HOW OBJECT-ORIENTED PROGRAMMING DEFEATED PROCEDURAL PROGRAMMING AND BECOME THE DOMINATOR IN INDUSTRY FIELD

Linh Dan Nguyen – 103488557

Faculty of Science, Engineering and Technology

Swinburne University of Technology

ABSTRACT

Currently, object-oriented programming and procedural programming are the two most popular paradigms. Procedural programming was thought to be the cornerstone of other programming paradigms before the invention of object-oriented programming (OOP). Due to many articles about programming paradigms, it is undeniable that nearly every developer used it at some point in their career. However, OOP is widely used in the industry instead of procedural programming. This research investigated the difference between object-oriented and structural programming to answer the question: Why do most developers prefer using OOP? Is OOP better than procedural programming in all aspects? If not, which are the strong points of each paradigm?

Keywords:

Object-oriented programming, procedural programming, Python language, popular, differences, industry

INTRODUCTION

In most languages, even object-oriented, dynamic programming, or any other types of programming languages, besides the main features of the language, all allow applying the procedural programming paradigm to create a program. This point has proved that procedural programming was the foundation of object-oriented development. In this research, the contrasts between each paradigm will be examined in several aspects or fields to explain why object-oriented programming is the dominant programming field. If developers love using OOP instead of procedural programming, what is the meaning of procedural programming existence? In this research, the question will be discussed through comparison tables and the investigation of other relevant articles. Example code written in both paradigms will also be attached to show the differences and support the explanation.

METHOD

How procedural languages work?

All programming languages were procedure-based on the very first day of computer science and programming. Ada Lovelace and Charles Babbage, two mathematicians, created the first computer in 1883, ushering in the modern era of computers. Following that noteworthy occasion, scientists start

looking into ways to speed up and improve computer operation. The heyday of procedural programming was in the middle of the 20th century, with the appearance of the first procedural languages like Assembly, FORTRAN, Autocode, LISP, Pascal, and Basic.

A programming language is a tool that instructs the computer on how to complete a task. It is a collection of statements that will instruct the computer step-by-step on how to accomplish a task. Procedural languages follow the top-down principle. The program starts from the first line of the instruction and executes each statement until it reaches the end of the instruction. Developers play an important role in determining the order of the instruction which is necessary to let the computer know which statement must be executed first. Procedural programming could be considered the precursor of OOP and consists of several well-structured modules and functions. The procedural programming paradigm is easier to modify and has a shorter execution time compared to OOP. However, the biggest drawback of procedural programming is that it is not flexible enough to solve all problems efficiently. All programs must follow a specific order that forces the developers to deal with the issues inactively because the issues will be viewed as a series of ordered steps.

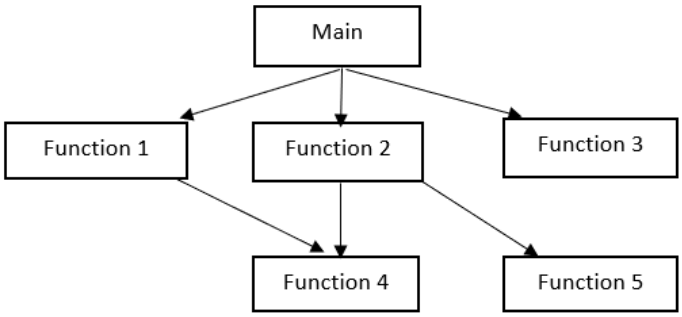

Figure 1: How procedural languages work

How object-oriented programming languages work?

OOP was developed based on the foundation of structured programming. Simula, the first high-level programming language, represented a breakthrough in computer science in 1967. The creation of that language by Kristen Nygaard had a significant influence on the development of other OOP languages like C#, Python, and Java. C++, one of the most popular OOP languages worldwide, was developed based on the C programming language.

