Universität Stuttgart
Institut für
Informationssicherheit

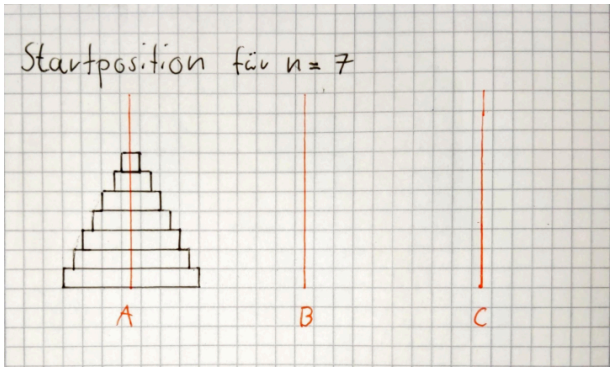
InformatikII - Programmierung in C++
Sommersemester 2023
Maria Unger-Zimmermann

Übungsblatt 7

05.06., 06.06. und 09.06

Problem 7.1: Rekursion: Türme von Hanoi

Die Türme von Hanoi ist ein Rätsel, das aus drei gleich großen Stäben A, B und C besteht, auf die mehrere gelochte Scheiben gelegt werden, welche verschieden groß sind. Zu Beginn liegen alle Scheiben auf Stab A, der Größe nach geordnet, mit der größten Scheibe unten und der kleinsten oben. Ziel des Spiels ist es, den kompletten Scheiben-Stapel von A nach C zu versetzen. Bei jedem Zug darf die oberste Scheibe eines beliebigen Stabes unter der Voraussetzung, dass sich dort nicht schon eine kleinere Scheibe befindet, auf einen der beiden anderen Stäbe gelegt werden. Folglich sind zu jedem Zeitpunkt des Spieles die Scheiben auf jedem Feld der Größe nach geordnet.



Ihre Aufgabe ist es nun eine Funktion zu schreiben, die als Eingabe die Anzahl an Scheiben auf dem ersten Stab (als `int`), Namen des ersten Stabes, den Namen des zweiten Stabes und den Namen des dritten Stabes (das Ziel) bekommt. Die Namen werden jeweils als char übergeben. Mit den Werten 'A', 'B' und 'C'.

```
void towerOfHanoi(int n, char start_stab, char ziel_stab, char hilfs_stab)
{
        ...
}
```

Die Funktion soll auf der Konsole die "Lösung"des Rätsels ausgeben. Die Lösung besteht aus einer Folge an Zügen der Form: «Start-Stab> -> <End-Stab>". Ein Solcher Zug bedeutet wir nehmen die oberste Scheibe von dem Start-Stab und legen ihn oben auf den End-Stab drauf. Beispielausgabe von towerOfHanoi(3, 'A', 'B', 'C'):

A -> C
A -> B
C -> B
A -> C
B -> A
B -> C
A -> C

Der Algorithmus soll wie folgt funktionieren:

- wenn auf dem start_stab eine Scheibe zu bewegen ist (n=1) dann soll diese Scheibe von dem start_stab zum ziel_stab bewegt werden.
- wenn auf dem start_stab mehr als eine Scheibe zu bewegen ist, dann sollen:

1. erst alle Scheiben bis auf den untersten (also n-1 Stück) auf den hilfs_stab bewegt werden. Dazu kann die Funktion towerOfHanoi sich selbst aufrufen (Rekursiver Aufruf). Damit die Scheiben auf den hilfs_stab verschoben werden und nicht auf den ziel_stab, muss die Funktion mit anderen Parametern aufgerufen werden.
   - als start_stab verwenden wir den aktuellen start_stab
   - als ziel_stab verwenden wir den aktuellen hilfs_stab
   - als hilfs_stab verwenden wir den aktuellen ziel_stab
2. Wenn dies geschehen ist, soll nun die unterste Scheibe vom start_stab auf den ziel_stab verschoben werden.
3. jetzt können wir noch die Scheiben, die jetzt auf dem hilfs_stab liegen (einer weniger, da wir ja schon einen auf den ziel_stab bewegt haben), auf den ziel_stab verschieben. Dazu rufen wir noch einmal die Funktion towerOfHanoi auf, dieses Mal mit den folgenden Parametern:
   - als start_stab verwenden wir den aktuellen hilfs_stab (da dort ja die Scheiben liegen)
   - als ziel_stab verwenden wir den aktuellen ziel_stab
   - als hilfs_stab verwenden wir den aktuellen start_stab

Um die Funktion zu testen, rufen wir die Funktion in der main Funktion auf:

```
int main()
{
        towerOfHanoi(3, 'A', 'B', 'C');
}
```

Sie können auch größere Zahlen ausprobieren oder das Programm so umschreiben, dass die Anzahl an Scheiben in der Konsole eingelesen wird.

