Dirk Praetorius,
Gregor Gantner

Sommersemester 2018
10.04.2018

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 4

Aufgabe 4.1. According to an old legend, there was a temple in Hanoi, which contained a large room with three time-worn posts in it surrounded by 64 golden disks of different diameters. When the temple was erected, the disks were arranged in a neat stack in ascending order of size on the first rod, the smallest at the top, thus making a conical shape. Since that time, the temple priests have been moving the disks with the objective of moving the entire stack to the third rod (preserving the ascending order). The second rode is auxiliary. All disk movements must be in accordance with the following immutable rules:

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack, i.e., a disk can only be moved if it is the uppermost disk on a stack.
- No disk may be placed on top of a smaller disk.

According to the legend, when the last move will be completed, the world will end.
The task can be accomplished with the use of a recursive algorithm. Let n denote the total number of disks ($n = 64$ in the original legend). In order to move the upper $m \leq n$ disks located on the i-th rode to the j-th rod ($i, j \in \{1, 2, 3\}$), proceed as follows:

1. Move the upper $m-1$ disks from the i-th rod to the k-th rod, with $k \notin \{i, j\}$;
2. The largest of the m disks is now on the top of the i-th rod and can be moved to the j-th rod;
3. Finally, the $m-1$ disks from in Step 1 can be moved from the k-th rod to the j-th rod.

The choice of $m = n$, $i = 1$ and $j = 3$ in the above algorithm solves the priest task. Please write a recursive function `void hanoi(int m, int i, int j)` which implements the algorithm. Any single movement of any disk must be printed out on the screen, e.g.,

```
Move a disk from Rode 2 to Rode 3.
```

Furthermore, write a main program that reads n from the keyboard and prints out the list of all movements. To test the algorithm, use $n \ll 64$, e.g., $n = 3, 4, 5$. Save your source code as `hanoi.c` into the directory `serie04`.

Aufgabe 4.2. Write a recursive function `papercut`, which visualizes all the possibilities for cutting a sheet of paper of integer length n into thin strips of length 1 and 2. Specifically, given a natural number n, the function should determine all the possible representations of n of the form $n = \sum_{j=1}^{k} \sigma_j$ with $\sigma_j \in \{1, 2\}$. Here, the ordering of the sequence is important. For istance, for $n = 4$, there are 5 possibilities, namely

- $4 = 2 + 2$
- $4 = 2 + 1 + 1$
- $4 = 1 + 2 + 1$
- $4 = 1 + 1 + 2$
- $4 = 1 + 1 + 1 + 1$

Moreover, write a `main` program, which reads n from the keyboard and prints to the screen all the possibilities. Save your source code as `papercut.c` into the directory `serie04`.

Aufgabe 4.3. Implement the function `int = binomial(int n, int k, int type)` which computes and returns the binomial coefficient $\binom{n}{k}$ using three different approaches:

- applying the formula $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, which exploits a function for computing the factorial (`type=1`),
- using the expression $\binom{n}{k} = \frac{n\cdot(n-1)\cdots(n-k+1)}{k\cdot(k-1)\cdots 1}$ driven by a suitable loop (`type=2`),
- in recursive form, exploiting the formula $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ (`type=3`).

The function should support all three options. Furthermore, write a main program, which reads n and k from the keyboard and prints out the resulting binomial coefficient. Save your source code as `binomial.c` into the directory `serie04`.

Aufgabe 4.4. A natural number $a \in \mathbb{N}$ is called Armstrong number if and only if the sum of its digits to the power of the number of digits coincides with a itself, i.e., $a := 10^{n-1}\cdot a_n + 10^{n-2}\cdot a_{n-1} + \ldots + 10^0 \cdot a_1$ with $n \in \mathbb{N}$, $a_i \in \{0, 1, \ldots, 9\}$, $0 < i \leq n$, $a_n \neq 0$, is an Armstrong number, if $a_1^n + a_1^n + \ldots + a_n^n = a$. For example, the first Armstrong numbers are $1, 2, \ldots, 8, 9, 153, 370, \ldots$ Write a `void` function `armstrong`, which, given a lower bound $b \in \mathbb{N}$ and an upper bound $c \in \mathbb{N}$ with $b \leq c$, computes and returns all Armstrong numbers within these bounds. Use `assert` to make sure that $b \leq c$. Write a main program which reads the bounds from the keyboard and calls the function `armstrong`. How did you test the correctness of your code? Save your source code as `armstrong.c` into the directory `serie04`.

Aufgabe 4.5. Write a main program which reads $n \in \mathbb{N}$ from the keyboard and prints to the screen the first n lines of Pascal's triangle: Every line starts and ends with 1. The remaining entries are the sum of two neighbouring entries from the line above. For $n = 5$, we obtain:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

See also:

`http://en.wikipedia.org/wiki/Pascal's_triangle`

Save your source code as `pascal.c` into the directory `serie04`.

Aufgabe 4.6. Write a function `energy` that returns the energy $e = \sum_{j=1}^{n} x_j^2$ of a given vector $x \in \mathbb{R}^n$. Further, write a main program which reads in the vector x and calls the function The dimension $n \in \mathbb{N}$ is a constant in the main program, the function `energy` should be implemented for arbitrary dimensions. How did you test the correctness of your code? Save your source code as `energy.c` into the directory `serie04`.

Aufgabe 4.7. The Frobenius-norm of a matrix $A \in \mathbb{R}^{m\times n}$ is defined by

$$\|A\|_F := \Big(\sum_{j=1}^{m}\sum_{k=1}^{n} A_{jk}^2\Big)^{1/2}.$$

Write a function `frobeniusnorm` which, given a matrix A as well as its dimensions $m, n \in \mathbb{N}$, computes and returns the Frobenius norm. Furthermore, write a main program that reads in the input parameters (matrix A and dimensions m, n), and prints out the corresponding Frobenius norm $\|A\|_F$. The matrix should be stored columnwise as a vector of length mn. The dimensions $m, n \in \mathbb{N}$ should be constant in the main program, but the function `frobeniusnorm` should be programmed for arbitrary dimensions. Save your source code as `frobeniusnorm.c` into the directory `serie04`.

Aufgabe 4.8. Let x be a sequence of 10 numbers (static array of type `int`) and y a combination of 3 numbers (array of type `int`) which are both read from the keyboard. Write a function `check` that obtains the two arrays as input and checks if the combination y is contained in the sequence x (return value 1) or not (return value 0). Further, write a main program which reads in the arrays x and y and calls the function. How did you test the correctness of your code? Save your source code as `check.c` into the directory `serie04`.