Syntax of Algorithms

In this chapter we describe the language we use in this book to present algorithms. The language we have chosen does not represent any particular programming language, because we do not want to delve too deeply into the technicalities involved in preparing a concrete computer program. The language we use is a combination of features found in present-day programming languages which we feel are most useful to keep the presentation reasonably detailed but simple. A person with some experience in programming should have no problem in rewriting our algorithms in a concrete programming language.

14.1 Overview

The purpose of this section is to acquaint the reader with the essential elements used to present the algorithms in this book. As an aid to the exposition we begin with the following trivial example, which will be referred to in the discussion that follows.

Algorithm 14.1 Average

function average( int m, n)
    return (m + n)/2;

Algorithms accept some data (input), manipulate the data, and provide some results (output). The input, intermediate data, and output are stored in variables, which can be thought of as containers. Each variable has a name by which the algorithm can assign data to it and later read it for further processing. Every algorithm has a name as well, so that it can be used in other algorithms.

In Algorithm 14.1 the input are two numbers that are assigned to the variables named m and n. The data are manipulated in the step (m + n)/2, and
the value of this operation represents the output. The name of this algorithm is \textit{average}.

To conveniently store and access the data needed in a particular algorithm, the data are organized into \textit{data structures}, which in turn are divided into \textit{data types}.

In a typical algorithm not all the data come from the input, but rather some are encoded directly in the form of \textit{literals}. In Algorithm 14.1 the number 2 is a literal. A more detailed discussion of data structures and literals is provided in Section 14.2.

One can declare a variable to be of a specific data type. This means that an error will occur when data of some other data type are assigned. For instance, the declaration

\begin{verbatim}
int m;
\end{verbatim}

forces an error every time noninteger data are assigned to m. Data-type declarations as well as some other language constructs require certain reserved words, called \textit{keywords}. To indicate them clearly, we write them in boldface. Also, observe from the first line of Algorithm 14.1 that we allow ourselves the luxury of simultaneously declaring a few variables of the same data type.

Note that we do not require declaration of variables unless it is not clear from the context what their proper data type should be. This helps to keep the algorithms reasonably short. However, one should be aware that in concrete programming languages declaration of variables simplifies memory management and facilitates early detection of possible errors.

Every algorithm (including Algorithm 14.1) presented in this book has the form

\begin{verbatim}
function name(list of arguments)
body
\end{verbatim}

Other algorithms may call a particular algorithm by using its name and supplying the required input data. The comma-separated list of arguments assigns names to the data required by the algorithm. In this list we declare the data types to which the arguments must belong.

The body of the algorithm consists of semicolon-separated statements. The statements may be simple or compound. A simple statement is the evaluation of an expression. The expression may contain variables, literals, operators, parentheses, and calls to other algorithms. Operators may be binary (for instance, the addition operator + or the inequality operator <) and unary (for instance, the negation operator \textbf{not}). The outcome of the expression is usually assigned to a variable by means of the assignment operator, as in the example

\begin{verbatim}
v := \textbf{not} (x + 1 < z);
\end{verbatim}

In the evaluation of expressions we assume the order that follows the standard mathematical conventions. In particular, the expressions in parentheses are evaluated first.