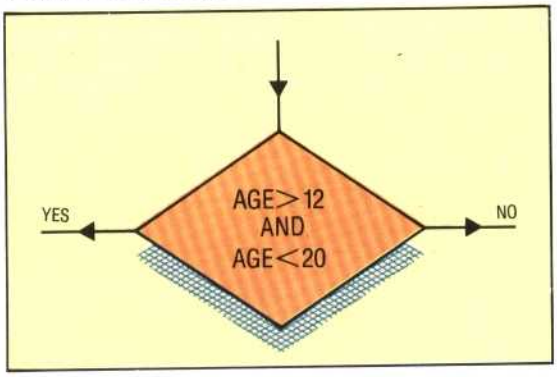
DECISIVE MOVES

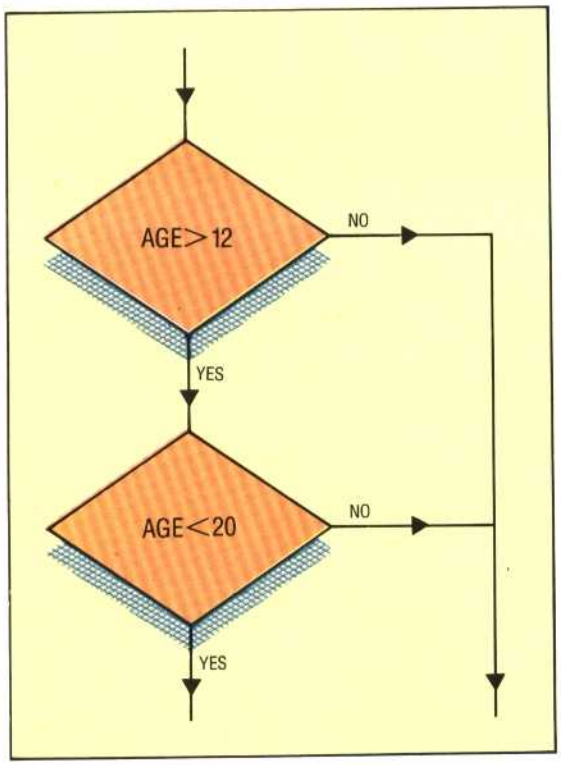
We continue our look at how to improve our use of flowcharts in the planning stages of program development. Here, we show you how compound decisions can be broken down into simple components, and look at the use of 'decision tables' in the more complex cases.

Programmers often want to use compound decisions in their programs such as:

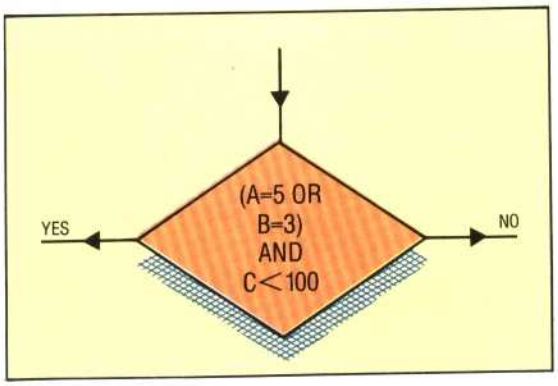
```
IF AGE>12 AND AGE<20 THEN STATUS=
  "TEENAGER"
```

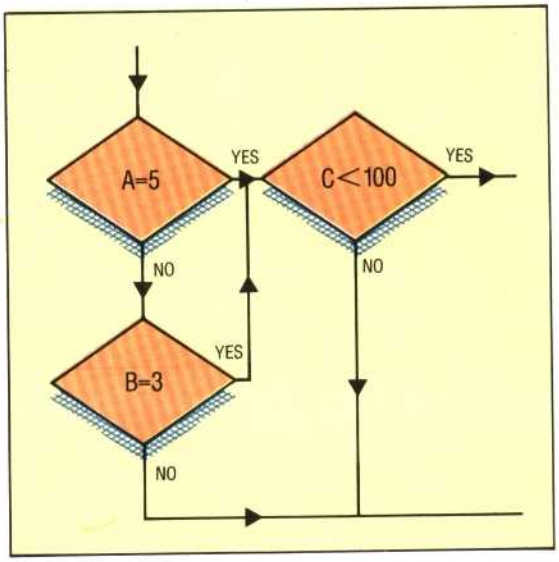
Algorithms with instructions like this are easier to understand if the compound decision is broken down into its component decisions.





We have represented our BASIC example in diagrammatic form to show how the compound version is less satisfactory than the simple version. Our second example (below), consisting of three component decisions, makes the flow of logic through the decision boxes far more intelligible than its compound counterpart. It also makes clear a similarity with the rules of Boolean logic, which enable the construction of complex circuits out of a combination of simple logic gates.





In the examples we have used, all the decisions have been binary ones, yet it is quite common for an algorithm to involve decisions with more than two possible outcomes. If we are reading an input from the keyboard that represents a selection from a menu, we would then want to branch to one of a number of different subroutines to take the requested action. To do this, most programming languages provide multiple branching constructs such as CASE..OF.. in PASCAL and ON...GOTO and ON...GOSUB in BASIC. The rules for binary decisions also apply to multiple decisions: only one route may be taken out of the decision box and