

Informatik für Mathematiker und Physiker HS14

Exercise Sheet 6

Submission deadline: 15:15 - Tuesday 28th October, 2014

Course URL: <http://lec.inf.ethz.ch/ifmp/2014/>

Assignment 1 (3 points)

a) Parenthesize the following expressions according to the rules of associativity and precedence.

- (i) $5 / 4 / 2.0 / 2 < 0.3125$
- (ii) $f > 3.5f \ || \ 5 / 4 / f == 1.25f \ \&\& \ ++f < 1$
- (iii) $3 / 2 > 1.0 \ \&\& \ d != 3 / 4$

b) Evaluate the expressions from part a) step by step. You may assume that the variable `f` is of type `float` and has initial value `0.0f`. And `d` is of type `double` and has initial value `0.75`.

Assignment 2 - Skript-Aufgabe 85 (4 points)

a) Write a function

```
// POST: return value is true if and only if n is prime
bool is_prime (unsigned int n);
```

and use this function in a program to count the number of *twin primes* in the range $\{2, \dots, 10000000\}$ (two up to ten millions). A twin prime is a pair of numbers $(i, i + 2)$ both of which are prime.

- b) Try to find an argument why your program is so slow.
- c) Can you think of a better (more efficient) approach than the one used in a)? (You don't have to implement it.)

Hint for a): The function `double sqrt (double x);` from the library `<cmath>` computes the square root of `x`. In order to use this function, you have to

- include the `<cmath>` library (add the line `#include <cmath>` below the `#include <iostream>` command in your code)
- call the function with the command `std::sqrt(x)`

Assignment 3 (4 points)

A perfect number is an integer which is equal to the sum of all of its proper divisors (a divisor not equal to the number itself). For example

$$6 = 1 + 2 + 3$$

is a perfect number. Write a program `perfect_numbers_2.cpp` which reads a given integer n from the user and outputs each perfect number between 1 and n . Write your program in a way such that it uses functions in a way that makes the program easy to read and easy to understand. Then argue for every function why you chose to use it.

Hint: In the exercise classes you saw the program `perfect_numbers.cpp` which consists of just the main-function. Where in that program could it make sense to use functions in order to improve its readability?

Assignment 4 (4 points)

The goal of this exercise is to write a function `cube_root` which approximates for a given number x its third root $x^{1/3}$. To solve this task we can apply the so-called Newton's method. This method gives us a sequence of points $(y_n)_{n \in \mathbb{N}}$ which converges to the third root:

$$y_n \xrightarrow{n \rightarrow \infty} x^{1/3}$$

The (y_n) can be computed as

$$\begin{aligned} y_{n+1} &= (2y_n + x/y_n^2) / 3 \\ y_0 &= x \end{aligned}$$

Notice that this approximation cannot be computed for $x = 0$. For $x \geq 1$ the sequence (y_n) is monotonously decreasing. We can thus let the approximation run as long as $y_n < y_{n-1}$. (This condition will always be false after a finite number of steps due to the finite precision of `double` and `float`.)

Implement the function `cube_root` for $x \geq 1$ by applying the above procedure. Then use your function in a program which reads a number $x \geq 1$ from the user and then outputs the results to the following two comparisons:

- absolute difference between `cube_root(x)` and `std::pow(x, 1.0/3)`
- absolute difference between x and the third power of `cube_root(x)`