

```

static int gcd (int m, int n) {
  // Return the GCD of positive integers m and n.
  int p = m, q = n;
  while (p % q != 0) {
    int r = p % q;
    p = q; q = r;
  }
  return q;
}

```

Program 1.7 Implementation of Euclid's algorithm as a Java method.

```

function gcd (m, n: Positive) return Positive is
-- Return the GCD of positive integers m and n.
  p, q, r: Natural;
begin
  p := m; q := n;
  while p mod q /= 0 loop
    r := p mod q;
    p := q; q := r;
  end loop;
  return q;
end gcd;

```

Program 1.8 Implementation of Euclid's algorithm as an Ada function.

1.3 Abstract data types and data structures

Complementary to the study of algorithms, and also a major theme of this book, is the study of data structures and abstract data types.

A data structure is a systematic way of organizing a collection of data. A familiar example of a data structure is the array, which you should have met in your first programming course. The array is sometimes called a *static data structure*, because its data capacity is fixed at the time it is created. This book will introduce the more flexible *dynamic data structures*, which are so called because they can freely expand and contract in accordance with the actual amount of data to be stored.

Every data structure needs a variety of algorithms for processing the data contained in it: algorithms for insertion, deletion, searching, and so on. In this book we shall study data structures and algorithms in conjunction.

Often there exist several data structures suitable for representing the same collection of data. Figure 1.9 illustrates this point by showing a character string represented by two different data structures. Figure 1.10 illustrates the point further by showing a set of words represented by three different data structures. We shall explore all these data structures in later chapters.