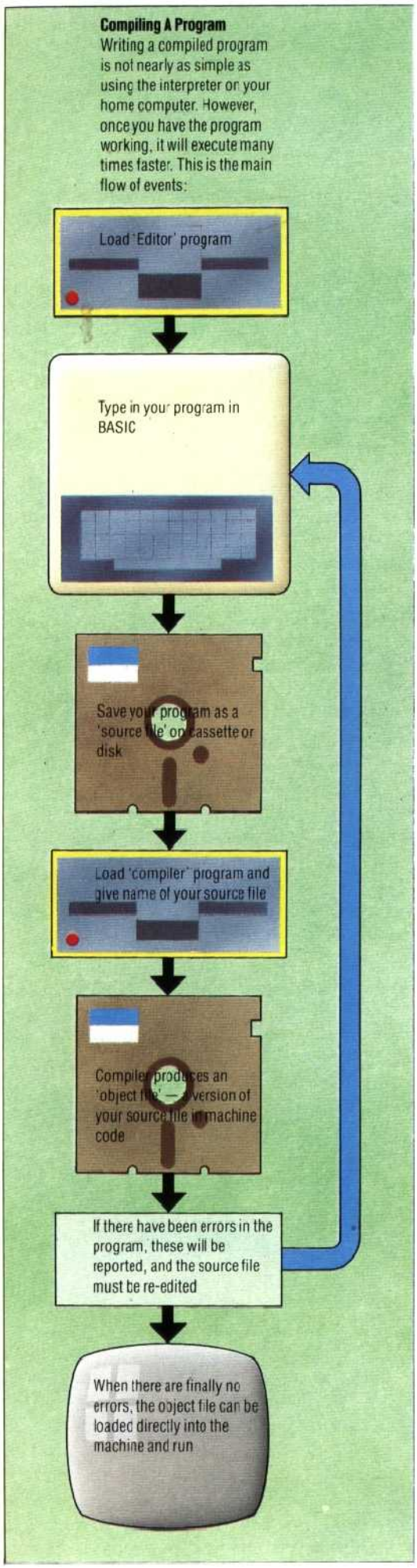

While the compiler translates your file, it checks it for syntax errors. If it finds any, then you'll get a message like this:

```
100 REED X:IF X=3(N-2) LET P=Q
    1            2    3

FATAL ERROR:-
1) //REED// UNRECOGNISED COMMAND
2) //(// ILLEGAL OPERATOR HERE
3) ??'THEN' OR 'GOTO' EXPECTED HERE
```

You get this kind of message for every line that contains an error. In other words, the error reporting is far more comprehensive than on a BASIC interpreter. Now you must load and run the Editor again, recall the source file from disk, make the changes and try to compile again. If there are no more errors you can type:

```
RUN MYPROG
```

and it either works as you expect or it doesn't. There are no syntax errors at this stage, because you've corrected them, but you might still want to change the program anyway, in which case you load and run the Editor, change the source file, recompile it...and so on.

The virtues of a compiler are not obvious in the program development stage, though informative error reporting is valuable. Compilers start to earn their keep after you've got a working program and typed RUN, which is precisely where interpreters start to let you down.

Compiled programs are fast — anything from five to 50 times faster than interpreted programs, depending on the efficiency of the compiler, but the compiled program's speed of execution is bought at the expense of its speed of program development.

Comparing compilers and interpreters by contrasting typical sequences of user commands like those above is unfair on compilers, since they are written mostly for more powerful, less specialised machines, the users of which might want to write and run programs in many different programming languages.

COBOL (for writing commercial data processing programs to handle accounts, payroll and inventory), for example, was invented with compilation in mind, whereas BASIC really demands an interpreter. If you're going to compare a Jensen with a Jeep, you ought to do so on both ploughed fields and metalled roads.

Once you've developed and compiled a program, you don't need the source file except for reference. So the source program can be fully commented on and written with readability in mind, while the object file may be a much smaller file, occupying less space on disk and memory.

The fact that the object files created by a compiler consist of unreadable machine code, can, surprisingly, be an advantage. If you're marketing software you don't sell the source file but only the object file, which makes it much harder to pirate, copy or alter.

Slow
In an Adventure-style game speed is not critical, and most of the program consists of manipulating strings of text. Therefore it is written in BASIC and interpreted as it runs on the computer

Faster
Many business programs (particularly spreadsheets) are difficult to write in machine code because they involve a lot of mathematics. However, an interpreted language would be too slow, so they are often written in BASIC and then compiled

Fastest
For fast action arcade-style games, which involve the manipulation of graphics, even a compiled program would not be nearly fast enough. Such packages have to be written directly in machine code — a slow and painful task