Ocean Graham

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

403IT – Problem solving and programming

CW2 Portfolio



Table of Contents

[Table of Figures 1](#_Toc190680724)

[Introduction 2](#_Toc190680725)

[Problem-solving techniques 2](#_Toc190680726)

[Decomposition 2](#_Toc190680727)

[Abstraction 3](#_Toc190680728)

[algorithm design 3](#_Toc190680729)

[software development lifecycle 4](#_Toc190680730)

[Waterfall and agile methodologies 4](#_Toc190680731)

[Algorithm Design and generalised Problem solutions 6](#_Toc190680732)

[Pseudocode 6](#_Toc190680733)

[flowchat 6](#_Toc190680734)

[programming language concepts 6](#_Toc190680735)

[Palindrome 6](#_Toc190680736)

[Tip Calculator 6](#_Toc190680737)

[conclusion 6](#_Toc190680738)

[Appendix 7](#_Toc190680739)

[References 7](#_Toc190680740)

# Table of Figures

403it – Problem solving AND PROGRAMMING

# Introduction

This portfolio displays a collection of problem-solving tasks, iterating the prompted tasks, how it was developed, resolved and ultimately completed to function. The primary goal of this coursework is to present a high degree of problem-solving skills, critical/analytical skills and understanding of techniques, concepts and methodologies relating to programming. “Programming is the process of drawing up the schedule of the sequence of individual operations required to carry out the calculation” (Hartree. 1950, p. 111). The programming tasks chosen in this portfolio were the ‘**Palindrome Checker**’ and ‘**Tip Calculator’** applications. These tasks were due to the variability in processing and purpose, ranging in the requirements to carry out their performance from pseudocode to finalised implementation.

# Problem-solving techniques

The first stage in resolving a problem is to identify the problem(s) in the first place. This is just the same in programming, “programming requires a hierarchy of skills like abstraction, generalization, transfer and critical thinking” (Gomes & Mendes, 2007, p. 18). The nature of programming is the adaptation, development and resolution of multiple versions of code. The two tasks displayed in this portfolio present this evidently, utilising the techniques explained onwards:

## Decomposition

Decomposition is the technique of breaking down a complex problem into a litany of smaller simpler problems which are easier to solve (GeeksForGeeks, 2022, ‘What is Decomposition Computational Thinking?’). This technique is fundamental in all complex programs, such programs with many moving parts can be difficult without a breakdown of the task into a selection of small sub-problems. These smaller problems can then be put together to find the larger original programs solution. The process typically can be broken down into a series of structured steps as suggested by (StudySmart, N.D., ‘Decomposition Computer Science’):

-Identification: defining what needs to be solved -Break down: breaking the large problem into smaller simpler problems -Analysing components: defining the purpose of each part (its requirements and function) -Resolution: creating solutions/answers to each fragment -Integration: combining all fragments into one whole system

## Divide and conquer

Divide and conquer is a technique that involves dividing the main problem into subproblems, these are then solved individually then combined to find the resolution to the original issue. This technique holds similarities to Decomposition and could be seen as a development on Decomposition.

## Abstraction

Abstraction is another important technique for computational thinking and is fundamental key aspect of object-oriented programming (Rouse, Techopedia, 2020, ‘Abstraction’). Abstraction as (Cambridge Dictionary, N.D., Cambridge Dictionary English: Abstraction) states, it is defined as (noun [C or U] ‘Removing’) “the [action](https://dictionary.cambridge.org/dictionary/english/action) of [removing](https://dictionary.cambridge.org/dictionary/english/remove) or [separating](https://dictionary.cambridge.org/dictionary/english/separate) something from a [place](https://dictionary.cambridge.org/dictionary/english/place) or [context](https://dictionary.cambridge.org/dictionary/english/context) (= the [situation](https://dictionary.cambridge.org/dictionary/english/situation), [facts](https://dictionary.cambridge.org/dictionary/english/fact), words, etc. that [exist](https://dictionary.cambridge.org/dictionary/english/exist) around something)”. Abstraction focuses on filtering through points, and ordering them on importance, and recognising similarities and ignoring differences. This allows a programmer to focus on the “elimination of the irrelevant and amplification of the essential” (Martin, 2003, Chapter 20 Section 3). This is important as it allows us to develop a general idea of the program, capturing the key functionalities and hiding irrelevant factors. The process instructing us to remove all specifics, and any pattern that will not aid us (BBC Bitesize, 2019, ‘Bitesize Abstraction’).

