refinement, down to the fine details of the program needed to start coding into the chosen high level language. We shall also try to adhere to the principles of so-called 'structured programming'. These principles will become clear during the course of developing this project.

The steps in the development of the program can be summarised like this:

1. A clear statement of the problem
2. The form of the input and output (first level description)
2.1 Refinement (second level description)
2.2 Further refinement (third to nth level description)
3. Coding into high level language

Before embarking on a major software project, it is essential to state the problem clearly. This is a far from trivial exercise. Let's try a few ideas for our computerised address book.

First we'll start with a list of desirable features — later we can decide which of these can be implemented with a reasonable amount of programming effort. We want to be able to:

1. Look up an address, telephone number and notes by entering a name at the keyboard
2. Get a list of names, addresses and telephone numbers by entering only part of a name (perhaps just the surname or first name)
3. Get a list of names, addresses and telephone numbers for a particular town or area
4. Get a listing of all names starting with a particular initial
5. Get a full listing of all names in the address book, sorted alphabetically
6. Add new entries at will
7. Change entries at will
8. Delete entries at will

Assume that the address book program has been written. What form should the input and output take? How would you like the program to work from the user's point of view? Broadly speaking, programs can be 'menu-driven', 'command-driven' or a combination of both. In a menu-driven program, at every point where a decision has to be taken the user is prompted with a list (menu) of options. The selection can usually be made by pressing just one key. With command-driven programs the user types in specific command words or phrases, usually without prompting. Some programs combine both techniques. The advantage of a menu-driven program is that it should be easy for a newcomer to use, making the program more 'user friendly'. A command-driven program should be faster for an experienced user to use. We will opt for a menu-driven approach, though you might decide to implement this program using command routines instead — the choice is yours.

Given that the program will be centred on a list of names, the first thing we must consider is what form those names should take. Will the computer understand all of the following formats, for example?

A. J. P. Taylor
Leonardo da Vinci
Glenda JACKSON
Liz T.
P. O'Toole
e. e. cummings
P Jackson
Twiggy
GROUCHO MARX
Sir Freddie Laker

This may seem like splitting hairs, but consider what would happen if you had entered `P Jackson` and then you asked the program to search for `P. Jackson`. Unless you had anticipated the problem, the computer would probably respond with `NAME NOT FOUND`.

There are two ways the problem could be tackled: we can either have 'fuzzy' input, allowing names to be input in any form, together with clever routines to allow for this when searches are made; or we can insist on names being input in a strictly defined form. Any name that did not conform to the convention would result in an error message such as `NAME FORMAT UNACCEPTABLE`. The choice is an arbitrary one, but we shall opt for very 'fuzzy' input and let the program worry about converting names to a standard form.

From the point of view of an alphabetic search, names can be thought of as having two parts — surnames and the rest. A surname is relatively easy to define: any string of upper or lower case alphabetic characters terminated by a Carriage Return and preceded by a space (ASCII 32). A problem immediately presents itself: what would happen if the name 'Twiggy' were entered without a space at the front? Presumably the program would reject it as an unacceptable format. We'd better change our definition.

A name comprises one or more parts: a surname or a surname and forename. The name may consist of either upper or lower case characters, full stops, apostrophes and hyphens. It will always start with an alphabetic character and be terminated with a Carriage Return (terminal full stops will not be allowed). If there is a space, the last group of characters — including apostrophes and hyphens — will be counted as a surname and other parts, including the space, will be counted as the forename. If there is no space, the whole name will be counted as the surname.

The surname needs special consideration because in any alphabetic search, it always takes precedence over forenames. Thus Albert Peterson would come after Zoltan Patel. If a name consists of only one group of characters, such as Trevanian, Twiggy or a nickname like Baldy, it can be considered as a surname for the purposes of our program.

In an alphabetic search, which name would come first — A. J. P. Taylor or Alfred Taylor? The decision is arbitrary, but the simplest solution would be to ignore the full stops and spaces before