the final space and to make the names equivalent to AJP TAYLOR and ALFRED TAYLOR. If we were to do this, both AJP and ALFRED could be considered as forenames, and so AJP would come first.

Part of our program would accept as an input a name and produce as an output a name, address and telephone number (note that we have not even begun to consider the meanings of 'address' and 'telephone number'). If we were to accept names with a 'fuzzy' format as input, with internal conversion to a standardised format, would we expect the output to be in the 'standardised' form, or in the same form as the original entry? The most 'user friendly' output would be for the name to be in the original form, but, as we shall see, this will complicate the programming.

As an initial programming task, let's suppose that a name has been assigned to the string variable NAME$ and that we have two other variables, FORENAME$ and SURNAME$. How will we assign the appropriate parts of NAME$ to FORENAME$ and SURNAME$? Ignoring, for the moment, the problem of keeping a record of the original form in which the name was entered (so that it can be retrieved when needed later), a simple statement of the program could be:

```
Convert all characters to upper case
Eliminate all non-alphabetic characters except
the final space
Assign all characters following a final space to
SURNAME$
Assign all characters preceding a final space to
FORENAME$
```

Before considering how this problem could be coded into BASIC, we'll see how the process of 'top down programming' can take us from a very broad statement of our objective to the point where coding into a particular programming language becomes possible. You will notice that we are using not only long variable names like SURNAME$, but command words like BEGIN, LOOP and ENDLOOP. These are constructions that we have invented to help us describe our program. At the final stage of development, they will be replaced with equivalent commands from BASIC. We'll explain more about these commands, and why we have indented some of the lines in the next instalment of the course.

1ST STATEMENT OF OBJECTIVES

```
INPUT
A name (in any format)
OUTPUT
1. A forename
2. A surname
```

1ST REFINEMENT

```
1. Read NAME$
2. Convert all letters to upper case
3. Find last space
4. Read SURNAME$
5. Read FORENAME$
6. Discard non-alphabetics from FORENAME$
```

2ND REFINEMENT

```
1. Read NAME$
2. (Convert all letters to upper case)
   BEGIN
   LOOP while unscanned characters remain in NAME$
     Read out characters from NAME$ in turn
     IF character is lower case
       THEN convert to upper case
       ELSE do nothing
     ENDIF
     Assign character to temporary string variable
   ENDLOOP
   LET NAME$ = temporary string variable
   END
3. (Find last space)
   BEGIN
   LOOP while unscanned characters remain in NAME$
     IF Character = "   "
       THEN note position in a variable
       ELSE do nothing
     ENDIF
   ENDLOOP
   END
4. (Read SURNAME$)
   BEGIN
   Assign characters to right of last space in NAME$
   to SURNAME$
   END
5. (Read FORENAME$)
   BEGIN
   LOOP while unscanned characters remain in NAME$
   up to last space
     SCAN characters
     IF character is not a letter of the alphabet
       THEN do nothing
       ELSE assign character to FORENAME$
     ENDIF
   ENDLOOP
   END
6. (Discard non-alphabetics from FORENAME$)
   (This has been handled in 5 above)
```

This second level refinement is now very near the stage where it could be coded into a programming language. Let's develop 2 (Convert letters to upper case) to a third level of refinement and then code it into BASIC. We've encountered an algorithm for doing this before (see page 212).

3RD REFINEMENT

```
2. (Convert all letters to upper case)
   BEGIN
   READ NAME$
   LOOP
   FOR L = 1 TO length of string
     READ character L
     IF character is lower case
       THEN subtract 32 from ASCII value of
         character
       ELSE do nothing
     ENDIF
     LET TEMPSTRING$ = TEMPSTRING$ + character
   ENDLOOP
   LET NAME$ = TEMPSTRING$
   END
```