SIM UOW CSCI251

ASSIGNMENT 3

Last updated: 13 February 2024

Timothy Mah

8750634

Lecture Group L01

Contents

[Introduction 2](#_Toc158814661)

[Usage 2](#_Toc158814662)

[Program flow 2](#_Toc158814663)

[Extra features 3](#_Toc158814664)

[Input validation 3](#_Toc158814665)

[Main menu 3](#_Toc158814666)

[Sorting menu 3](#_Toc158814667)

[All shape data input 3](#_Toc158814668)

[Learnings 3](#_Toc158814669)

[Compiler will omit trailing decimal point of a double if not needed 3](#_Toc158814670)

[Good diagram for reference 4](#_Toc158814671)

[Thoughts and reflections 4](#_Toc158814672)

[Purpose? 4](#_Toc158814673)

# Introduction

Hello this is my documentation for SIM UOW CSCI251 ASSIGNMENT 3.

# 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\_a2.app

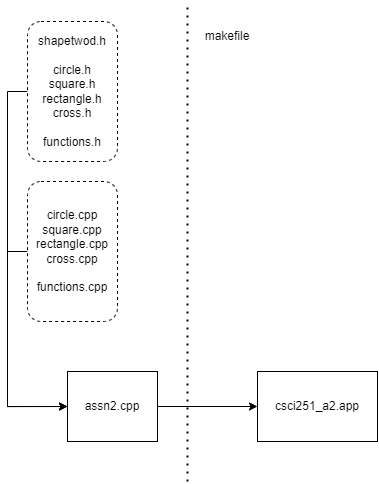
To compile,

make

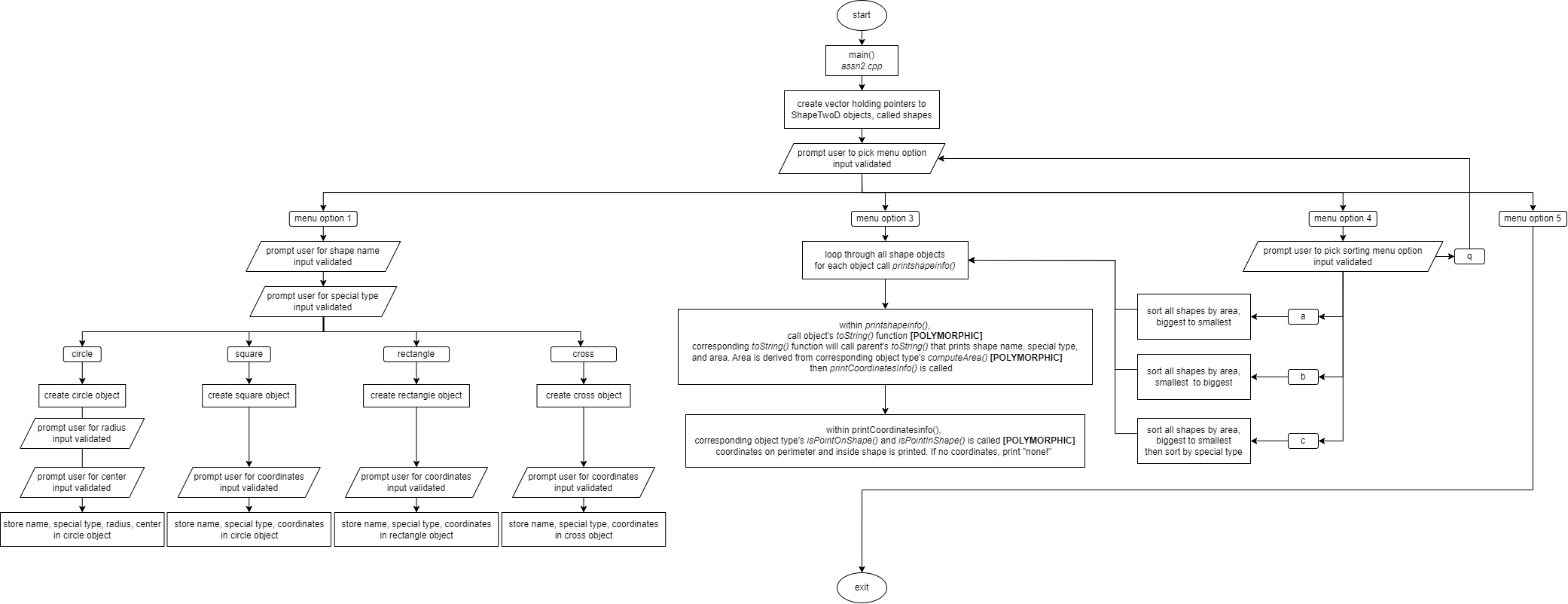
In same directory as makefile. Makefile should be in same directory as main file and the other helper files.

