**Multi-Paradigm Programming – Shop Assignment Report**

**Student: Niamh O’Leary**

**ID number: G00376339**

**Introduction**

The assignment for the Multi-Paradigm Programming module focused on producing a model of a shop which could be applied to C, Java and Python, with the aim of investigating how different languages deal with performing similar tasks. “A programming paradigm is the concept by which the methodology of a programming language adheres to. Paradigms are important because they define a programming language and how it works.”[[1]](#footnote-1) There are lots of programming languages that are well-known but all of them need to follow some strategy when they are implemented. And that strategy is a paradigm.

Types of programming paradigms

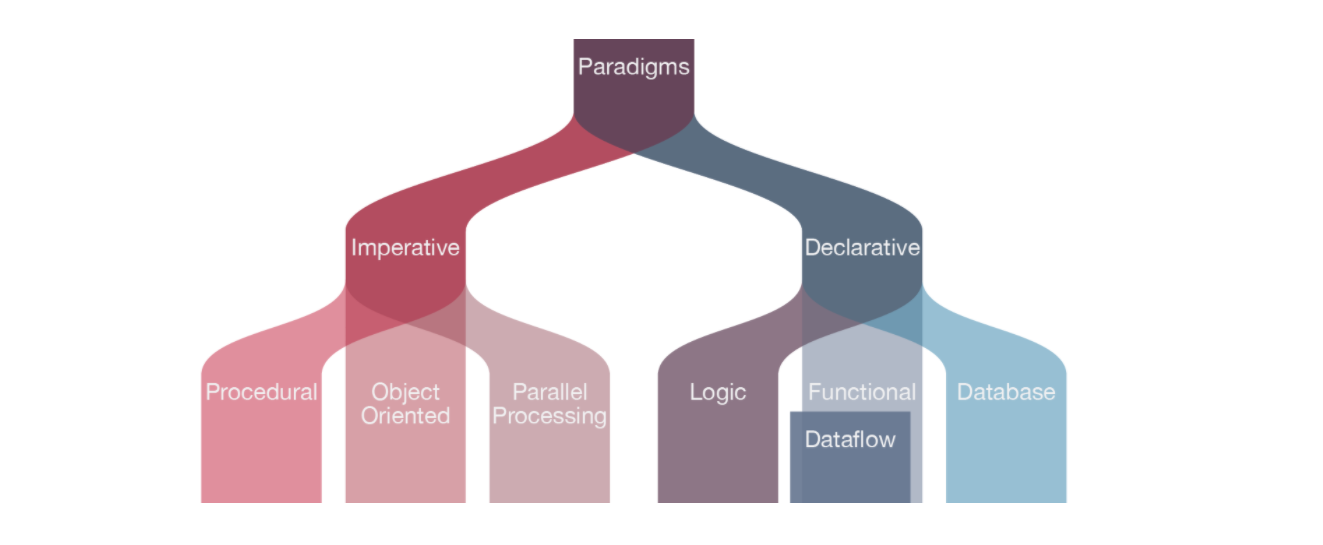


Figure 1 Types of Programming Languages [[2]](#footnote-2)

**Imperative**

“Imperative programming defines the solution to a problem as a series of steps—first do this, then do that, then do the next thing, and so on. The computer steps through each line of code, executing it and moving on to the next step.” [[3]](#footnote-3) Imperative programming explicitly tells the computer "how" to accomplish a task and move onto the next step. Programs written in the imperative style often resemble recipes—first crack the eggs, then mix in the flour, then add water. Imperative programs often change the state of the program on each line, assigning new variables and referring to or changing old ones. Though intuitive for solving small problems, imperative programs quickly become unmanageable as they become larger.

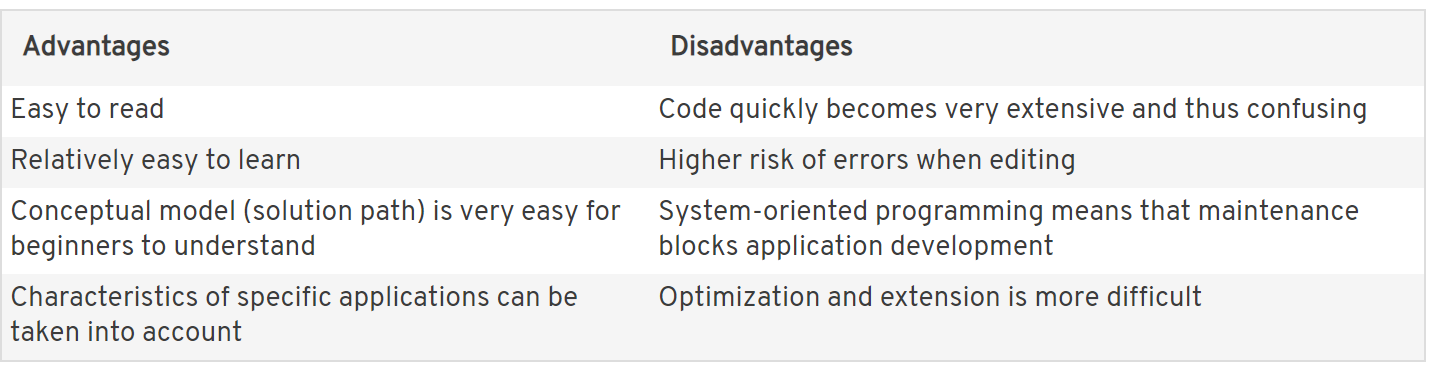


Table 1 Advantages and Disadvantages of Imperative Programming Languages [[4]](#footnote-4)

Imperative programming is divided into three broad categories: Procedural, OOP and parallel processing.

