Multi- Paradigm Programming

Katie O’Brien G00398250

This report aims to compares the solutions achieved using a procedural approach versus an Object-oriented approach to the project at hand.

In this project the developer is asked to create a shop program in both C programming language and Python, in the case of procedural programming, and in python for the Object-Oriented programming. We will compare the similarities and differences between the 2 approaches in programming.

C is a general-purpose computer programming language, created in the 1970’s by Dennis Richie at Bell Labs in the USA. It was designed to provide low-level access to memory and language constructs that can map efficiently to machine instructions with minimal runtime support; and as a result, C’s features are deliberately designed to reflect the capabilities of the targeted CPUs.

Because of this functionality, it has found lasting use in operating systems and device drivers; and is commonly used on computer architectures that range from the largest supercomputers to the smallest microcontrollers and embedded systems.

C is an imperative procedural language supporting structured programming, lexical variable scope, and recursion, with a static type system. Despite its low-level capabilities, the language was designed to encourage cross-platform programming and standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code. Because of this, and the added controllability over memory making it ideal for space-limited machines; C has consistently ranked in the top 2 languages in the TIOBE index, which measures the popularity of programming languages since the late 1980’s. (source: <https://www.tiobe.com/tiobe-index/>)

The TIOBE index ranks Python as the most popular programming language as of November 2022. It is a general-purpose language that supports multiple programming paradigms- most popularly procedural programming and Object-Oriented Programming.

Python was first released in the early 1990’s and is a high-level language that emphasizes code readability and comes with comprehensive standard libraries.

It is important to note that programming paradigms are not language- specific, and while some languages can support many types of paradigms, many, such as SQL which is declarative programming, are specific to just one. A **Programming paradigms** are a way to classify [programming languages](https://en.wikipedia.org/wiki/Programming_language) based on their features. Languages can be classified into multiple paradigms.

Some paradigms are concerned mainly with implications for the [execution model](https://en.wikipedia.org/wiki/Execution_model) of the language, such as allowing [side effects](https://en.wikipedia.org/wiki/Side_effect_(computer_science)), or whether the sequence of operations is defined by the execution model. Other paradigms are concerned mainly with the way that code is organized, such as grouping a code into units along with the state that is modified by the code. Yet others are concerned mainly with the style of syntax and grammar.

Common programming paradigms include:[[1]](https://en.wikipedia.org/wiki/Programming_paradigm#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Programming_paradigm#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Programming_paradigm#cite_note-symbolic-programming-lisp-3)

* [imperative](https://en.wikipedia.org/wiki/Imperative_programming) in which the programmer instructs the machine how to change its state,
  + [procedural](https://en.wikipedia.org/wiki/Procedural_programming) which groups instructions into procedures,
  + [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) which groups instructions with the part of the state they operate on,
* [declarative](https://en.wikipedia.org/wiki/Declarative_programming) in which the programmer merely declares properties of the desired result, but not how to compute it
  + [functional](https://en.wikipedia.org/wiki/Functional_programming) in which the desired result is declared as the value of a series of function applications,
  + [logic](https://en.wikipedia.org/wiki/Logic_programming) in which the desired result is declared as the answer to a question about a system of facts and rules,
  + [mathematical](https://en.wikipedia.org/wiki/Mathematical_programming) in which the desired result is declared as the solution of an optimization problem
  + [reactive](https://en.wikipedia.org/wiki/Reactive_programming) in which the desired result is declared with data streams and the propagation of change

[Symbolic](https://en.wikipedia.org/wiki/Symbolic_programming) techniques such as [reflection](https://en.wikipedia.org/wiki/Reflection_(computer_programming)), which allow the program to refer to itself, might also be considered as a programming paradigm. However, this is compatible with the major paradigms and thus is not a real paradigm in its own right.