Problem 7.2: Funktionen und Strukturen

Strukturen (`struct`) hatten wir schon in Aufgabe 4.1; in dieser Aufgabe überzeugen wir uns davon, dass Strukturen problemlos als Parameter und Rückgabewerte von Funktionen auftreten können und studieren das Zusammenspiel von Strukturen und Vektoren (vgl. Aufgabe 4.2).

Wir werden einen virtuellen Zoo bauen – das wird leider alles sehr künstlich werden, damit in einem Programm überschaubarer Länge alle relevanten Dinge vorkommen; wenn Sie es lieber realistischer haben, spricht nichts dagegen, das Programm auszubauen…

1. Definieren Sie eine Struktur mit Namen `tierart` und Elementen
   - `name` (Datentyp `string`) und
   - `anzahl` (Datentyp `int`).

   Schreiben Sie Funktionen
   - `druckeArt` mit einem Parameter vom Typ `tierart` und ohne Rückgabewert, die die Elemente auf dem Bildschirm ausgibt und
   - `eingabeArt` ohne Parameter, dafür mit einem Rückgabewert vom Typ `tierart`, mit der die Elemente von `cin` eingelesen werden (es soll dazu auch jeweils eine passende Aufforderung ausgegeben werden).

2. Definieren Sie nun eine Struktur mit Namen `zoo` und Elementen
   - `name` (Datentyp `string`) und
   - `arten` (ein Vektor mit Elementen vom Typ `tierart`).

   Auch hier soll es wieder je eine Funktion zum Ausdrucken und Einlesen geben – dabei soll der Name des Zoos und die Anzahl der Tierarten ausgegeben bzw. abgefragt werden; die Tierarten sollen dann natürlich mit einer Schleife und den Funktionen aus Teil 1 gedruckt bzw. eingelesen und hinten an den Vektor angefügt werden.
3. Schreiben Sie nun noch eine Funktion `zaehleTiere` mit einem Parameter vom Typ `zoo` und Rückgabewert `int`, die die Gesamtzahl aller Tiere im Zoo berechnet (also die Summe über die Tierarten).

   Diese Gesamtzahl soll beim Drucken des Zoos mit ausgegeben werden.
4. Jetzt üben wir noch mal Referenzparameter und probieren das *Überladen* von Funktionen (Breymann: Kap. 2.2.5) aus.

   Schreiben Sie eine Funktion `ausbruch` mit einem `tierart`-Referenzparameter und ohne Rückgabewert, die einen Ausbruch dieser Tierart aus dem Zoo simuliert: Die Anzahl der Tiere dieser Tierart soll halbiert werden (falls ungerade: Abrunden).

   Schreiben Sie eine weitere Funktion `ausbruch`, diesmal mit einem `zoo`-Referenzparameter, aber ebenfalls ohne Rückgabewert. In dieser Funktion sollen alle Tierarten des Zoos (jeweils mittels der eben geschriebenen Funktion) ausbrechen.

   Simulieren Sie damit einen (allgemeinen) Ausbruch aus Ihrem Zoo!

So könnte ein Dialog mit dem Programm aussehen (Eingaben unterstrichen):

```
Name des Zoos: Wilhelma
Anzahl der Tierarten: 3
Name der Tierart: Affe
Anzahl: 5
Name der Tierart: Okapi
Anzahl: 2
Name der Tierart: Uhu
Anzahl: 3

Zoo Wilhelma
3 Tierarten:
Affe: 5
Okapi: 2
Uhu: 3
Insgesamt 10 Tiere

Nach dem Ausbruch:
Zoo Wilhelma
3 Tierarten:
Affe: 2
Okapi: 1
Uhu: 1
Insgesamt 4 Tiere
```