# The software development lifecycle

The Software Development Lifecycle (SDLC) is the framework used within software companies/teams. The SDLC has 7 main phases according to (Jackson, IBM, 2024, ‘What is the software development life cycle?’), Each phase has a specific purpose to maximise design: Planning & Analysis, Design, Development (Implementation), Testing, Deployment, Maintenance. The SDLC proves a valuable structure to teams, allowing them to “manage the development process systematically with clear goals and deliverables at every stage.” (Clark, The Product Manager, 2024, ‘What is the Software Development Life Cycle?’).

## Waterfall and agile methodologies

There are a range of different SDLC models, however there are two approaches with higher prominence in relation to the others, these two approaches are:

* A diagram of a process

  AI-generated content may be incorrect.Waterfall – The waterfall model is a software development methodology, which is generally believed to have first been introduced in Winston W. Royce’s 1970s paper. This model entails a linear, sequential, document-focused model, having a clear roadmap of each stage.

Figure 1 - Image of Waterfall Model

Source: (Wieczorkowski & Polak, 2012, ‘An approach to analysis and implantation. From the waterfall model to the two-segmental model of information systems lifecycle’)

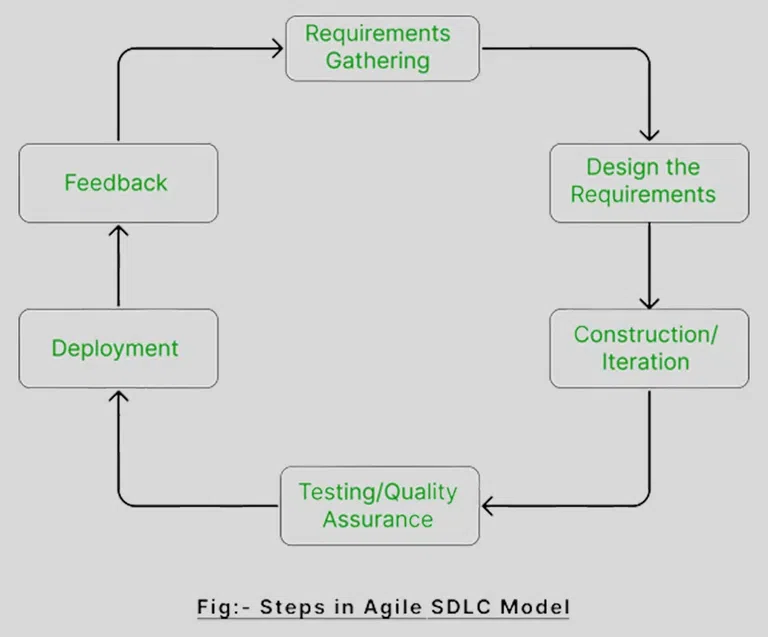
As figure 1 shows, the waterfall model is a step by step model, requiring the completion of the current phase before being able to continue to them to the next phase (GitHub, 2024, ‘What is the SDLC’). This model is best for projects that require little adaptation and changing requirements.

* A diagram of a process

  AI-generated content may be incorrect.Agile – The agile method is a flexible iterative approach that emphasises collaboration, adaptability and incremental deliveries. This model is more focused on communication between a team, rather than waterfall’s documentation driven model, It has the ability to break a large project into smaller more comfortable pieces, facilitating faster project completion and business efficiency (Xero, N.D. ‘How to implement agile methodologies in your business’)

Figure 2 - Steps in the Agile Model

Source: Own Elaboration



The methodology used for the programs was taken into account as part of the Planning & Analysis stage. For this project, the Agile method was more favourable due to the nature of the project, having the flexibility to alter the requirements and demands to adapt to new processes and needs for the algorithms is a crucial factor that the Waterfall method does not provide due to its rigidity. The Planning & Analysis stage involved using abstraction to break the main program into subsections and identifying the key features and challenges. For example, the Tip calculator, needed to calculate tips, however, there needed to be a thing to calculate tips from, therefore the order menu was developed to order from, to then promote a purpose of the tip calculator.

# Algorithm Design and generalised Problem solutions

Algorithms are a plan and set of steps for solving problems, they act as a template and foundation for programs. As said by (Paris, Inside Algorithms, 2024, ‘What is an algorithm? Definition, structure and examples’) “Algorithms are the beating heart of modern computing”, An algorithm is a composition of control structures; examples of such are defined by the ‘304IT Understanding PowerPoint’ within the appendix. Algorithms are needed as they help a program respond appropriately to its directive and utilise its recourses more accordingly. (Chris, freeCodeCamp, 2022, What is an Algorithm? Algorithm Definition for Computer Science Beginners’). Some representation includes Pseudocode and Flowcharts.

