# Parameter PAssing in Modular Code

## Visit <http://www.tinyurl.com/GlowCompSci> click on the Higher Folder in Documents and then Parameter Passing. You will need your Glow Login and Password.

## Overview

This activity allows students to gain an understanding of parameters and parameter passing techniques**.**This is an introductory lesson. The focus of the lesson is to introduce passing parameters; at this stage, topics such as difference between functions and procedures are not considered.

## Suitable for

Higher Computing Science

## Key concepts

Sub-programs/routines, defined by their name and arguments (inputs and outputs) and parameter passing (value and reference) and scope- local and global variables.

## Learning outcomes

* Understand the difference between a non-modular and a modular program.
* Understand what a parameter is and how they help us to create modular programs.
* Understand different types of data flow that a sub-program can have.
* Understand the difference between passing a parameter by reference and by value.
* Understand what happens to a local variable when the subprogram it’s created in finishes.

## Success criteria

* I can describe what a parameter is and how it is used to create modular programs.
* I am able to explain how pieces of data are passed into and out of a subprogram.
* When given a problem description and a set of main steps I can identify the pieces of data needed by each subprogram and whether they should be passed in, out or in/out.
* I can explain the difference between passing a parameter by value and passing it by reference.
* When given a modular program I can identify when a local variable will be created and destroyed.

## Time required

1 single period

## Preparation

1. Print 1 set of the pupil handouts preferably on card so they can be reused.
2. Separate these into four piles with one sheet of instructions, some blank paper and some marker pens per pile. For the main program sheet you should include five long pieces of string that can stretch between the main program table and any other table and some pieces of blue tack.
3. Make sure the class is arranged into four groups at different tables or areas.
4. Place the bin within easy reach of the tables.

## Prior learning assumed

Pupils should have a basic understanding of modular code, what scope means and the scope of local and global variables.

# Outline of Activity

* Students will work in one of four groups and will simulate the flow of data between different sub-programs in a program to calculate the area of a rectangle. Each subprogram will be completely self-contained with no global variables used.

## Introduction

Outline the learning intentions and success criteria for this activity and mention that this will give them a good understanding of why we need parameters and how parameter passing works.

**4 mins**

**8 mins**

Introduce the programming problem to the class: in this example, a program is to be written to enter the length and breadth of a rectangle. The program should calculate the area of the rectangle before displaying its dimensions and area on the screen.

Demonstrate the working program but don’t show the code for it so they can see the problem the program is solving. In groups pupils discuss what they think the main steps are in this program.

Lead a class discussion on the main steps and write these on the whiteboard so that each step can be “ticked off” as it’s completed later.

|  |
| --- |
| **Main Sub-program** |
| 1. **Declare variables** 2. **Enter dimensions of rectangle *(Enter\_Details)*** 3. **Calculate area of rectangle *(Calculate\_Area)*** 4. **Display dimensions and area of rectangle *(Display\_Results)*** |

Allocate a different subprogram to each group by handing out one pile to each table trying to keep the main sub-program group near the middle so that it’s fairly close to all of the other groups.

**Parameter Passing**

**4 mins**

**Question-** In their groups discuss “If all variables are local variables, is it possible for another subprogram to see, share and update the data stored in memory?”

**Answer-** No it’s not possible as they can only be accessed in the subprogram they were created in.

Introduce the need for parameters as a way to pass data between different sub-programs.

**Question-** In their groups discuss “What pieces of data will be needed by more than one sub-program?”

**Answer-** Rectangle Length, Breadth and Area

## Main Activity (20 mins)

You will run the program which will involve the groups helping to execute parts of the program. Ask the class “What will be the first sub-program that executes? The class should say the main sub-program.

**Step 1– Main subprogram declares variables**

**Question-** Ask the main subprogram group “What happens next?”

**Answer-**The three variables Rect\_length, Rect\_breadth and Rect\_area are created.

The main sub-program 'creates’ these variables by filling in the blank variable cards with a name and an initial value.

Tick off the declare variablesstep on the whiteboard.

**Question-** Ask the class “What happens next?”

**Answer-**The main sub-program calls the Enter\_Details sub-program.

**Step 2- Enter\_Details subprogram**

Group discussions to try to answer the following questions.

* **Question-** What pieces of data will the Enter\_Details subprogram need?

**Answer-** Rect\_Length, Rect\_Breadth

* **Question-** Will an updated value need to be passed back out or does the subprogram just need to read the value?

**Answer-** An updated value needs to be passed out.

The main module will now attach one end of a string to the local Rect\_Length and Rect\_Breadth variables and then pass the other end to the Enter\_Details subprogram group. They will attach this to the value part of the Length and Breadth parameter cards.

