Vorkurs Programmierung

– Übungsblatt 5 (Tag 8) –

Fachgruppe für Computergraphik und Multimediasysteme
Andreas Kolb, Nicolas Cuntz, Felix Heide

Aufgabe 1 (Tannenbaum)

Es soll ein Programm geschrieben werden, welches Tannenbäume der Höhe n in das Terminal ausgibt. Ein Tannenbaum besteht nur aus Leerzeichen und #-Zeichen. Dabei ist die Vorschrift, nach der sich der Tannenbaum aufbaut, durch folgende Beispiele gegeben:

```
n=3:     n=7:       n=11:

                          #
                         ###
                          #
                        #####
           #             #
          ###          #######
           #             #
         #####        #########
  #       #             #
 ###    #######      ###########
  #       #             #
```

Damit ein vollständiger Tannenbaum ausgegeben wird muss für die Höhe n gelten n = 2*k + 1, wobei k eine natürliche Zahl (ohne Null) sei. Die Zahl k kann als die Anzahl der Baumelemente im Tannenbaum verstanden werden. Ein Baumelement besteht dabei aus zwei Ästen an einem in der Mitte liegenden Stammelement und einem darüberliegenden einzelnen Stammelement.

a) Schreiben Sie zunächst eine Funktion `void drawtree(int n)`, die für eine zulässige Höhe n einen Tannenbaum ausgibt. Testen Sie Ihre Funktion ausgiebig.

b) Das Programm soll nun so erweitert werden, dass das komplette Wachstum eines Baumes bis zum Alter von 14 Jahren simuliert wird. Im Winter wächst ein Tannenbaum nicht. Im Sommer wächst der Baum um ein Baumelement in die Höhe.

   Tipp: Dabei kann ”der Winter” durch einen Aufruf der Funktion `sleep(1)` realisert werden, die das Programm für eine Sekunde anhält. Unter Nutzung der Escape-Sequenz

```
cout << "\033[2J";
```

Aufgabe 2 (Primzahlen) **(optional)**

Schreiben Sie ein Programm, das eine Tabelle mit allen Primzahlen a im Intervall $2 \leq a \leq n$ mit $n \in \mathbb{N}$ erzeugt und anschließend ins Terminal ausgibt.

Tipp: Sie können (müssen aber nicht) den Algorithmus Sieb des Eratosthenes zur Erzeugung der Primzahltabelle nutzen. Die Grundidee dabei ist, dass es zu jeder Zahl m, die keine Primzahl ist, zwei Zahlen i, k mit $2 = i, k = m$ und $i \cdot k = m$ gibt. Damit sieht der Algorithmus wie folgt aus:

1) Schreibe alle Zahlen von 2 bis n in eine Liste.
2) Bilde alle Produkte i k, wobei i und k Zahlen zwischen 2 und n sind.
3) Streiche alle Ergebnisse, die in der Liste vorkommen, aus der Liste.

Aufgabe 3 (Silbentrennung) **(optional)**

Schreiben Sie ein Programm, das nach einfachen Regeln die möglichen Trennstellen eines Wortes, welches über das Terminal eingelesen werden soll, sucht und ausgibt.
Zunächst werden dabei zusammenhängende Folgen von Konsonanten und Vokalen innerhalb des Wortes bestimmt. Vor jedem letzten Konsonanten einer Konsonantenfolge gibt es eine mögliche Trennstelle. Die Folgen ”ch” und ”sch” gelten jeweils als ein Konsonant, so dass eine Trennstelle vor dem letzten Konsonanten einer dieser Gruppen vor die ganze Gruppe gelegt werden muss. Die Folge ”ck” wird durch die Folge ”k-k” ersetzt. Ein einzelner Buchstabe am Anfang oder Ende des Wortes darf nicht abgetrennt werden.
Beispiel:

```
D  o  n  a  u  d  a  m  p  f  s  c  h  i  f  f  Eingabe
k  v  k  v  v  k  v  k  k  k  k  k  k  v  k  k  Folgen von Vokalen und Konsonanten
-     -        -                    -        -  Mögliche Trennstellen
-     -        -              -              -  "sch" gilt als ein Konsonant
      -        -              -                 Einzelne Buchstaben nicht abtrennen
```

Ergebnis: Do-nau-dampf-schiff

Hilfe: Eine sehr ausführliche Übersicht mit Beispielen zu der Klasse `string` findet sich unter `http://www.cplusplus.com/reference/string/string/`.

Glossar:

http://www.cplusplus.com/reference/string/string/
Ausführliche Information zur string-Klasse