

every day. So the team proceeded to build and use a machine, Colossus, that could break the codes hundreds of times faster than mere humans.

The first general-purpose digital computers were built in the USA and Great Britain. You are probably familiar with the story of computers since then. Computers are now ubiquitous, whether as general-purpose computers or as components embedded in devices as diverse as digital watches, kitchen appliances, and aircraft. And every one of these computers is a machine designed to perform algorithms.

We are also surrounded by algorithms intended for performance by humans. Whether or not a device has a computer inside it, it probably comes with a user's manual. Such manuals contain algorithms necessary for us to use the devices effectively: to operate a washing machine, to set a digital watch to give an alarm signal or to record lap times, to change a car wheel, to change a light bulb, to assemble a piece of furniture from a flat-pack, and so on.

## 1.2 Algorithms and programs

So what exactly *is* an algorithm? We shall attempt a precise definition in Chapter 2. For the time being, we can think of an algorithm as a step-by-step procedure for solving a stated problem. An algorithm may be intended to be performed by a human or a machine. In either case, the algorithm must be broken down into steps that are simple enough to be performed by the human or machine.

Algorithms have many things in common with programs, but there are also important differences between them.

Firstly, algorithms are *more general* than programs. An algorithm may be suitable to be performed by a human, or by a machine, or by both. A program *must* be capable of being performed by a (suitable) machine. In this book, the machine will always be a general-purpose computer (as opposed to a robot or weaving machine, for example).

Secondly, algorithms are *more abstract* than programs. An algorithm can be expressed in any convenient language or notation. A program must be expressed in some programming language. If an algorithm is intended to be performed by a computer, it must first be coded in a programming language. There may be many ways of coding the algorithm and there is a wide choice of programming languages. Nevertheless, all the resulting programs are implementations of the same underlying algorithm. For example, Programs 1.7 and 1.8 are both implementations of Euclid's algorithm, coded in two different programming languages: compare them with Algorithm 1.3.

In this book we will express algorithms in English, and code them as methods in Java. The very same algorithms could be (and undoubtedly have been) equally well expressed in French or Chinese, and coded in C or Pascal or Ada. The point is that we can study these algorithms without paying too much attention to the language in which they are expressed.

Often there exist several different algorithms that solve the same problem. We are naturally interested in comparing such alternative algorithms. Which is fastest? Which needs least memory? Fortunately, we can answer such questions in terms of the qualities of the algorithms themselves, without being distracted by the languages in which they happen to be expressed. We shall study algorithm efficiency in Chapter 2.