When designing algorithms, preparation before the construction of actual code, allows a more refined quality code in the long term, as you can more easily clarify problems and challenges during practised development compared to real code, as tools such as pseudocode and flowcharts are easier to resolve than the true script. The tools such as Pseudocode and Flowcharts are essential during the design phase of the SDLC, planning the structure of the script before the design and implantation.

## What is psuedocode?

Pseudocode is “an informal contrived way of writing programs in which you represent the sequence of actions and instructions in a form humans can easily understand” said (Ubah, freeCodeCamp, 2021, ‘What is Pseudocode? How to use Pseudocode to solve coding problems’). In simple terms, it is a rough design structure for a potential algorithm which is made for understanding rather than function. Pseudocode helps us focus on the structure and logic to break the problem down, rather than focusing of details such as Syntax.

## Pseudocode For palindrome checker

A computer program with text

AI-generated content may be incorrect.

Figure 3 - Pseudocode for Palindrome Checker

Figure 3 shows the pseudocode for the Palindrome Checker, it illustrates a simple template for an application that reverses words. It presents the offer to input a word, that word is then taken and reversed and compared to itself to see if it is a palindrome. This is designed to highlight the variability of inputs and outputs.

## pseudocode for Tip calculator

Below with figure 4 shows the Pseudocode for the Tip Calculator, it presents a framework for a program that presents an order menu to the user. This then allowing them to order items and add it to a list. Continuing, this is then prompted with a price from the accumulation of the ordered items, this next offers the opportunity to tip a selected amount. This is designed to allow a high amount of variability and decision-making for the user.

A close-up of a document

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Figure 4 - Pseudocode for Tip Calculator

## A diagram of a company AI-generated content may be incorrect.flowchart for palindrome checker

Figure 5 - Flow Chart for Palindrome Checker

## Flowchart for tip calculator

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 6 - Flow Chart for Tip Calculator

The two flowcharts above (Figure 5 & 6) present a visual representation of how both programs could handle user inputs and the subsequent outputs. These maps show the flow of an algorithm using different shapes and symbols, Rectangles for processes, Diamonds for decision points, arrows indicate the direction of the path, parallelograms showing inputs and outputs.

The palindrome checker presents as said priorly, how the program may handle user inputs, the flowchart is a development on the pseudocode however, including the decision point of an invalid input, this is due to the palindrome checker not allowing special characters, numbers or spaces, In the case of this it prompts an invalid input, however minus the invalid input feature added to the flow chart, the process remains the same, prompting an input, if a valid input it goes on to compare the input word to itself reversed, and if both equal the same, then it will be flagged and output that it is a palindrome.

The Tip calculator is similarly, the closely the same as priorly said during the pseudocode, however the flow chart presents the options of ‘5’, ‘10’, ‘20’, ‘Custom’ and No Tip have been added, rather than a flat tip amount input. This version uses percentage calculations of the ordered items list added on top of the bill if a selected tip option is added.

# programming language concepts

The language used to develop the Tip Calculator and Palindrome Checker was Python, this is due to it’s simplicity and easy of use, supporting multiple different types of code, including Procedural, Functional and Object-Oriented Programming (OOP)

Procedural programming is a paradigm that involves following linear steps in an ordered path, similar to following a recipe. Procedural was used in both the Tip Calculator and Palindrome Checker to make the scripts follow loops to handle certain tasks, for example ordering items from a menu list (In the Tip Calculator), In the case of the Palindrome checker, it was used to handle the inversion of the input word. Functional programming is a when everything is tied to mathematical functions, this programming is a declarative style (GeeksForGeeks, 2024, Functional Programming Paradigm) this was not used for any of the algorithms designed. Finally, Object-Oriented Programming (OOP) is focused around objects. These objects represent real things such as in the case of the two programs, items such as an order list, menu, tip and a word that is to be inverted. OOP is used throughout both programs.

A vast amount of practises and learning of Python coding tasks and functions, from simple to complicated programs, have aided in the imperative development of a high degree of understanding that has been used to develop the algorithms using Python. These include syntax elements, functions, variables, blocks, control structures (if-else statements, for and while loops) and data structures (dictionaries and lists). The syntax listed is illustrated within the ‘403IT\_Understanding\_PP’ in the appendix and are some of the concepts and techniques used within the code.

## Palindrome

## Tip Calculator

# conclusion

# Appendix



# References