# Ceebot Programing

bucks new university

## Programming Concepts CO452


## Study Pack for Ceebot

# Part B
# Weeks 5-8

Original work by Brian Ward

## Contents



## Learning Outcomes

# for the Programming Concepts module: CO452

**On successful completion of the module the student will be able to:**

- Analyse a simple requirement in a structured manner in order to establish a strategy to solve the current problem
- Design, document, implement and test reliable, maintainable programs as solutions to simple problems
- Use structured techniques of design and implementation and good documentation practice

Make effective use of software development tools when implementing fit-for-purpose solutions

## General Introduction

### Welcome to Programming Concepts

- The CO452 module plan and method of assessment for this module is detailed at the back of this booklet, and you will also find the assessment criteria there.
  We want you to enjoy this module and achieve a good result. Therefore it is important that you read the module plan and assessment criteria at your leisure.

- You will need an electronic A4 logbook to record your work. Please get this up and running as soon as possible.
  Classwork will be checked each week, and should be recorded in your logbook.

- You start this module by using Ceebot, a highly visual environment for learning fundamental programming concepts. It uses a C++/C#/Java programming style, so principles learnt here will transfer easily to other programming areas. Hopefully you will also find Ceebot fun to use. The first few exercises should be fairly easy, but you will find that they get more challenging in later weeks. Near the end of the module you will be introduced to the C# language .Good Luck!

- In the first introductory session you will learn the essentials of working with **Ceebot**. Then you will progress by using the most important principles of programming.

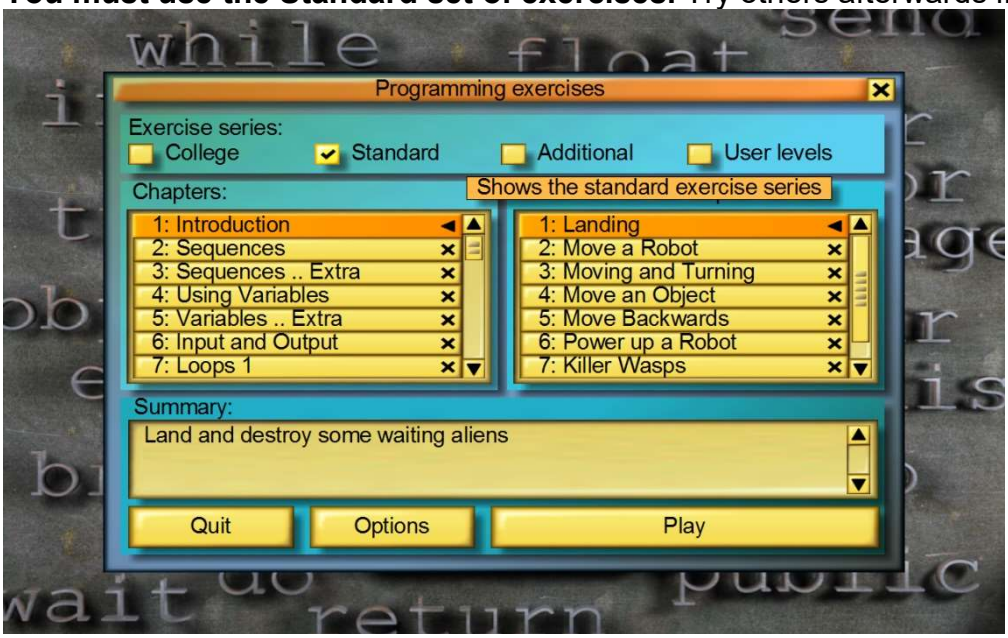
- Please Note: There are hundreds of exercises in the Ceebot package. **You are NOT expected to complete them ALL!** The exercises that you need to complete will be explained here in this document. Of course you CAN do the others if you want to!

- For your convenience, the details of each task are summarised for you in this booklet, but you will also find information by pressing the **[F1]** key during an exercise. Using the **[F2]** key will bring up general support for the current chapter.
  - If you wish, you will be able to purchase a copy of Ceebot and the exercises for your own personal use (for a nominal charge).

