Praktikum 4

Ziele

1. Strukturierung von Programmen in Unterprogramme
2. Einfache statistische Auswertungen
3. Stochastische Simulationen
4. Umgang mit Listen (Befüllung, Auslesen, Sortieren)

Aufgaben

Aufgabe 1

Als Refactoring wird die Vereinfachung oder Verbesserung von bestehendem Quellcode bezeichnet, bei der jedoch keine funktionale Änderung des Programms herbeigeführt wird. Ziel ist die Erhöhung von Wartbarkeit, Verständlichkeit und Erweiterbarkeit.

Führen Sie ein Refactoring Ihres Programms zur Bestimmung des Binominalkoeffizienten (Praktikum 3, Aufgabe 2) mit einem Unterprogramm namens `factorial()` durch.

Aufgabe 2

Programmieren Sie zunächst ein Unterprogramm `randomPositiveInteger()`, welches eine ganze Zufallszahl zwischen eins und einem zu übergebenen Maximalwert erzeugt. Die Grenzen 1 und der Maximalwerte sind dabei eingeschlosssen (können als Rückgabewert also tatsächlich auftreten). Der Aufruf `randomPositiveInteger(4)` erzeugt also eine der folgenden Zahlen: 1, 2, 3 oder 4.

Implementieren Sie anschließend ein Unterprogramm `rollD10()`, welches das Wurfergebnis eines zehnseitigen Würfels ermittelt. Verwenden Sie hierzu das Unterprogramm `randomPositiveInteger()` mit einem sinnvollen Parameter.

Simulieren Sie mit diesen Unterprogrammen 1.000 unabhängige Würfe. Führen Sie über die Ergebnisse Buch (Stichwort: Strichliste) und werten Sie anschließend die Ergebnisse in einem Unterprogramm `frequency()` tabellarisch aus: Für jede mögliche Augenzahl soll die absolute und die relative Häufigkeit zu erkennen sein.

Für eine übersichtliche Anordnung ist es sinnvoll, die drei Spalten Augenzahl, absolute und relative Häufigkeit jeweils mit einem Tabulator (Zeichenkette `"\t"`) auszurichten.

Aufgabe 3

Erstellen Sie ein neues Programm zur Simulation von Würfen mit mehreren Würfeln. Die Augenzahlen einer Runde werden dabei für die Auswertung zusammengezählt. Übernehmen Sie sinnvolle Unterprogramme von Aufgabe 2.

Ihr Programm sollte am Ende folgende Bestandteile und Funktionen haben:

- Ein Unterprogramm `simulateDiceGames()` mit Parametern für die Anzahl der Runden, Anzahl der Würfel und Maximalaugenzahl eines Würfels – idealerweise gesteuert durch sinnvolle und gültige Benutzereingaben.
- Ein Unterprogramm `rollDice()` mit Parametern für die Anzahl der Würfel und der Maximalaugenzahl eines Würfels zur Simulation einer Spielrunde mit mehreren Würfeln.
- Nach der Würfelsimulation soll durch ein Benutzerdialog die Wahl der Analyse (Tabelle oder Histogramm) ermöglicht werden.
- Es wird ein Unterprogramm `frequency()` für eben diese tabellarische Anzeige benötigt.
- Für die Anzeige der Ergebnisse als Histogramm programmieren Sie ein Unterprogramm `graph()`. Erstellen Sie mit Hilfe des Asteriskzeichens (`"*"`) ein um 90 Grad gekipptes Säulendiagramm (wie in Praktikum 1, Aufgabe 4). Bedenken Sie, dass sich die absoluten Häufigkeiten schlechter eignen als die mit einen Streckungsfaktor multiplizierten relativen Häufigkeiten (in der Praxis hat sich ein Wert zwischen 60 und 120 bewährt).

Aufgabe 4

Erzeugen Sie mit Hilfe der bekannten Unterprogramme zunächst eine Liste mit vierzig verschiedenen Zufallszahlen zwischen 1 und 1.000 (ganze Zahlen, beide Grenzen eingeschlossen). Geben Sie die Liste lesbar auf dem Bildschirm aus.

Implementieren Sie den Sortieralgorithmus „Selection Sort“ in einem Unterprogramm `selectionSort()` und ordnen Sie die Zahlen damit aufsteigend an. Kontrollieren Sie die sortierte Liste durch eine gut lesbare Anzeige auf dem Bildschirm.

Ermitteln Sie schließlich Mittelwert und Median für die erzeugte Zahlenliste.

Randnotiz – im Internet finden sich interessante Animationen diverser Sortieralgorithmen:

- https://www.toptal.com/developers/sorting-algorithms
- https://www.youtube.com/watch?v=kPRA0W1kECg
- https://www.youtube.com/watch?v=y9Ecb43qw98
- https://www.youtube.com/watch?v=ZZuD6iUe3Pc
- https://www.youtube.com/watch?v=bcwwM6EoveA
- https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
- http://sorting.at/

Aufgabe 5

Füllen Sie mit Hilfe der bereits entwickelten Unterprogramme eine Liste mit 8.000 Zahlen, die mit dem folgenden Verfahren ermittelt werden:

```
sub generateSkewedRandomInteger {
  my $result = (rollD40() % rollD20()) + (rollD60() % rollD40());
  return($result);
}
```

Das Unterprogramm `rollD20()` erzeugt ganzzahlige Zufallsergebnisse aus dem Bereich von eins bis zwanzig; das Unterprogramm `rollD40()` aus dem Bereich von eins bis vierzig. `rollD60()` erzeugt aus dem Bereich von ein bis sechzig. Auch hier gilt wieder, dass die beiden Grenzen der Intervalle eingeschlossen sind.

Ermitteln Sie für die erzeugte Zahlenliste Mittelwert, Median und Modus.

Aufgabe 6

Implementieren Sie ein Unterprogramm `greatestCommonDivisor()`, welches für zwei übergebene ganzen Zahlen den größten gemeinsamer Teiler mit dem Euklidischen Algorithmus ermittelt.