2024-05-17 / Reading Comprehension / English for Software Development

Chapter 10. Weather, Rise, Repeat

[Everything under Control; second to third paragraphs]

A loop controls how often its body is executed, but the effect of the loop is obtained solely by the steps of its body. In other words, a loop □ does not have any effect directly but rather indirectly through the repetition of the steps of its body. Since it generally matters how often an algorithmic step is executed, a loop exerts its influence through the number of times it executes its body. Since a loop controls the effect of its body (through the termination condition), it is called a *control structure*. A loop is a control structure for repeatedly executing a group of algorithmic steps. The other two major control structures are sequential composition and the conditional.

Sequential composition connects two steps into an ordered sequence of steps. I have previously used the word *and* for this purpose, but to indicate that both steps are to be executed in sequence, not □ □, it might be better to use a keyword such as *andThen* or *followedBy*. However, for simplicity and succinctness I □ the notion used in most programming languages, □, connecting two steps by a semicolon. […]

[A Loop Is a Loop Is a Loop; fifth to seventh paragraphs]

All the different loop descriptions □ so far (repeat, while, and recursion) have in common that their termination is controlled by a condition that is reevaluated before or after each execution of the body. The termination of the loop depends on the body to have an effect that □ makes the termination condition true (or the entry condition false, in the case of a while loop). This means that it is not known □ how many iterations a loop will go through; it is not even clear that any such loop will terminate □. This uncertainty is actually an important part of the Groundhog Day loop that Phil Connors experiences.

For some computations described by loops, □, it is clear how many times the loop should be executed. For example, if the task is to compute the square of the first ten natural numbers, it is clear that this computation can be achieved by a loop that repeats the squaring operation exactly ten times. Or, □ the algorithm to fold a piece of paper to fit into an envelope, which is described by a loop that is executed exactly twice. For case like these, we employ for loops, which have the following general form:

for *numbers* **times do** *step*

Using this schema, the paper-folding loop would be □ as “**for** 2 **times do** *fold*”. The advantage of a for loop is that it is absolutely clear even before it is executed how many iterations will be □. This is not the case for the other loops because one finds out only when executing the loop. This is an immensely important difference, since the for loop is □ to terminate, while the other loops may run forever.

Closely related is the question of the runtime of loops. It is clear that a loop that is executed, □, 100 times takes at least 100 steps. In other words, a loop is linear in the number of its iterations, and this is the case for every kind of loop. In addition to the number of iterations we also have to consider the runtime of the loop’s body. The runtime of a loop is the number of its iterations times the runtime of the body. For loop is linear in the size of the list, and the body takes, on average, time proportional to half of the list length. Therefore, the runtime of selection sort is quadratic in the size of the list.

Sentences To Memorize

1. Note that even in a loop that produces varying outcomes, the loop body is the same for each iteration.
2. The control structures presented here are those used by algorithms for single computers.
3. Yet another description of loops can be achieved using recursion.