GMIT
INSTITIÚID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

# Higher Diploma in Computer Science with Data Analytics

## Multi-Paradigm Programming

## Shop Assignment (70%)

**Lecturer: Dominic Carr**

**Damien Connolly**

**G00340321**

**16/12/2020**

## INTRODUCTION

The object of this report is to compare the shop programs which were created using two different programming paradigms within two different languages, C and Python.

A programming paradigm refers to a style of programming. It does not refer to a specific language, but rather it refers to the way in which a problem is approached and affects how the code is written and structured [1].

In this project we will be discussing the procedural paradigm created using C and Python and the object orientated paradigm also created in Python.

## PROCEDURAL PROGRAMMING

Procedural programming, also known as inline programming, takes a top down approach. It is derived from the imperative approach, but it has extra functionality that allows for procedural calls to be made. It is typically a list of instructions that tell the computer what to do by executing one after the other. It directly tells the computer how to carry out the task using these specific steps and will write procedures or methods to perform operations on the data. The data and methods are two separate entities [2].

Procedures, also known as functions, simply consist of a series of computational steps to be carried out. During a program's execution any given procedure can be called at any time, this allows for code and functions to be reused. Based on the concept of a procedure call, procedural programming divides the program into procedures, which are also known as routines, sub-routines or functions, that simply contain a series of steps to be carried out [3].

Procedural programming is very similar to how you would expect a computer to work, by providing it with step by step instructions on how to complete the given task.

## OBEJECT – ORIENTATED PROGRAMMING

Object-orientated programming is a programming paradigm that is based on the concept of objects. Objects can contain data in the form of attributes and code in the form of methods, meaning they contain both state and functionality. Object-orientated programming models real world entities as software objects that have some data and can perform certain functions [4]. Using the example of a person as an object, a name would be a property of the object while walking would be a method within the object.

A method in object-orientated programming is like a procedure or function in procedural programming. In object-orientated programming you begin your code by creating objects and then by giving those objects properties and adding methods [2]. Objects can communicate with each other by means of message passing, whereby a method can be called by another object. Another aspect of object-orientated programming is the use of classes.

A class of an object can be thought of as its template and an object held within the class then becomes an instance of that class. Object-orientated programming uses many techniques not available in procedural programming by using encapsulation, abstraction, inheritance and polymorph. These are seen as the fundamentals of object-orientated programming.

Encapsulation is one of the fundamentals of OOP. It is used to hide the values or state of a structured data object inside a class preventing unauthorised access. Encapsulation is achieved when each object keeps its state private inside a class [5].

Abstraction is one of the key concepts of OOP**.** Its main goal is to handle complexity by hiding unnecessary details from the user. Abstraction has to do with displaying only the relevant aspect to the user and aims to ensure simplicity [5].

Inheritance defines the relationship between classes or objects. They can interact with each other using inheritance. The properties of a class can be inherited and extended by other classes or functions. The superclass is inherited from while the subclass is the inheritor as it receives the properties of the superclass. Inheritance is a major pillar in Object-Oriented programming and helps to ensure that code is reused [6].

Polymorphism means existing in many forms and variables, functions, and objects can exist in multiple forms. With polymorphism, a method or subclass can define its behaviours and attributes while retaining some of the functionality of its parent class. This means you can have a class that displays date and time, and then you can create a method to inherit the class but should display a welcome message alongside the date and time. The goals of Polymorphism in object-oriented programming are to enforce simplicity, making codes more extendable and easily maintaining applications [6].

The main idea behind object-oriented programming is simplicity, code reusability, extendibility, and security and these are achieved through encapsulation, abstraction, inheritance, and polymorphism.

## SIMILARITIES

There are some similarities between procedural programming and object-orientated programming:

- Both paradigms operate by mutating their state, either inside a data structure in a procedural style or an object in an object-orientated style
- They both provide step by step instructions on how to return an output.
- Data structures (procedural) and objects (OOP) are both used for storing data.
- In both paradigms the code is separated into smaller blocks, functions in procedural programming and objects in object-orientated programming.
- Methods and functions can both be called and reused within the two paradigms.

- Both paradigms can be created using different programming languages although procedural is more common in older or low-level languages.

## DIFFERENCES

There are many differences between procedural programming and object-orientated programming:

- Procedural programming follows a top down approach while object-orientated programming follows a bottom up approach
- In procedural programming data and functions are two separate concepts while in object-orientated programming these two concepts are held together in objects
- Procedural programming is good for general purpose programming and on shorter programs while object-orientated programming allows you to develop large, modular programs that can instantly expand over time and are easier to maintain due to their structure
- Due to encapsulation, abstraction, inheritance, and polymorphism there is better security, flexibility and cooperation available in object-orientated programming
- There is no access specifier in procedural programming while object-orientated programming has access specifiers like private, public, protected etc.
- Object orientated programming provides data hiding through encapsulation making it more secure than procedural programming which does not have any way for hiding data
- Procedural programming handles selection using the familiar if/else syntax while objects prefer to handle selection using polymorphism
- In procedural programming function is more important than data while in object-orientated programming data is more important than function. This can be seen by the way in which the programs are created
- Object-orientated programming is based on real world while procedural programming is based on unreal world

## <u>CONCLUSION</u>

This project was interesting to be able to create the shop program in two different paradigms. Coding the project in C I found to be time consuming and challenging, but I believe this made the process easier to complete in Python for the procedural and object-orientated style.

There are many different paradigms that can be used, selecting the right paradigm is about selecting the paradigm that is best suited for a program. It is clear to see how the procedural style, especially in C, can begin to look confusing as the program grows. Making an edit to the code can have a knock-on effect within the rest of the program so it may take a while to do so. The procedural paradigm does seem slightly easier to follow as it has a top down approach, but the object-orientated approach does allow for easier edits to be made and objects created or added. This would be very important when working in a collaborative environment.

After using both paradigms for this project it would seem the procedural approach, although easy to follow, is best suited for smaller programs while an object-orientated style would suit bigger programs that may need to keep expanding over time.

Through its use of encapsulation, abstraction, inheritance, and polymorphism, object-orientated programming offers better code reusability, extendibility, security and simplicity than procedural programming. OOP's error handling is also superior to that of procedural programming, especially when using C as it does not have a built-in mechanism for dealing with this, so errors can become time consuming.

From this project I believe I have gained great experience within C and Python. Creating the project in C (first time using the language) was an important experience as it gave me the opportunity to use a low-level language and helped me with creating the project in Python. There is more to the code within C, but the ideas are both the same and within the procedural paradigm both are very similar.

I found the object-orientated approach a bit harder to fully understand at first, therefore I decided to write the program in Python rather than trying a new language by using Java. I can see why the object-orientated approach is so popular. This style is suited to bigger projects with more collaboration and provides easier options for growth or edits to be made. I am sure I will become a lot more familiar with this paradigm in the future.

Overall, this project has given me important knowledge of two different paradigms across two different languages. Although at times I found this project difficult and time consuming (especially C) I believe I have learned a lot from it, and it will be of huge benefit to me in the future.

## References:

1. https://www.freecodecamp.org/news/what-exactly-is-a-programming-paradigm/
2. https://study.com/academy/lesson/object-oriented-programming-vs-procedural-programming.html
3. https://hackr.io/blog/procedural-programming
4. https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/
5. https://press.rebus.community/programmingfundamentals/chapter/encapsulation/
6. https://www.nerd.vision/post/polymorphism-encapsulation-data-abstraction-and-inheritance-in-object-oriented-programming