Brandenburgische
Technische Universität
Cottbus - Senftenberg

Brandenburgische Technische Universität Cottbus – Senftenberg
Institut für Informatik, Informations- und Medientechnik
PD Dr.-Ing. habil. Thomas Hinze

Wintersemester 2015 / 2016

Abgabe der Lösungen: bis Mittwoch, 06.01.2016

Einführung in die Programmierung
Übungsblatt 4

Hinweise

- Erstellen Sie bitte eine schriftliche Lösung der nachfolgenden Aufgaben und erzeugen Sie daraus eine pdf-Datei. Sie können auch Ihre (lesbare!) handschriftliche Lösung als pdf einscannen. Achten Sie bitte darauf, dass am Beginn der ersten Seite des pdfs Ihr *Name*, Ihre *Matrikelnummer* und der Name Ihres *Tutors* (Laborübungsgruppe) vermerkt ist. Jede Aufgabe, die durch **(Laborübung)** gekennzeichnet ist, beinhaltet als Lösung auch einen C-Quelltext. Bitte packen Sie Ihr pdf gemeinsam mit allen C-Quelltextdateien in eine zip-Datei, die Sie termingemäß im moodle hochladen.
- Sie bekommen eine Bewertung Ihrer Lösung mit der Information, ob sie ausreichend zum Bestehen des Übungsblattes ist. Im Falle einer nichtausreichenden Lösung haben Sie Gelegenheit, eine verbesserte Version zeitnah, aber spätestens bis 01.02.2016 im moodle nachzureichen. Plagiate werden nicht als ausreichende Lösungen anerkannt.

Aufgabe 1

(Laborübung) Nach der *ergänzten Gaußschen Osterformel* lässt sich das Datum das Ostersonntags für das Jahr x im Gregorianischen Kalender (erstmals zu Ostern gültig 1583) mithilfe der nachfolgenden Funktionen schrittweise berechnen (div ist die ganzzahlige Division).
Implementieren Sie jede dieser mathematischen Funktionen als eigenständige Funktionen in C, der die benötigten Parameter übergeben werden und die den jeweiligen Funktionswert zurückgibt. Schreiben Sie darüber hinaus eine `main`-Funktion, die eine Jahreszahl (z.B. 2016) als Tastatureingabe einliest, auf Plausibilität prüft, bei einem gültigen Wert die Ostersonntagsberechnung per parametrisiertem Funktionsaufruf anstößt und das Ergebnis im Format `dd. Maerz` bzw. `dd. April` ausgibt.

1. die Säkularzahl ..... $k(x) = x \operatorname{div} 100$
2. die säkulare Mondschaltung ..... $m(k) = 15 + (3k + 3) \operatorname{div} 4 - (8k + 13) \operatorname{div} 25$
3. die säkulare Sonnenschaltung ..... $s(k) = 2 - (3k + 3) \operatorname{div} 4$
4. den Mondparameter ..... $a(x) = x \bmod 19$
5. den Keim für den ersten Vollmond im Frühling ..... $d(a, m) = (19a + m) \bmod 30$
6. die kalendarische Korrekturgröße ..... $r(d, a) = (d + a \operatorname{div} 11) \operatorname{div} 29$
7. die Ostergrenze ..... $og(d, r) = 21 + d - r$
8. den ersten Sonntag im März ..... $sz(x, s) = 7 - (x + x \operatorname{div} 4 + s) \bmod 7$

9. die Entfernung des Ostersonntags von der Ostergrenze . . . . . $oe(og, sz) = 7 - (og - sz) \bmod 7$
10. das Datum des Ostersonntags als Märzdatum (32. März = 1. April usw.) . . . . . $os(og, oe) = og + oe$

Testen Sie Ihr Programm für mindestens vier verschiedene gültige Jahreszahlen aus, darunter 2016. Fällt von 2000 bis zum Jahr 2099 der Ostersonntag auf das frühestmögliche Datum am 22. März und auf das spätestmögliche Datum am 25. April und falls ja, wann ist dies jeweils das nächste Mal der Fall? Ergänzen Sie dafür Ihre `main`-Funktion in geeigneter Weise.

Aufgabe 2

(Laborübung) Im Begleitmaterial finden Sie das Modulfragment `feiertage.c`. Erweitern Sie dieses Fragment zu einem vollständigen Modul einschließlich zugehöriger Headerdatei `feiertage.h`. Das Modul soll für jeden bundeseinheitlichen Feiertag eine Funktion bereitstellen, der Tag (`t`), Monat (`m`) und Jahr (`j`) eines Kalenderdatums übergeben werden und die genau dann das Ergebnis `1` liefert, wenn das übergebene Datum auf den entsprechenden Feiertag fällt. Anderenfalls wird der Ganzzahlwert `0` als Ergebnis zurückgegeben. Die im gegebenen Modulfragment `feiertage.c` schon hinterlegte Funktion `istGueltig(int t, int m, int j)` prüft, ob die übergebenen Parameterwerte ein gültiges Kalenderdatum darstellen (Rückgabewert `1`) oder nicht (Rückgabewert `0`). Bitte entnehmen Sie die Spezifikationen der von Ihnen zu implementierenden Funktionen der nachfolgenden Tabelle:

