Declaration on Plagiarism
Assignment Submission Form

This form must be filled in and completed by the student(s) submitting an assignment

Name(s): **Andrew Finn**
Student Number: [redacted]
Programme: **CASE3**
Module Code: **CA341**
Assignment Title: **Report: Comparing Object Oriented vs Imperative style programming.**
Submission Date: **November 2020**
Module Coordinator: **Dr. David Sinclair**

I/We Declare that this material, which I/Wenow submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/ourwork. I/We Understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I/We Have read and understood the Assignment Regulations. I/We Have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the sources cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/usor any other person for assessment on this or any other course of study. I/Wehave read and understood the referencing guidelines found at http://www.dcu.ie/info/regulations/plagiarism.shtml, https://www4.dcu.ie/students/az/plagiarismand/or recommended in the assignment guidelines.

Name: **Andrew Finn** Date: **Nov 4th 2020**

Comparing Imperative and Object-Oriented Programming

Andrew Finn - 18402034

For the completion of this assignment I used Python3 as my Object-Oriented language and C for my Imperative language. Source code is attached with my submission, src - Imperative containing my Imperative C solution and src - OOP containing my Python Object-Oriented Solution. For the remainder of this report I will refer to Object-Oriented Programming as OOP.

Object-Oriented Programming Approach:

As my primary coding language is Python and I am more familiar with Object-Oriented Programming I began by writing my Python code first.

In my implementation I used 4 classes, Node, Tree, Contact and Phonebook. I could have accomplished this with less but for readability and to showcase OOP.

The node class is the base of any Tree it contains a key, data which in this case is an instance of a Contact Class, a left and right child node and its parent node if any. It also contains functions to ascertain certain properties of a given node such as “has_children()” and “is_root()” these functions are added to reduce repetition later.

The Tree class is the heart of the program. The tree class is where all the logic is contained it has functions such as “delete(key)” , “insert(key)” and “find(key)” which will delete and reformat the tree, insert a new node into the tree in the correct position and return a node given its key respectively.

The Contact class is a simple class which stores a contact's data in an object. It stores an Name, Address and Phone number. It also has a __str__ function for easy formatted printing.

The PhoneBook class makes this generic Tree class into a PhoneBook. It first starts by initializing two trees, a tree added by the hash of a Name and the other by the hash of a Phone number. It contains functions such as “Add(Contact)” which, given a Contact class, will add the contact to both trees by the hash value of the contacts Name and Phone number. The “search_name(name)” and “seach_number(number)” function will return a Contact object after searching by name and number hashed respectively. The “remove_name(name)” and “remove_number(number)” function will remove a given entry from both trees simultaneously.

These 4 classes make up the vast majority of the program. Each class is useless without its companions. The PyTest function at the end of the program showcases how the programming functions and also tests to ensure that the programming is working properly by testing all functionality.

To utilize this program a user can initializing a book “book = PhoneBook()” and from there using this new phone book to add (“book.add()), remove (book.removeName/Number()) and search (book.searchName/Number()) the phone book for records.

Imperative Programming Approach:

Writing my approach imperative was not as straightforward. Python? No. Trying to implement a Binary Search Tree using only int, strings, and booleans (and maybe arrays but only to store data) was more of a challenge than was necessary. After some research, I decided that it was overdue for me to learn C. As i had created my Python implementation before creating my C one i knew how to solve the problem so it was just a matter of learning the syntax and nuances of C.

The main struct of my implementation is the Node struct which contains its key in the form of a hash value, and pointers to its left and right children if any and a Contact Struct. The contact Struct contains values for a given Contacts Name, Address and Phone number.

The remainder of the program is made up using functions the main functions are:

The “searchNode()” function is a recursive function that will return the contact struct of function given a hash value for the Node and its respective tree,

The “searchHash()” function works in a similar way to the search node function however it differs as it can handle events such as if a contact doesn't exist and automatically prints the respective contact when found.

The “insertNode()” function will add a given key in the form of a hash and contact struct to the respective tree in the correct position.

The “deleteNode()” function will delete and reformat the tree given node given its hash key value and its Tree.

The remainder of the functions are either helper function or tying the above function together in order to simplify the function required of the program for example:

The “newContact” function will create a new instance of a Contact struct and add the contact to both trees by their respective hash keys

The “removeNumber/Name” function when given both trees and corresponding name/number will remove from both trees the corresponding contact.

The “searchNumber/Name” void functions when given both trees and corresponding name/number will search for the contact and automatically print the contact details as an error message if the contact doesn't exist.

A showcase on how to utilize and initialize the program can be seen in the “main()” function of the program along with test cases showing its functionality.

Comparison

I think it's important for my comparison to focus on the difference in terms of approach rather than the differences that occur as a result of using a different lower level language as both implementations successfully implement the program.

The OOP style of programming revolves upon the concept of objects. An object 'encapsulates' its data and contains various methods and functions which can act upon and manipulate its data. Data can only be accessed after an object is initialized. Code is structured in such a way as to emphasise this data format and it's associated actions that can be called upon. A class can be equated to a template, it groups the code based upon its purpose. The four classes that make up my implementation hold data and functions relating to a certain part of the solution Eg. The Contact class holds details relating to an instance of a contact such as Name and Address and the Tree class holds data and functions relating to searching and storing a tree. Writing programs in this form allows the developers to segment their solution into different parts, solving one problem at time and building on top of other classes rather than repeating code.

However implementing a solution in the imperative programming style is more straightforward (in small solutions) and concise, every line of code has a function. The methodology behind this paradigm is to provide an exact and logically flowing set of instructions on what needs to be done in order for the desired effect to be achieved. Within my imperative solution, Order of declaration of functions in my Imperative solution is mandatory you can not call a function which has been declared after the line in which you call it. Python on the other hand will find the correct function regardless of when it's declared.To but the difference into an analogy coding in an imperative style is like an instruction manual you follow each step, page by page, coding in an OOP style would involve jumping from page to page in the instruction manual, doing steps multiple times, jumping from instruction point to instruction point in a non sequential order.

In the OOP the end product is very clean and simple to use, it's a structure, the top floor being the PhoneBook class and the bottom floor being the Node class. To program such a solution is easier as it can be segmented each floor can be built separately and then interlinked. Classes can be shared among programs, can be initialized easily multiple times and can have methods and functions corresponding to them.
In the Imperative C solution the code is more streamlined each line has a purpose there is very little extra code. The imperative solution does not have the same building like solution there is floors in its structure instead there is only one floor, the program calls other functions however these functions are simple minilpating and changing the same underlying data they act more as shortcuts. This approach makes each function very tailor made, making them very short and quick but can only be used for that exact purpose.

In relation to the problem of the phonebook I am of the opinion that an OOP style is a better approach; its class based building blocks allow for easier development in this situation as well as facilitating easy expansion of the program should more functionality be required of the program. However the imperative style is a quicker and more concise implementation

with very little unnecessary code. I feel that an imperative approach would be a better for approach for a smaller simpler problem.

References

The hash function in the C program was referenced from -
https://stackoverflow.com/questions/20462826/hash-function-for-strings-in-c
https://www.youtube.com/watch?v=KJgsSFOSQv0
https://www.educba.com/advantages-of-oop/
https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/