For example, languages that fall into the **imperative paradigm** have two main features: they state the order in which operations occur, with constructs that explicitly control that order, and they allow side effects, in which state can be modified at one point in time, within one unit of code, and then later read at a different point in time inside a different unit of code. The communication between the units of code is not explicit. Meanwhile, in **object-oriented** programming, code is organized into [objects](https://en.wikipedia.org/wiki/Object_(programming)) that contain a state that is only modified by the code that is part of the object. Most object-oriented languages are also imperative languages. In contrast, languages that fit the **declarative paradigm** do not state the order in which to execute operations. Instead, they supply a number of available operations in the system, along with the conditions under which each is allowed to execute.[[4]](https://en.wikipedia.org/wiki/Programming_paradigm#cite_note-4) The implementation of the language's execution model tracks which operations are free to execute and chooses the order independently. More at [Comparison of multi-paradigm programming languages](https://en.wikipedia.org/wiki/Comparison_of_multi-paradigm_programming_languages).

**Procedural Programming**

Procedural programming is a programming paradigm that has evolved from imperative programming. It is based on the idea of procedures, or function, calls. Functions are a group of specific instructions that perform a specific task, packaged as a unit.

Fundamentally, the procedural code is the one that directly instructs a device on how to finish a task in logical steps. This paradigm uses a linear top-down approach and treats data and procedures as two different entities. Procedural Programming divides the program into procedures, which are also known as routines or functions, simply having a series of steps to be carried out.

Procedural programming is often likened to the idea of following a recipe. Using this concept, it becomes easy to explain the importance of a clearly defined set of rules when creating a program or script in procedural style. For example, when considering how to bake a cake, we can often use a certain amount of common sense when following a recipe. If, for example, a recipe calls for the user to “add flour” it would be implied that this is to a mixing bowl. With computers they can only do exactly what they are told so the instruction “add flour” would also have to include further details including: “to where”, “with what”, “how quickly”. Without this explicit instruction the task would be unable to be completed and this is no different with a computer program. Computer processors can provide hardware support for procedural programming through a stack register and instructions for calling procedures and returning from them.

Some of the key features of procedural programming are given below:

Predefined functions: A predefined function is typically an instruction called by name. Usually, the predefined functions are built into higher-level programming languages, but they are derived from libraries, or registries, rather than the program itself. One example of a pre-defined function is “charAt()”, which searches for a character position in a string.

Local Variable: A local variable is one declared in the main structure of a method and limited only to the local scope it is given. The local variable can only be used in the method it is defined in, and if it were to be used outside the defined method, the code will cease to work. It is better to not mix the names of variables being used for local and global variables.

Global Variable: A global variable is declared outside functions in the code. Due to this, global variables can be used in all functions, unlike a local variable.

Modularity: Modularity is when two dissimilar systems have two different tasks at hand but are grouped together to conclude a larger task first. Every group of systems then would have its own tasks finished one after the other until all tasks are complete.

Parameter Passing: Parameter Passing is a mechanism used to pass parameters to functions, subroutines or procedures. Parameter Passing can be done through ‘pass by value’, ‘pass by reference’, ‘pass by result’, ‘pass by value-result’ and ‘pass by the name’.

As with any programming paradigm, there are disadvantages and advantages to Procedural Programming. While excellent for general-purpose programming it can be harder to write in true Procedural. It is often not reusable from program to program, which can mean having to recreate the code if it is needed in another application; but can be reused in a single program without having to copy it. The source code is portable and can be used to target different CPU’s, along with ease of implementation of compilers and interpreters.  
Through Procedural Programming technique, the program flow can be tracked easily, however, the importance is given to the operation rather than the data, which might pose issues in some data-sensitive cases. Finally, it can be difficult to relate with real-world objects

Object-Oriented Programming

References:

* <https://en.wikipedia.org/wiki/C_(programming_language>)
* <https://en.wikipedia.org/wiki/Python_(programming_language>)
* <https://en.wikipedia.org/wiki/Procedural_programming>
* <https://hackr.io/blog/procedural-programming>
* <https://peps.python.org/pep-0008/>
* <https://en.wikipedia.org/wiki/Programming_paradigm>