When Enter\_Details carry out their instructions they will follow the string to update the original variable when they receive a SET.. TO instruction.

Enter detailsis now complete*.*Tick off Enter\_Detailsstep on the whiteboard.

**Question-** Ask the class “What happens next?”

**Answer-**Length and Breadth parameters are deleted and the main sub-program calls the Calculate\_Area subprogram*.*

## **Step 3- Calculate\_Area subprogram**

Group discussions to try to answer the following questions.

* **Question-** What pieces of data will the Calculate\_Area subprogram need?

**Answer-** Rect\_Length, Rect\_Breadth, Rect\_Area

* **Question-** Will an updated value need to be passed back out or does the subprogram just need to read the value?

**Answer-** Rect\_Length and Rect\_Breadth won’t be changed but Rect\_Area will be updated.

The main module will get the two parameter cards length and breadth from the Calculate\_Area group and copy the values onto them and pass them back. They’ll then attach one end of a string to the local Rect\_Area variable and then pass the other end to the Calculate\_Area subprogram group. They will attach this to the value part of the Area parameter card.

When Calculate\_Area carries out their instructions they will follow the string to update the original variable when they receive a SET.. TO instruction.

Calculate*\_*Area is now complete. Tick off Calculate\_Areastep on the whiteboard.

**Question-** Ask the class “What happens next?”

**Answer-** Length, Breadth and Area parameters are deleted and the main sub-program calls the Display Results subprogram.

**Step 4- Display\_Results subprogram**

Group discussions to try to answer the following questions.

* **Question-** What pieces of data will the Display\_Results subprogram need?

**Answer-** Rect\_Length, Rect\_Breadth, Rect\_Area

* **Question-** Will an updated value need to be passed back out or does the subprogram just need to read the value?

**Answer-** Rect\_Length and Rect\_Breadth and Rect\_Area values will be read but not changed.

The main module will get three parameter cards Length, Breadth and Area from the Display\_Results group and copy the values from Rect\_Length, Rect\_Breadth and Rect\_Area onto them and pass them back.

Display\_Results will carry out its instructions.

Display\_Results is now complete. Tick off Display\_Resultsstep on the whiteboard.

**Question-** Ask the class “What happens next?”

**Answer-** Length, Breadth and Area parameters are deleted and we return to the Main sub-program.

**Step 5- Program terminates**

**Question-** Ask the class “What happens next?”

**Answer-** There are no more instructions so the program terminates

**Question-** What happens to the variables in the main sub-program?

**Answer-** Rect\_Length, Rect\_Breadth and Rect\_Area are deleted

## Review

**Question-** What were the parameters and dataflow for each subprogram?

**4 mins**

**Answer-** 2. Enter\_Details(out: Length, out: Breadth)

3. Calculate\_Area(in: Length, in: Breadth, out: Area)

4. Display\_Results(in: Length, in: Breadth, in: Area)

**Question-** Do parameters work in a similar way to variables or are they different?

**Answer-** The work in a similar way as they are named locations that store a value. The only difference is they are declared and initialised automatically when a sub-program is called.

**Question-** What happens to a local variable when the sub-program it was created in finishes?

Students should write down in their own words a definition for the terms modular program, parameter and what the terms pass by value and pass by reference mean. They should share their definitions with their group in order to come up with the best definition. The teacher will ask one person from each group to share this with the class.

**10 mins**

# Suggested follow up work

* Show the implementation of the code within your preferred language, and illustrate differences between the conceptual model and the actual implementation. A potential follow-up discussion here is that different languages deal with discarded data in different ways.
* Pupils could take a non-modular program they’ve worked with in the past and rewrite it so that there is a main program and then subprograms with parameters that each carry out one step.
* Pupils could study example problems and decide for each step what data is required and whether it needs to flow into, out of or into and out of each step.

# Acknowledgements and Copyright

This idea and associated resources were developed by

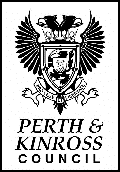


Morag Shaw- Clydeview Academy in Inverclyde

David Stott- Kirkland High School & Community College in Fife

Colin Chisholm- Morgan Academy in Dundee City

Dr Mark Zarb- Robert Gordon University in Aberdeen City

With final refinements, improvements and formatting by

Peter Donaldson- Crieff High School in Perth and Kinross

During a Craft the Curriculum for Higher Computing Science event

A joint event held by PLAN C and Education Scotland







© Crown copyright 2015. You may re-use this information (excluding logos) free of charge in any format or medium, under the terms of the [Open Government Licence](http://www.nationalarchives.gov.uk/doc/open-government-licence/).

Where we have identified any third party copyright information you will need to obtain permission from the copyright holders concerned.