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](#_Toc190352659)

[Introduction 2](#_Toc190352660)

[Problem-solving techniques 2](#_Toc190352661)

[Decomposition 2](#_Toc190352662)

[Abstraction 3](#_Toc190352663)

[algorithm design 3](#_Toc190352664)

[software development lifecycle 4](#_Toc190352665)

[Waterfall and agile methodologies 4](#_Toc190352666)

[Algorithm Design and generalised Problem solutions 4](#_Toc190352667)

[Pseudocode 4](#_Toc190352668)

[flowchat 4](#_Toc190352669)

[programming language concepts 4](#_Toc190352670)

[Palindrome 4](#_Toc190352671)

[Tip Calculator 4](#_Toc190352672)

[conclusion 4](#_Toc190352673)

[Appendix 6](#_Toc190352674)

[References 6](#_Toc190352675)

# Table of Figures

**No table of figures entries found.**

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

## 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’).

## algorithm design

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’). Representation of Algorithms are Pseudocode and Flowcharts.

# software development lifecycle

## Waterfall and agile methodologies

# Algorithm Design and generalised Problem solutions

## Pseudocode

## flowchat

# programming language concepts

## Palindrome

## Tip Calculator

# conclusion

# Appendix



# References