Übungen zur Vorlesung
„Grundlagen der Informatik 2“
Studiengang E-Technik

SoSe 2002

AG Betriebssysteme, FB 3
Jan Bredereke

ausgegeben: 28.5.2002
Abgabe und Besprechung:
in der Übung am **6.6.2002**

Blatt 2

Stacks

Aufgabe 1: Nicht-begrenzter Stack

Implementiert einen Stack, dessen Fassungsvermögen nur durch den verfügbaren Speicher des Rechners begrenzt ist, und zwar mithilfe einer einfachen Liste. Schreibt die Dateien `unboundedStack.h` und `unboundedStack.c`, die Typdefinition `typedef struct stack *stackH;` sowie die Funktionen `stackH createStack()`, `void push(stackH s, struct stackItem si)`, `struct stackItem pop(stackH s)`, `struct stackItem peek(stackH s)` und `int isEmpty(stackH)` zur Verfügung stellen. Die Funktionalität des Stacks soll die übliche sein, wie auch in der Vorlesung vorgestellt: `push()` legt ein neues Elemen auf den Stack, `peek()` liefert eine Kopie des obesten Stackelements zurück, ohne den Stack zu verändern, `pop()` liefert das oberste Element vom Stack zurück und entfernt es dabei vom Stack, und `isEmpty()` liefert `1` zurück, falls der Stack leer ist und sonst `0`. (`pop()` und `peek()` dürfen nicht auf einem leeren Stack aufgerufen werden.) Die Elemente auf dem Stack sollen immer vom Typ `struct StackItem` sein. `stackItem` soll eine Record mit drei Feldern sein: `char artikeltyp[100]`, `int artikelId` und `int artikelvolumen`.

Löst das Problem, indem Ihr zuerst einen Record `struct stackList` schreibt, der eine einfache Liste über `stackItem`s implementiert. Die Records sollen jeweils zwei Komponenten haben, nämlich einen Zeiger `item` auf das zugehörige Objekt im Stack und einen Zeiger `prev` auf das vorige Listenelement. Unter Verwendung dieser Listenstruktur implementiert Ihr anschließend `unboundedStack`.

Aufgabe 2: Stapel und Roboter

Betrachtet das folgende Szenario: In einem kleinen Unternehmen werden aus Kostengründen (Regale sind teuer ...) alle Lagerartikel einfach auf große Stapel gelegt. Um dennoch in der Lage zu sein, einen Artikel halbwegs leicht wiederzufinden, sind diese Stapel sortiert. Dabei sollen die größten Objekte immer unten angeordnet werden (sonst kippt der Stapel zu leicht um). Leider werden im Wareneingang neue Lieferungen immer in unsortierten Stapeln angeliefert. Glücklicherweise hat sich aber in der letzten Woche ein manisch-depressiver Roboter vorgestellt, der eigentlich nur auf das Ende der Welt wartet (nennen wir ihn Marvin). Er hat sich bereit erklärt, nebenbei auch noch (kostenlos) immer die Artikel der unsortierten Eingangsstapel unter Zuhilfenahme eines Hilfsstapels im Lager jeweils auf einem neuen Stapel sortiert aufzuschichten.

Schreibt für Marvin ein C-Programm `marvin.c`, welches ihm sagt, wie seine Sortieraufgabe für einen gegebenen unsortierten Stapel durch hin- und herstapeln erledigt werden kann. Die grundlegende Idee besteht darin, daß der Eingangsstapel schrittweise abgebaut wird und auf

den Hilfsstapel umgeschichtet werden kann, wobei man sich in einer internen Variablen merkt, wie groß das bislang größte Objekt ist. Wenn der Stapel komplett übertragen ist, so kennt man das momentane Maximum und kann es beim Zurückschichten des Hilfsstapels in den Eingangsbereich auf den sortierten Stapel legen. Dies wiederholt man so lange, bis der Eingangsstapel leer ist, der Lagerstapel ist dann sortiert. (Das Verfahren ist nicht das effizienteste, aber Marvin hat genug Zeit ...)

Der Wareneingangsstapel soll zu Anfang aus einer Datei `eingang.txt` in einen `unbounded-Stack` eingelesen werden (siehe Aufgabe 1). In der Datei soll in jeder Zeile der Typ des Artikels (Schraube, Motorblock, Pferd, ...), eine firmeninterne Kennnummer (z.B. 100042) und das Volumen (als ganze Zahl, die größer als Null ist) angegeben werden, durch jeweils ein Blank getrennt. Für das Einlesen aus der Datei und die Zerlegung einer Zeile benutzt Ihr die Funktionen `fopen()` und `fscanf()`. Die geneuen Beschreibungen dieser Funktionen bekommt man mit `man fopen` bzw. mit `man fscanf`.

Der Sortieralgorithmus soll auf dem Bildschirm Anweisungen für Marvin ausgeben, was er tun soll, also z.B. „x mit Id y und Volumen z: Eingangsstapel `->` Hilfsstapel“ (Die Angabe des Objekts ist nur eine Hilfsinformation zum Nachvollziehen des Algorithmus'; durch die Struktur des Stapels ist dieses Objekt bei einer konkreten Stackbelegung immer eindeutig definiert).