# Program flow

This is my program structure. shapetwod.h holds the parent class declaration. The 4 child classes have their declaration in their corresponding .h files. Functions.h is the holds the function prototypes of all my functions. The implementation of the child classes is in their corresponding .cpp files. The implementation of all my functions is in functions.cpp. Main() is in assn2.cpp. The compiled program is in csci251\_a2.app.



This is my program flowchart



# 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.

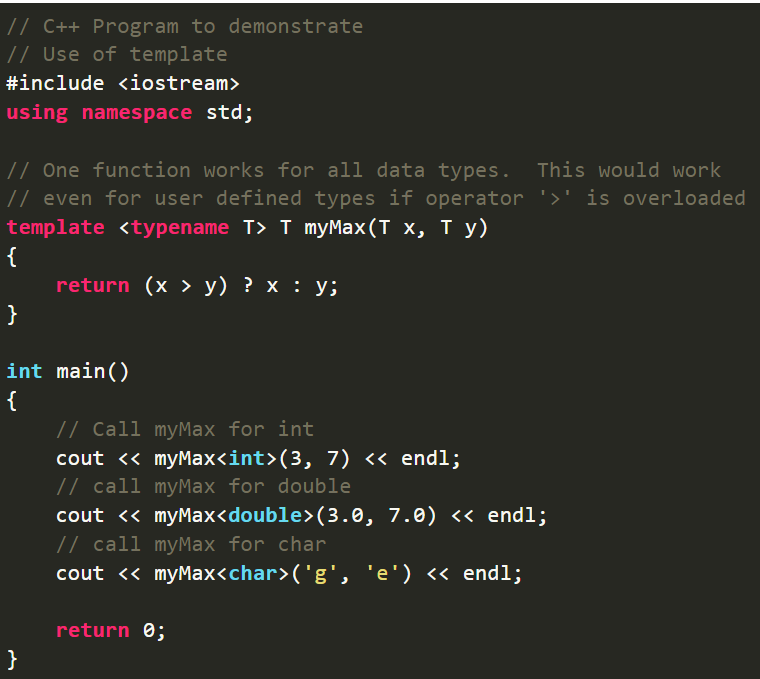
For shape name, only square, rectangle, cross, or circle is accepted.

# Learnings

Some cool stuff I learnt

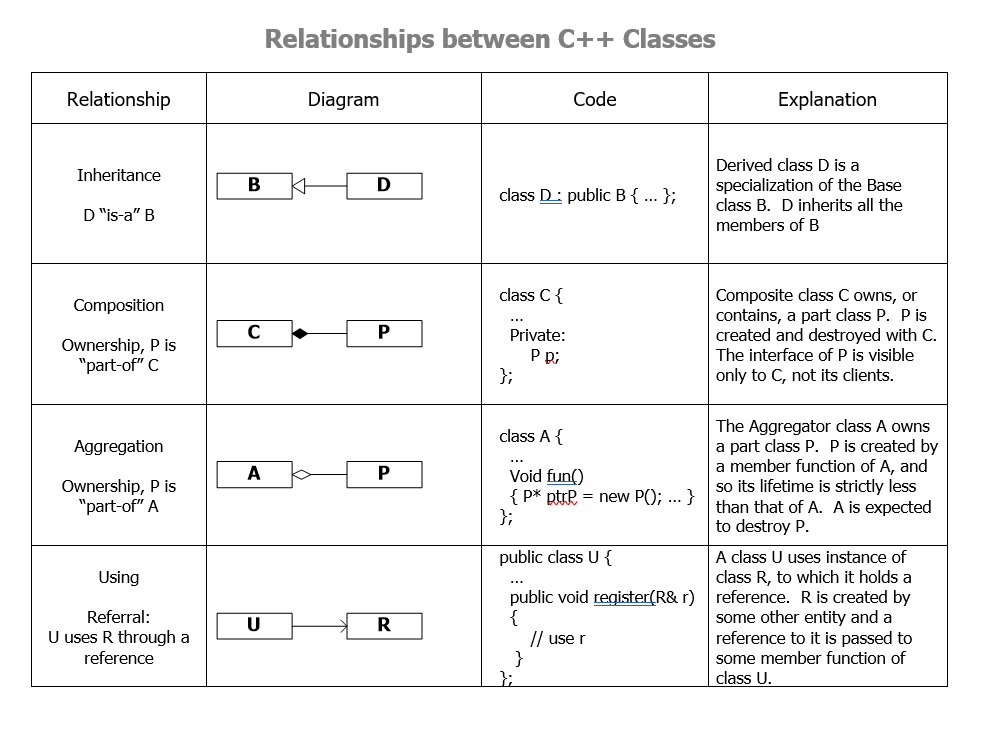
## Compiler will omit trailing decimal point of a double if not needed

