language to language. BASIC uses the REM statement. The word REM must appear at the front of your comment and then the interpreter will ignore everything it finds up to the next end-of-statement marker (: or (cr)). In other languages (PASCAL, PL/1, PROLOG, etc.) comments are bracketed by /* and */ (sometimes { and }), and anything between the marks is ignored by the compiler. An advantage of this system is that comments can run over more than one line. The disadvantage is that, if you forget the second */, the rest of your program is taken to be a comment and will be ignored!

Use comments wherever you feel some explanation may be needed: when you are defining constants, initialising variables, beginning a program, beginning a new procedure (subroutine), defining a function, or writing some code that isn't readily understood because of its complexity. Comments need not be long or wordy, and often just a reminder is needed. When you are trying to understand the logic of last year's adventure program, large blocks of general comment that break up the code and do not give enough detail can be more of a hindrance than a help, so keep comments short and to the point. Put them before tricky sections of code, and only put them inside the code when their presence is not likely to interfere with reading the logical structure of the program. Our final program (Listing 3) shows some examples as guidelines.

External documentation, in the form of handbooks and written specifications, is the hardest and most tedious to produce. For programmers, studies have shown that written documentation is usually only consulted as a last resort. However, when it is used, it can save a lot of effort. If your program is not too long and is well documented internally, it is unlikely that you will ever find a need for external *program* documentation. *User* documentation is another matter and will be discussed later in the series. Nonetheless, it is often useful to have some written documentation to hand when it comes to revising an old program or to debugging a new one. One of the ways in which the so-called 'fifth generation' languages aim to improve programmer productivity is by generating the documentation automatically. This will be achieved by using information from the design phase of a program's development. Not surprisingly, one of the best ways to document your own programs is to use this same trick.

Keep a file on your programs as you write them. Put into it all the notes you make as you design the program, including drafts of algorithms and flowcharts. Most importantly, keep the final version of the flowchart you have used to write the code from. If you have a printer, keep a listing of the finished program. Note that, in our completed version of the program, the first comment includes the program name and a date. Whenever you modify a program, change the date on it so that you know that it is the latest version.

Properly Documented

Listing 1

BASIC

```
(a) 10 INPUT A,B
    20 C=A*31536000
    30 D=B*2592000
    40 E=C+D
    50 PRINT E
```

PASCAL

```
(b) program abcde (input,output);
    var a,b,c,d,e:integer;
    begin
    read(a,b);
    c:=a*31536000;
    d:=b*2592000;
    e:=c+d;
    writeln(e);
    end.
```

Listing 2

BASIC

```
(a) 10 AYEAR=31536000
    20 AMONTH=2592000
    30 PRINT"Enter your age (years then months separated by a comma) ":
    40 INPUT NYEARS,NMONTHS
    50 YSECS=NYEARS*AYEAR
    60 MSECS=NMONTHS*AMONTH
    70 AGEINSECS=YSECS+MSECS
    80 PRINT"Your age in seconds is (approximately) ";AGEINSECS
```

PASCAL

```
(b) program ageinseconds (input,output);
    const
      ayear=31536000;
      amonth=2592000;
    var
      nyears,nmonths,ysecs,msecs,ageinsecs:integer;
    begin
      write('Enter your age (years then months separated by a comma) ');
      read(nyears,nmonths);
      ysecs:=nyears*ayear;
      msecs:=nmonths*amonth;
      ageinsecs:=ysecs+msecs;
      writeln('Your age in seconds is (approximately) ',ageinsecs);
    end.
```

Listing 3

BASIC

```
(a) 10 REM "AGEINSECONDS"   June 1984
    20 REM INPUTs age in years and months (y,m) and
    30 REM uses an approximate conversion (month = 30 days)
    40 REM to give age in seconds.
    50 REM
    60 AYEAR=31536000:REM          seconds in 365 days
    70 AMONTH=2592000:REM          seconds in 30 days
    80 PRINT"Enter your age (years then months separated by a comma) ";
    90 INPUT NYEARS,NMONTHS
   100 REM age in secs is (age in years * secs in year) plus
         (months since last birthday * secs in month)
   110 YSECS=NYEARS*AYEAR
   120 MSECS=NMONTHS*AMONTH
   130 AGEINSECS=YSECS+MSECS
   140 PRINT"Your age in seconds is (approximately) ";AGEINSECS
```

PASCAL

```
(b) program ageinseconds (input,output);
    /*   June 1984
       reads age in years and months (y,m) and uses an
       approximate conversion (month = 30 days) to give
       age in seconds.                                  */

    const
      ayear=31536000;      /* seconds in 365 days */
      amonth=2592000;      /* seconds in 30 days  */
    var
      nyears,nmonths,ysecs,msecs,ageinsecs:integer;
    begin
      write('Enter your age (years then months separated by a comma) ');
      read(nyears,nmonths);
      /* age in secs is (age in years * secs in year) plus
         (months since last birthday * secs in month)     */
      ysecs:=nyears*ayear;
      msecs:=nmonths*amonth;
      ageinsecs:=ysecs+msecs;
      writeln('Your age in seconds is (approximately) ',ageinsecs);
    end.
```