Stefan Kurtz,
Dirk Willrodt

Universität Hamburg
Zentrum für Bioinformatik

Programmierung in der Bioinformatik
Wintersemester 2013
Übungen zur Vorlesung: Ausgabe am 09.12.2013

Aufgabe 9.1 Ruby hat eine recht freundliche Konvention für Bezeichner. Die Konventionen wurden in der Vorlesung besprochen. Wenn Sie einen Editor mit Syntax-Highligting benutzen, wird dieser die verschiedenen Formen der Bezeichner unterscheiden können.

Identifizieren Sie in der Datei `varnames.rb` 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 9.2 Implementieren Sie ein Ruby-Programm `backwards.rb`, das entweder eine Datei mit dem Namen `testfile` oder Dateien deren Name auf der Komandozeile übergeben wurden einliest und die Zeilen in dieser/n Datei(en) in umgekehrter Reihenfolge und rückwarts wieder ausgibt. Schauen Sie sich dazu an, welche Methoden Objekte der Klassen `File` und `Array` zur Verfügung stellen.

Sie können `ARGF` zum Lesen der Dateien verwenden, wenn Parameter übergeben wurden. Die Methode `String.chomp` könnte Ihnen auch helfen.

Ein Beispiel:

input:

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

output:

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

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

Aufgabe 9.3 Implementieren Sie zwei verschiedene Ruby-Programme, die bei Eingabe zweier positiver ganzer Zahlen m und n die folgende Summe berechnen und ausgeben:

$$\sum_{i=m}^{n} i$$

Falls $m > n$, dann ist diese Summe 0. Falls $m = 0$, dann gilt die folgende Gleichung, die als „Gaußsche Summenformel“ bekannt ist.

$$\sum_{i=0}^{n} i = \frac{n \cdot (n+1)}{2}$$

Das erste Programm soll die Lösung iterativ in einer Laufzeit proportional zur Größe des Intervalls $[n..m]$ berechnen. Das zweite Programm soll die oben angegebene Formel benutzen und die Summe in konstanter Zeit berechnen.

Was passiert, wenn das Intervall $[n..m]$ sehr groß wird?

Aufgabe 9.4 Implementieren Sie ein Ruby-Programm `chardistribution.rb`, das eine Datei einliest, und für alle Zeichen (außer `newline`) die Anzahl der Vorkommen dieses Zeichens ausgibt, sowie die relative Häufigkeit des Vorkommens dieses Zeichens. Zusätzlich soll Ihr Programm noch die Anzahl der Zeichen (außer `newline`) insgesamt ausgeben sowie die Anzahl der Zeilen. Beachten Sie, dass Ihr Programm beliebige ASCII-Zeichen verarbeiten soll (und nur diese zu verarbeiten braucht). Wenden Sie Ihr Programm auf die beiden Dateien `ecoli.fna` und `swiss` aus dem Verzeichnis `/projects/lehre/programming/data` an.

Die Lösungen zu diesen Aufgaben werden am 06.01.2014 besprochen.