Programmierübungen

Wintersemester 2006/2007

1. Übungsblatt

25. Oktober 2006

Abgabe bis Freitag, 3. November 23:59 Uhr.

Die Abgabe Ihrer Bearbeitung können Sie im eClaus-System durchführen. Quelltexte, die Teil der Aufgabenstellung sind, werden auf der Webseite zur Veranstaltung zur Verfügung gestellt. 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.
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/WS0607/inf-prokurs`

Aufgabe 1.1: Stellenwertsysteme (5 Punkte)

Aus der Vorlesung Einführung in die Informatik kennen Sie Stellenwertsysteme. Schreiben Sie ein Programm, das eine ganze Zahl aus einem Stellenwertsystem in ein anderes umwandeln kann, mit folgendem Ablauf:

a. Das Programm gibt den Text „Basis der Eingabe: “ aus.

b. Der Benutzer gibt eine Dezimalzahl im Bereich $2 - 36$ ein. Ist die eingegebene Zahl außerhalb dieses Bereichs, so darf das Programm unkontrolliert abbrechen.

c. Das Programm gibt den Text „Eingabezahl: “ aus.

d. Der Benutzer gibt eine Zahl in dem Stellenwertsystem der Basis aus Schritt b ein. Als Ziffern dienen die arabischen Ziffern ‚0‘–‚9‘ und die Buchstaben ‚A‘–‚Z‘ sowie ‚a‘–‚z‘ nach Wahl des Benutzers. Groß- und Kleinbuchstaben dürfen auch gemischt werden. Sollte die Eingabe andere als diese Zeichen enthalten, so gibt das Programm die Meldung „Eingabe nicht verstanden.“ aus und bricht ab.

e. Das Programm gibt den Text „Basis der Ausgabe: “ aus.

f. Der Benutzer gibt eine Dezimalzahl im Bereich $2 - 36$ ein. Ist die eingegebene Zahl außerhalb dieses Bereichs, so darf das Programm unkontrolliert abbrechen.

g. Das Programm gibt die in Schritt d eingegebene Zahl als Zahl im Stellenwertsystem zur Basis aus Schritt f aus.

Ihr Programm soll mindestens die Zahlen im Bereich $-$Integer'Last .. Integer'Last verarbeiten können. Stellen Sie sicher, dass Zwischenergebnisse Ihrer Berechnungen keine Überläufe verursachen (nicht zu groß oder zu klein werden).
Beispielabläufe:

```
Basis der Eingabe: 16
Eingabe: -02A3f
Basis der Ausgabe: 3
Ausgabe: -112211120

Basis der Eingabe: 36
Eingabe: ABC
Basis der Ausgabe: 10
Ausgabe: 13369

Basis der Eingabe: 2
Eingabe: x
Eingabe nicht verstanden.
```

Aufgabe 1.2: Münzauswahl (5+5+5 Punkte)

Gesucht wird eine möglichst kleine Anzahl verschiedener Münzen, deren Wert genau einen bestimmten Betrag hat. Die Stückelung der zur Verfügung stehenden Münzen soll beliebig festgelegt werden. In den folgenden Teilaufgaben sollen Sie zu diesem Zweck ein Ada-Programm entwickeln.

a) Laden Sie die Paketspezifikation und die Gerüst-Implementierung des Pakets Denominations von der Webseite herunter. Das Paket enthält Datentypen, die Sie verwenden sollen und Unterprogramme, deren Implementierung Sie ergänzen sollen.

   Verändern Sie die Paketspezifikation („.ads“-Datei) nicht! Ihre Implementierung muss mit dem von uns ausgegebenen Quelltext übersetzt und ausgeführt werden können. Schreiben Sie zunächst die Benutzungsschnittstelle:

