Solutions to Exercises 1

* **1.2.2** The longevity of FORTRAN and COBOL is explained mainly by a legacy of software, coded in these languages from the 1960s onwards, and still regarded as business-critical by its owners. The enormous volume of this legacy software, and the problems of maintaining it, were vividly exposed by the "millennium bug" scare of the late 1990s. Nevertheless, the owners have individually decided, rightly or wrongly, that it is more cost-effective to maintain their legacy software rather than rewrite it using modern programming languages (and modern software engineering methods).

FORTRAN and COBOL have not stood still. Both have been extended to support more advanced concepts. For example, FORTRAN supports parallel processing, and COBOL even supports object-oriented programming.

* **1.2.3** Object-oriented programming has become the dominant paradigm for several reasons. Objects are a very natural way to model real-world objects with state. Novice programmers seem to take to object-oriented programming more comfortably than imperative programming. More experienced programmers find that classes are excellent building blocks for large programs. Classes are also highly reusable, and object-oriented languages invariably have much richer libraries than other languages.

Functional and logic programming are very well suited for certain kinds of applications, but not for general-purpose programming. In particular, pure functional and logic languages cannot naturally model real-world objects with state (and an impure functional or logic language that can model state tends to be an uneasy compromise).