Rank And File

Continuing our programming project to develop a computerised address book, we now look at how our file of data will need to be split up into records and fields

We ended the previous instalment of the Basic Programming course by setting the task of refining the elements of the programming exercise through one or more layers of 'pseudo-language', up to the point where the examples could be coded into BASIC. We will start by revising this exercise and giving some possible solutions. The first 'Statement of Objectives' for the exercise was:

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

In our first level refinement we found that this could be broken down into six steps (later we found that the last step could be dispensed with). These were:

1. Read the name (* READ *)
2. Convert all the letters to upper case (* CONVERT *)
3. Find the last space (* SPACE *)
4. Read the surname (* READSURNAME *)
5. Read the forename (* READFORENAME *)
6. Discard the non-alphabetics from the forename

We are treating all of these activities as subroutines and the name we have assigned to each subroutine is given in brackets. Unfortunately, most versions of BASIC are unable to call subroutines by name and it will be necessary when writing the final program to insert line numbers after the respective GOSUBs. During the development phase, however, it is much easier to refer to subroutines by name. These names can then later be incorporated in REM statements. We are indicating this use of named subroutines by putting the names within asterisks. In languages that can call subroutines by name (such as PASCAL), subroutines like these are usually referred to as 'procedures'.

Even though your BASIC may not be able to handle procedures, it is recommended that you pretend it can while programming at the pseudo-language stage. Similarly, your version of BASIC may not be able to handle long variable names such as COUNT or STREETNAME$, but at the pseudo-language level it is easier and clearer to assume that it can. Try to make them descriptive. It is much clearer to call a temporary variable for a string TEMPSTRING$ than to call it XV$. Fortunately, many versions of BASIC now allow longer variable names.

We have already developed the second of the steps (Convert all the letters to upper case) through a second and third level of refinement and created a short program in BASIC to do this task. We will now attempt this for the other steps:

2ND REFINEMENT

```
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
```

3RD REFINEMENT

```
3. (Find last space)
BEGIN
READ FULLNAME$
LOOP (while unscanned characters remain)
  FOR L = 1 to length of FULLNAME$
  READ character from FULLNAME$
  IF character = "  "
    THEN LET COUNT = position of character
    ELSE do nothing
  ENDIF
ENDLOOP
END
```

We are now in a position to code from pseudo-language into programming language:

```
10 INPUT "INPUT FULL NAME "; FULLNAME$
20 FOR L = 1 TO LEN (FULLNAME$)
30 LET CHAR$ = MID$ (FULLNAME$,L,1)
40 IF CHAR$ = "  " THEN LET COUNT = L
50 NEXT L
60 PRINT "LAST SPACE IS IN POSITION ";COUNT
70 END
```

Note that line 10 is a dummy input for testing the routine; line 60 is a dummy output, also for testing; and line 70 will have to be changed to RETURN when the routine is used as a subroutine.

Now let's try the same process for step four:

2ND REFINEMENT

```
4. (Read surname)
BEGIN
Assign characters to right of last space to SURNAME$
END
```

3RD REFINEMENT

```
4. (Read surname)
BEGIN
  READ FULLNAME$
```