   1. Die Benutzerin soll die Stückelung in aufsteigender Reihenfolge eingeben. Das Programm gibt wiederholt den Text „`Münze` n`: `“ aus, wobei n ein Zähler ist, der von 1 an aufwärts zählt. Das Programm erwartet die Eingabe einer natürlichen Zahl (inkl. 0). Gibt die Benutzerin nacheinander zwei Münzwerte ein, wobei der zweite kleiner als der erste ist, so gibt das Programm die Warnung aus „Warnung: Eingabe ist nicht aufsteigend sortiert. Ergebnisse werden falsch sein.“, akzeptiert die Eingabe aber trotzdem.
   2. Gibt die Benutzerin den Wert 0 ein, so endet die Eingabe der Münzwerte. Die 0 zählt selbst nicht als Münzwert.
   3. Sie dürfen in Ihrem Programm eine maximale Anzahl für die Münzwerte festsetzen. Falls Sie eine solche Obergrenze festgesetzt haben, so muss Ihr Programm die Meldung „Maximale Anzahl Münzen erreicht.“ ausgeben sobald die Benutzerin diese Anzahl an Werten eingegeben hat. Dann fährt das Programm mit Schritt 4 fort. Ihr Programm muss mindestens 100 Münzwerte akzeptieren.
   4. Das Programm gibt den Text „Geldbetrag: “ aus und wartet auf Eingabe einer nicht-negativen ganzen Zahl.

5. Das Programm gibt zwei Zeilen aus: „Ergebnis b) ...“ und „Ergebnis c) ...“, wobei „...“ jeweils durch das Ergebnis der entsprechenden Teilaufgaben ersetzt wird. Implementieren Sie dazu die Prozedur Put_Usage aus dem Paket Denominations und verwenden Sie diese zur Ausgabe. Beachten Sie die Hinweise in den Quelltext-Kommentaren der Prozedur Put_Usage genau, diese definieren das Ausgabeformat. Rufen Sie die Funktionen Split_Greedy auf, um das Ergebnis von Teilaufgabe b zu berechnen, rufen Sie Split_Backtracking auf um das Ergebnis von Teilaufgabe c zu berechnen.

Beispielabläufe (nach Implementierung der Teilaufgaben b und c):

```
Münze 1: 3
Münze 2: 6
Münze 3: 13
Münze 4: 0
Geldbetrag: 35
Ergebnis b) 2x13 1x6 1x3
Ergebnis c) 2x13 1x6 1x3

Münze 1: 2
Münze 2: 1
Warnung: Eingabe ist nicht aufsteigend sortiert. Ergebnisse werden falsch sein.
Münze 3: 2
Münze 4: 5
Münze 5: 2
Warnung: Eingabe ist nicht aufsteigend sortiert. Ergebnisse werden falsch sein.
Münze 6: 0
Geldbetrag: 15
Ergebnis b) ...
Ergebnis c) ...
```

Hinweis: „...“ im vorigen Beispiel darf durch beliebigen Text ersetzt werden. Es wird nur bei korrekt sortierter Eingabe erwartet, dass Ihr Programm sich sinnvoll verhält.

```
Münze 1: 2
Münze 2: 5
Münze 3: 7
Münze 4: 0
Geldbetrag: 20
Ergebnis b) Nicht darstellbar.
Ergebnis c) 4x5
```

b) Implementieren Sie die Funktion Split_Greedy. Ziel der Funktion ist es zur Darstellung eines gesuchten Betrags eine Auswahl von Münzen zu finden. Verwenden Sie den folgenden Algorithmus:

1. Wählen Sie den größten Münzwert, der kleiner oder gleich dem gesuchten Betrags ist. Gibt es keinen solchen Münzwert hören Sie auf und prüfen ob der gesuchte Betrag erreicht werden konnte oder nicht.

2. Verwenden Sie so viele Münzen des gewählten Münzwerts wie möglich, ohne dass die Summe größer als der gesuchte Betrag wird.
3. Zurück zu Schritt 1

Nutzen Sie aus, dass die Stückelung bereits sortiert eingegeben wurde.

c) Implementieren Sie die Funktion Split_Backtracking. Ziel der Funktion ist es zur Darstellung eines gesuchten Betrags eine Auswahl von Münzen zu finden. Dabei soll die Anzahl der benötigten Münzen minimal sein. Überlegen Sie sich einen Algorithmus, der alle möglichen Kombinationen der Münzen aus der Stückelung durchprobiert. Versuchen Sie Kombinationen, die nicht zum Ziel führen können, auszuschließen um die Rechenzeit gering zu halten.

Beachten Sie, dass es Beträge gibt, die mit einer bestimmten Stückelung gar nicht dargestellt werden können, z.B. kann mit den Münzen 5 und 3 der Betrag 7 nicht dargestellt werden.

Testen Sie Ihr Programm!