ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Informatik I - D-ITET

Wintersemester 2003/2004

Prof. Markus Gross, Bruno Heidelberger, Richard Keiser, Nicky Kern, Edouard Lamboray, Christoph Niederberger, Tim Weyrich, Felix Eberhard, Manuel Graber, Nathalie Kellenberger, Marcel Kessler, Lior Wehrli

Uebung 10 - Klassen I

Ausgabe:	19. Januar 2004
Abgabe:	26. Januar 2004
Autor:	Bruno Heidelberger

1. Klasse für rationale Zahlen — 7 Punkte

Es soll eine kleine und unkomplizierte Bibliothek zum Rechnen mit rationalen Zahlen geschrieben werden. Rationale Zahlen sind Zahlen, die durch einen Bruch darstellbar sind, wobei Zähler und Nenner ganzzahlig sein müssen. In dieser Version der Bibliothek werden die Rechenergebnisse noch in **ungekürzter** Form vorliegen.

Implementiere eine Klasse `Rational` für die Verarbeitung von rationalen Zahlen. Die Klasse soll zwei ganzzahlige Membervariablen besitzen, nämlich `zaehler` und `nenner`. Weiter soll sie die folgenden Memberfunktionen definieren und implementieren:

- `getZaehler()` und `getNenner()`, um Zähler resp. Nenner abzufragen.
- `set()`, um Zähler und Nenner neu zu setzen.
- `add()`, `sub()`, `mult()` und `div()`, um die entsprechende Rechenoperation durchzuführen.

Die Methoden `getZaehler()` und `getNenner()` sollen den entsprechenden ganzzahligen Wert zurückliefern. Mit Hilfe der Methode `set()` soll es möglich sein, den Zähler und Nenner gleichzeitig neu zu setzen. Die vier Rechenoperationen `add()`, `sub()`, `mult()` und `div()` kriegen als Parameter eine konstante rationale Zahl, die dann z.B. zu der aufrufenden rationalen Zahl dazu addiert wird.

In welcher Section (`public` oder `private`) deklarierst du die Membervariablen? Wo die Memberfunktionen? Begründe deine Wahl!

Achte darauf, dass die Klassendeklaration in einer Headerdatei (`.h`) und deren Implementation in einer C++-Datei (`.cpp`) gespeichert ist.

Schreibe ausserdem ein kleines Testprogramm, welches zwei rationale Zahlen einliest, diese nach Wunsch addiert, subtrahiert, multipliziert oder dividiert und das entsprechende Ergebnis ausgibt.

2. Gekürzte rationale Zahlen — 3 Punkte

Schreibe eine Methode `kuerzen()` für die Klasse `Rational`, die am Ende jeder mathe-

matischen Operation aufgerufen wird, um sicher zu stellen, dass `zaehler` und `nenner` am Schluss in gekürzter Form und mit korrektem Vorzeichen vorliegen. Verwende dazu eine Version des Euklid-Algorithmus zur Berechnung des grössten gemeinsamen Teilers:

```
int ggt(int x, int y)
{
      while(y > 0)
      {
            int rest = x % y;
            x = y;
            y = rest;
      }
      return x;
}
```

3. Unix-Einführung – Archivierung und Dateiaustausch fakultativ

Das Archivierungsprogramm `tar`

Gerade, wenn man Dateien verschicken oder archivieren möchte, ist es hilfreich, mehrere Dateien oder ganze Verzeichnisbäume in einer einzigen Datei zu bündeln. Eine solche Datei nennt man ein *Archiv*. In der UNIX-Welt sind vor allem tar-Archive[1] verbreitet. Ein tar-Archiv wird mit dem Programm `tar` erstellt und erhält die Endung `.tar`. Der Aufruf lautet:

```
tar cvf archivName.tar Datei-/Verzeichnisliste...
```

Das Kommando packt alle angegebenen Dateien und Verzeichnisse in das Verzeichnis mit dem Namen `archivName.tar`[2]. Um zum Beispiel Eure Arbeit zum Miniprojekt in ein Archiv zu packen, könnt Ihr aus dem `painter`-Verzeichnis mit `cd ..` ein Verzeichnis hoch gehen und dort

```
tar cvf mein-painter-backup.tar painter
```

aufrufen. `tar` packt dann das Verzeichnis `painter` mit all seinen Dateien und Unterverzeichnisse in das Archiv `mein-painter-backup.tar`.
Auch zum Entpacken eines solchen Verzeichnisses wird wieder `tar` verwendet, nur diesmal mit den Optionen[3] `xvf` anstatt `cvf`:

```
tar xvf meinArchivName.tar
```

Es werden dann alle in meinArchivName.tar enthaltenen Dateien/Verzeichnisse in das aktuelle Verzeichnis geschrieben. Falls sich an dieser Stelle bereits Verzeichnisse und Dateien gleichen Namens befinden, werden diese ohne Warnung überschrieben.

