# 3 Übung 2

## Aufgabe 2.1

Gegeben sei eine Zeichenkette des Typs `string`, die eine natürliche Zahl darstellen soll und daher nur aus Ziffern besteht; Beispiel: "17462309".

1. Wandeln Sie den String in eine Zahl z vom Typ `long` um, wobei keine Bibliotheksfunktionen benutzt werden sollen!
2. Berechnen Sie die Quersumme von z und geben Sie beides auf dem Bildschirm aus.

## Aufgabe 2.2

Das folgende Problem ist klassisch, und es haben sich schon viele Menschen damit beschäftigt:

## Wenn Zahlen Achterbahn fahren

Gegeben sei eine natürliche Zahl > 0.

1. Wenn die Zahl gerade ist, teile sie durch 2. Wenn nicht, multipliziere sie mit 3 und addiere 1.
2. Wenn die sich ergebende Zahl größer als 1 ist, wende Schritt 1. auf diese Zahl an. Wenn nicht, ist das Verfahren beendet.

Es zeigt sich, dass die Zahlen erheblich anwachsen können und auch wieder kleiner werden – daher der Name Achterbahn. Schreiben Sie ein Programm, das eine Startzahl als Eingabe erwartet und den obigen Algorithmus durchführt. Lassen Sie sich die erreichte Zahl und das erreichte Maximum anzeigen. Am Ende des Programms soll ausgegeben werden, wieviele Iterationen bis zum Ende des Programms benötigt werden. Mit der Anweisung

```
cin.get(); // weiter mit Tastendruck
```

können Sie die Ausgabe nach Erreichen eines neuen Höchstwertes anhalten. Falls sich durch die Eingabe der Zahl noch unerwünschte Zeichen im Eingabe-Puffer befinden, können diese mit

```
cin.ignore(1); // Lösche ein Zeichen aus dem Eingabepuffer
```

gelöscht werden. Versuchen Sie die Startzahlen 4096, 142587, 1501353. Bei der ersten Zahl (4096) ist klar, dass der Algorithmus schnell endet, weil 4096 eine Zweierpotenz ist. Die Frage ist letztlich: Gibt es eine Startzahl, mit der der Algorithmus nicht irgendwann endet? Dieses Problem tritt auch unter einer Reihe anderer Namen auf: *Syracuse*-Problem, *Ulams Problem* oder *Collatz*-Problem.

Hinweis: Bei großen Zahlen wie der letzten angegebenen wird der `int`-Zahlenbereich überschritten; nehmen Sie in diesem Fall `long long`.

## Aufgabe 2.3

Schreiben Sie ein Progamm, das den Inhalt einer Datei hexadezimal ausdruckt (Erzeugung eines sog., häufig verwendeten ' *Hex-Dump*'). Neben jeder Zeile mit der hexadezimalen Darstellung von 16 Bytes sollen die 16 entsprechenden Buchstaben dargestellt werden. Wenn der entsprechende ASCIICode nicht darstellbar ist, soll ein '.' ausgegeben werden. Die Ein- und Ausgabe könnte so aussehen:

```
1 Dateiname: Hallo.sec
2 44 61 73 20 68 69 65 72 20 69 73 74 20 65 69 6E   Das hier ist ein
3 20 6B 75 72 7A 65 72 20 54 65 78 74 2E 0A 0A 0A    kurzer Text....
4 55 6E 64 20 6A 65 74 7A 74 20 6B 6F 6D 6D 65 6E   Und jetzt kommen
5 20 65 69 6E 20 70 61 61 72 20 5A 65 69 63 68 65    ein paar Zeiche
6 6E 2C 20 64 69 65 0A 6E 69 63 68 74 20 64 61 72   n, die.nicht dar
7 73 74 65 6C 6C 62 61 72 20 73 69 6E 64 3A 0A 0A   stellbar sind:..
8 F0 E5 B3 AF FE DE AD BE EF F0 00 AF BC D6 D7 E8   ................
```

Hier können Sie die Datei zum Testen herunterladen: Hallo

## Aufgabe 2.4 (*)

Schreiben Sie ein Programm, das eine einzugebende natürliche Zahl in römischer Darstellung ausgibt. Die römischen Ziffern seien in einem konstanten `string ZEICHENVORRAT = "IVXLCDM"` gegeben. Die entsprechenden Wertigkeiten können Sie in einem `vector` speichern. Die syntaktische Regel lautet: Keine Ziffer außer ’*M*’ darf mehr als dreimal hintereinander stehen. Das heißt, ein vierfaches Vorkommen wird durch Subtraktion vom nächsthöheren passenden Wert ersetzt. Subtraktion geschieht durch Voranstellen des kleineren Werts. So wird 4 nicht zu *IIII*, sondern zu *IV*, und 9 wird nicht zu *VIIII*, sondern zu *IX*.

Tipp: Iterieren Sie über den oben angegebenen Zeichenvorrat, und prüfen Sie, wie oft die entsprechende Wertigkeit in die eingegebene Dezimalzahl bzw. den Restwert passt. Beginnen Sie dabei (naturgemäß) mit der größten Wertigkeit.