**Procedural**

In procedural programming, programs are made up of procedures, also known as routines, subroutines, or functions. In this topic procedures will be referred to as subroutines. The term procedure is conventionally used to classify subroutines that do not return any value(s), whereas a function always has a return statement.[[5]](#footnote-5)

The procedural paradigm originates from the programming of imperatives. PPP, contrasted with the imperative. The paradigm offers additional enhancement that makes function calls possible. It is possible to interpret functions (also known as procedures) as sets of instructions. Idea and a definition Good practice behind tasks is that they specialize in a single task and do nothing. Calls are used to communicate between the functions. The functions may be renamed, as necessary.

**OOP (Object Oriented Programming)**

The program is written as a collection of classes and object which are meant for communication. The smallest and basic entity is object and all kind of computation is performed on the objects only. More emphasis is on data rather procedure. It can handle almost all kind of real life problems.[[6]](#footnote-6)

OOP takes advantage of many principles, such as inheritance, polymorphism, encapsulation and abstraction, which are not applicable to the procedural model. Inheritance describes the hierarchical, family-tree-like structure of the relationship between those objects. After the parent class, children's classes pick up all properties and all features. The two paradigms allow the code to be split into smaller blocks (procedural and object functions in OOP), that carry out a specific task.

In order to monitor program flow, the code blocks can be evoked as required. Between functions and between objects, cross-reference is possible. In both of these paradigms, the

Data structures and functions are separate entities in procedural paradigms, while data structures and functions are separate entities in OOP data.

Structures and approaches are merged into a single object-entity.

OOP provides additional features through inheritance, encapsulation, polymorphism and principles of inheritance.

Abstraction. Abstraction In the procedural model, no equivalent exists. The effects of these are

Concepts (there is greater scalability, sustainability, cooperativity, stability but security for OOP like

The sophistication of the code as well).

Advantages:

Data security

Inheritance

Code reusability

Flexible and abstraction is also present

**Parallel**

Shop

Project Layout & Code Maintainability

For the procedural paradigm the project layout was quite straightforward. The entire project consists of one program script (shop.c), a csv file for the shop and stock information and a few test orders used to test the functionality of the code. The script was written from start to finish, with care taken to ensure that any procedures required were defined first in the file before they are called – otherwise the program won’t compile. The layout of the object oriented project was much more complex. Each class required its own file, this needed to be named exactly the same as the class name. However the methods in each class can be written and called in any order without affecting the compilation of the program. Even though the object oriented layout is a bit more complex – it is much easier to navigate when writing code. For example methods relating to the Customer are in the customer file and are very easy to locate and update when maintaining the code, whereas the functions in the C program don’t explicitly relate to any one data structure but must be located in the file in such a way that they are defined before they are called. It can be difficult to find them and update them when maintaining the code.

Data Types In Java everything (except primitive data types – ints, doubles, chars etc.) are modeled as objects. This means that it is possible to create methods to interact with them. The benefit of this is that there is some built in functionality in Java using these methods. In C there are no objects, so instead structs are used to model data types. They are similar to classes in that they define something that has many attributes (like a shop or a customer), but they don’t have dedicated methods for performing dedicated operations on the fields in that data.

Creating New Items

When using the object oriented approach a special method is created for each class called a constructor. This defines how to create an instance of the class and set iinitial values for the attributes. So in the shop program we created a constructor for the shop class that took a csv filename as a parameter and then read in the initial values for the object from the file. We also created a similar one in the customer class that was used for creating customers by reading data in from a csv file. One of the benefits to using an object oriented approach is that we can create many different constructors for the same class that are called depending on the data passed as a parameter. This process is known as overloading. Overloading was used in our program to create customer objects in a different manner depending on whether or not we were processing a csv file or live order. As a privacy conscious individual, I don’t believe that a shop needs to know a customers name or budget when they go into a shop to buy items. They just need to be able to process the shopping list and determine the total cost. In Java this was a trivial problem to solve by using an extra constructor, however in the C program a new struct (liveOrder) and 2 new functions (createLiveOrder and processLiveOrder) were required. Overall this was much easier to accomplish using the object oriented approach. Another issue encountered regarding creating new items was related to the way strings are handled in C. As stated above there are no strings in C, instead there are ‘character arrays’’which are lower-level are require the programmer to manually allocate memory of sufficient length for the character array. This is quite challenging to do dynamically as it requires clever algorithms anytime a user is going to enter a string to ensure that enough memory will be allocated to that string. This was also easier to accomplish in Java due to having both native String data types and automatic memory allocation.

