The do while loop

The **do .. while … loop** is different from the **while loop** as it always executes the block of instructions at least **once** because the condition is checked at the **end** of the loop.

```
do
{
        // ---- put instructions to be repeated here
}
while (condition);    // while the condition is true
```

1 Ceebot Task 7.6: Calculator 2

Your task is to use a **for loop** to input 4 numbers. When the loop has finished, output the total of the 4 numbers

- You will need to use the **dialog(…)** and **strval(…)** instructions as well as the **message(…)** instruction to output to the screen.
- And, of course, you will need a **for loop**.
- Note: your input prompts should look like this:
  Enter Number 1
  Enter Number 2
  etc.

Testing
Design a suitable **test plan** for your program, with at least 3 different tests.

Extra
Can you also display the **average** (of the numbers entered) in the same message as the total?

2 Ceebot Task 7.2: Massacre

Your task is to destroy the 10 spiders that surround you.

- As there are 10 spiders you need a loop that repeats 10 times.
- Before you start, you will need to set up the variables you need.
- A suitable **algorithm** for the program is:
- You should note that the spiders have conveniently arranged themselves **15 degrees apart**.

Algorithm
```
1. Loop  10 times
        a. fire cannon for 0.1 seconds
        b. turn to next spider
        c. pause for 1.5 seconds
   End Loop
```

Extra
Now also get the program to display a message after each firing:
"Spider 1 Destroyed" etc.

3

Ceebot Task 11.3: Shooting Practice

Your task: Using **one** series of statements, destroy **all** of the targets

Hint
Using **nested loops** is the most efficient way. Create a loop (the inner loop) that destroys one side of targets, and then repeat that loop 4 times (the outer loop) turning after each side is destroyed.

4

Ceebot Task 7.3: Blasted Ants

This exercise uses an **infinite loop** to destroy all the attacking ants. First set up a **while loop** with **true** condition .. this makes the loop continue forever!

- Inside the loop you need to keep firing as you turn your robot. Try using **fire()** and **turn()** .. it doesn't work. You can turn or fire, but not both at the same time!
- What you need is a new instruction: **drive(..)** .. which has 2 parameters, the **speed** and the **direction** of turning (both can vary from -1 to +1)
- **drive(0, 1);** will starts the robot turning on the spot in an anticlockwise direction .. other instructions such as **fire(..);** will still work as the robot turns.
- Put all this together to create your program

Extra: Can you do another program that uses your radar to detect the ants before firing?

5

Ceebot Task 12.1: Testing, Testing!

Power Cells are not what they used to be! You have been asked to design a test program for them that can be run in a loop **any** number of times.

Your task:
Put together a sequence of robot instructions to run in a loop. Your test sequence should have a few movements, turns and firing. An example is given in the algorithm below. The test will use up energy from the power cell and you can then display this energy level (how?... see the note below)

- Start the program by asking the user to **input** how many tests are to be run
- then use a **while loop** to run that number of tests.
- You should **output** messages at the beginning and end of each test:
  Starting Test 1 (etc.) and
  Finished Test 1: Energy Level = …… (etc.)

A partial **algorithm** for the program is:

Note: **energyCell.energyLevel** gives the current level of the power cell.

When the program is working, **test both robots** to see which uses most energy for the same number of tests.

```
Algorithm (partial)
1.  Set counter to zero
2.  Input the number of tests to run
3.  Loop while counter < number of tests
        a. output starting test message
            ▪ move forward 3 metres
            ▪ turn 180 degrees
            ▪ fire for 0.5 seconds
            ▪ move back 3 metres
        b. output finish test message
        c. output current energy level of power cell
        d. add 1 to counter
    End Loop
```

6

Ceebot Task 8.4: Exchange Posts 2

This exercise uses **Information Exchange Posts** which store information that can be picked up by any robot within 10 metres. Your robot is on a very dangerous path through a lava lake. You may just be able to see your goal on the right .. a tall **Lightning Conductor** in the distance across the lake.

Your task .. reach the goal destination without falling into the lake!
But how can you know what direction to take and how much to move your robot?

- This information is stored in each of the 9 **Exchange Posts**.
- If you click on one of these you will see that it stores a **Number**, a **Direction** and a **Length**. Each Exchange Post tells you how to get to the next one.
- There are 9 Exchange Posts altogether so use an appropriate **for loop** for this.

Program Guidance
What must you do inside the loop?

- You need to **receive** the Direction and Length information from an Exchange post.
- Then use **turn(…)** and **move(…)** with these received values.

How do you receive the information?.

- First set up two float variables to hold the information ..

```
float  dir;     // declare variable to hold direction
float  dist;    // declare variable to hold distance
```

- then use the **receive(…)** instruction twice …

```
dir = receive("Direction");   // pick up Direction info from Exchange Post
dist = receive("Length");    // pick up Length info from Exchange Post
```

Design an algorithm for the program. Then use it to produce a working program

Note: Speed up your program using **x2** or **x3** buttons. Fly the astronaut to watch the action.