Unlike procedural programming which views the program as a list of steps, OOP is a paradigm that approaches the solutions of the issues as a series of classes and objects. OOP is more flexible than procedural programming because of the interaction among different objects in different parts of the program. OOP is not required to follow any order. The objects will be called flexibly whenever it needs to be used to continue the execution process. OOP is a method based on real-life entities that focus on the result of working with the tasks rather than how it works. Each entity will have two parts: attributes and behaviors. All OOP languages have four characteristics: Encapsulation, Abstract, Inheritance, and Polymorphism. The data hiding feature is also supported, therefore the program will become more secure and easy to gain users' trust.

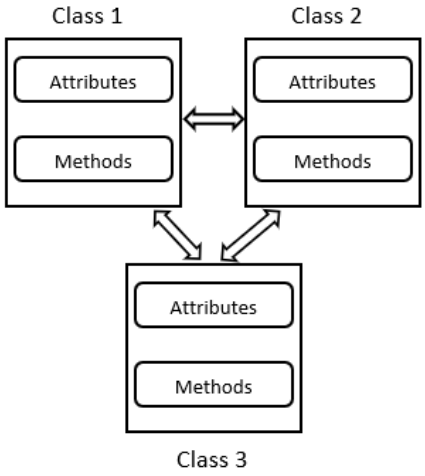

Figure 2: How object-oriented languages work

Differences in code syntaxes

In this section, a comparison of OOP and procedural programming code syntax will be implemented with Python to explain why OOP is chosen over procedural programming in computer science. In this research, we implemented the test in the Visual Studio environment. We took the box volume calculation as the problem to analyze the code syntaxes.

First, the mathematics formula will be provided to simplify the analysis:

Volume = height x breadth x length

We compared the following cases:

1. **Structure**

```
length = 3
height = 1
breadth = 2
volume = length * height * breadth
print(volume)
```

Procedural Programming

```
class Box:
def _init_ (self, length, height, breadth):
        self. length = length
        self. height = height
        self. breadth = breadth

def calculateVolume(self):
        volume = self. length * self. height * self. breadth
        return volume

newBox = Box (3,1,2)
print (newBox.calculateVolume())
```

Attributes

Methods

Object-Oriented Programming

When reflecting on the program's structure, the code of the procedural programming approach executes from top to down, but the OOP approach's code has two major parts: attributes and methods of an object. In the OOP code above, the first section of the OOP code provides the answers to the How questions: How do the objects work? In Box class, the volume is not calculated based on specific numbers. Instead, they use the derived attributes to outline the operation of the objects. The values are only given in the main part of the program when instances are created. The second part, on the other hand, provides the answers to the following questions: What are the values of the objects, what are they required to accomplish, and what are the outcomes?

In calculateVolume, why the method only returns the value of volume instead of printing it to the screen? This is a unique feature of OOP, and it may be used for many other purposes than just printing the result to the program., and only returning the value will increase the program's flexibility. Moreover, showing the result is not the target of the class, its target is to show how the objects work.

2. Modification

```
volume1 = length1 * height1 * breadth1 * 100
volume2 = length2 * height2 * breadth2 * 100
```

Procedural Programming

```
def calculateVolume(self):
        volume = self. length * self. height * self. breadth
        return volume * 100


newBox1 = Box (3,1,2)
newBox1 = Box (2.4.6)
```

Object-Oriented Programming

Consider the box volume described in the section before being measured in centimeters, both the assigned values and the result. Now, the requirement changed, and the volume must be calculated in millimeters., which means the initial volume must be multiplied by 100.

When a procedural programming paradigm is employed, each volume must multiply by 100 manually. If many boxes need to be calculated, it will take a lot of time to modify and guarantee the program's consistency. Sometimes, human errors might occur, and if that occurs, it will be challenging to find the errors among hundreds or thousands of similarly written lines of code. However, when the OOP paradigm is applied, the code only needs to be changed once in the method, this is more convenient and avoids making unwanted mistakes.