I was reading up about function templates and I came across this code snippet



It’s a simple program to show the ability of function templates to pass datatype as a parameter when function is being called. I understand how it works but when I run this code snippet, the second cout output is 7. I had assumed it would be 7.0 since the return value is a double. However, upon further research, I discovered that it is because most C++ compilers, including g++ apparently, would omit trailing zeroes from a double when it is being cout. In the program the variable is stored as double yes, but the compiler will remove the trailing zeroes. I found this quite interesting behaviour

## Good diagram for reference

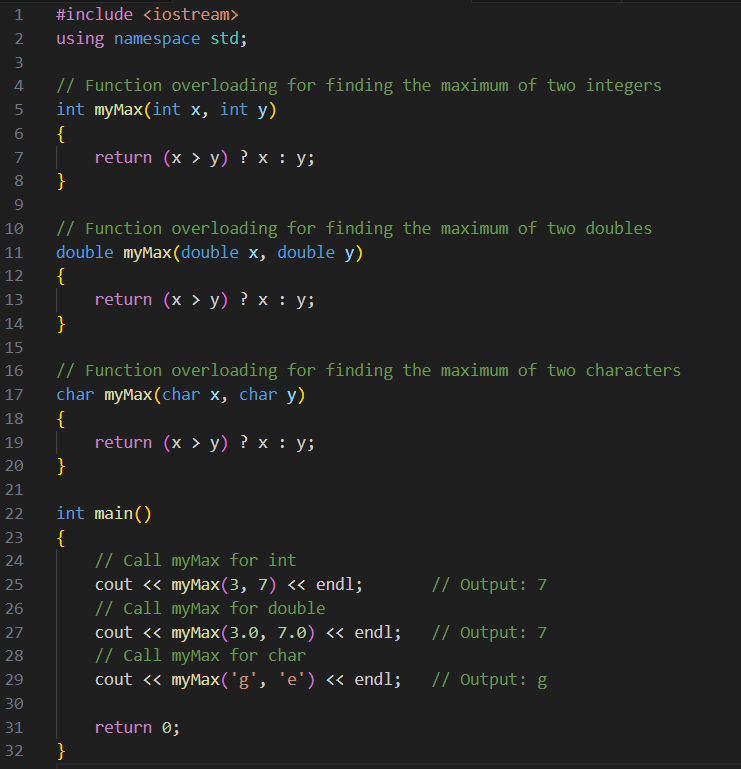


## Generic programming is like python?

Generic programming is an approach where generic data types are used as parameters in algorithms so that they work for variety of suitable data types. This is one component of the assignment. I suppose this is like python.

## Function template vs function overloading

This is how it would look like if the above code snippet about function templates, were to be re-written using function overloading

