Dr. Matthias Grabert Abteilung Angewandte Informationsverarbeitung 21. November 2002
Johannes Mayer Blatt 5

UNIVERSITÄT ULM · SCIENDO · DOCENDO · CURANDO ·

Objektorientierte Softwareentwicklung mit C++ (WS 2002/2003)

Abgabetermin: 28. November 2002

Lesen Sie zuerst alles durch bevor Sie zu Arbeiten beginnen. Sie könnten es sonst bereuen. Zu Risiken und Nebenwirkungen fragen Sie bitte ihren Dozenten oder Übungsleiter.

1 Erdös-Zahlen (2 Punkte)

Paul Erdös (1913 – 1996) war ein berühmter ungarischer Mathematiker, ein richtiger Exzentriker, der Zeit seines Lebens eine ganz Menge publiziert hat. Weil es eine Ehre ist, gemeinsam mit so einem berümten Mann zu publizieren, wollten natürlich möglichst viele Leute mit ihm zusammen veröffentlichen. Und diejenigen die eine solche Gelegenheit verpasst haben, haben sich im Nachhinein ganz schön darüber geärgert. Leute, die nicht direkt mit ihm publiziert haben, sind auch schon stolz, wenn es einen Weg zwischen ihnen und Erdös im „Publikationsgraphen“ gibt. (Dieser enthält für jeden Menschen einen Knoten und zwischen zwei Knoten gibt es genau dann eine Kante, wenn diese beiden Personen miteinander publiziert haben.) Weil Erdös sich auch viel mit Graphentheorie beschäftigt hat, kam die Idee der Erdös-Zahlen auf (siehe *The Erdös Number Project*). Die Erdös-Zahl einer Person gibt die Länge des kürzesten Weges zwischen Erdös und ihr im Publikationsgraph an. Gibt es keinen solchen Weg, so ist sie unendlich. Herr Prof. Maier ist meines Wissens nach der einzige in unserer Fakultät mit der Erdös-Zahl 1, d. h. er hat schon einmal mit Erdös zusammen publiziert. Die Datei *erdoes.txt* enthält zu jeder Person mit Erdös-Zahl 1 (diese stehen jeweils linksbündig in der Datei) die Personen mit denen sie publiziert hat (diese sind um einen Tabulator eingerückt in der Datei) – außer Erdös. Die Personen mit Erdös-Zahl 1 sind mit Großbuchstaben geschrieben. Bei den anderen handelt es sich also um „Erdös-2-Leute“.

Bestimmen sie zu allen „Erdös-2-Leuten“ jeweils die Anzahl von „Erdös-1-Leuten“, mit denen sie publiziert hat und geben sie die Personen mit Erdös-Zahl 2 alphabetisch sortiert aus. Hinter der Person soll – durch einen Tabulator getrennt – die Anzahl der Personen mit Erdös-Zahl 1 stehen, mit denen sie schon eine Veröffentlichung hatte. Die Datei *erdoes.txt* soll ihrem Programm über die Standardeingabe zugeführt werden.

2 Lustiges Wörterzählen (2 Punkte)

Schreiben sie ein Programm, das Schiller's Bürgschaft (*schiller.txt*) von der Standardeingabe liest. Alle Großbuchstaben in der Eingabe sollen beim Lesen in Kleinbuchstaben konvertiert werden. Ihr Programm soll eine alphabetische Liste der vorkommenden Wörter ausgeben. Durch Tabulator getrennt soll hinter jedem Wort die Anzahl der Vorkommen stehen.

3 Map ma mal (6 Punkte)

Verwenden sie für die vorigen beiden Aufgaben die *selbst geschriebene* Klasse `map`. Diese soll ein assoziatives Array implementieren. Mit einem solchen Objekt kann man wie mit einem Array arbeiten und es mit Schlüsselwerten indizieren. Die zu implementierenden Methoden lassen sich aus dem Beispielprogramm *test.cc* ersehen. (Die Klasse `map` befindet sich dabei im Namensraum `lib`.) Als Anforderung an den Schlüsseldatentyp (der erste Template-Parameter wofür im Beispiel die Klasse `string` verwendet wurde) ist, dass er *nur* die totale Ordnung `<` besitzen muss (mehr ist zwar schön aber nicht notwendig).

Tipps:

- Ihre `map`-Implementierung darf natürlich die `vector`-Klasse aus den Übungen verwenden.
- Der Iterator muss wohl eine innere Klasse von `map` sein (analog zu den `error`-Klassen aus der Vorlesung). Dann kann dieser auch auf die privaten Elemente von `map` zugreifen!
- Für diese Aufgabenstellung bietet es sich an, die Elemente (der `map`) intern sortiert zu halten (nach dem Schlüssel versteht sich). Dann kann man die Elemente mit dem Iterator schon in sortierter Reihenfolge durchlaufen. Das Einfügen eines neuen Elements bzw. die Suche nach einem vorhandenen sollte dann mittels binärer Suche erfolgen. (Dabei darf aber nur der `<`-Operator von Schlüsseln verwendet werden!)
- Zum Zerlegen von Strings in Wörter kann man gut `strtok()` aus `string.h` gebrauchen.