# Classwork

Remember, you do NOT have to do all the Ceebot exercises in the package.
The ones you need to attempt are detailed here in this booklet.

**You must use the Standard set of exercises.** Try others afterwards if you want to.




The first thing to keep in mind is that when you have selected an exercise and it has loaded:

- the **[F1]** key will always bring up the instructions for the current exercise.
- the **[F2]** key will always give you more general help with the chapter and the instructions you may need to use to complete a task..

This week you are to try to complete some tasks in class from the first few **Ceebot** chapters:
Ask for help if you need it.

- Show your solutions to your lecturer as you complete them
- always include **comments** in your code (ask your lecturer how to do this)
- You should include the following information in comments at the top of each program:
  - **// Programmer's name: and ID:**
  - **// Course:**
  - **// Week No: and Exercise No:**
  - **// Date:**
- print out your finished code, and put this into your logbook with appropriate **headings**
  (Note .. you can cut and paste into MS Word or WordPad for later printing)

# Your Log Book

You should try to organise your log book clearly and logically.

- Put **Unit Headings** and **Task Headings**.
- Give a brief description of the task
- Stick in your commented source code solution
- You may sometimes need other documentation such as algorithms or test plans.
- Add brief comments as to your success or otherwise and any problems that occurred

This will become more important in later chapters.

# Week 5
# Functions

## The Technical Bit

## Functions using Ceebot

Functions can be used to break up large programs into smaller, more manageable units that can be reused over and over again. They can also be used to return a result back to the main program,.

e.g. Here we define a function called **DoSomething()** and call it from the MainProgram.

```
extern  void  object::MainProgram()
{
      DoSomething() ;          // call the DoSomething() function
}
//**************************
void  object::DoSomething()     // define the DoSomething() function
{
    message ("Now entering the DoSomething function");
       // put statements to be done here
}
```

Functions are particularly useful when designing LARGE programs because the program can be divided up into smaller, easier-to-handle units (i.e. functions)

1

# Ceebot Task 18.1: A Square Function

Design a function called **OneSquare()** and use it to draw several different squares, all having 3 metre sides.

**Your task:**

- First of all write a function that will draw a single square. You should put the function underneath the main part of the program. (see below for an incomplete version)

```
extern void object::Task18_1()
{
      //  XXX   call the function here
}
//***************************
void object::OneSquare()   // define OneSquare() function
{
    pendown();
    for (int i=0; i<4; i++)
    {
          // put 2 more instructions here
    }
    penup();
}
```

- Now in the main program (at XXX above) you need to put the instruction to **call** the OneSquare() function. To do this you just use the function **name** with its brackets … like this:

  **OneSquare();**

- Try running the program .. you will need to put 2 more instructions inside the OneSquare() **for loop** to get it to work correctly.

- Now use the OneSquare() function to draw 3 different colour squares, each separated by 4 metres. Here is the **algorithm** to use in your main program

- Your solution should have **one** OneSquare() function that is used 3 times.
- Test it to see if it works

**Algorithm**

1. set colour to red
2. draw one square
3. move 4 metres
4. set colour to blue
5. draw one square
6. move 4 metres
7. set colour to green
8. draw one square

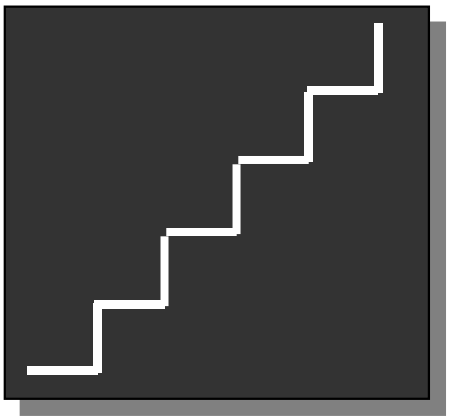
# 2 Ceebot Task 18.2: The Step Function (1)

Use a Step() function to draw a rising staircase.

