Exercise Sheet #6

Bulcsú Sándor <sandor@th.physik.uni-frankfurt.de>
Hendrik Wernecke <wernecke@th.physik.uni-frankfurt.de>
Laura Martin <lmartin@th.physik.uni-frankfurt.de>
Christopher Czaban <czaban@th.physik.uni-frankfurt.de>

Problem 1 (*Template functions*) 7 Pts

For the following tasks, implement the requested functions and call them in the `main` function.

(a) Implement a function with a template type for the arguments and the return value, which takes two numbers of the same template type (e. g. `int`, `double`) and returns the product, which is of the same type. (2 Pts)

(b) Extend the function so that it additionally has a templated integer parameter and takes an array of template type `T` and length `N` as argument. The function shall print all entries of the array and the size of the array. **Hint:** The function then looks like
`template<typename T, int N> func_name (T (&array)[N]) {...}`. (2 Pts)

(c) Re-write the function such that you can pass a two dimensional $n \times m$ array. (3 Pts)

Note: In the case of a qudratic 2d array the function becomes a bit simpler (but thus more confusing) and can take the form
`template<typename T, int N> func_name (T array[N][N]) {...}`.

Problem 2 (*Function pointer and header files*) 6 Pts

(a) For many purposes, such as integration methods, one has functions that take a function as an argument. Write a function named `eval` that evaluates a given function `f` at `N` points on the interval `[a,b]`, where `f, N, a, b` are given as arguments. More specific, pass `f` as a function pointer. Test your code with any polynomial or trigonometric function and plot it in gnuplot. (4 Pts)

(b) For more complex project and especially when developing code in a group, it can be useful to create a multi-file structure. Then there is exactly one executable file containing the `main` function and all functions, classes, structs etc. are loaded from external `C++` files and header files. E. g. a function is defined in a separate `C++` file (e. g. `test.cpp`)

and only the function body, i. e. with name, argument types, return type, is defined in a header file of same name (`test.h`). The function in the external file can be loaded by including the corresponding header file (`#include "test.h"`). Re-write the code from (a) into such a multi-file project by creating a external `C++` file and a corresponding header file for the function `eval`. (2 Pts)

Problem 3 (*Recursive functions*) 7 Pts

(a) Write a programm that calculates the factorial of a given number recursively, i. e. by involving function that calls itself. (3 Pts)

(b) Now write the same code by recursively calling the `main` function. (4 Pts)
Disclaimer: Mind that recursively calling the `main` function is declared illegal in the `C++11` documentation §3.6.1/3. We request this here for academic purpose only.

Problem 4 (*Final confusion – facultative*) 0 Pts

`C++` offers a variety of different ways to do one and the same thing. For instance refering to an array via pointer can be done by creating a pointer that points to an array, but also without a separate pointer. This exercise should motivate you to face the many faces of `C++` and getting rid of some confusion. Therefore create an array `arr` and a pointer `ptr` that points to the array.

(a) What is the difference between the following two expressions
`int* ptr = arr` and `int* ptr = &arr`?

(b) Print out the pointer `ptr`, the address of the pointer `&ptr` and the value that the address of the pointer points to `*&ptr` .

(c) Repeat this for the array `arr`. What do you expect?

(d) Iterate the pointer `ptr++` and try to do this with the array.

(e) Using reference types one can refer to a variable,
e. g. `int k; int& ref = k`.
What does the following expression: `int* const& ref = arr` ?