3. Extensibility

```
length = 3
height = 1
breadth = 2
volume = length * height * breadth
area = 2 * self. breadth * (2 * self. length + self. height)
```

Procedural Programming

```
class Box:
def _init_ (self, length, height, breadth):
        self. length = length
        self. height = height
        self. breadth = breadth

def calculateVolume(self):
        volume = self. length * self. height * self. breadth
        return volume

def calculateSurfaceArea(self)
        area = 2 * self. breadth * (2 * self. length + self. height)
        return area

newBox = Box (3,1,2)
print (newBox.calculateVolume())
print (newBox. calculateSurfaceArea ())
```

Extended method

Object-Oriented Programming

The capacity of a class to be extended is referred to as extensibility. In OOP code, the class that defines an object can easily have a new method added to it. When the extended function is requested, the caller only needs to call it after declaring the instance. However, in procedural programming, declaring a new method and using it in a program can be done in two different ways. First, the developers can write a direct statement in the main program to instruct it on what to do. Second, separate functions can be written to save time and increase code efficiency.

4. Code Reusability

```
def calculateVolume(l, h, b)
        return l * h * b

def calculateSurfaceArea(l, h, b)
        return 2*b*(2*l+h)

length1 = 3
height1 = 1
breadth1 = 2
volume1 = calculateVolume(length1, height1, breadth1)

length2 = 2
height2 = 4
breadt2 = 6
volume2 = calculateVolume(length2, height2, breadth2)
```

Procedural Programming

```
class Box:
def _init_ (self, length, height, breadth):
        self. length = length
        self. height = height
        self. breadth = breadth

def calculateVolume(self):
        volume = self. length * self. height * self. breadth
        return volume

def calculateSurfaceArea(self)
        area = 2 * self. breadth * (2 * self. length + self. height)
        return area

newBox1 = Box (3,1,2)
newBox2 = Box (2,3,5)
print(newBox1.calculateVolume())
print(newBox2.calculateVolume())
```

Object-Oriented Programming

The most crucial characteristic that determines the power of OOP is code reusability. Although procedural programming allows reusability, the reused code in different parts of a program does not interact with each other. The calculateVolume method was reused in the code above to determine the volume of two boxes. Except for the names of the variables, there was no other evidence that the outcome belonged in a certain box. If we switch the name of box 1 and box 2, an error will appear. Procedural programming also has the drawback of requiring new parameter passes each time a function is called.

When solving the problem with the OOP approach, the misunderstanding of the variable name will not appear because we need to create a new instance before calling their sub-methods to do something. As the result, the returned outputs will depend on the provided data of a specific object.

5. Handling invalid inputs

```
def calculateVolume(l,h,b)
        if (l<=0 or b<=0 or h<=0)
                raise Exception("Invalid")
        return l * h * b

def calculateSurfaceArea(l,h,b)
        if (l<=0 or b<=0 or h<=0)
                raise Exception("Invalid")
        return 2*b*(2*l+h)
```

Procedural Programming

```
class Box:
def _init_ (self, length, height, breadth):
        if (length<=0 or breadth<=0 or height<=0)
                raise Exception("Invalid")
        self. length = length
        self. height = height
        self. breadth = breadth
```

Object-Oriented Programming

To prevent the program from terminating or performing erroneously, all data must be validated before being used. Procedural programming is a collection of separate functions. Therefore, checking all functions and throwing the exception is the only way to validate data. It will lead to code repetition and make it hard to maintain.

Unlike procedural programming, OOP is not required to validate data in each function; instead, passed parameters will be validated at the initialization time. If the value is invalid, an exception is thrown, and the instance will not be created. Handling invalid inputs is a superior feature in programming because preventing problems at the entry level makes the code more robust.

Data security

```
def calculateVolume(l,h,b)
        if (l<=0 or b<=0 or h<=0)
                raise Exception("Invalid")
        return l * h * b

def calculateSurfaceArea(l,h,b)
        if (l<=0 or b<=0 or h<=0)
                raise Exception("Invalid")
        return 2*b*(2*l+h)

length = 3
height = 1
breadth = -2
volume = calculateVolume(length,height,breadth)
breadth = 2
area = calculateSurfaceArea(length,height,breadth)
```