**Your task:**

- First of all write a function that will just draw **one** step. This is a horizontal line of 2 metres followed by a vertical line of 2 metres, like :

- Now using a for loop in your main program, call the step function 5 times so that you draw a red staircase consisting of 5 steps:



- Your solution should have **one** Step() function that is called **5 times** from the main program
- Use the **fill()** or **fillall()** instruction to fill your stage with a suitable colour before drawing (see unit 3.7)

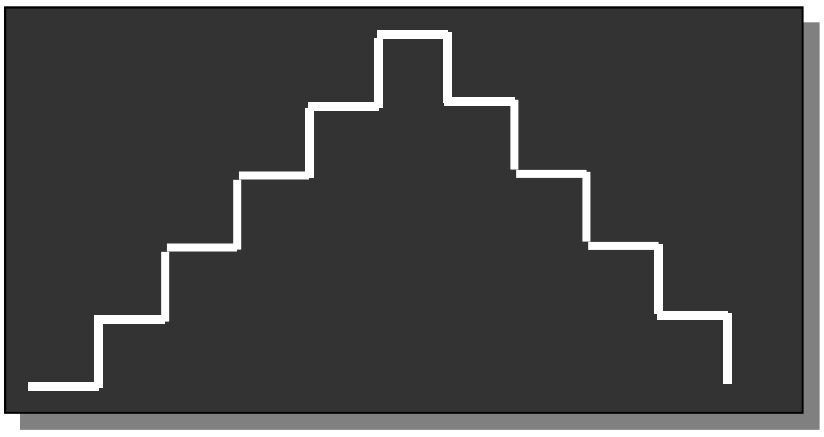
# 3 Ceebot Task 18.2: The Step Function (2)

Modify the previous program to draw a step pyramid

**Your task:**

- This program requires 2 functions.
  - **StepUp()** is the same as the Step() function in the previous exercise. It draws **one** upwardstep.
  - **StepDown()** draws **one** downward step, like this:

- Now using for loops in the main program, draw 5 upward steps followed by 5 downward steps, producing a step pyramid like this:



- Your solution should have **one** StepUp() function and **one** StepDown() function, each of them being called **5 times** from the main program
- Test it to see if it works

# 4 Ceebot Task 18.5: Power Cell Functions

In this program you are to produce a PowerCell from a piece of TitaniumOre

**Your task:**

This is a larger program that divides naturally into 4 parts. Use **functions** for each part.
The 4 parts of the program are :

**1. Get the TitaniumOre**
**2. Convert this into Titanium using the Converter.**
**3. Produce a PowerCell from the Titanium using the PowerPlant**
**4. Deliver it to the GoalArea**

- The four functions can be given the following names (choose others if you want to):
  **GetOre();**
  **ConvertOre();**
  **ProduceCell();**
  **Deliver();**
- It is up to you to decide what to put into each function. Your best strategy is to program one function at a time and test it to see if it works.
- Hint: You will need to use the **radar()** instruction in each function and you can use this to find the **TitaniumOre**, the **Converter**, the **PowerPlant** or the **GoalArea**

**How Long Must you Wait?**

The Converter takes time to produce Titanium, so you must wait before picking it up.
Use a while loop with radar .. a **null** result means NO Titanium detected yet (see below)

```
while (radar(Titanium) == null)
{
      wait(0.1);        // wait until radar detects Titanium
 }
```

- include this wait loop in the appropriate function
- use a similar method to wait for the **PowerPlant** to do its job

# 5 Ceebot Task 18.6: Fly Using Functions

**1. Take Off**
**2. Fly to the island**
**3. Land**

Get to the island GoalArea, using **functions** to program the 3 main parts

**Your task:**

- Write a program that calls 3 functions
- Use these algorithms to help:

**TakeOff**
1. Start Climbing
2. Loop while altitude < 10 metres
   - pause

   End Loop
3. Stabilise altitude

**FlyToIsland**
1. Use radar to detect GoalArea
2. Turn in this direction
3. Calculate distance to GoalArea
4. Move this distance

**Land**
1. Start Descending
2. Loop while altitude > 0 metres
   - pause

   End Loop
3. Stop Jet

**More Functions**

- When you have got this program to work, add another function, called **FlyToMe()**
  - This should detect you (the astronaut) by using the radar with category **Me** and then fly towards you before flying to the island
- Then add a fifth function called **BuzzMe()** that will give you a scare:
  - This should descend slowly to about 5 metres above you (see hint below)

