zu einer Verklemmung (engl. *deadlock*) der Prozesse führen. Dies ist die Bezeichnung für einen Zustand, in dem alle Philosophen ohne weiteren Fortschritt verharren. Ein weiteres Problem kann auftreten, wenn manche der Philosophen besonders schnell beim Denken und Aufheben ihrer Essstäbchen sind. In diesem Fall kann es passieren, dass ein einzelner oder mehrere Philosophen nie zum Essen kommen, weil sie schlicht zu langsam sind. Dieses Problem wird als Aushungern (engl. *starvation*) bezeichnet.

Fortsetzung auf der Rückseite

(i) Erläutern Sie, wie es bei dem oben beschriebenen naiven Vorgehen zu Deadlocks bzw. Starvations kommen kann. Geben Sie jeweils Beispiele an.
(ii) Nehmen Sie an, dass es unter den Philosophen Rechts- und Linkshänder gibt, wobei ein Rechtshänder immer zuerst nach seinem rechten Stäbchen greift (und dieses auch zuerst wieder beiseitelegt), während ein Linkshänder zuerst nach seinem linken Stäbchen greift (und dieses zuerst beiseitelegt). Weisen Sie formal nach, dass es zu keinem Deadlock mehr kommen kann, wenn unter den Philosophen am Tisch sowohl Rechts- als auch Linkshänder sind.
(iii) Werden durch das Vorhandensein eines Linkshänders auch Starvations verhindert? Begründen Sie Ihre Antwort.

Aufgabe 3 (10 Punkte)**.** Die folgende Tabelle zeigt den *American Standard Code for Information Interchange* (ASCII), eine 7-Bit-Zeichenkodierung, die das lateinische Alphabet in Groß- und Kleinschreibung, die zehn arabischen Ziffern, einige Satz-, Wort- und Sonderzeichen sowie 33 weitere, nicht druckbare Steuerzeichen umfasst.

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	␣	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Jedes Zeichen in dieser Tabelle wird mit einer Folge von 7 Bits identifiziert, deren Wert genau der Hexadezimalzahl entspricht, die zu der entsprechenden Zeile und Spalte gehört, z. B. entspricht das Leerzeichen dem Wert 0x20 = 0100000_2 = 32. In C++ kann der Dezimalwert des ASCII-Codes eines Zeichens `c` vom Typ `char` mittels der Konvertierung `int(c)` ermittelt werden. Schreiben Sie ein Programm, welches eine Textdatei mit dem *Cäsar-Algorithmus* je nach Benutzerwunsch entweder ver- oder entschlüsselt. Dieser sehr einfache Algorithmus verschlüsselt einen Text, indem jeder Buchstabe des Klartexts auf einen eindeutigen Geheimtextbuchstaben abgebildet wird. Diese Abbildung ist gegeben durch eine zyklische Verschiebung des Alphabets um k Stellen nach rechts, wobei die natürliche Zahl $k < 26$ den Schlüssel darstellt. Ziffern sowie Sonderzeichen werden dabei nicht verschlüsselt. Ihr Programm sollte die Groß- und Kleinschreibung in der Datei beibehalten und den Schlüssel k vom Benutzer abfragen.

Aufgabe 4 (10 Punkte)**.** Das Bisektionsverfahren zur Approximation einer Nullstelle einer gegebenen Funktion $f\colon \mathbb{R} \to \mathbb{R}$ ist wie folgt definiert:

(1) Wähle eine Genauigkeit $\varepsilon > 0$ und zwei Startwerte $a_0, b_0 \in \mathbb{R}$, sodass $a_0 < b_0$ und $f(a)$ und $f(b)$ unterschiedliches Vorzeichen haben. Setze $k = 0$.
(2) Falls $b_k - a_k < \varepsilon$, so ist $[a_k, b_k]$ das gesuchte Lösungsintervall: Breche das Verfahren ab.
(3) Falls $f(\frac{a_k+b_k}{2})$ das gleiche Vorzeichen wie $f(a_k)$ hat, setze $a_{k+1} = \frac{a_k+b_k}{2}$ und $b_{k+1} = b_k$. Andernfalls setze $a_{k+1} = a_k$ und $b_{k+1} = \frac{a_k+b_k}{2}$.
(4) Erhöhe k um 1 und gehe zu (2).

Implementieren Sie dieses Verfahren in MATLAB als Funktion

```
function nst = bisektion(f,a,b,eps),
```

wobei die als Parameter übergebene Funktion `f` ein sogenannter *function handle* ist, d.h. eine Variable, welche den Verweis auf eine Funktion speichert. Die Syntax zur Definition solcher Variablen ist dabei wie im folgenden Beispiel, in dem die Funktion `f(x)` den Wert von $\texttt{x}^2 - 1$ berechnet:

```
f = @(x) x*x-1;
```

Der Rückgabewert `nst` Ihrer Funktion sollte der Mittelpunkt des Lösungsintervalls sein. Testen Sie Ihr Programm, indem Sie damit die Nullstelle der Funktion $x \mapsto x^3 - 2$ approximieren.

Abgabe der Programme per E-Mail, (handschriftlich) kommentierte Ausdrucke der Programme und Rechnungen auf gehefteten, mit Namen versehenen Zetteln in die Briefkästen

Homepage zur Vorlesung: `https://aam.uni-freiburg.de/agba/lehre/ss18/einfprog`