

Programmierübungen

Wintersemester 2007/2008

5. Übungsblatt

30. November 2007

Abgabe bis Freitag, 7. Dezember 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>

Aufgabe 5.1: Ausnahmen

(6 Punkte)

Ein Paar natürlicher Zahlen a, b kann durch die Funktion f mit: $f(a, b) = 2^a \cdot 3^b$ in einer Zahl kodiert werden. Schreiben Sie zwei Unterprogramme „Encode“ und „Decode“. Encode soll als Parameter zwei natürliche Zahlen übergeben bekommen, daraus die kodierte Zahl berechnen und zurückgeben. Decode bekommt als Parameter eine kodierte Zahl, berechnet daraus die beiden ursprünglichen Zahlen und gibt diese zurück.

Schreiben Sie ein Hauptprogramm Code, das als Kommandozeilen-Argumente entweder eine Zahl z_1 übergeben bekommt und das dekodierte Zahlenpaar a_1, b_1 mit $z_1 = f(a_1, b_1)$ ausgibt, oder das zwei Zahlen a_2, b_2 übergeben bekommt und die kodierte Zahl $z_2 = f(a_2, b_2)$ ausgibt.

Wie Sie wissen, wird in einem Ada-Programm bei Zuweisung eines Werts, der den zulässigen Wertebereich verlässt, die Ausnahme Constraint_Error erhoben. Falls die Berechnung der Kodierung oder Dekodierung einen Constraint_Error zur Folge hatte, so gibt das Programm einen sinnvollen Text als Fehlermeldung aus. Verwenden Sie zum Erzeugen dieser Meldung einen Exception-Handler. Das Programm soll auch sinnvolle Meldungen ausgeben, falls zu viele oder zu wenige Kommandozeilen-Argumente übergeben werden.

Beispiele (\$ steht für die Eingabeaufforderung):

```
$ ./code 2 5
972
$ ./code 384
7 1
$ ./code 17
Unqueltige Kodierung!
$ ./code
Bitte Kodierung oder zu kodierendes Zahlenpaar uebergeben!
$ ./code 1 2 3
Bitte Kodierung oder zu kodierendes Zahlenpaar uebergeben!
```

Hinweis₁: Entscheiden Sie, ob Sie eine Prozedur oder eine Funktion verwenden sollten.

Hinweis₂: Der Exception-Handler kann entweder in dem Unterprogramm selbst stehen oder in dem aufrufenden Unterprogramm. Überlegen Sie welche Variante sinnvoller ist. Nehmen Sie dazu an, dass die beiden Unterprogramme auch in anderen Programmen wiederverwendet werden könnten.

Aufgabe 5.2: Abstrakte Datenobjekte

(14 Punkte)

Sicher kennen Sie das Galgenraten-Spiel. Darin denkt sich der Spielleiter ein Wort aus und schreibt so viele Unterstriche wie das Wort Buchstaben hat auf ein Papier. Der Spieler darf dann wiederholt einen Buchstaben raten und alle Positionen des Worts, an denen dieser Buchstabe vorkommt, werden aufgedeckt. Rät der Spieler einen Buchstaben, der im gesuchten Wort nicht vorkommt, oder der bereits aufgedeckt wurde, so wird ein Strich an eine Galgenzeichnung angefügt. Sobald der Galgen samt gehängtem Strichmännchen fertiggestellt ist, hat der Spieler verloren. Hat der Spieler jedoch das Wort vollständig aufgedeckt, so hat er gewonnen.

Eine unvollständige Implementierung des Spiels können Sie von der Übungs-Webseite herunter laden. Entpacken Sie das Zip-Archiv. Darin finden sich das Hauptprogramm Galgenspiel sowie das Paket Hangman. Das Paket bietet durch seine Unterprogramme Zugriffsmöglichkeit auf einige globale Variablen. Zur Laufzeit des Programms dient das Paket als Abstraktion für den Spielzustand. Die Unterprogramme des Pakets führen die Spielaktionen durch und lassen nur regelkonforme Veränderungen des Spielzustands zu.

galgenspiel.adb: Hauptprogramm des Spiels. Dies können Sie nach Ihren Wünschen verändern. Es dient nur als zusätzliche Erklärung der Funktionsweise, sowie zum Testen des Pakets Hangman. Beachten Sie, dass das Hauptprogramm nicht alle Verwendungsmöglichkeiten des Pakets ausreizt und deshalb kein vollständiger Test ist. Richten Sie sich beim Schreiben Ihres Quelltexts nach der Anforderungsspezifikation und nicht nach diesem Beispiel.

hangman.ads: Die Spezifikation des Pakets ist das Kernstück der Aufgabe. Sie darf nicht verändert werden. Hier finden Sie eine genaue Anforderungsspezifikation der erwarteten Funktionalität.

hangman.adb: Die Implementierung des Pakets soll von Ihnen geschrieben werden. Es sind bereits einige globale Variablen-Deklarationen vorhanden. Diese sind als Hinweis auf eine mögliche Implementierung gedacht. Beachten Sie die mit „TODO“ gekennzeichneten Kommentare. Diese kennzeichnen Stellen an denen auf jeden Fall Hand angelegt werden muss.

Hinweise (mögliches Vorgehen):

- Überlegen Sie sich zunächst, welche verschiedenen Spielzustände auftreten können und wie sich während des Spiels der aktuelle Spielzustand mit Variablen modellieren lässt.
- Dokumentieren Sie möglichst exakt, wie Sie die Variablen verwenden wollen.

- Prüfen Sie für jedes Unterprogramm, welche Veränderungen an den Werten der Variablen durchgeführt werden müssen und ob diese Veränderungen konsistent zu Ihrer Planung durchgeführt werden können.
- Schreiben Sie den Quelltext für die Unterprogramme.
- Testen Sie Ihre Implementierung. Modifizieren Sie auch das Galgenspiel-Programm um weitere Tests durchzuführen.
- Sie dürfen die Deklarationen der globalen Variablen im *Body* des Pakets Hangman verändern oder ersetzen. Nur die Datei `hangman.ads` darf nicht verändert werden.