Institut für Informatik

Lehrstuhl für Technische Informatik
Prof. Dr. Björn Scheuermann

Digitale Systeme 2015

C-Programmierprojekt

1 Aufgabenstellung

In einem Hafenterminal werden Container für die Verladung auf Schiffe gelagert. Immer zu Beginn eines Tages ist das Terminal leer. Dann werden zunächst alle Container eines Tages auf der Straße oder per Bahn angeliefert und im Terminal für die Verladung gestapelt. Danach werden die Schiffe eines nach dem anderen beladen. Um dies möglichst einfach zu gestalten, sollen die Container gleich bei ihrer Anlieferung so gestapelt werden, dass beim Beladen der Schiffe keine Umschichtungen mehr erfolgen müssen. Auch während des Ablegens der Container darf nicht umgeschichtet werden.

Das Containerterminal benötigt deshalb einen Plan für das Ablegen der Container. Dieser Plan muss es ermöglichen, dass alle Schiffe nacheinander beladen werden können, wobei immer nur Container entnommen werden, die oben liegen. Außerdem soll – um Platz zu sparen – die Zahl der Stapel minimiert werden. Nehmen Sie an, dass die Stapel beliebig hoch werden können.

Für diese Planungsaufgabe sind folgende Daten verfügbar:

- Die Reihenfolge, in der die Schiffe beladen werden sollen. Alle Schiffe sind durch eine positive ganze Zahl bezeichnet (32 Bit Integer). Die Schiffe werden in der Reihenfolge dieser Nummern beladen.
- Die Reihenfolge, in der die Container ankommen. Beispielsweise könnte es sein, dass zunächst ein Container für Schiff 4 eintrifft, danach einer für Schiff 7, und schließlich zwei hintereinander für Schiff 1.

Ihr Programm soll die Anzahl der mindestens benötigten Stapel berechnen. Außerdem soll eine mögliche Reihenfolge der Container in diesen Stapeln ausgegeben werden.

2 Ein- und Ausgabeformate

Ihr Programm soll seine Eingabe aus einer Datei lesen. Der Name der zu lesenden Datei wird als erste (und einzige) Kommandozeilenoption beim Aufruf übergeben. Die Eingabedatei ist eine Textdatei. In ihr ist die Reihenfolge der eingehenden Container für eine Reihe von Tagen codiert, jeder Tag in einer Zeile. Die Tage (d. h. die Zeilen) werden unabhängig voneinander einer nach dem anderen, aber in einem einzigen Programmaufruf bearbeitet. Das Format für einen der Tage ist eine kommaseparierte Liste der Schiffsnummern der eingehenden Container, ohne Leerzeichen. Für obiges Beispiel wäre die Eingabezeile also

```
4,7,1,1
```

Verwenden Sie für die Ausgabe des Ergebnisses eine Zeile für jeden Tag in der Eingabedatei. Geben Sie zunächst die Zahl der mindestens benötigten Stapel als vorzeichenlose Ganzzahl aus, und hängen Sie dann eine mögliche Anordnung der Container in den Stapeln an. Trennen Sie den ersten Stapel von der Anzahl durch genau ein Leerzeichen; zwischen zwei Stapeln verwenden Sie ebenfalls genau ein Leerzeichen. Die Schiffs-IDs der Container innerhalb eines Stapels trennen Sie durch Kommata, wobei die Stapel von unten nach oben ausgegeben werden. Für die Eingabe `4,7,1,1` gibt es also sechs korrekte Ausgaben:

```
2 4 7,1,1
2 7,1,1 4
2 4,1,1 7
2 7 4,1,1
2 4,1 7,1
2 7,1 4,1
```

Ihr Programm soll *eine* dieser Möglichkeiten ausgeben.

3 Beispieldaten

Wir stellen ihnen auf der Homepage der Lehrveranstaltung einige Beispiel-Eingabedateien zum Testen Ihres Programms zur Verfügung. Auch eine "Musterlösung" für jeden Eingabedatensatz finden Sie dort. Denken Sie beim Vergleich der Ausgabe Ihres Programms mit diesen Lösungen daran, dass die minimale Zahl der Stapel immer übereinstimmen muss, für die konkrete Reihenfolge der Container in den Stapeln aber mehrere zulässige Möglichkeiten existieren können.

Für die Überprüfung Ihrer Abgabe werden andere Datensätze zum Einsatz kommen.

