Stefan Kurtz **Universität Hamburg**
Mathias von Behren **Zentrum für Bioinformatik**

Programmierung in der Bioinformatik
Wintersemester 2015
Übungen zur Vorlesung: Ausgabe am 07.12.2015

Punktevergabe:

- Aufgabe 9.1: 5 Punkte
- Aufgabe 9.2: 1 Punkte
- Aufgabe 9.3: 2 Punkte
- Aufgabe 9.4: 2 Punkte

Aufgabe 9.1 Schreiben Sie ein C-Programm, das beim Aufruf genau zwei Argumente erhält:

- das erste Argument ist entweder der String `-d` oder der String `-a`, der als Option zu verstehen ist.
- das zweite Argument ist der Name einer Datei, die zeilenweise ganze Zahlen vom Typ **`long`** enthält.

Das Programm soll diese Datei in ein Array einlesen und entsprechend der Wahl der Option dieses Array sortieren, und zwar aufsteigend, falls die Option `-a` verwendet wird und absteigend, falls die Option `-d` verwendet wird. Anschliessend soll überprüft werden, ob die Elemente korrekt sortiert sind, indem man zwei aufeinanderfolgende Elemente mit einander vergleicht und das erwartete Ergebnis mit `assert` testet. Schliesslich soll das Array ausgegeben werden und zwar Zeile für Zeile. Zum Einlesen der Zahlen können Sie die entsprechenden Programmzeilen aus dem Skript (Abschnitt Dynamic Memory Management) verwenden. Zum Sortieren verwenden Sie bitte die C-Funktion `qsort`. Beachten Sie, dass Sie zwei verschiedene Vergleichsfunktionen für `qsort` verwenden müssen, je nachdem ob aufsteigend oder absteigend sortiert werden soll. Zum Ausgeben des Arrays können Sie die Funktion `array_show` aus dem Skript verwenden. Vergessen Sie nicht, am Ende Ihres Programms den Speicher für das Array wieder frei zu geben. In Stine finden Sie die Datei `longnumbers.txt` mit 1 000 Zahlen im Wertebereich von $-50\,000$ bis $50\,000$, die Sie zum Testen Ihres Programms verwenden sollen.

Aufgabe 9.2 Ruby hat einige Konventionen für Bezeichner. Die Konventionen wurden in der Vorlesung besprochen. Wenn Sie einen Editor mit Syntax-Highlighting benutzen, wird dieser die verschiedenen Formen der Bezeichner unterscheiden können.

Identifizieren Sie in der Datei `varnames.rb` (siehe Stine) die Bezeichner und korrigieren Sie diese. Achten Sie dabei auch auf Konventionen der Schreibweise, die nicht von Ruby vorgegeben sind, also z.B. die Verwendung von MixedCase (auch CamelCase) oder Unter_strichen.

Aufgabe 9.3 Implementieren Sie ein Ruby-Skript, das für alle Temperaturen von 1 bis 100 Grad Celsius die entsprechende Temperatur in Fahrenheit ausgibt. Zur Erinnerung: wenn t_C die Temperatur in Celsius ist, dann ist $t_F = (t_C \cdot 9)/5 + 32$ die entsprechende Temperatur in Fahrenheit. Verwenden Sie dazu die Klasse `Range` und die Funktion `each` oder die Methoden `Integer.times` oder `Integer.upto` um die Werte im genannten Temperatur-Bereich aufzuzählen.

Beachten Sie, dass Sie eine ganze Zahl `i` zuerst in eine Fließkommazahl umwandeln müssen, damit die Rechenoperationen korrekt funktionieren. Das erreicht man mit der Methode `Integer.to_f`.

Um dieses zu veranschaulichen, werten Sie die folgenden Anweisungen in `irb` aus:

```
puts 6 / 4
puts 6.0 / 4.0
puts 6.to_f / 4.to_f
```

Sie sehen, dass der erste Ausdruck den ganzzahligen Anteil der Division liefert, und dass die zweite und dritte Zeile das erwartete Ergebnis `1.5` liefern.

Formatieren Sie die Ausgabe mit `IO.printf`. Verwenden Sie dazu die Anweisung

```
printf("c = %.2f f = %.2f\n", celsius, fahrenheit)
```

Vergleichen Sie die Ausgabe Ihres Programms mit der Datei `celsius_out.txt`, die im Material zum Übungsblatt in Stine zur Verfügung steht. Zum Vergleich können Sie die Programme `diff`, `sdiff`, oder `vimdiff` verwenden.

Aufgabe 9.4 Implementieren Sie ein Ruby-Programm `backwards.rb`, das Dateien deren Name auf der Kommandozeile via `ARGV` übergeben wurden einliest und die Zeilen in dieser/n Datei(en) in umgekehrter Reihenfolge und rückwärts wieder ausgibt. Schauen Sie sich dazu an, welche Methoden und Objekte der Klassen `File` und `Array` zur Verfügung stehen.

Sie sollten auf die eingelesenen Zeilen die Methode `chomp` aus der Klasse `String` anwenden, um das Zeichen \n zu entfernen.

Ein Beispiel:

Inhalt der Dateien:

```
dies ist Zeile 1
dies ist Zeile 2
```

Ausgabe:

```
2 elieZ tsi seid
1 elieZ tsi seid
```

Die Ausgabe entspricht der Kombination der Linux-Kommandos: `tac <datei> | rev`

Die Lösungen zu diesen Aufgaben werden am 04.01.2016 besprochen.