Week 7
Arrays

The Technical Bit

What are arrays?

An array is a data structure, able to store a collection of items of the same type. It is like a numbered list or set of numbered boxes. You can put data in each separate box in the array.

Defining an Array

An array is defined using square brackets [ ] which come either after the type or after the variable name. We can define an array like this .. **float myArray[20];** .. this sets up an array of 20 'boxes', each able to store one float number.
Once an array has been defined, it can be used to store and retrieve many similar items.
Arrays always start their numbering from 0, so in this example we have **myArray[0]** up to **myArray[19]** but **NOT** myArray[20].
Here are some more examples:

int[] nums;	.. nums is an array of integers
int nums[12];	.. nums is an array of 12 integers
string names[3];	.. names is an array of 3 strings
float numbers[20];	.. numbers is an array of 20 float numbers.
object items[30];	.. items is an array of 30 objects.

Using an Array

- The separate 'boxes' of an array can be accessed using its **index** number e.g. **nums[3] = 5.7;** which puts 5.7 into the 4th 'box' (n.b. numbering starts at 0)

- You can often use loops to put a lot of data **into** an array, e.g.

```
for (int i=0; i<12; i++)    // store 12 items in nums[] array
{
        nums[i] = strval(dialog("Enter a number please"));
}
```

- Similarly, you can use loops to **output** data stored in an array, e.g.

```
for (int i=0; i<12; i++)    // output 12 items from nums[] array
{
        message( "Number " + i + " is " + nums[i] );
}
```

Using Arrays with functions

You can pass a whole array as a parameter to a function by just using its **name** as a reference. Any changes made inside the function, will automatically change the outside array too. e.g.

```
extern void::object::ArrayExample()
{
	float  numbers[20];    // define an array called numbers
	numbers[0]=0;          //  initialise first element (this is an extra step required in Ceebot)

	inputNums(numbers);  // call the inputNums function and pass the array name
	outputNums(numbers);  // call the outputNums function and pass the array name
}
//************************************
void object::inputNums( float[ ]  n)    // n is an array parameter
{
	for (int i=0; i<20; i++)    // store 20 items in nums[] array
	{
		n[i] = strval(dialog("Enter a number please"));
			// changes made here to array n also change array numbers
	}
}
//************************************
void object::outputNums( float[ ]  n)    // n is an array parameter
{
	for (int i=0; i<20; i++)    // ouput 20 items in nums[] array
	{
		message( "Number " + i + " is " + n [i] );
			// array n has been passed the data from array numbers
	}
}
```

1

Ceebot Task 22.1: Exchange Posts

Here is an awkward narrow trail consisting of 6 sections, each of length 20 metres. There are 6 turns to make, but the angles are all different. How can you avoid plunging into the lava lake?

Your task:
Your robot must reach the **finish** platform, near the Derrick you can see in the distance .. and of course survive the lava lake! Fortunately, the six directions that you need are all stored in the nearby **Exchange Post**. If you click on it, you will see these directions.

Program Guidance

1. Before it starts, the robot must pick up the 6 directions from the exchange post.

- To store this information it needs to use an **array**
- As we need to store **6** directions, we must declare an array with 6 cells:

  float dir[6];

  This creates an array called **dir** with 6 cells, numbered from 0 to 5, each able to store 1 float number.
- Your robot must read the directions from the exchange post. You could write:

  dir[0] = receive("Direction0");
  dir[1] = receive("Direction1"); etc.

  But it is much better to use a loop, where the counter **i** will go from 0 to 5:
  Then we can use ..

  dir[i] = receive("Direction"+i);

  inside the loop, using **i** to step through the cells of the array.

2. Program a **for-loop** to receive all the directions from the Exchange Post.

3. Then add another **for-loop** to the program, to **move** the robot 20 metres and then **turn** using the directions stored in the array. Hint: use **dir[i]** again.

2

Ceebot Task 22.2: Exchange Posts 2

This task is exactly the same as a previous one (22.1) but you are required to complete it this time using **functions** with array parameters

Your Task
Write the program using 2 functions: **GetData(…)** and **UseData(…)**
You pass an array to the two functions..

- The first function **GetData(…)** will pick up all the Exchange Post data and store it in the array.
- The second function **UseData(…)** will move the robot along its path using the array data picked up by GetData(…).

Program Guidance
Read the notes: Using Arrays with functions in the 'Technical Bit' at the start of this unit.
You may also find advice by using [F1] in the normal way.

3 Ceebot Task 22.3: Secret Picture

The information required to draw a secret picture is contained in **2 arrays**.
One array contains 7 **distances** and the other 7 **angles**.

Your task:
Use the distance and angle information to **draw** the picture in red.

- You will find that the information has already been given to you as part of the program in slot 1 of the editor. You just need to complete the program.

