Einleitung

In der Informatik kommt es beim Speichern und Übertragen von Dateien häufig darauf an, die zu speichernden Dateien möglichst klein zu halten. Dazu verwenden Nutzer Programme um die Redundanz (Daten die keine Information enthalten) der enthaltenen Daten zu reduzieren und diese Dateien zu komprimieren (zip, gzip, rar...)

Der Runlength-Algorithmus

Ein recht einfaches jedoch häufig wirkungsvolles Verfahren zur Reduzierung der Dateigröße stellt der sogenannte Runlength-Algorithmus dar. Hierbei wird gezählt, ob Zeichen mehrfach hintereinander vorkommen. Diese werden dann durch ein "Ausscheidungszeichen", einen Zähler und das entsprechende Zeichen ersetzt. Das Ausscheidungszeichen sollte dabei so gewählt werden, dass es möglichst selten im Original-Text vorkommt.

Beispiel 1:

Es werden nur ASCII-Zeichen (im Bereich 0-127/0x0-0x7F) verwendet. Der Zahlenbereich 0x80-0xFF wird für den Zähler verwendet. Sich wiederholende Zeichen werden gespeichert als Zähler und Zeichen. Der Zähler der Wiederholungen muss somit um 0x80 erhöht werden. Fünf Wiederholungen werden gespeichert als 0x80 + 5 = 0x85. Auf diese Weise können maximal 128 Wiederholungen mit einem Zähler gespeichert werden.

Übungsaufgabe:

Wie wird die Datei mit dem folgenden Inhalt komprimiert?

0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x24 0x24

Um die Effektivität zu erhöhen wird erst ab dreimaliger Wiederholung komprimiert. Deswegen kann man festlegen, dass der Zählerstand 1 das 3-malige Wiederholen angibt. Der Zähler 1 bedeutet also 3 Wiederholungen. Der Zähler 128 bedeutet somit 131 Wiederholungen.

Beispiel 2:

Sich wiederholende Zeichen werden gespeichert als 0x90 + Zähler und Zeichen. Zählerstand 1 bedeutet erneut 3 Wiederholungen. Die Zahl 0x90 ist somit beim späteren Expandieren das Ausscheidungskriterium zwischen komprimierten und unkomprimierten Daten.

Sollte das Zeichen 0x90 tatsächlich im Quelltext vorkommen, so wird dies durch das Zeichen 0x90 mit einer folgenden 0 im Zieltext gespeichert. Für die Speicherung der 0x90 benötigt man also in der kodierten Datei ein Byte mehr. Dadurch sind Missverständnisse ausgeschlossen. 0x90 + 0 werden beim Dekodieren also wieder zu 0x90. Alle anderen Kombinationen mit 0x90 stellen Zähler dar.

Übungsaufgabe:

Wie wird die Datei mit dem folgenden Inhalt komprimiert?

0x12 0x12 0x12 0x12 0x12 0x12 0x12 0x90 0x90 0x24 0x24 0x100

1 Aufgabe

Erstellen Sie ein Programm zur Reduzierung der Größe von Dateien welches nach dem Prinzip des sogenannten „Runlength-Encoders“ funktioniert. Das Programm soll zusätzlich Parameter aufnehmen. Ein Aufruf ihres Programms soll sich wie folgt darstellen:

Runlength –c/-u –f <filename.ext> [-e Extension] [-q]

Runlength —compress/--uncompress --file=<filename.ext> [--extension= Extension] [--quiet]

Der erste Parameter dient dazu festzulegen, ob die Datei komprimiert (-c/--compress) oder expandiert (--uncompress) werden soll.

Der zweite Parameter –f gibt den Namen bzw. Speicherpfad der Datei an. Der Name wird für die Zieldatei übernommen, jedoch ohne die extension. Der dritte Parameter (-e/--extension) ist optional. Mit ihm legt man die Namenserweiterung der komprimierten Datei fest. Standardmäßig ist diese "rld". Der Name der Datei bleibt erhalten.

Der letzte optionale Parameter (-q/--quiet) steht für den stillen Modus. In diesem Modus werden keinerlei Bildschirmausgaben erzeugt, weder Hilfetexte noch Meldungen über den Erfolg oder Misserfolg der Umwandlung.

Um die erstellten Dateien mit dem eigenen Datenformat zu kennzeichnen und später wieder erkennen zu können, werden wir

- die Bytes 1-3 der neuen Datei für die eigene Kennung „rl2“ bzw. „rl3“ nutzen
- das Byte 4 nutzen, um die Anzahl der Buchstaben der Extension zu speichern
- die Byte 4-x verwenden, um die Extension der Datei zu speichern (z.B. „bmp“ oder „sqlite“.)

Im Anschluss folgt der Inhalt der Datei.

Beispiel für eine komprimierte .sqlite-Datei:

```
0x72 0x6c 0x64 0x05 0x73 0x71 0x6c 0x69 0x74 0x65    -> Inhalt
 r    l    3          s    q    l    i    t    e     -> Inhalt
```

Die gespeicherte Namenserweiterung wird später zum Herstellen der Originaldatei verwendet.

Ihr Programm soll durch einen entsprechenden Rückgabewert darüber informieren, ob der Programmaufruf erfolgreich war. Dabei gelten folgende Vereinbarungen:

Rückgabewert	**Meldung/Bedeutung**
0	compress/expand successful
1	wrong usage of parameters
2	file error: source file
3	file error: destination file
4	wrong file format
5	other error

Es ist an der Zeit sich mit Bibliotheken/Modulen vertraut zu machen. Sollten Sie c++ verwenden so probieren Sie für des Auslesen der Kommandozeilenparameter mal die Bibliothek CgetOpt (a commandline parameter parsing class) aus. Zu finden diese unter:

http://www.codeproject.com/useritems/CGetOpt.asp#xx1389307xx

Eine andere Möglichkeit wäre die C++ Bibliothek "boost" die neben vielen anderen Funktionalitäten auch mit Kommandozeilenparametern umgehen kann:

Siehe: http://www.boost.org

oder genauer: http://www.boost.org/doc/html/program_options.html

Unter Python steht Ihnen des Modul argparse zur Verfügung.

Um die korrekte Funktion ihres Programms zu überprüfen sind mehrere Tests erforderlich, die sowohl die richtige Nutzung der Parameter als auch das korrekte Komprimieren / Expandieren überprüfen. Ein Minimum an erforderlichen Tests ist in der folgenden Liste dargestellt:

Durchzuführende Tests:

- Falsche Anzahl der Parameter übergeben
- Falsche Parameter übergeben
- Fehlende Parameter übergeben
- Korrekte aber vertauschte Parameter übergeben
- Rückgabewerte und Quite-Modus kontrollieren
- Die Datei zum Lesen lässt sich nicht öffnen
- Die Datei zum Lesen hat das falsche Dateiformat
- Eine größere Datei erfolgreich komprimieren und expandieren
- Eine kleinere Datei erfolgreich komprimieren und expandieren
- Eine Datei komprimieren und expandieren die ein 0x90 Zeichen enthält
- Eine Datei komprimieren und expandieren die mehrere 0x90 Zeichen hintereinander enthält
- Eine Datei komprimieren und expandieren die die zu komprimierenden Zeichen am Dateianfang enthält
- Eine Datei komprimieren und expandieren die die zu komprimierenden Zeichen in der Mitte enthält
- Eine Datei komprimieren und expandieren die die zu komprimierenden Zeichen am Dateiende enthält
- Eine Datei komprimieren und expandieren die mehr als 255 zu komprimierende Zeichen hintereinander enthält

Viel Erfolg!