SIM UOW CSCI251

ASSIGNMENT 2

Last updated: 31 January 2024

Timothy Mah

10258663

Lecture Group L01

Contents

[Introduction 2](#_Toc157620971)

[Usage 2](#_Toc157620972)

[Program flow 2](#_Toc157620973)

[Extra features 3](#_Toc157620974)

[Input validation 3](#_Toc157620975)

[Main menu 3](#_Toc157620976)

[Sorting menu 3](#_Toc157620977)

[All shape data input 3](#_Toc157620978)

[Data visualisation 4](#_Toc157620979)

[Dynamic shape range 4](#_Toc157620980)

[Limitations 4](#_Toc157620981)

[Learnings 4](#_Toc157620982)

[Override 4](#_Toc157620983)

[Virtual functions 5](#_Toc157620984)

[Abstract functions 5](#_Toc157620985)

[Unique\_ptr 5](#_Toc157620986)

[static variable vs static function vs static variables in a class 5](#_Toc157620987)

[ternary conditional operator 5](#_Toc157620988)

[Thoughts and reflections 5](#_Toc157620989)

[Purpose? 5](#_Toc157620990)

[Assignment difficulty 5](#_Toc157620991)

# Introduction

Hello this is my documentation for SIM UOW CSCI251 ASSIGNMENT 1.

# Usage

The final program file can only be run in Ubuntu because of the makefile. However, the main file can be compiled to support Windows by compiling as .exe.

To run,

./ csci251\_a1.app

To compile,

make

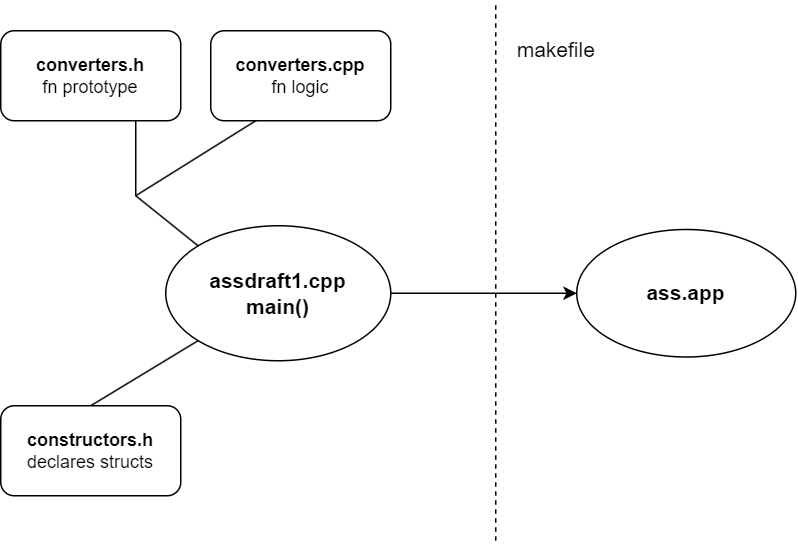
In same directory as makefile. Makefile should be in same directory as main file and the other helper files.

# Program flow

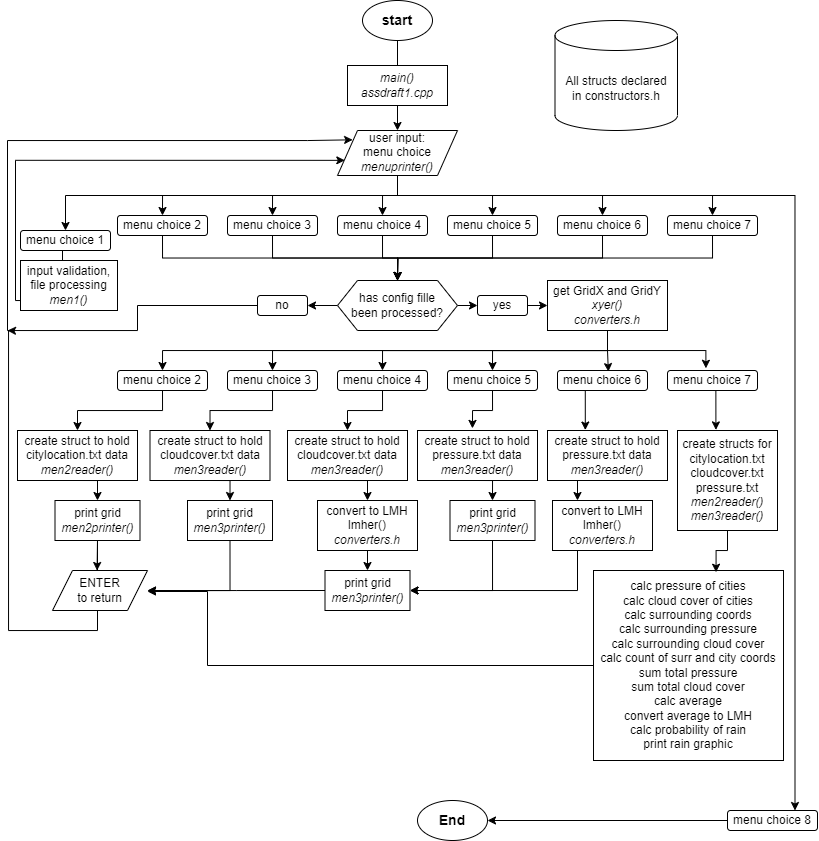
This is my program structure. assdraft.cpp is the main program file while converters.h and converters.cpp declare functions that transform values. All structs are declared in constructors.h.

