Ceebot Programing

bucks new university

Programming Concepts CO452

Study Pack for Ceebot

Part B
Weeks 5-8

Original work by Brian Ward

Week 6
Functions and Parameters

The Technical Bit

Parameters using Ceebot

Parameters are used to make functions more flexible. Using **parameters** allows you to pass **values** into a function that can then be used by that function. The values are passed in by using the **brackets** as a kind of letterbox.
Functions can also be used to **return** a value back.

e.g. Here we are defining a function called **CalcAverage()** which has 2 parameters and returns a float result .. the average of the 2 numbers passed in.

```
extern void object::MainProgram()
{
        float  num1=5;
        float  num2=12;
        float  avg;

      avg = CalcAverage(num1, num2) ; // call CalcAverage function

      message("The average of the numbers is " + avg);
}
//*************************************************************************
float object::CalcAverage(float a, float b)   // define the  function
{
      float result;
      result = (a + b)/2;     // work out average
      return result;          // return value back
}
```

Now we have produced a function, **CalcAverage()** that will work out the average of **any** 2 numbers and we could use it in other programs too.

1 Ceebot Task 20.1: Better Square Function

Design a better Square() function with one **parameter** for the side length, so that it can be used to move the robot around squares of different sizes.

Your task:

- You have to move the robot around a **15 metre** square, followed immediately by a **25 metre** square. You must use **one** function to complete **both** squares.
- First of all design the function with a single parameter for the side length. You should put the function underneath the main part of the program (see below)

```
extern void object::Task20_1()
{
      //   xxx  call the function here
}
//**************************
void object::Square( float side )
{
       // put code here to move around a square using side
}
```

- Put the code inside the function to move the robot through a complete square. Use a **for loop** and remember to use **side** which holds the value of the length to be used.
- Now in the main program (at XXX above) you need to put the instructions that will move the robot around the track .. a **15 metre** square, then a **25 metre** square.
- Your solution should have **one** Square() function, used **twice** in the main program.

2 Ceebot Task 20.3: Scaleable Rectangles

Your task(1):

- Design a **DrawRectangle()** function with 2 **parameters** for the height and width, so that it can be used to draw rectangles of any size.
- Next design a main program to test your function by using the **dialog()** instruction to enter a width and height.
- Test your program with by drawing a red rectangle with a width of 15 metres and a height of 10 metres. Then see **task(2)** below.

DrawRectangle(…): Algorithm

1. Put the pen down
2. **loop** 2 times
   a. move the width passed
   b. turn 90 degrees
   c. move the height passed
   d. turn 90 degrees
   end loop
3. lift pen up

Your task(2):

- Improve your test program by using a **do .. while** loop to allow you to run several tests during one run of the program.
- At the end of each test, you should ask: "Any More?" and repeat while the answer is not equal to "n".
- See the partial algorithm here:

```
Main test program
  1. do loop
        a. enter width
        b. enter height
        c. convert values
        d. DrawRectangle
        e. Any More?
     while answer not "N"
  2. output "Finished"
```

3 Ceebot Task 20.7: Calculator Function

Use a **GetNum()** function to input and return a float number instead of a string. Then use it to input loads of numbers .. and total and average them.

The GetNum() function

- It is a nuisance to have to keep using dialog() and strval() to input numbers, so we shall design a function to do this. Here it is:

```
float object::GetNum(string prompt)
{
      string input;
      float num;
      input = dialog(prompt);
      num = strval(input);
      return num;
}
```

- Notice that instead of **void**, the function starts with **float** .. this shows that a float number is being returned (**void** means nothing is returned)
- You can use the function like this:
  num = GetNum("Enter a Number");
- Now include this function after your main program and add comments to explain how the function works

Your task

- After you have added the **GetNum()** function, you need to design a main program to use it to do the following:
  - Repeatedly input numbers until a negative number is input
  - Keep a running **count** and **total**
  - Display the **total**, **count** and **average** of the numbers
- An algorithm is included here to help
- Get the program to work, then modify it so the input prompt is :
  Enter Number 1
  Enter Number 2 etc.
- Then **test** the program .. see next section

```
1. set count to zero
2. set total and average to zero
3. input first number using GetNum
4. loop while number >= 0
     a. add number to total
     b. add 1 to count
     c. input next number using GetNum
   end loop
5. if count > 0
     average = total/count
   end if
6. Display count, total, average
```

Testing the Calculator Program

A **Test Plan** has been partially completed for you (see below).