Feiertag	Funktion	Definition
Neujahr	`istNeujahr(int t, int m, int j)`	01.01.`j`
Karfreitag	`istKarfreitag(int t, int m, int j)`	Ostersonntag -2 Tage
Ostersonntag	`istOstersonntag(int t, int m, int j)`	siehe Aufgabe 1
Ostermontag	`istOstermontag(int t, int m, int j)`	Ostersonntag $+1$ Tag
Tag der Arbeit	`istTagDerArbeit(int t, int m, int j)`	01.05.`j`, falls `j` ≥ 1946
Christi Himmelfahrt	`istChristiHimmelfahrt(int t, int m, int j)`	Ostersonntag $+39$ Tage
Pfingstsonntag	`istPfingstsonntag(int t, int m, int j)`	Ostersonntag $+49$ Tage
Pfingstmontag	`istPfingstmontag(int t, int m, int j)`	Ostersonntag $+50$ Tage
Tag der Deutschen Einheit	`istTagDerDtEinheit(int t, int m, int j)`	03.10.`j`, falls `j` ≥ 1990 17.06.`j`, falls $1954 \leq$ `j` ≤ 1990
Erster Weihnachtsfeiertag	`istErsterWeihnachtstag(int t, int m, int j)`	25.12.`j`
Zweiter Weihnachtsfeiertag	`istZweiterWeihnachtstag(int t, int m, int j)`	26.12.`j`

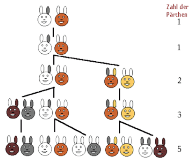
Anmerkung: Im Jahr 1990 wurde der *Tag der Deutschen Einheit* auf dem heutigen Gebiet der alten Bundesländer tatsächlich zweimal begangen.

Programmieren Sie in einer gesonderten Datei `feiertagtests.c`, die das Modul `feiertage` nutzt, eine `main`-Funktion, die einen tagweisen Jahreskalender 2016 mit Anzeige der entsprechenden bundesweiten Feiertage ausgibt. Ein Ausschnitt der Ausgabe könnte wie folgt aussehen:

```
30. 4.2016
 1. 5.2016  Tag der Arbeit
 2. 5.2016
 3. 5.2016
 4. 5.2016
 5. 5.2016  Christi Himmelfahrt
```

Ergänzen Sie Ihre `main`-Funktion, um die Frage zu beantworten, in welchen Jahren zwischen 2000 und 2200 der *Tag der Arbeit* und *Christi Himmelfahrt* auf das gleiche Kalenderdatum fallen.

Aufgabe 3



www.wikipedia.de

Ein Kaninchenpaar wird in einem vollständig abgegrenzten Gebiet ausgesetzt. Von Natur aus zeugt jedes Kaninchenpaar ab dem zweiten Lebensmonat ein weiteres Paar pro Monat. Dieses wiederum beginnt vom zweiten Lebensmonat an, sich auf gleiche Weise fortzupflanzen (siehe Grafik). Wir nehmen an, dass keine Kaninchen sterben.

- Entwickeln Sie eine rekursive Bildungsvorschrift, aus der sich die Anzahl Kaninchenpaare $\mathrm{f}(n)$ nach n Monaten berechnen lässt.
- Recherchieren Sie, unter welchem vordefinierten Namen die rekursive Bildungsvorschrift bereits bekannt ist.
- Schätzen Sie den Werteverlauf Ihrer rekursiven Funktion nach oben ab.
- Wie lässt sich Ihre rekursive Bildungsvorschrift rein *iterativ* (mittels Schleifen) und *explizit* (durch eine nichtrekursive Formel) abbilden?
- **(Laborübung)** Schreiben Sie sowohl ein C-Programm für die rekursive als auch ein C-Programm für eine nichtrekursive Berechnung und vergleichen Sie anhand von selbstgewählten Fallstudien, wieviele Berechnungsschritte (Anzahl rekursive Aufrufe einerseits und Anzahl arithmetische Operationen andererseits) jeweils nötig sind.

Aufgabe 4

Die *Quersumme* einer natürlichen Zahl im Dezimalsystem bietet eine elegante Möglichkeit, die Teilbarkeit durch 3 und 9 zu testen. Darüber hinaus finden Quersummen als Prüfsummen in fehlererkennenden bzw. fehlerkorrigierenden Übertragungscodes Verwendung. Die Quersumme einer natürlichen Zahl ist definiert als Summe ihrer einzelnen Dezimalziffern, beispielsweise besitzt die Zahl 582 die Quersumme 15. Eine rekursive Berechnungsvorschrift ist gegeben durch:

$$\mathrm{quersumme}(n) = \begin{cases} n & \text{falls} \quad n \leq 9 \\ \mathrm{quersumme}(\lfloor \frac{n}{10} \rfloor) + (n \mod 10) & \text{sonst} \end{cases}$$

- **(Laborübung)** Implementieren Sie diese rekursive Berechnungsvorschrift in C und testen Sie sie anhand mehrerer Fallstudien aus.
- Ist der Speicherplatzbedarf zur Ausführung des rekursiven C-Programmes abhängig vom eingegebenen Wert für n? Begründen Sie bitte Ihre Entscheidung.
- **(Laborübung)** Entwickeln Sie ein C-Programm zur Quersummenberechnung, das ohne Rekursion auskommt und vergleichen Sie verbal seinen Speicherplatzbedarf mit der rekursiven Variante.