

Design Report for Practical Coursework 2: Run Logger

CS1D461 C++ Programming

Simon Payne – 12345678

University of South Wales

[simon.payne@southwales.ac.uk](mailto:simon.payne@southwales.ac.uk)

Contents

[Introduction 3](#_Toc63168957)

[Design 3](#_Toc63168958)

[Flow Chart 3](#_Toc63168959)

[Pseudocode 3](#_Toc63168960)

[UML 4](#_Toc63168961)

[Classes 4](#_Toc63168962)

[Class: Person 4](#_Toc63168963)

[Class: Run 4](#_Toc63168964)

[Data Types 4](#_Toc63168965)

[Implementation 5](#_Toc63168966)

[Testing 6](#_Toc63168967)

[Test 1: Adding a new Person 6](#_Toc63168968)

[Conclusion. 7](#_Toc63168969)

[Code Review 7](#_Toc63168970)

# Introduction

In this section, write about which client brief you have chosen. Then describe what the report will tell the reader.

For example: “In this report the design, implementation and testing of the Run Logger system will be detailed”.

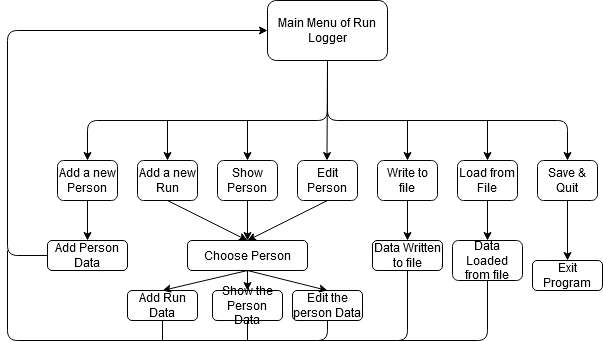
Write about why you picked the client brief you have chosen to work on.

# Design

In this section talk about how you have designed your software. You can use any sort of technique you like. Best options are diagrams.

For example, flow chart:

## Flow Chart



## Pseudocode

You could also write some pseudocode do describe certain pieces of logic.

## UML

You could also use UML diagrams and describe the way the application will fit together. Class or state diagrams would be fine. These will be necessary for practical coursework 2 (the 4th assessment)

## Classes

### Class: Person

Fields:

* Name
* Age
* Collection of Runs

Functions:

* Add a run
  + Function to allow a run to be added to the person.
  + Etc

### Class: Run

Fields:

* Distance
* Time
* Terrain

Functions:

* Calculate Pace
* Etc….

## Data Types

A table could be used for data types:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Data Type | Example Data | Reasoning |
| Name | String | Simon Payne | Simple string to hold name data |
| Age | Int | 30 | Ages are stored as whole numbers only |
| Distance in KM | Double or float | 5.5 | KM distance could include partial KM’s so need a decimal point |
| Etc….. |  |  |  |

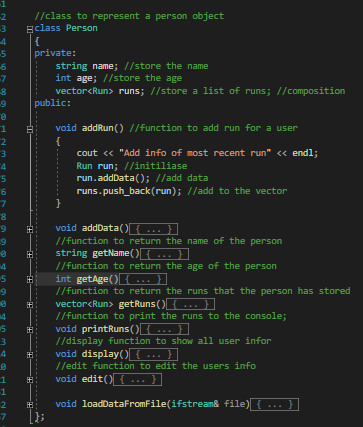
Describe classes or structures you will need:

# Implementation

In this section show screenshots of code, and a description of the code you have written to fulfil the client brief that is guided by your design.

e.g.

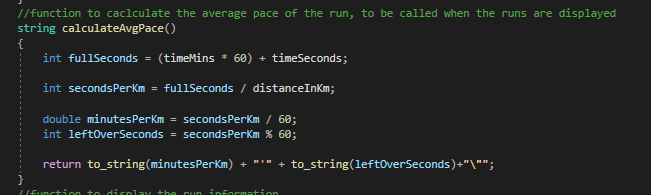
Implementation of the Person Class:



This is the person class. The fields and functions have been implemented as per the design. The runs are stored in a vector so that an unlimited number of runs can be stored against the person object.

Highlight any section of code you are pleased with, or want to show off a bit:

For example:

This code calculated the average pace of the run. The value is not stored but calculated on the fly when needed.

