01 - Die Welt der Programmiersprachen

The lecture about the world of programming languages hold by Prof. Schwill began by illustrating the importance of programming languages in general. They are not only a simple tool to accomplish certain tasks, but also a medium for fundamental ideas with influence on our thinking. It was even suggested that Wittgenstein's ideas behind his famous quote "The limits of my language mean the limits of my world." and the Sapir-Whorf hypothesis are applicable to programming languages in contrast to the usual application to ordinary languages. This means, when creating an algorithm to solve a particular problem, programmers could be heavily influenced by the language constructs available in the languages they know best. To avoid being trapped in some kind of language cage it is best to know at least about the concepts offered by other programming languages.

That's why the lecture continued with the two major programming paradigms: imperative and declarative programming. The former makes the algorithm explicit and leaves the goal implicit, while the latter makes the goal explicit and leaves the algorithm implicit. That means, in imperative programming you specify in detail how your problem is to be solved by the computer, while in declarative you just describe how your problem looks like and let the computer find a way to solve it. Almost every imperative language is also procedural, so the source code is split up into several procedures or subroutines which make it possible to use code more than just once in the same program or even in different programs.

The declarative languages can be separated into functional and logical ones. In functional languages computation is nothing more than the evaluation of mathematical functions and new values are always produced from existing ones unlike from modification of states as in imperative programming. Developed in the 1930s by Church, the lambda calculus founded the basis of functional programming and languages like LISP, ML and Miranda. It was mathematical logic especially the predicate calculus which did the same thing for the other half of declarative languages: logic programming. A logic program does not consist of commands as imperative programs, but of axioms or facts that just describe and don't compute. The actual computation takes place when a query was made to the program. Examples for logic languages are Prolog and Gödel. Although declarative languages are of great use in science and areas like artificial intelligence and expert systems, the most popular and widespread language paradigm is the imperative one. Starting in 1940 with Assembler, imperative programming made its way with languages like Algol60, Simula67, Pascal and C. In recent years it was enhanced by concepts like object and aspect orientated programming that increased its usability and popularity even more.