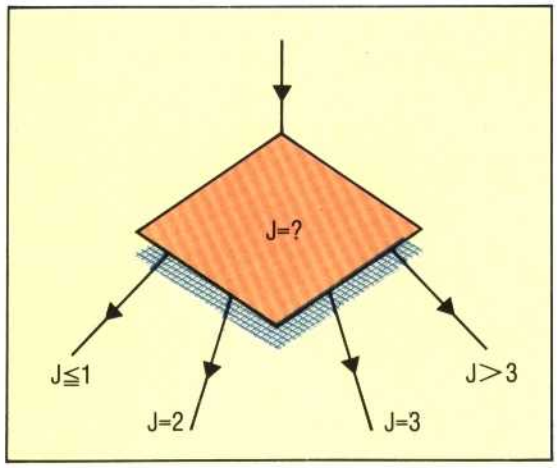
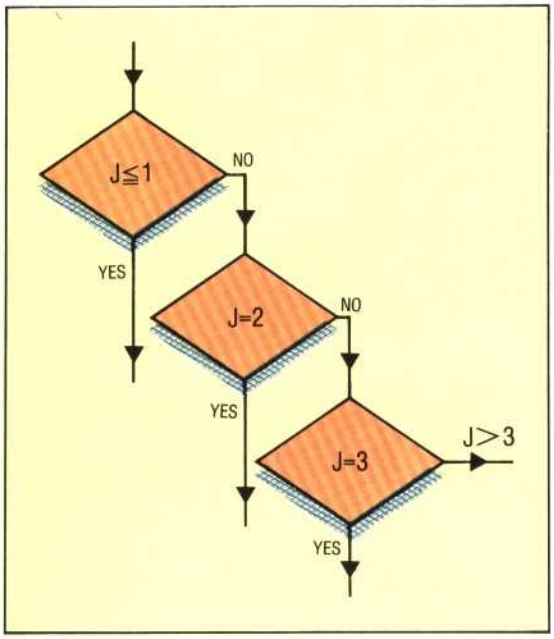
all exits must be well labelled, with all possible routes being mutually exclusive and covering all possibilities. A multiple decision may be drawn, as in our example, with a set of exit paths leading from the same decision box. However, it is rare to see this and, more often, the decision will be broken down into binary decisions, as shown.





All multiple decisions can be represented as a set of binary decisions in this way.

THE DECISION TABLE

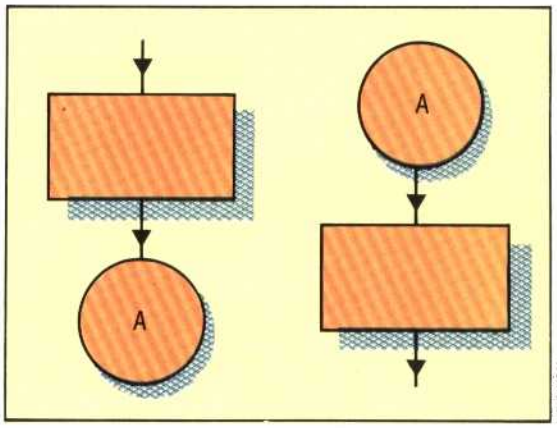
As an alternative to flowcharts, especially where there may be many multiple decisions, we can recommend the use of decision tables. We give an example of such a table, which represents a set of rules for making decisions. The table has four main sections: text describing the conditions for the rules, text describing the actions to be taken, a grid showing how the conditions fit into the rules, and a grid showing which actions are appropriate to each rule. In the 'conditions/rules' grid, values of variables appear in the cells, while in the 'actions/rules' grid below it, a tick indicates what action should be performed and a value acts as an input parameter for that action. Rule 4, for example, reads as: 'If the fire button is pressed and the game level is 2 and the player level is novice, then activate random blaster shields and reduce energy level by two per cent.'

	RULES							
CONDITIONS	1	2	3	4	5	6	7	8
FIRE BUTTON PRESSED	✓	✗	✗	✗	✓	✗	✓	✗
GAME LEVEL	1	1	2	2	1	1	2	2
PLAYER LEVEL	NOVICE	NOVICE	NOVICE	NOVICE	EXPERT	EXPERT	EXPERT	EXPERT
ACTIONS								
ALIEN EVASIVE ACTION							✓	✓
RANDOM BLASTER SHIELDS			✓	✓			✓	✓
REDUCE ENERGY LEVEL BY	1%	1%	2%	2%	2%	2%	4%	4%

Decision tables also serve to combine simple decisions into compound decisions and, in simpler forms than the one given here, are exactly equivalent to the truth tables used for predicting the output of logic gates.

One final point about the use of flowcharts. Wherever possible, restrict your flowcharts to one page. It can be irritating and time consuming leafing through many pages of paper. If your algorithm becomes too large, try and break it down into smaller algorithms. Remember that each algorithm can be used as a single instruction in some other algorithm. In this way, each routine in a program could be written as a single process box in a flowchart of the whole program, even if that routine uses other routines that in turn use others, and so on.

Inevitably something will go wrong now and then and a need will arise for a flowchart to continue beyond one page. If this happens, divide the flowchart at a suitable point (a decision, say) and use a circle with an identifying symbol inside it to point to the place where the flow of control continues on the next page (represented by another circle with the same symbol inside it, as shown below). If control returns to the main program, use the circles again to point back. Another solution is to view the missing portion as a separate algorithm, refer to it in a process box and represent it with its own separate flowchart.



LIZ DIXON