Comparison of Template Metaprogramming and Functional Programming

By José Mendes (xmende04) and Ricardo Cebrian (xcebri00)

Metaprogramming in the context of C++ is used to preform computation at compile-time instead of runtime. Templates create temporary source code which is then used to compile the complete program. While having a strange syntax compared to other languages, it is still Turing complete.

Metaprogramming can be used with function overloading in order to create generic programs that work on most clients as the implementation is mostly handled by the compiler. This reduces the amount of code needed to be written and helps maintainability. This language was not totally planned by the developers, therefore useful things like error messages and step debugging can't be done easily in the standard compiler.

Functional Programming computes problems as expressions with arguments and a result. This is different from other programming paradigms as it is devoid of outside factors, such as machine state, we always get the same result for the same arguments.

Metaprogramming is closely tied with functional programming as it can be a tool to achieve it, as it creates the implementation by itself and removes client specific details.

In the presentation we'll give a quick overview of C++ template metaprogramming, its history, some examples, a quick overview of functional programming and its relation to the template metaprogramming.