Preface

We've known about algorithms for millennia, but we've only been writing computer programs for a few decades. A big difference between the Euclidean or Eratosthenes age and ours is that since the middle of the twentieth century, we express the algorithms we conceive using formal languages: programming languages.

Computer scientists are not the only ones who use formal languages. Optometrists, for example, prescribe eyeglasses using very technical expressions, such as "OD: -1.25 (-0.50) 180° OS: -1.00 (-0.25) 180°", in which the parentheses are essential. Many such formal languages have been created throughout history: musical notation, algebraic notation, etc. In particular, such languages have long been used to control machines, such as looms and cathedral chimes.

However, until the appearance of programming languages, those languages were only of limited importance: they were restricted to specialised fields with only a few specialists and written texts of those languages remained relatively scarce. This situation has changed with the appearance of programming languages, which have a wider range of applications than the prescription of eyeglasses or the control of a loom, are used by large communities, and have allowed the creation of programs of many hundreds of thousands of lines.

The appearance of programming languages has allowed the creation of artificial objects, programs, of a complexity incomparable to anything that has come before, such as steam engines or radios. These programs have, in return, allowed the creation of other complex objects, such as integrated circuits made of millions of transistors, or mathematical proofs that are hundreds of thousands of pages long. It is very surprising that we have succeeded in writing such complex programs in languages comprising such a small number of constructs — assignment, loops, etc. — that is to say in languages barely more sophisticated than the language of prescription eyeglasses.

Programs written in these programming languages have the novelty of not only being understandable by humans, which brings them closer to the scores used by organists, but also readable by machines, which brings them closer to the punch cards used in Barbarie organs.

The appearance of programming languages has therefore profoundly impacted our relationship with language, complexity, and machines.

This book is an introduction to the principles of programming languages. It uses the Java language for support. It is intended for students who already have some experience with computer programming. It is assumed that they have learned some programming empirically, in a single programming language, other than Java.

The first objective of this book will then be to learn the fundamentals of the Java programming language. However, knowing a single programming language is not sufficient to be a good programmer. For this, you must not only know several languages, but be able to easily learn new ones. This requires that you understand universal concepts like functions or cells, which exist in one form or another in all programming languages. This can only be done by comparing two or more languages. In this book, two comparison languages have been chosen: Caml and C. Therefore, the goal is not for the students to learn three programming languages simultaneously, but that with the comparison with Caml and C, they can learn the principles around which programming languages are created. This understanding will allow them to develop, if they wish, a real competence in Caml or in C, or in any other programming language.

Another objective of this book is for the students to begin acquiring the tools which permit them to precisely define the meaning of the program. This precision is, indeed, the only means to clearly understand what happens when a program is executed, and to reason in situations where complexity defies intuition. The idea is to describe the meaning of a statement by a function operating on a set of states. However, our expectations of this objective remain modest: students wishing to pursue this goal will have to do so elsewhere.

The final objective of this course is to learn basic algorithms for lists and trees. Here too, our expectations remain modest: students wishing to pursue this will also have to look elsewhere.

http://www.springer.com/978-1-84882-031-9

Principles of Programming Languages

Dowek, G.

2009, XII, 159 p., Softcover
ISBN: 978-1-84882-031-9