

Programmierübungen

Wintersemester 2007/2008

4. Übungsblatt

24. November 2007

Abgabe bis Freitag, 30. November 23:59 Uhr

Die Abgabe Ihrer Bearbeitung können Sie im eClaus-System durchführen. Erarbeiten Sie Lösungsideen zu den Aufgaben möglichst in Kleingruppen. Es wird jedoch von Ihnen erwartet, dass jeder Teilnehmer eine eigene Lösung abgibt. Sollten kopierte Quelltexte abgegeben werden, so werden grundsätzlich alle Kopien mit 0 Punkten bewertet. In den Vortragsfolien der Programmierübungen oder im Skript zur Einführung in die Informatik abgedruckte Quelltexte können verwendet werden, müssen aber der Programmierrichtlinie entsprechend formatiert und kommentiert werden.

Beachten Sie die Programmierrichtlinie und kommentieren Sie Ihren Quelltext! Dokumentieren Sie unbedingt Ihre Lösungsidee in den Quelltext-Kommentaren.

<http://www.iste.uni-stuttgart.de/ps/Lehre/WS0708/inf-prokurs>

Hinweis: Eine kurze Einführung zur Arbeit mit Textdateien und Zufallszahlen wird am Montagmorgen (26.11.2007, 8:00 Uhr) in der Vortragsübung gegeben.

Aufgabe 4.1: Berechnung von π

(6 Punkte)

Die Zahl π bezeichnet die Fläche des Einheitskreises (der Kreis mit Mittelpunkt $(0, 0)$ und Radius 1). Mit Hilfe eines Zufallsexperiments kann die Zahl π geschätzt werden. Hierzu wählt ein Programm n -mal einen zufälligen Punkt innerhalb des Quadrats $Q = \{(x, y) \mid x \in [0..1), y \in [0..1)\}$ und zählt, wie oft dieser Punkt innerhalb des Einheitskreises liegt; das sei s mal der Fall. Aufgrund der Tatsache, dass der Flächeninhalt des Quadrats 1 ist und der Flächeninhalt des Schnitts mit dem Einheitskreis $\frac{\pi}{4}$ lässt sich π schätzen. Man erwartet bei großem n :

$$\frac{\frac{\pi}{4}}{1} \approx \frac{s}{n}$$

Schreiben Sie ein Programm, das dieses Experiment durchführt und lassen Sie es nach je 1.000 Punkten die aktuelle Näherung an π ausgeben.

Verwenden Sie den vordefinierten Subtyp Float. Einen Pseudozufallszahlen-Generator gibt es in dem generischen Paket `Ada.Numerics.Float_Random`.

Hinweis₁: Die Prozedur `Ada.Numerics.Float_Random.Reset` darf in jedem Programmablauf nur einmal aufgerufen werden, um den Generator zu initialisieren.

Hinweis₂: Falls Sie diese benötigen, finden Sie Funktionen zur Berechnung von Quadratwurzeln in dem Paket `Ada.Numerics.Elementary_Functions`.

Aufgabe 4.2: Datei-Statistiken

(7 Punkte)

Schreiben Sie ein Programm „File_Stats“, das Text-Dateien analysiert. Ihr Programm gibt für eine analysierte nicht-leere Datei folgende Statistik aus:

```
Analyisierte Datei: fraction_helpers.adb
Durchschnittliche Zeilenlaenge: 21 Zeichen
Kuerzeste Zeile: 0 Zeichen
Laengste Zeile: 79 Zeichen
Anzahl Zeilen: 65
Anzahl ';' : 20
```

Sollte jedoch die Datei keinen Inhalt besitzen, so erfolgt nur die folgende Ausgabe:

```
Analyisierte Datei: empty.txt
Datei ist leer.
```

1. Ihr Programm fragt den Benutzer nach einem Dateinamen, liest die Datei dieses Namens ein und analysiert sie. Behandeln Sie auftretende Fehler (z. B. Datei nicht vorhanden) und geben Sie im Fehlerfall sinnvolle Meldungen aus.
2. Erweitern Sie Ihr Programm aus Teilaufgabe 1, so dass die Namen von Dateien als Kommandozeilen-Argumente übergeben werden können.
 - Wird kein Kommandozeilen-Argument angegeben, so soll weiterhin die Eingabe vom Benutzer verlangt werden.
 - Wird mindestens ein Kommandozeilen-Argument angegeben, so soll der Benutzer nicht mehr zur Eingabe eines Namens aufgefordert werden.
 - Werden mehrere Kommandozeilen-Argumente angegeben, so sollen die Dateien nacheinander analysiert werden. Es wird für jede Datei eine separate Statistik ausgegeben.

Hinweis: verwenden Sie zur Durchführung der Analyse eine Prozedur „Analyze“, der ein Dateiname als Parameter übergeben werden kann.

Aufgabe 4.3: Mehrdimensionale Felder

(7 Punkte)

In dieser Aufgabe soll die Frage untersucht werden, wie viele Spielzüge ein Springer mindestens braucht, um von einem bestimmten Startfeld ausgehend ein beliebiges Feld eines Schachbretts zu erreichen. Ein Springer zieht in jedem Zug zwei Felder weit in eine beliebige Richtung (westlich, östlich, nördlich oder südlich) und ein Feld weit in eine der Richtungen, die im rechten Winkel abzweigen (z. B. 2 Felder westlich und 1 Feld südlich).

Laden Sie von der Übungswebseite die Spezifikation des Pakets Chess herunter und implementieren Sie die Prozedur `Knight_Distance`. In Werten des zwei-dimensionalen Arraytyps `Chess_Board` wird zu jedem Feld des Schachbretts gespeichert, in wieviel Zügen dieses Feld erreicht werden kann.

Schreiben Sie ein Programm `dist_matrix`, das den Benutzer zur Eingabe eines Buchstabens und einer Zahl auffordert und diese Eingabe als Startfeld des Springers interpretiert. Als Ausgabe erzeugt daraufhin das Programm eine Matrix, die für jedes Feld des Schachbretts die Anzahl der benötigten Züge angibt.

Beispiel mit Startfeld C3:

Start-Position horizontal: C

Start-Position vertikal: 3

	A	B	C	D	E	F	G	H
1	4	1	2	1	4	3	2	3
2	1	2	3	2	1	2	3	4
3	2	3	0	3	2	3	2	3
4	1	2	3	2	1	2	3	4
5	4	1	2	1	4	3	2	3
6	3	2	3	2	3	2	3	4
7	2	3	2	3	2	3	4	3
8	3	4	3	4	3	4	3	4