Multi Paradigm-Programming Shop Project Report

By Sarah McNelis – ID: G00398343

Introduction

This report contains comparisons between procedural programming and object orientated programming (OOP). This analysis is based on my shop project which involved me creating a shop simulation in c procedural programming, python procedural programming and python object orientated programming. Procedural programming is based on calling different procedures or functions (geeksforgeeks, 2022). Whereas OOP programming is based on classing related data together and containing the code within using methods (geeksforgeeks, 2022). The question is which is the best approach?

Comparisons

Firstly, I should explain that for c procedural programming, I use structures also known as structs. Structs are a system where one can group numerous variables together in one place (w3schools, 2022). For example, a society with it's members. The struct is the society and the variables are it's members.

Next, for python procedural programming I use dictionaries also known as dicts. Dicts are used to store data using keys and values (w3schools, 2022). In other words, a key would be considered the label and the value would be the actual content. For example, key = name, value = Sarah. By calling a key you can access the value.

Finally, for object orientated programming I use classes. A class is like a template for creating objects (hackerearth, 2022). For example, a class is like a map of Ireland. In that map there are 32 counties. Each one has different elements including size, population, and infrastructure. But they are all classed as a county of Ireland.

Now let's look at some of the key differences between my programmes.

Creating and stocking the shop:

For creating the shop and it's stock, OOP uses the class shop and opens the stock csv, reads in the cash amount and loops through the rest in order to get the stock, prices and quantities and appends them to a list. Python procedural has a function on its own for this which uses a dictionary to store the shop and it's stock in. C procedural is similar, expect the values are assigned using pointers. In c you can assign a variable and point to that variable using an asterisks (*). And you can access that variable's memory using the ampersand operator (&) (tutorialspoint, 2022). C has other useful built in functions for example, atoi which converts a string into a integer (tutorialspoint, 2022), atof which converts a string to a floating point number (tutorialspoint, 2022) and strcpy which returns a pointer to a destination string (programiz, 2022). This is a useful way of assigning values to strings with ease. Within my c code you'll find arrows. The reason for this is that the arrow can access elements of a struct using the pointer (geeksforgeeks, 2022).

Reading the customer order:

For reading in the customer order, OOP uses a method within the customer class. So this information is linked and can be accessed with ease from all the members of this class. Similarly to creating and reading the shop stock python procedural uses an individual function to read in the customer's order and c procedural uses the same foundation as reading the stock to read its customers.

Update the shop:

Again for OOP, the update shop function lies within the shop class. This allows elements of the shop to be accessed and amended with ease. Python procedural uses a separate function with the shop passed in as a parameter. This parameter was set whilst calling the main function. This link allows the shop to be updated from a separate function. Similarly, c procedural passes the shop struct in as a parameter into the update shop function. This is what allows the shop to be updated using c.

Update the customer csv

Again the update customer happens within the customer class in OOP. The difference between python OOP and python procedural is that in the procedural shop there is a separate function to update the customer. And, the customer has to be set and passed in as a parameter. Whereas with OOP, because

it is all happening within the same class the parameter doesn't need to be assigned. It just needs to call itself to make the required changes. Interestingly, in c procedural, the update to the customer is quite subtle and happens in the customer check out function. It is done using the strcpy method. This can be good in that it avoids creating extra functions. However, this could be perceived as negative if there were to be changed made to this programme in the future. It could become complicated to separate and re-write code.

Live shop:

In OOP, my live shop is a class of it's own. I have done this as it is an interactive platform with the user and was easier to keep separate from the actual shop. However, once the user goes through the live shop, their order is processed, and their csv updated same as it would be when reading from a csv. Similar to this, python procedural uses a separate function for the live shop. In c procedural, the live shop is a separate function which is linked to the shop struct. Scanf is used to retrieve the user input while fprintf writes the data to the csv file for us (ibm, 2022).

Customers check out:

Finally, the customer check out was the trickiest part of all three programmes in my opinion. The idea for all three was to loop through the customer's shopping list, check the stock, quantity, check the budget and process the order. The difficulty in this was allowing for the lack of stock, quantity, and budget. In OOP, this method was contained within the shop class. The advantage of this is that we are already in the shop it's just a case of passing in the customer. Whereas for procedural both the shop and the customer had to be passed in. I ran into trouble when attempting to access the price of the product the customer desired within this function. It was much easier to access this in OOP as the checkout was happening within the shop class. Whereas with the python procedural, I had to create extra variables to retrieve the price. With c procedural, my programme became difficult when checking the customer's budget. I had to create a variable to use in my budget check as my programme was completely skipping this if/else statement. In any case, this problem was resolved.

Conclusion

Having completed a shop simulation in two programming languages and two programming styles, I believe the best option for creating a good shop simulation is to use Python OOP. Classes are a great

resource for organising related data together under one umbrella. In my opinion, it is far more of an advantage to programmers being able to access related data for use within a class than having to create separate functions and calling each one every time. Comparing the two procedural approaches I would recommend c over python. Structs are a useful way of accessing data and the option to use pointers makes it easier to manipulate and access the data. However, I wouldn't recommend a procedural approach for this type of programme. Therefore, first place goes to a python object orientated approach.

Bibliography

geeksforgeeks. (2022, November 29). Retrieved from https://www.geeksforgeeks.org/arrow-operator-in-c-c-with-examples/

geeksforgeeks. (2022, November 29). Retrieved from https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/#:~:text=In%20procedural%20programming%2C%20the%20program,follows%20a%20bottom%2Dup%20approach.

hackerearth. (2022, November 28). Retrieved from https://www.hackerearth.com/practice/python/object-oriented-programming/classes-and-objects-i/tutorial/#:~:text=A%20class%20is%20a%20code,the%20instance%20of%20the%20class.

ibm. (2022, November 29). Retrieved from https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-fprintf-printf-sprintf-format-write-data

programiz. (2022, November 29). Retrieved from https://www.programiz.com/c-programming/library-function/string.h/strcpy

programiz. (2022, November 29). Retrieved from https://www.programiz.com/c-programming/c-input-output

tutorialspoint. (2022, November 29). Retrieved from https://www.tutorialspoint.com/cprogramming/c_pointers.htm

tutorialspoint. (2022, November 29). Retrieved from https://www.tutorialspoint.com/c_standard_library/c_function_atoi.htm

tutorialspoint. (2022, November 29). Retrieved from
https://www.tutorialspoint.com/c_standard_library/c_function_atof.htm

w3schools. (2022). Retrieved November 28, 2022, from
https://www.w3schools.com/c/c_structs.php

w3schools. (2022, November 28). Retrieved from
https://www.w3schools.com/python/python_dictionaries.asp