**In the following diagrams, csci251\_a1.cpp is renamed to assdraft1.cpp. csci251\_a1.app is renamed to ass.app**

*I created these diagrams before renaming my files to the assignment’s required naming convention*



This is my program flowchart



There are primarily 5 functions in main(), and the rest are either my converter functions that are all declared in converters.h, or

# Extra features

Please may I have more marks ☺

## Input validation

### Main menu

User can only input 1,2,3,4, or 5. All else will be rejected and the program will not continue until valid user input is received. Program will continuously prompt for valid user input.

### Sorting menu

User can only input a,b,c, or q. All else will be rejected and the program will not continue until valid user input is received. Program will continuously prompt for valid user input.

### All shape data input

For all input that require numbers, only integers are allowed. All else will be rejected and the program will not continue until valid user input is received. Program will continuously prompt for valid user input.

Only exception is radius must be positive.

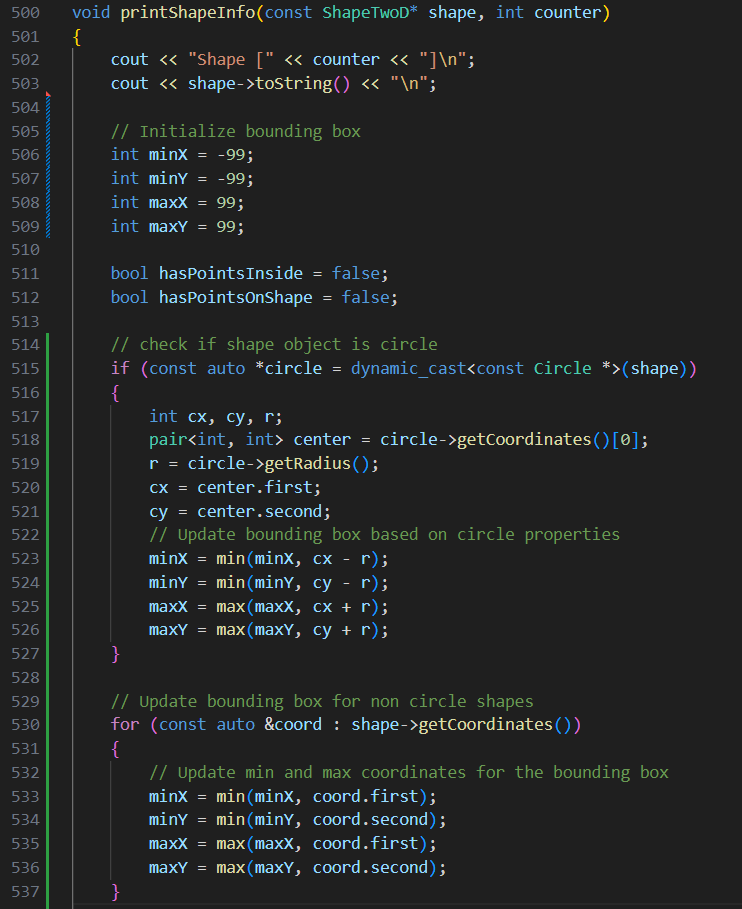
For special type input, only ws or ns is accepted.

For shape name, only square, rectangle, cross, or circle is accepted.

## Data visualisation

## Dynamic shape range

The shape coordinates can be between -99 and 99 for both x and y axis. However, the program will only iterate through a “bounding box” that fits the shape. Hence better performance than iterating through -99 to 99 x and y coordinates.