- Run the program 3 times using the input data for the 3 existing tests and fill in the results.
- Then devise two more tests, work out the expected values and complete the testing for these.

Calculator
Test Plan

Test No.	Input Numbers				Expected			Actual		
					Count	Total	Average	Count	Total	Average
1	12.5	23.4	-1		2.00	35.90	17.95			
2	-1				0.00	0.00	0.00			
3	0	75.6	12	-1	3.00	87.60	29.20			
4										
5										

4 Ceebot Task 20.8: Flexible Square Function

Use the **GetNum()** function from the previous exercise to input the square size and colour (a number from 1-15) .. then draw a Square using these 2 parameters

Your task

- Copy over the **GetNum()** function from the previous exercise so you can use it here.
- Design a new **Square()** function with 2 parameters:
  - the **size** of the square
  - the **colour** (a number from 1 to 15)
- Design the function to use these 2 parameters.
- Note: the **pencolor(…)** instruction uses a number to set the colour … look up the details and use it in your Square() function
- Now get your main program to use **GetNum()** to input a size and pencolor. Then call your **Square()** function, passing in the 2 values.

Validation

- If the square **size** is over 15 metres you will hit the barriers
- **Colour** values can only be from 1 to 15
- Use **do .. while loops** in your main program to validate the 2 inputs so that only values in these ranges are allowed .. an **error message** should be produced for values outside the range

Week 6: Independent Study (4 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

5 Ceebot Task 21.6: The FlyTo Function

In this exercise there is a **WingedGrabber** robot that must fetch various items from nearby islands and drop them at the coloured flags.

What the robot needs to do is :

Drop the **Titanium**	at the **RedFlag**
Drop the **TitaniumOre**	at the **YellowFlag**
Drop the **FuelCell**	at the **BlueFlag**
Drop the **PowerCell**	at the **GreenFlag**

You must complete this exercise using a **function** (see below)

Your Task 1

Design a function that can fly off to **any** object category (e.g. Titanium, RedFlag etc.)

- the function may be called **FlyTo()** and should have one integer parameter for the category of the object that you want to fly to
- inside the function, do the following:
  - use the robot's radar to detect the object (use the parameter here)
  - display a message saying "I'm off to the ……" (use **item.category** here)
  - use goto() to travel to the object
- Now you can use your new FlyTo() function in the main program like this:
  FlyTo(Titanium);
  FlyTo(RedFlag); etc.

Your Task 2

Now you need to get the FlyTo() function to either **grab()** or **drop()** depending on what the object is.

- Hint: the y-coordinates of all the grabbable objects are more than zero
- The y-coordinates of all the flags are less or equal to zero
- **Item.position.y** gives you the y-coordinate of an object (after using radar)

6 Ceebot Task 21.6: The FlyTo Function (Extra)

- Add code to the function so that it displays additional messages .
  "I'm taking the …. to the …." at the appropriate time
- Example

  "I'm off to the Titanium"
  Then later: "I'm taking the Titanium to the RedFlag"
  "Titanium now deposited at the RedFlag"

Hints: When a robot is not carrying anything, its **load** is **null**
What a robot is carrying is described by **this.load.category**

7 Ceebot Task 21.8: Arithmetic Quiz

You will find a program already waiting in slot 1 for this exercise. If you run it you will see that it asks you a simple maths question and waits for your answer.
If your answer is correct you get a "Well Done" message. Look at the program code carefully and examine the 2 functions **askQuestion()** and **checkAnswer()**.
Make sure you understand how the program generates 2 random numbers and then passes them as parameters into these functions.

Before you Start

Copy the Quiz program from slot 1 into slot 2 and work on this slot 2 version
(you will lose any slot 1 work because it is overwritten when you reset)

Your Tasks

1. Modify the program so that multiplications are done with numbers up to 20. Also add some code to the checkAnswer() function so that if the answer is wrong a suitable message is displayed:

Sorry that's wrong! ... times ... is ...

2. At the moment the program only asks one random question! Add some more code so the program repeats by asking the user if they wish to continue.
The program should stop when they say no.
Also modify **askQuestion()** so a Question number is displayed as well as the question itself.

3. Add a new function to the program, called **showResults()** which displays the number of correct answers in the form:

Final Results

==========

You got ... correct answers out of ...

Hint: modify **checkAnswer()** so the score is also passed into it and an updated score returned.back.

8 Ceebot Task 18.3: Alternating Colours

Your task is to recreate the picture below. You should use functions and parameter passing where appropriate. The colours of the circles must alternate.

Hint: see the slides on modulus from this week's presentation if you get stuck