Updating Items In C, in order to update the fields of an instance of a struct, we need to create a function to do this and pass pointers to the memory location of the particular struct to the function, so that the method can be update. This was done in the processOrder function where pointers to a customer and shop are passed so they can be updated, and also processLiveOrder where a pointer to a shop is passed. This is a little bit tricky as it is very ‘computer focused’ - taking the programmer back into the memory of the computer. In the Java program, the objects are modeled on ‘real world’ phenomenon such as customers and shops, and there are methods defined to both fetch values from the objects, and also to update values in them while abstracting the memory location away from the programmer. For this reason I believe that the object oriented approach is more intuitive for even problems of reasonable complexity such are our shop example

## Object-Oriented Programming

When creating the shop in Java we use object-oriented programming. Object-oriented Programming uses classes and objects. This language model is based on the idea that programs will be organized around data, or [objects](https://searchmicroservices.techtarget.com/definition/object), as opposed to functions and logic. Programmers will identify what objects they need to manipulate and how they relate to each other ([Data Modelling](https://searchdatamanagement.techtarget.com/definition/data-modeling)). Object-oriented Programming is much easier to modify and maintain. The ease of development and efficiency of the language make it a better choice than a procedural language.

When creating the shop in java each class has its own file which makes it easier to pull information and update method or information relating to that class. This is great benefit as I can jump from different classed generating getter, setters and toStrings, so I can pull the information that I need into the main class. Each class has a constructor which initializes a newly created object and will be called when an object of the class is created. I didn’t realise until I had a quick google search that Java does not support any header file. Loops in Java for me were very straight forward as I was able to iterate an array by calling the array or using array.length (For Live Mode). I used various for loops to pull information from arrays that are checked and validated either from the user input of CSV file. The main issue that I seem to have come across was a FOR loop for the live shop. I created a Boolean variable and set it to false and when true is found it will accept the user input, but when run sometimes it works first time and other times the same input will have to be entered in a few times before the response is true.

## Procedural Program

When creating the shop in C we used a procedural programming approach and takes more of a top down approach to programming. The program takes a problem and breaks that problem into sub-procedures. These sub-procedures are kept broken down until the sub-procedure can be solved. This can be tricky when a developer needs to edit the program because when more and more changes made to the code it can becom very difficult to locate and edit all related elements.

When creating the shop in C the ability to use classes in separate files like in OOP is not used. All the code in within a single file and considering that my program was not huge I can see how quickly the language can get very untidy. In C at the beginning of the file we need to create a struct or structs, we also need to add stdio.h which is the header file that is used in C. Struct are like what a class is in Java and contain different information such as the shops products stuck will hold a char for name and a double for price. C also uses pointers that are used store and or manage the addresse of dynamically allocated blocks of memory. Loops in C I found more difficult as the index needed to be defined in the struct. Overall the C section of this assignment was the hardest for me as a lot of external research needed to be completed. I am happy that I completed this section as I have gained great knowledge of the Language

Summary

There are a number of programming paradigms that exist. Within the context of this project, three different

The methods reviewed were (C procedural, Python procedural, Python OOP). After the completion of

I may draw the following conclusions from the project.

Each paradigm has pros and cons, with a field of application that is preferable. For a given, a given

programming problem, the most suitable paradigm should be chosen based on the

Requirements, complexity level, resources available, etc.

Languages in both C and Python

References

1. Medium. 2020. What Is A Programming Paradigm?. [online] Available at: <https://towardsdatascience.com/what-is-a-programming-paradigm-1259362673c2> [Accessed 5 December 2020].
2. TypesnUses.com. 2020. Types And Differences Between Programming Languages. [online] Available at: <https://www.typesnuses.com/types-of-programming-languages-with-differences/> [Accessed 5 December 2020].
3. GC Digital Fellows. 2020. *An Introduction To Programming Paradigms*. [online] Available at: <https://digitalfellows.commons.gc.cuny.edu/2018/03/12/an-introduction-to-programming-paradigms/> [Accessed 5 December 2020].
4. Imperative Programming: Overview Of The Oldest Programming Paradigm. [online] IONOS Digitalguide. Available at: <https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/> [Accessed 5 December 2020].
5. Isaac Computer Science. 2020. Isaac Computer Science. [online] Available at: <https://isaaccomputerscience.org/topics/procedural\_and\_structured\_programming> [Accessed 5 December 2020].
6. GeeksforGeeks. 2020. *Introduction Of Programming Paradigms - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/introduction-of-programming-paradigms/> [Accessed 5 December 2020].

1. https://towardsdatascience.com/what-is-a-programming-paradigm-1259362673c2 [↑](#footnote-ref-1)
2. https://www.typesnuses.com/types-of-programming-languages-with-differences/ [↑](#footnote-ref-2)
3. https://digitalfellows.commons.gc.cuny.edu/2018/03/12/an-introduction-to-programming-paradigms/ [↑](#footnote-ref-3)
4. https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/ [↑](#footnote-ref-4)
5. https://www.geeksforgeeks.org/introduction-of-programming-paradigms/ [↑](#footnote-ref-5)
6. https://www.geeksforgeeks.org/introduction-of-programming-paradigms/ [↑](#footnote-ref-6)