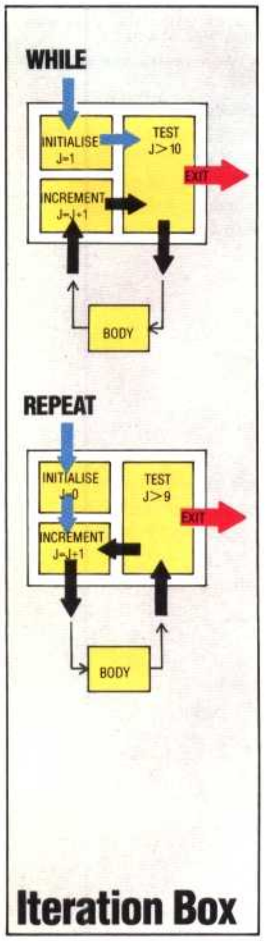
LOOP LINES



Iteration Box

Flowcharts are an important technique in program design, but the commonplace notation is not always sufficiently precise, especially when dealing with loop structures. We consider various ways of classifying loops, and introduce a new flowchart symbol — the iteration box.

Iteration, or looping, is one of the essential structures of any programming language. Earlier in the course, we said that a loop was used in an algorithm whenever a decision switched the flow of control onto a path that brought it back, eventually, to the same initial decision. This adequately describes the structure: the repeated performance of a body of code. However, it doesn't define it in all its many forms. Since loops are such an essential primitive structure, comprising probably 60 per cent of all processor activity, it is extremely useful to look at them in more detail. We will pay particular attention to their effect on general program/algorithm structure, and to the various ways in which they are constructed and classified.

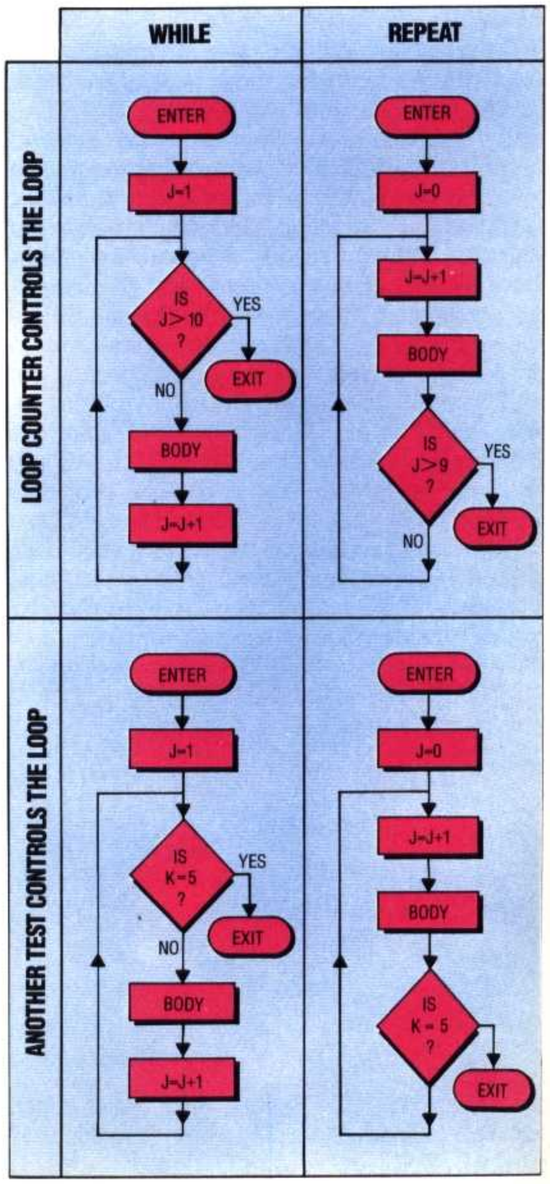
Loops are often divided into two classes, depending on their similarity to the two high-level language loop structures, REPEAT... UNTIL and WHILE... ENDWHILE; both types are used in PASCAL, and the REPEAT loop is implemented in BBC and Oric BASIC. The two types differ in their positioning of the loop exit test: in a REPEAT loop the test comes at the end of the loop body, whereas in a WHILE loop it comes at the start. This means that the body of a REPEAT loop, once entered, will always be executed at least once, whereas that of a WHILE loop need not be. This first difference can be seen quite clearly in the flowchart diagram.

Another way of classifying loops is according to whether the variable that acts as the loop counter is used in the loop exit test, or whether some other test condition controls the loop exit. This is less clearly seen in a flowchart of the conventional linear kind. In fact, the kind of 'commonsense' flowchart that we have become familiar with portrays loops in an unhelpful way. In these flowcharts, a loop looks exactly the same as a simple branch, and it is often necessary to examine the algorithm in detail to distinguish between them.

A clearer notation, called the 'iteration box', exists, which clearly marks the start of a loop, and eliminates the loop/branch confusion. It consists of three linked boxes: the first box shows the initialisation of the counter, the second shows the incrementing of the counter, and the third contains the loop exit test. REPEAT and WHILE loops can both be shown in this form, but differ in the flow of control through the boxes, while the test box indicates whether the loop is counter-controlled or not. These points can be clearly seen in the diagram.

In a REPEAT loop the flow of control is 'initialise-body-test-body-test', whereas a WHILE loop goes 'initialise-test-body-test-body'. This can be seen in the way that the control lines leave and re-enter the iteration box, with the body of the loop 'dangling' from them.

Loop Classification


WHILE
REPEAT
LOOP COUNTER CONTROLS THE LOOP
ANOTHER TEST CONTROLS THE LOOP
ENTER
J=1
IS J>10 ?
YES
EXIT
NO
BODY
J=J+1
ENTER
J=0
J=J+1
BODY
IS J>9 ?
YES
EXIT
NO
ENTER
J=1
IS K=5 ?
YES
EXIT
NO
BODY
J=J+1
ENTER
J=0
J=J+1
BODY
IS K=5 ?
EXIT