Institut für Informatik — TUM
Scientific **C**omputing in **C**omputer **S**cience
R. Wittmann

WT 2015

Tutorial
(Advanced Programming)
Worksheet 4:

Assignment 1: Code analysis (exam training)

Consider the following code from a nerd programmer:

```
#include <iostream>
int main(int argc, char *argv[]) {
    int xvalues[10];
    for (char i=0; i < 128; i++) {
        std::cin >> xvalues[static_cast<int>(i)];
    }
    double a = static_cast<double>(1/3);
    double b = 5;
    double c = 10;

    for (int i=0; i < 128; i++) {
        int x = xvalues[i];
        double result = a * ((x+1)^2) + b * x + c;
        std::cout << "x=" << xvalues[i]
                  << ",result=" << result
                  << std::endl;
    }
    return 0;
}
```

What was he trying to accomplish and what did he do wrong? Why does it even compile? Explain how the compiler actually translates and evaluates the code. Fix the program and improve the code.

Assignment 2: Static lookup table using function pointers

In scientific applications it is not always reasonable to fit complex datasets with a single function either due to numerical issues or due to expensive computations. Instead, one could split up the dataset in different parts which are then approximated by simple functions. However, this approach requires a way to reconstruct the original function using these smaller pieces. One possible solution to this is to define a lookup table which maps an interval to the respective function. Please download the the stub implementation in `lookuptable.cpp` from moodle.

- Consider the implementation of the `LookupTable_entry` data structure. Explain the structure with respect to syntax and semantics. Also, think about what advantages such a statically allocated data structure could have.
- Implement the missing functionality in `query_lookuptable` which maps a query point to a suitable interval. If there is no suitable interval, then return `0.0`.
- Explain the expression in line 61.

Homework Assignment 3: Accurate and efficient computations

Consider the following polynomial of order n with $n+1$ coefficients:

$$p(x) = \sum_{i=0}^{n} a_i \cdot x^i \tag{1}$$

We can also reorder the computation of the polynomial by factoring of x in the following way, which is called "Horner's method":

$$p(x) = a_0 + x \cdot (a_1 + x \cdot (a_2 + \cdots + x \cdot (a_{n-1} + a_n \cdot x) \cdots)) \tag{2}$$

Implement both methods and evaluate the polynomial at point $x = 2.5$ for the following coefficients:

$$a_i = \frac{1}{i+1} \tag{3}$$

$$a_i = \frac{i}{n+1} \tag{4}$$

Compute the differences between both methods for different orders up to 30 and compare the results with respect to floating point accuracy. Which method is more efficient in terms of arithmetic operations?