R 17 WHILE, REPEAT-UNTIL Loops - Reading

Computers are great at doing the same task over and over again. In programming, structures that repeat sections of code are known as loops or **iteration**. There are three types of loop and we will look at two of them here. Any type of loop can be programmed to go on forever. This is known as an **infinite loop**. Programmers don't normally want loops to go on forever, so infinite loops are normally caused by a programming error. When a program locks up and seems to be doing lots of work but achieving nothing it is often due to an infinite loop.

WHILE loops

WHILE loops check a **condition** at the start of the **loop**. If the condition is True then they will loop, if not they will go to the end of the loop.

```
WHILE condition
      Run this code
ENDWHILE
```

The example to the right shows a WHILE loop that will **repeat** the code between WHILE and ENDWHILE. Each time the user will be asked to enter something. If they enter "quit" then the **Boolean variable** gameOver will be assigned the value True. The condition in the WHILE loop needs to be True to repeat. As gameOver is True, NOT gameOver will be False. So the WHILE loop will stop repeating.

```
gameOver ← False
WHILE NOT gameOver
    OUTPUT 'Playing game'
    userChoice ← USERINPUT
    IF userChoice = "quit" THEN
        gameOver ← True
    ENDIF
ENDWHILE
```

We could use a WHILE loop to help find an average for class scores. The program first **initialises** variables that it needs to use. It receives a score from the user and adds this to totalScore. It also adds 1 to numOfStudents. If the user next enters "no", then moreStudents is set to False. Otherwise it will remain True. At the start of the next iteration moreStudents is checked. If it is False then the loop will finish and the average will be calculated.

```
totalScore ← 0
numOfStudents ← 0
moreStudents ← True
WHILE moreStudents
    score ← USERINPUT
    totalScore ← totalScore + score
    numOfStudents ← numOfStudents + 1
    addAnother ← USERINPUT
    IF addAnother = "no" THEN
        moreStudents ← False
    ENDIF
ENDWHILE
average ← totalScore / numOfStudents
OUTPUT average
```

REPEAT-UNTIL loops

REPEAT-UNTIL loops check the condition at the end of the loop. They will always run the first iteration.

```
REPEAT
      Run this code
UNTIL condition
```

This may be more useful for instance in a game where you wish to give the player a free game before checking the condition to see if they have paid.

```
moneyPaid ← 0
REPEAT
    Play game
    moneyAdded ← USERINPUT
    moneyPaid ← moneyPaid + moneyAdded
UNTIL moneyPaid ≤ 0
```