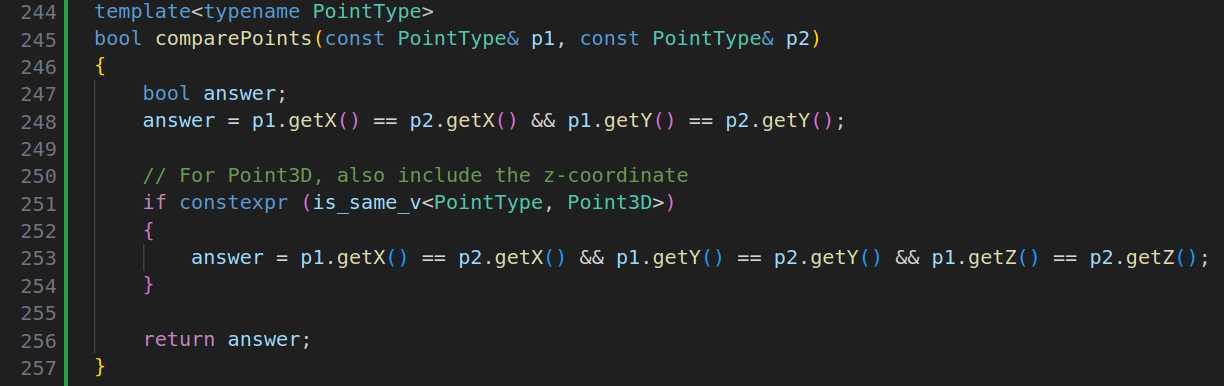


We can see that using templates is a cleaner solution since I don’t have to specify the function to accept the different data types as needed when performing function overloading.

This shows that although function templates and function overloading allows a function to behave differently based on parameter’s data type, function templates allow any data type, while function overloading requires the function declaration/ implementation to specify every scenario of data type to work.

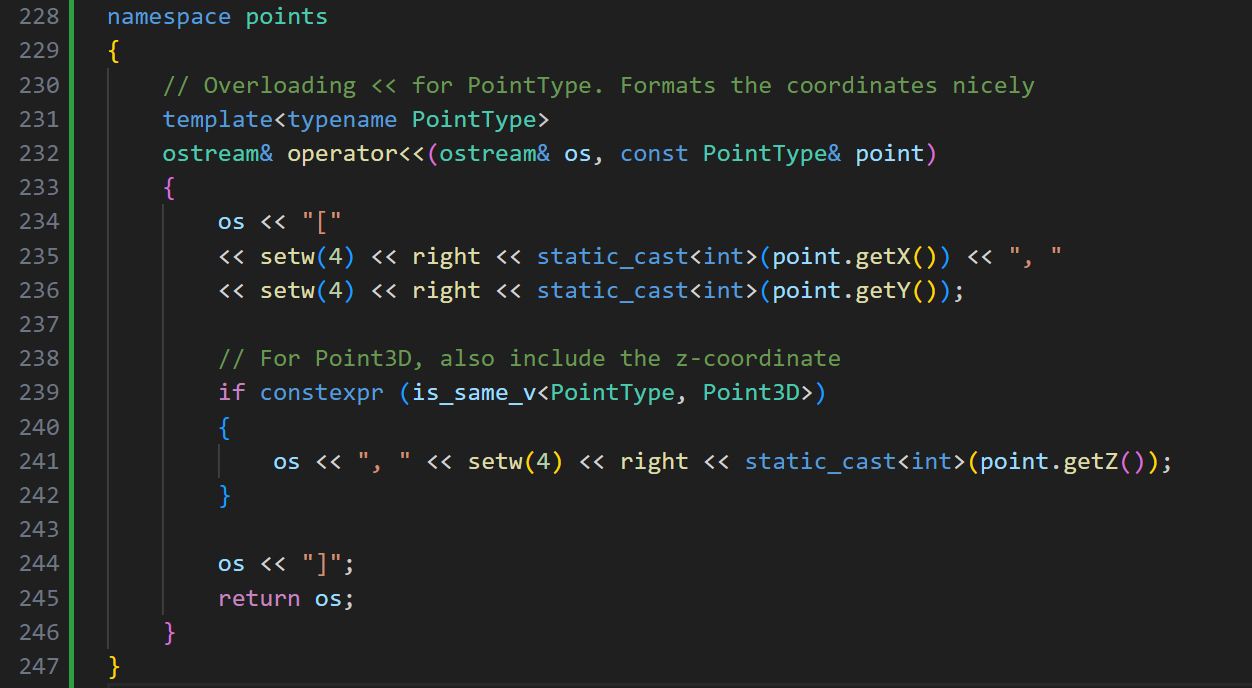
## if constexpr and is\_same\_v

These components allow me to create if statements within my template function to perform differently when a certain object type is used as template



## Cannot merge template functions and operator overloading together

Initially I wanted to write my function something like this. This function is an operator overload for << and it uses template as well. After many hours of debugging I realised that the problem was not so much with the code, but my system design. I realised that if my << overload takes in a template as parameter, it basically can take it any value thus it also has the same signature as a normal <<.



# Thoughts and reflections

## Purpose?