Program Guidance

- These are the definitions of the 2 arrays that you have already been given:

```
float  dist[7];         // an array for 7 distances
float  angle[7];        // an array for 7 angles
```

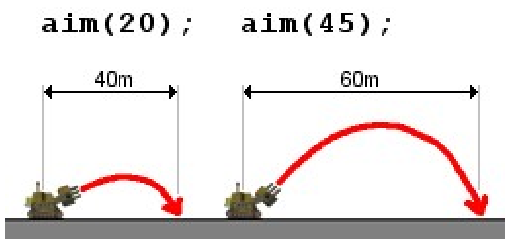
- The information you need has already been loaded into the 2 arrays:

```
dist[0] = 4;              angle[0] = 90;
dist[1] = 6;   etc.       angle[1] = -90;    etc.
```

- Now all you have to do is program a **for-loop** that will use this array information to **move** and **turn** while drawing using the robot pen.
- The drawing should be **red**, so add the appropriate instructions to do this.

4 Ceebot Task 22.4: Ballistic Fire

The planet is infested with giant spiders. A giant shooter is needed to handle them. The **PhazerShooter** is a powerful robot that can rotate and aim at angles of up to 45 degrees. If you know the distance to a spider, you can set the angle for firing.



Your Task
Destroy all the **AlienSpiders** that are running around at various distances from the robot.

Program Guidance
1. How can you know the correct angles to aim the shooter?

- You could perform a very complex calculation, but an **array** has already been set up for you as a **lookup-table** with all the angles stored for every possible distance.
- Whenever you start a new program, the editor sets up the array with all the data:

```
float  angle[64];      // set up a large array for storing 64 float angles

// store firing angles for various distances in the angle array
angle[0] = 0.0;     angle[1] = 0.45;  angle[2] = 0.90;  angle[3] = 1.35;
angle[4] = 1.80;    angle[5] = 2.25;  angle[6] = 2.70;  angle[7] = 3.15;
angle[8] = 3.61;    angle[9] = 4.06;  angle[10] = 4.52; angle[11] = 4.97;   etc.
```

2. How can you pick the right angle to use?

- the robot must first use its **radar** to find an **AlienSpider**
- the spider **direction** can then be found and the robot **turn** to point in that direction
- the **distance** between this robot and the spider can be found:
  for example, **dist** = **distance(this.position, item.position);**
- then use **aim(…)** using the correct angle for this distance (try using **angle[dist]**)

3. This will need to be done in a repeating loop.

If you are still have problems, find an algorithm that might help you, by pressing the **[F1]** key.

5 Ceebot Task 22.6: There and Back Again

A dangerous explosive robot has been left behind on this planet of lakes and hills. You must find and destroy it. Fortunately the expedition that discovered the explosive has marked a path to it using **WayPoints.**.

Your Task

Destroy the explosive using your LeggedShooter robot and then return back to base camp. All you have to do is follow the WayPoints and you will get into a suitable position to destroy the explosive device. But …

Program Guidance

How will the robot get back after destroying the explosive? The **WayPoints** disappear as you cross over them! So you will need a way to store your route if you are to get back safely.

- The secret of success is to use an **array** to store the details of your movements.
- There are 8 **WayPoint**s to track and you can use your **radar** to detect them.
- You should then store the position of each **WayPoint** as you move along the path.
- When you reach your destination, you can destroy the explosive device
- Then retrace your steps using the stored details in the array.

Hints

There are several ways of tackling this exercise, but this is probably the most sensible:

- Create an array of **point** to store the position of each WayPoint as you detect it. (point variables are used to store the coordinates of any point in space)
  ```
  point  path[8];            // declare path as an array of point
  ```
- Then within a **loop**, you can use your radar to detect a **WayPoint**, **goto** the **position** of the WayPoint and also store its position in the array:
  ```
  path[i] = item.position;    // store its position point
  ```
- When you have reached your destination, you can destroy the **TargetBot** there .. use your **radar**, then **turn** and **fire**.
- Later, to retrace your steps, you can step through the array in **reverse** and **goto** each position point: **goto(path[i]);** // goto the current path point

6 Ceebot Task 22.8: Secret Slaughter

In this exercise you are required to destroy only the 'shooter' robots that are travelling along the misty road ahead of you. The problem is there are 4 different types of robot so how can you use the radar? Use an integer **array** to store all the categories you need to detect, then you can use this with the radar instruction.

e.g.
```
int [ ]  list;                        // set up an array called list

list[0] = WheeledShooter;             // add some items
list[1] = TrackedOrgaShooter;         // etc.

item = radar (list, 0, 10);           // detect the list using a 10 degree beam
```

You should keep a count of how many robots are destroyed and display a message after each shooting, saying what category of robot was destroyed.

e.g.

WheeledShooter destroyed! Total Shooters Eliminated = 5

Use the [F1] key to get more information

Week 7: Independent Study (5 Tasks)

The following exercises will be marked. Attempt them outside of class, and copy your code, as well as screenshots, and algorithms into a logbook. You will be required to submit this logbook electronically

Instead of doing new exercises, your task this week is to convert class exercises 3, 4, 5 and 6 so that they use functions with arrays passed as parameters. You should also include Task 22.2 (exercise 2) in your logbook.