Inconsistent data

Procedural Programming

```
class Box:
def _init_ (self, length, height, breadth):
        if (length<=0 or breadth<=0 or height<=0)
                raise Exception("Invalid")
        self. length = length
        self. height = height
        self. breadth = breadth
(…)
newBox = Box (3,1,2)
newBox._length = 5
print(newBox.calculateVolume())
print(newBox. calculateSurfaceArea ())
```

Syntax error

Object-Oriented Programming

When validation code is added to each function to handle invalid inputs, each time the function is called the time data will be validated again. In the given code, the initial input of the "breadth" variable was invalid because it was a negative number. Therefore, when the calculateVolume function was called, an exception was thrown. After that, the value of "breadth" was changed to a positive number which satisfies the conditions of valid data. When another function is called, it used a new valid value instead of the invalid value. Due to that reason, the calculateSurfaceArea executed normally without throwing any exception. Inconsistent data may result in wrong results.

On the other hand, assigning a new value to an existing object in OOP will result in syntax error since the encapsulation hides the private data from getting an error at compile time or an exception at runtime. This feature is necessary because it reveals the data inconsistency problem instead of allowing it to be passed.

RESULT

The previous part has analyzed both programming paradigms on different aspects that developers will deal with when working in the industry. Procedural programming is the first paradigm that all developers must use on the first day of their programming learning process, its existence is a foundation of OOP. Although procedural programming could complete most tasks, its convenience, security, and flexibility could not defeat the power of OOP. Details of the comparison will be pointed out in the below tables:

Procedural programming	Object-oriented programming
Consist of several modules and separated functions	Consist of several object definitions (classes). Each class has many attributes and methods
The program is divided into sub-program	The program works based on the interaction of objects
Difficult to modify data when requirements are changed	Easy to modify data
Data can be attacked or misunderstood	Data has been protected thanks to the encapsulation
No access specifier	Many access specifiers (scope) such as public, private, protected
Function call is used	Message passing is used
Follow the Top-Down approach	Follow the Bottom-Up approach
It is impossible to resuse code	It is possible to reuse code
Easy to manage the code because it follows the ordered sequence	Hard to manage because the interaction between objects is dense
Data is insecure	Data is secure

Table 1: Basic differences between OOP and procedural programming

Aspect	Procedural programming	Object-oriented programming
Modification	Hard to modify, must be carried out manually	Easy to modify
Extensibility	Code can be extended. The extensibility is not effective as it cannot automatically pass parameters	Easy to extend the program. The extended methods will execute based on data provided to the objects
Code reusability	Cannot reuse code. In some cases, code reusability is allowed but might produce wrong results	Code reusability is allowed. Most data declared in a class is general and can be used for many similar child classes.
Data security	Unwanted data can be accidentally accessed without any warnings	Private data will be hidden to avoid accidental access
Data consistency	Do not satisfy this requirement	Good Data will be passed at the time of initialization and cannot be changed

Table 2: Compare the efficiency OOP and procedural programming based on industrial aspects

DISCUSSION

In this section, we will discuss the value of analyzed programming features and how they are used in practice. Next, we will discuss the existing reasons for procedural programming. If it is not used in industry, which field could it be applied to highlight its strong points in programming?

How OOP could be applied in the industry?

The programming paradigm must satisfy specific business requirements to have significant advantages when applied in the industry. In the technological era, the application of programming in business is wide, most industries try to hire as many expert programmers as possible to minimize the effort that needs to be used to accomplish a task while maximizing the efficiency of the working process. Education, healthcare, agriculture, design, and other fields all use programming. To satisfy the requirements of completing tasks with the best result, some factors will be used to demonstrate that OOP produces better outcomes compared to procedural programming:

1. Modification

In real life, the requirements of all industries change from time to time to satisfy customers' demands. To boost the efficiency of working in the fast pace of the industry, a powerful tool that could quickly modify the code is required to meet the expectations of industry life. In this case, OOP is the optimal choice that allows changing code related to one object only by rewriting some lines of code in the methods of that object rather than fixing each line of code manually.