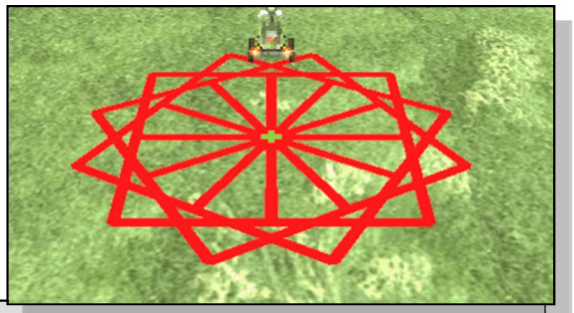
**Hint for BuzzMe()**

the **z-coordinate** is the height above sea level, so

- **this.position.z** is the z coordinate of the robot
- **item.position.z** is the z coordinate of any object (item) detected by radar.

## 6 Ceebot Task 18.4: Flower Power

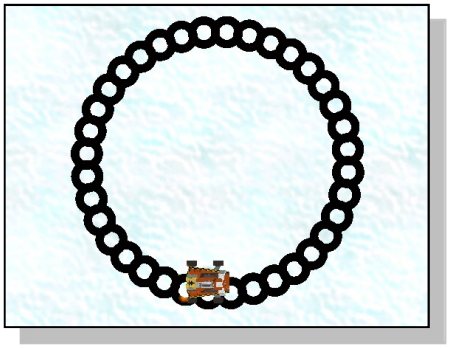
Call a **Square()** function (see 18.1) inside a loop to draw a flower (sort of!)
Press **[F1]** to see instructions for this.



# Week 5: Independent Study (3 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

# 7 Ceebot Task 19.3: One More Flower



Use a function **Circle** that draws one small circle and use it to produce a circle of petals like the one shown here. Press **[F1]** for details

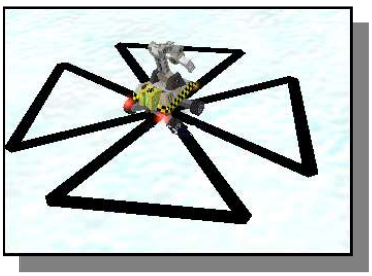
**Your task:**

1. First design the Circle function to draw one circle
2. Then use a loop in your main program to call the function in a circular way

**Extra**

- Try to get the program to draw alternate red and yellow circles for the petals.
- Can you design a better petal shape (see **19.5**)

# 8 Ceebot Task 19.4: Triangle Function



You are to draw a Maltese Cross, using a Triangle function

**Your task:**

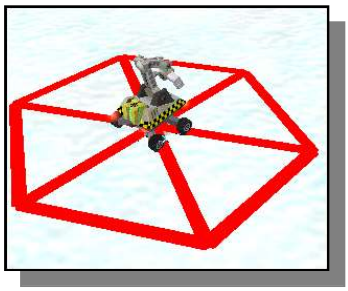
1. You should start by creating 2 functions:

   **MoveToCentre()**
   This function should move the robot approximately to the centre of the floor space.
   The robot should end up facing in the same direction as it started.

   **Triangle()**
   This should use a for loop to draw **one** equilateral triangle with sides 5 metres long.
2. You should then test your 2 functions by calling them from your main program.
3. Now modify your main program so it uses a loop to call the Triangle() function 4 times in order to draw a Maltese Cross:

**Extra**

- Copy your program into another editor slot and modify it to draw a **Hexagon**:



**Put commented source codes and screenshots into the log book**

# 9 Ceebot Task 18.7: Lurking Ants

There are 5 AlienAnts lurking on some distant islands. You must program a flying shooter robot to destroy them using a function called

**DestroyOneAnt**

**Your task:**

Write a function to destroy one ant. Then use a loop in the main program to call it 5 times. The function should:

- Use radar to detect an AlienAnt
- Fly up or down to reach the same height as the ant (use **z-coordinates**, see above)
- Turn towards the ant
- Move close (say within 10 metres)
- Fire to destroy the ant

**Put commented source code into the logbook**