Multi-Paradigm Programming

Autumn/Winter 2021 Assignment

“THE SHOP”

REPORT

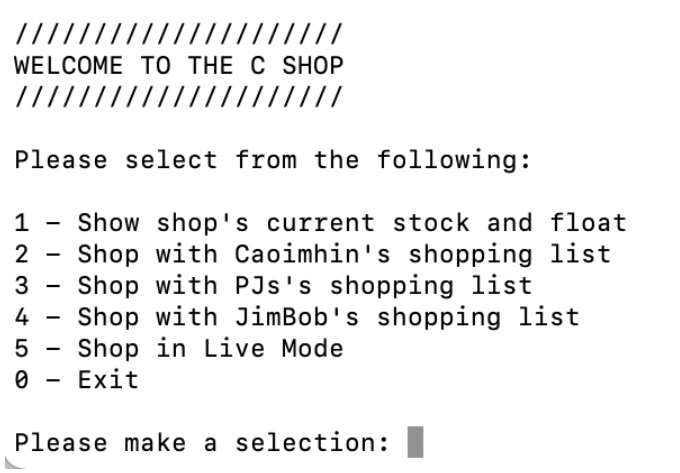
Introduction

The brief was to create a shop simulation using, and demonstrating an understanding of, both the **Procedural** and **Object-Orientated** programming paradigms. The initial development of the programs was carried out in lectures with Dr. Dominic Carr, after which we were requested to add additional functionality and bring the programs to completion. The procedural approach was to be completed in both the **C** and **python** programming languages while we had a choice for the object-orientated version between **java** and **python**.

My Solution

As requested, my versions of the simulation/application using procedural programming are in C and python, while I decided on python for the object-orientated approach. Once compiled (for the C program) and run, the programs all feature an identical user interface, with the same menus, functionality, and each accesses the same csv data files.

The user is firstly presented with the main menu as follows:



Option 1 prints the current shop stock, price per item, and cash float. Initially this will be as read from the csv file (stock.csv), but will be updated accordingly as the program runs, i.e. when items get bought, the levels of individual stock items will reduce and the shop's cash float will increase.
Options 2, **3**, and **4** read in 'shopping lists' from different customers. Each shopping list contains the customer's name and budget, and then a list of items and the quantities required. Customer1 is straight forward in that he has a sufficient budget, and the shop has sufficient stock, to complete the transaction smoothly. Customer2 however has a small budget so can't quite afford a few of the items on his list! A warning message is returned for these items and the program moves on to evaluate the following item. Customer3 on the other hand has lots of money and wants to buy more than the shop has in stock. The program checks the stock for each requested item, warns him if the shop doesn't have enough, and if so, limits him to that. Once each customer's order has been processed, the user is presented with their final bill, and the shop's cash float and stock are updated accordingly.

Option 5 is a live interactive mode. The user is asked for their budget, before being presented with a list of the shop's stock items and their price. They are prompted to choose an item and a quantity of that item. The program evaluates if there is sufficient stock or if the user has enough money and, if successful, the item is added to their order. Otherwise, a warning is returned. When the shopping is finished, the user can choose to finish and print their bill and the shop's float and stock is then updated. The following is an example of a 'live shopping mode' interaction:

```
//////////////////////////////////////////
You are now entering our LIVE SHOPPING MODE!
//////////////////////////////////////////
What is your budget?
25

//////////////////////////////////////
Welcome to the C shop LIVE SHOPPING MODE!
//////////////////////////////////////

Your current budget is €25.00.

Please select what you would like to buy from the list below:

1 - Cartons of milk @ €1.49 each
2 - Loaves of bread @ €1.99 each
3 - Pats of butter @ €3.45 each
4 - Bottles of wine @ €12.99 each
5 - Cans of beer @ €2.79 each
6 - Bottles of tequila @ €29.99 each
7 - Limes @ €0.49 each
8 - Packets of tortilla chips @ €2.99 each
9 - Jars of salsa @ €2.50 each
10 - Mandarin oranges @ €0.40 each
11 - Sausages @ €0.30 each
12 - Black puddings @ €3.00 each
13 - Packets of rashers @ €2.30 each
*98* - Finish shopping and print total bill
*99* - Exit Live Mode

Please make your selection: 4
How many Bottles of wine would you like to purchase? 1
---------------
PRODUCT NAME: Bottles of wine
PRODUCT PRICE: €12.99
- - - - - - - - -
QUANTITY REQUIRED: 1
QUANTITY PURCHASED: 1
TOTAL ITEM COST: €12.99
ADJUSTED BUDGET: €12.01
(ADJUSTED SHOP FLOAT: €1013.29)
- - - - - - - - - - - - - -

TOTAL BILL SO FAR: €12.99
Your current budget is €12.01.
```