2. **Extensibility**

In large projects, to decide which requirements will be the priorities, which functions should be removed, and which should be added, all requirements must be discussed repeatedly. In the process of adding a new method to a class called extensibility, the program must be extended appropriately to prevent data from being inconsistent with the execution of the program. Since OOP is a paradigm that strongly encourages extensibility, new methods are typically inserted into the class that defines the object. This will ensure that extended features will be linked to other parts of the program without dealing with corruption.

3. **Data consistency**

Data is considered one of the most crucial factors that may affect the success of businesses. Data will be used to predict upcoming trends and identify opportunities by providing insight into customer behaviors and current strategies. Therefore, ensuring data protection and consistency is compulsory in programming. All related data must be packaged, and access should be limited to avoid unwanted access. Procedural programming includes a collection of separate functions, some statements may use the same functions, but their result is not connected. Inconsistent data which is incoherent and hard to maintain will harm the business development. OOP is the best paradigm that could protect the consistency of data. Each class in OOP consists of several attributes and methods, all data related to an object will be private to avoid outside access. The encapsulation principle of OOP will make all data remain unchanged throughout the program which could increase the level of data consistency.

4. **Data security**

The data hiding feature is not supported in procedural programming languages while it is strongly supported in OOP languages. The integration of this feature will make data more secure and easier to maintain. Most projects carried out in the industry are the collection of sub-programs, therefore ensuring the security of data is important to prevent the project from corruption because of data insecurity or misunderstandings.

5. **Code reusability**

Code reusability is an indispensable factor that contributes to the success of OOP languages in the industry. In large projects, the methods should be reusable for developers to call whenever they need to implement them in the program. All objects in large projects usually have interaction and connection with each other. An important object in the program may be called several times to do several tasks during the execution time. It is inconvenient if similar lines of code have to be rewritten multiple times when the developers need to implement them in the main program. Code repetition is risky because it may make some accidental mistakes. Code usability will create clean code and cut down the time needed to fix code when changes appeared. (If similar code is rewritten, when a small change appeared, the programmers must change the code in different places manually).

Where is a place of procedural programming?

After nearly a century since the year that the first procedural programming language was introduced, the development of computer science and programming languages has overcome many breakthroughs.

At this moment, procedural programming has no longer been the dominator of programming, this position was taken by OOP – the most popular paradigm used in industry.

However, as I have mentioned at the beginning of the research, the procedural programming paradigm was the precursor of OOP which means its appearance significantly contributes to the success of OOP nowadays. OOP was born later than procedural programming, a new paradigm must be better and more complete than the old one. The importance of procedural programming cannot be erased by the popularity of OOP, each paradigm has its strengths and weaknesses, if OOP is implemented in industrial areas to maximize working efficiency, procedural programming focuses on building a solid foundation of programming. The real power of procedural programming is often shown in the educational environment. All experts in programming used to be procedural programmers to clearly understand basic concepts of programming, how the program will be executed, and so on. The reason to explain why procedural programming is taught in the first university is because of its clear syntax and its top-down approach which is simple to absorb for beginners. Although procedural programming is not the best programming paradigm, it is still the first instructor for all experienced programmers at the start of their careers.

CONCLUSION

Each paradigm has its meaning, both have huge contributions to the development of the computer science field. Although it is undeniable that object-oriented programming is preferred over procedural programming in the industry nowadays, procedural programming still has a special position in the development history of object-oriented programming. OOP is more widely used and practical, which is a prerequisite for being famous in the industry. On the other hand, key components of OOP were inherited from its precursor - procedural programming. The existence of OOP is also the existence of procedural programming, the original paradigm has never been outdated, it just became a part of OOP's success.

REFERENCE

1. Kuan C. Chen. (2004). Comparison of object-oriented and procedure-based computer languages
2. HP company. (2018). Computer history: A Timeline of Computer Programming Languages
3. Lott, S., Phillips, D. (2021). Python object-oriented programming - fourth edition. Packt Publishing.
4. Asagba, P. O., Ogheneovo, E. (2010). A comparative analysis of structured and object-oriented programming methods. Journal of Applied Sciences and Environmental Management, 11(4).
5. Singh, N., Chouhan, S., Verma, K. (2021). Object oriented programming: Concepts, limitations, and application trends.