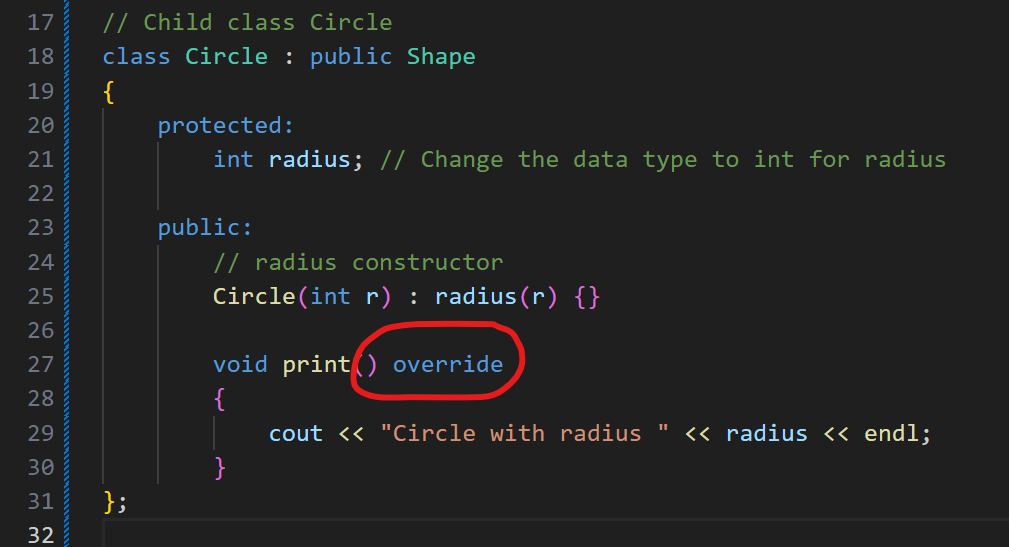
# Limitations

# Learnings

Some cool stuff I learnt

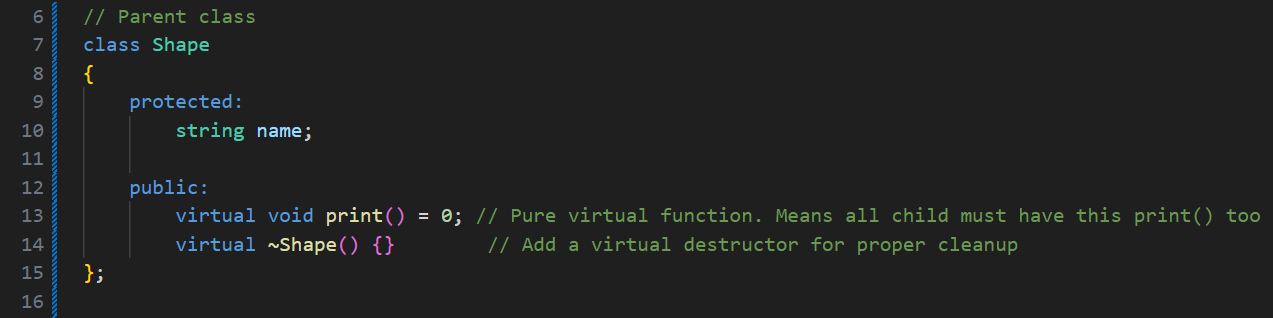
## Override

I understood from the lectures what is function overriding. Essentially function call of same name can invoke different behaviour based on the object type. However, I don’t remember this Override syntax being used. Maybe it’s a new C++ feature. Regardless, it’s something I had learnt while doing this assignment. This override keyword explicit signals that the specified function is being overridden. The code would work as per normal even if the override keyword is not used. It’s more for ease of understanding.



## Virtual functions

I think I did not understand fully in class but after writing out the code and taking reference from online sources, I have a better understanding of what virtual functions are. In the following context, basically print() in parent class shape does nothing, but declaring this virtual function means that all children classes must have a print() function. This also enables polymorphism because if there’s no print() in parent class Shape, then shape -> print() would not work.



## Abstract functions

Abstract function means that I cannot create an instance of it. In my context it means I cannot create an object of ShapeTwoD type directly.

## Unique\_ptr

Smart pointers! It’s basically a pointer pointer but used with vectors.

## static variable vs static function vs static variables in a class

I discovered that actually the single keyword static has multiple meanings depending on where it is used. If used simply in a function when declaring a variable, aka static variable, it means that the value of the variable is not destroyed when the function ends. The value remains persistent between function calls.

Meanwhile, when used while declaring a function, it has several characteristics. First, static functions in a class can be called without creating an object of the class. Second, static functions in a class cannot interact with non-static variables or functions.

Lastly, static variables in a class share its value among all derived objects.

Apparently when static is used correctly, it can save memory and optimise performance but upon further reading, it seems modern compilers can optimise the code in the background already thus making the use of static to save memory a bit redundant.

## ternary conditional operator

basically the ? operator. It’s a quick way to write an if statement in a single line. I would not use often since it is not very easily readable but it is useful in my assignment because it made the compareShapesByArea() function much cleaner. And the whole purpose of creating this function was to make my code cleaner.

The syntax is

return value ? (A < B) : (A> B);

If value is true, it will return the expression on the left of the : . if value is false, it will return the expression on the right of the : .

# Thoughts and reflections

## Purpose?

What is the purpose of function overriding and polymorphism? Perhaps for optimising games code for more fps? This feature of OOP seems so specific and niche I can’t foresee myself using this feature often unless I have a specific problem that requires it.

## Assignment difficulty

I don’t even need to hide this, I relied on online sources EXTENSIVELY for all logic. I could apply what I had learnt about classes, inheritance and polymorphism in class, however logic for is point in shape, is point on shape, and all the other logic for cross, these were extremely difficult to create on my own. Moreover, this goal of this assignment is to test understanding of classes and polymorphism so why is the logic segment so difficult? The goal is not to test algorithms and maths right? I don’t even understand some of the algorithms I used. I treated them like a black box.