---

1. Die Abkürzung tar kommt noch aus der Zeit, als man vornehmlich Magnetbänder zur Archivierung verwendet hat, und steht für **t**ape **ar**chive.
2. *Achtung:* Wenn Ihr den Archivnamen anzugeben vergesst, interpretiert `tar` die erste Datei aus der Dateiliste als Archivnamen und überschreibt sie mit einem Archiv, das die restlichen Dateien enthält. Damit geht der Inhalt der ersten Datei verloren!
3. Eine ausführliche Erklärung der `tar`-Optionen findet Ihr — wie immer — in der man-page zu `tar`.

Komprimieren von Dateien mit `gzip/gunzip`

In einem tar-Archiv werden alle Dateien unkomprimiert gespeichert. Das heisst, sie werden 1-zu-1 in die Archivdatei kopiert. Tatsächlich liesse sich das aber platzsparender gestalten, da sich insbesondere Text-Dateien effizient *komprimieren* lassen. Dabei gehen keine Daten verloren, und die Datei kann jederzeit wieder *dekomprimiert* werden.
Mit `gzip` können beliebige Dateien komprimiert werden. Möchte man zum Beispiel eine Datei `main.cpp` „kleinkriegen", schreibt man:

```
gzip main.cpp
```

Anschliessend ist die Datei `main.cpp` verschwunden, und es liegt ein `main.cpp.gz` im aktuellen Verzeichnis. Diese Datei ist im Normalfall wesentlich kleiner als das „Original". Um aus `main.cpp.gz` wieder `main.cpp` zu rekonstruieren, muss man

```
gunzip main.cpp.gz
```

aufrufen. Besonders häufig setzt man `gzip` ein, um tar-Archive zu komprimieren. Nach der Kompression hat ein solches Archiv entsprechend die Endung `.tar.gz`[1].

`gtar`, das modernere `tar`

Auf Dauer ist es etwas mühsam, .tar.gz-Archive jeweils zuerst mit `tar` erstellen, um sie dann mit `gzip` zu packen. Das Programm `gtar`[2] ist in der Lage, beides auf einmal zu erledigen. Statt `cvf` müssen dafür aber die Optionen `czvf` übergeben werden. Beispiel:

```
gtar czvf meineProgrammDateien.tar.gz foo.cpp foo.h Makefile
```

Anschliessend befinden sich `foo.cpp`, … in `meineProgrammDateien.tar.gz`. Zum Entpacken des Archivs reicht analog ein

```
gtar xzvf meineProgrammDateien.tar.gz
```

Eine Auflistung aller Dateien eines .tar.gz-Archivs bekommt Ihr mit

```
gtar tzvf meineProgrammDateien.tar.gz
```

Ein alternatives Archivierungsprogramm: `zip`

Ein anderes recht verbreitetes Archivformat ist das zip-Format. Im zip-Format werden alle Dateien sofort komprimiert abgelegt, so dass sich ein nachträgliches `gzip`en erübrigt. Archive werden mit `zip` erstellt und mit `unzip` wieder ausgepackt. Das obige Beispiel würde bei der Verwendung von `zip`

```
zip -r meineProgrammDateien.zip foo.cpp foo.h Makefile
```

lauten. zip-Archive können mit `unzip` ausgepackt werden:

---

1. Da manche Betriebssysteme Probleme mit „geschachtelten" Dateiendungen haben, findet man manchmal auch .tgz statt .tar.gz vor. Dabei handelt es sich dann aber nach wie vor um ein gzip-komprimiertes tar-Archiv.
2. Unter Linux ist `tar` in Wirklichkeit ein `gtar`, so dass dort alles für `gtar` Gesagte auch für `tar` gilt.

```
unzip meineProgrammDateien.zip
```

Eine Liste aller Dateien im zip-Archiv erhält man liefert

```
unzip -l meineProgrammDateien.zip
```

Konvertierung unterschiedlicher Textformate

Beim Austausch von Textdateien zwischen DOS, Windows und UNIX stellen sich immer wieder Probleme ein: So verwendet DOS eine andere Zeichensatzkodierung als UNIX und neuere Windows-Versionen. Dadurch werden insbesondere Umlaute nicht korrekt übertragen. Ausserdem kennzeichnen DOS und Windows Zeilenenden mit den beiden Sonderzeichen `\r\n`, während UNIX nur `\n` verwendet.
Mit den Programmen `dos2unix` und `unix2dos` lassen sich Texte zwischen den verschiedenen Formaten konvertieren. Eine genauere Dokumentation der beiden Kommandos findet Ihr in ihren man-pages.
Schliesslich tritt gerade beim Programmieren oft noch ein weiteres Problem auf: Obwohl laut Standard die Tabulatorweite auf 8 Zeichen festgelegt ist, verwenden die meisten Windows-Editoren (insbesondere die von Entwicklungsumgebungen) 4er-Tabulatoren. Dadurch wird beim Wechsel zwischen den Systemen oft die Einrückung von Programmcode verunstaltet. Der beste Weg, die zu vermeiden ist, Leerzeichen statt Tabs zu verwenden. Die meisten Editoren verfügen über die Einstellungsmöglichkeit, dass beim Druck auf die Tab-Taste statt des Tabulators entsprechend viele Leerzeichen eingefügt werden.