Procedural vs Object-Orientated Programming

“Procedural programming is a programming paradigm, derived from imperative programming, based on the concept of the procedure call. Procedures (a type of routine or subroutine) simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.”
https://en.wikipedia.org/wiki/Procedural_programming

“Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields

(often known as attributes or properties), and code, in the form of procedures (often known as methods).”
https://en.wikipedia.org/wiki/Object-oriented_programming

Procedural programming is built around the concept that a program is a sequence of instructions to be executed. It is heavily focussed on splitting up programs into named sets of instructions called procedures or functions. A procedure can store local data that is not accessible from outside the function’s scope and can also access and modify global data variables. Object-oriented programming (OOP) relies on the concept of classes and objects. It is used to structure a software program into simple, reusable pieces of code called classes, which are used to create individual instances of objects. A class is like a blueprint used to create more specific, concrete objects. Classes can represent broad categories, like Shop or Customer, that share attributes. Classes can also contain functions available only to objects of that type.

The most important distinction is that while procedural programming uses procedures to operate on data structures, object-oriented programming bundles the two together, so an "object", which is an instance of a class, operates on its "own" data structure.”
https://en.wikipedia.org/wiki/Procedural_programming

Procedural programming involves a series of instructions, top to bottom, given to the computer to carry out a task step by step. OOP on the other hand is a collection of objects, grouped together by attributes and behaviour, which perform certain actions and know how to interact with each other. In my program for example, ProductStock is a class. The class provides the framework, i.e. the ProductStock can have a name and a price. An object or instance of that class might be a particular product called x that costs y and that has certain functions available to it such as a calculation of the total cost of multiple purchases of that item. The product can then interact with objects from the Shop or Customer class.
The main difference between my programs is while the procedural version is a series of functions which operate on the data and can be called from any part of the program, the OOP version includes the functions and data within the objects (encapsulation). In theory this should reduce the amount of code required (although this wasn’t the case for me!), but also improve reusability of the code. It also means ‘developers’ could possibly work on the different classes without affecting the other too much.

Examples of languages that use procedural programming are BASIC, C, FORTRAN, and Pascal. Examples of languages that use object-orientated programming are Java, C++, Ruby and C#. Python is a very versatile language can use several different paradigms including both procedural and object-orientated.

Conclusions

The first challenge in this assignment was getting to grips with the **C** language. It is a much older and significantly lower-level language than python, which has been the focus of our teaching to date. Due to its age and I guess lack of popularity, there isn’t a huge amount of resources online to assist, so a lot of my coding was trial and

error! We had the ‘head-start ‘worked on in class, so it was a matter of deciding what functionality I wanted to incorporate and then finding ways to complete that. This section of the assignment took up the longest amount of time by far. In hindsight, completing the procedural version in python first, a language I’m slightly more familiar with, might have been more sensible. I was eventually able to find enough snippets online to adapt and develop, alongside what we had already done in lectures to complete a logical solution that seems to work.
When it came to the procedural python version, it began firstly as an exercise in translation, albeit a poor ‘google-translate’ attempt! It quickly became apparent though that the more intuitive syntax and handy built-in functions in python made this a far more comfortable experience. The code reads and runs in a much more logical fashion, at least to me, and I was able to introduce a few more complex bits of code adding extra functionality to the application. This however, meant I had to revisit the C version and work out how to interpret them there!
The object-orientated version again began as an exercise in finishing what had been developed in class. I was able to copy and paste significant pieces of the other python program and was quickly able to find a solution that worked. The final result however may come across as a bit of a hybrid! I suppose the main difficulty I had was in how to access data and functions from different classes. I guess though, this is one of the main features of the paradigm, and is seen as an advantage in terms of security, etc. i.e. some information should not be accessible to all. In my case, perhaps the shop float and stock data is private information which could be kept in a separate file. Or data about the product stock should only be able to be modified from within that class. This is known as abstraction.
Another feature which I came to become familiar with was inheritance where objects can inherit attributes from their ‘parents’ in other classes. A ProductStock object will inherit characteristics from the Product class.
It probably wasn’t until I had finished all of the coding that I fully appreciated the actual differences between the two paradigms. I suppose I had treated the project as an exercise in finishing the given code in class, and just making it work. If I was to attempt the program in an object-orientated manner again in isolation, the result may be quite different.

References

https://www.geeksforgeeks.org/c-programming-language/
https://www.w3schools.com/python/
https://www.tutorialspoint.com/cprogramming/index.htm
https://www.programiz.com/c-programming/c-switch-case-statement
https://www.tutorialspoint.com/python/python_classes_objects.htm
https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/
https://medium.com/@LiliOuakninFelsen/functional-vs-object-oriented-vs-procedural-programming-a3d4585557f3
https://en.wikipedia.org/wiki/Object-oriented_programming
https://en.wikipedia.org/wiki/Procedural_programming
https://en.wikipedia.org/wiki/Programming_paradigm
https://www.educative.io/blog/object-oriented-programming