# Testing

This section should refer to your design and implementation sections. This should also describe the tests you have carried out with evidence. E.g

## Test 1: Adding a new Person

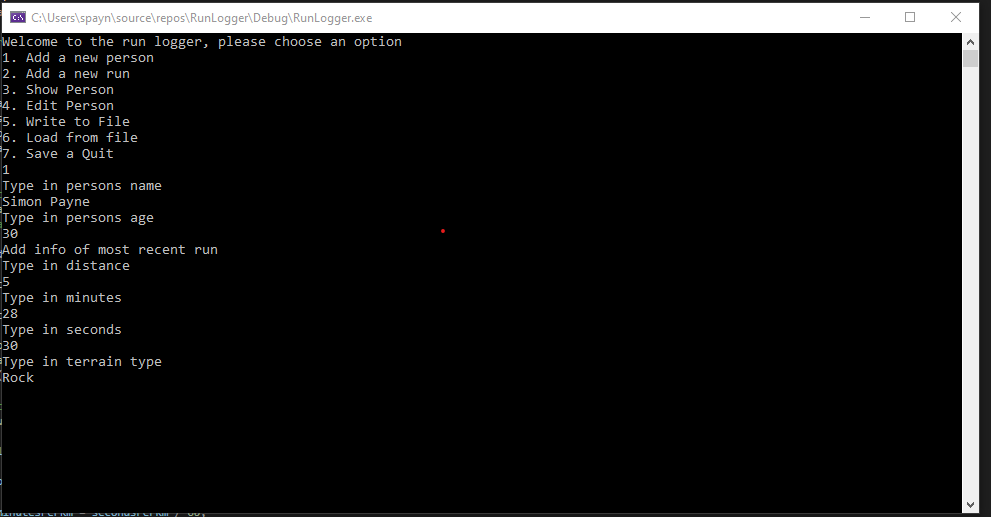
Step 1: Select Add A new person from the menu:

Step 2: enter the User details including run information.

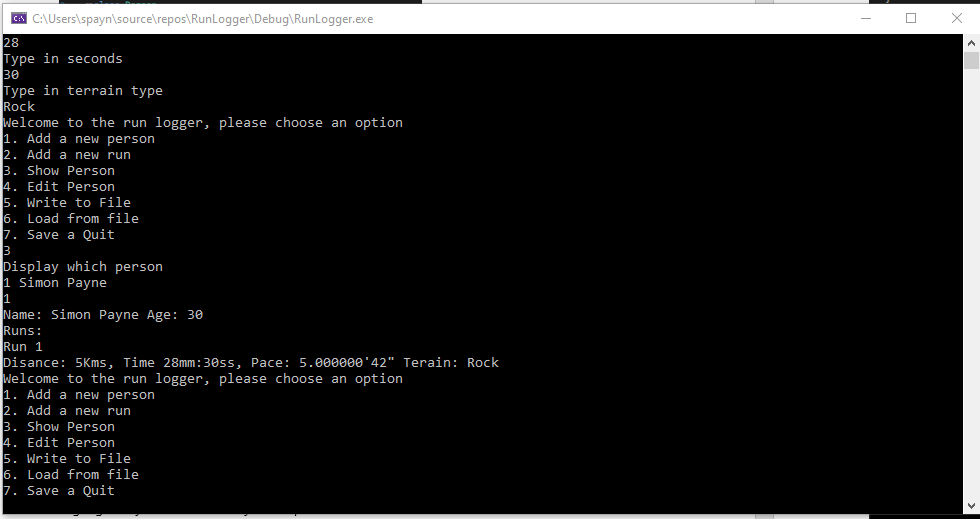
Step 3: View the user to check that data entry as worked.

Evidence of Test Pass:

Adding Data:



Viewing Data:



Etc..

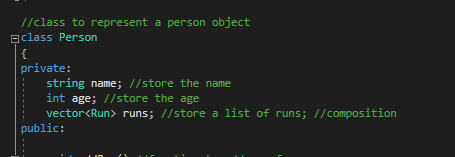
You should at least test each menu item.

# Conclusion.

In your conclusion sum up the report. Describe if you feel you have met the client specification. Highlight any areas of your code you think could be done better Ort could be cause for concern.

E.g.:

## Code Review



The runs are stored as a vector, which is good because an unlimited number of runs can be stored, however it does make it difficult to read the data from the file as the program will not know how many runs need to be retrieved. This has led to some complicated load from file code:

