Christmas In Basic

We introduce new commands for dealing with data and write a program to work out the number of days until Christmas

This program revises all the topics covered so far in our programming course, and also introduces several new and powerful BASIC statements. The purpose of the program is to calculate the number of days remaining until Christmas.

If you look at the program listing, you will see that it starts with a list of the variables used. This practice is certainly not essential, but is advisable as it can make your programs much easier to understand when you come to look at them later. Some versions of BASIC allow variables to have long names, DAY for example, rather than the single letters we have been using. If you are lucky enough to have a BASIC that allows long variable names, choose meaningful names. DAY, MONTH or DAYNUM are much better than A, X or D. If you have no choice in the matter because your BASIC does not allow long variable names, listing the variables at the top of the program makes it almost as 'readable'.

When the program is run, the first thing that will appear on the screen will be the PRINT statements starting at line 230. These state briefly what the program will do and then prompt the user to type in the date in the form shown, using commas to separate the day, month and year.

The first unfamiliar statement will be in line 300. This is a DIMension statement. It is used to set the number of items or elements allowed in the array labelled X. An array, sometimes called a subscripted variable, is like an ordinary variable except that the box contains several compartments. In line 300 we are creating a variable called X with 13 compartments inside the box. We shall return to the subject of arrays and the DIM statement in more detail later in the course.

```
310 INPUT D, M$, Y
```

This line is an ordinary INPUT statement except that it expects three inputs. D is a numeric variable that will contain today's date. Y is another numeric variable for the year. M$ is slightly different. It is called a 'string variable' and this is indicated by the $ (dollar) sign. A string variable accepts characters from the keyboard as well as numbers. If, for example, we type 23, JANUARY, 1983, variable D will be assigned the value 23, variable M$ will be assigned the character string JANUARY and variable Y will be assigned the value 1983.

```
330 GOSUB 560 REM 'NO OF MONTH' ROUTINE
```

This statement instructs the program to branch to the subroutine starting at line 560. Note, also that a REMark has been inserted on the same line. If there is room on the line, it is not always necessary to put REMs on a new line. This particular subroutine is only used by the main program once, and strictly speaking could just as easily have been incorporated into the main program. Making it into a subroutine just keeps this part separate from the rest of the program.

When the program was originally written, a number was used for the month and this part of the program was not needed. Later it was decided to allow the month to be entered as a typed word spelled out in full. In order to convert the spell-out month into its equivalent number, the extra program now forming this subroutine was written separately. The only change needed to the main (original) program was to add a single GOSUB statement. This subroutine illustrates the ease with which programs can be built up in blocks and linked together using the GOSUB and RETURN statements.

The subroutine itself is very simple, but illustrates how clever BASIC is at manipulating character strings. Suppose we had entered JANUARY as the month part of the INPUT statement. Variable M$ would then be assigned the character string JANUARY. The first line of the subroutine is:

```
560 IF M$ = "JANUARY" THEN LET M = 1
```

This statement compares the contents of M$ with the characters inside the double quotation marks. If they are the same (as they are in this case) the line goes on to set the value of numeric variable M to 1. Do not confuse variable M with variable M$. They are different. Only one can contain a string variable, the one with the $ sign! After checking to see if M$ is the same as the string JANUARY, the program moves to the next line and checks to see if the contents of M$ are the same as FEBRUARY. It is not, so M is not set to 2. Only where the match is correct will variable M be set to a value, and that value is the same as the number of the month — 1 for January, 3 for March and so on.

On getting to line 680 BASIC RETURNs to the main program, to the line after the GOSUB statement. This is line 340. It contains a REM but no comment. It is inserted simply to space out the program and to make it easier to read.

Lines 350 to 370 are a FOR—NEXT loop. This increments the value of I, starting with 1 and counting up to 13. The variable I is used as the