4 Hinweise zum Algorithmus

Es ist möglich, das Problem folgendermaßen zu lösen: Für jeden eingehenden Container kommen ein oder mehrere bereits existierende Stapel in Frage, außerdem kann ein neuer Stapel angelegt werden. Ihr Programm könnte der Reihe nach alle Möglichkeiten ausprobieren und diejenige wählen, die mit der geringsten Anzahl an Stapeln auskommt. Die bestehenden Ablagemöglichkeiten für einen eingehenden Container hängen dabei von der Ablage der vorangegangenen Container ab – eine rekursive Implementation bietet sich an. Diese Vorgehensweise benötigt unter Umständen viel Rechenzeit, wird aber zu einem korrekten Ergebnis führen.

Wenn Sie keine andere Lösung finden, dann können Sie dieses Brute-Force-Verfahren implementieren. Berücksichtigen Sie dann auf jeden Fall die folgenden beiden Beobachtungen:

- Wenn ein Container mit einer höheren Schiffs-ID auf einem mit einer niedrigeren Schiffs-ID abgelegt wird, so kann dies niemals zu einer gültigen Lösung führen. Diese Möglichkeit muss deshalb nicht überprüft werden.
- Wenn bereits ein Stapel existiert, dessen oberster Container zum selben Schiff gehört wie der gerade eingehende, dann ist es immer die beste Option, den Container auf diesen Stapel hinzuzufügen. Andere Ablagemöglichkeiten müssen also nicht getestet werden.

Es gibt allerdings auch (mindestens) eine sehr viel elegantere Lösung für das Problem, die außerdem auch etwas einfacher umzusetzen ist. Sie dürfen jedes (korrekte) Lösungsverfahren

einsetzen, das zwei grundlegende Bedingungen erfüllt:

- Es soll nicht noch ineffizienter sein als das oben skizzierte Brute-Force-Verfahren.
- Sie müssen es uns bei der Vorstellung Ihres Programms erläutern und begründen können.

5 Zeitplan

Datum	Beschreibung
13.05.2015	Bekanntgabe der Aufgabenstellung
15.07.2015	Empfohlene Deadline für eine erste Abgabe; bei Abgabe bis zu diesem Termin können wir im Falle von Problemen mit der Abgabe gewährleisten, dass nach einer ersten Rückmeldung noch Zeit zur Nachbesserung bis zur endgültigen Deadline bleibt
16.08.2015	Spätestmögliche Deadline für die Abgabe der Endversion, die alle Kriterien vollständig erfüllen muss
14.09.2015 - 23.09.2015	In diesem Zeitraum finden die mündlichen Abnahmen in Einzelsitzungen statt. Eine frühere Abnahme ist nach individueller Absprache möglich.

6 Wichtige Hinweise

Beachten Sie bei der Erstellung Ihres Programms **unbedingt** folgende Punkte:

- Kommentieren Sie Ihren Quelltext in angemessenem Umfang.
- Halten Sie sich strikt an das oben definierte Ausgabeformat und die Aufrufkonventionen. Wir überprüfen Ihr Programm halbautomatisch auf die Korrektheit der ausgegebenen Lösungen. *Eine falsch formatierte Ausgabe wird nicht als korrekt anerkannt!* Im Zweifelsfall fragen Sie *rechtzeitig vor Abgabe* nach!
- Fangen Sie falsch formatierte Eingabedaten mit einer Fehlermeldung ab.
- Ihr Programm muss auf der Kommandozeile ohne grafische Oberfläche lauffähig sein.
- Verwenden Sie keine Bibliotheken außer der C-Standardbibliothek.
- Achten Sie darauf, dass Ihr Programm auf unterschiedlichen Architekturen lauffähig ist (z. B. mit unterschiedlich langen Integer-Typen).
- Verwenden Sie eine einzige C-Quelltextdatei. Ihr Programm muss sich mit einem einzelnen GCC-4.9.1-Compileraufruf ohne weitere Optionen fehlerfrei übersetzen und linken lassen.
- Legen Sie Ihr Programm so aus, dass es mit beliebig großen Eingabedaten umgehen kann. Hierfür ist es unbedingt notwendig, dass Sie dynamische Speicherverwaltung einsetzen. Wir werden Ihr Programm mit *großen* Datensätzen testen! (Das bedeutet auch sehr lange (d. h.: beliebig lange) Ein- und Ausgabezeilen!)
- Ihr abgegebenes Programm wird automatisch auf Speicherlecks überprüft. Stellen Sie sicher, dass es keine Speicherlecks aufweist. Dies können Sie beispielsweise mit dem Werkzeug Valgrind bewerkstelligen. Geben Sie jeglichen dynamisch allozierten Speicher vor der Terminierung des Programms korrekt wieder frei.