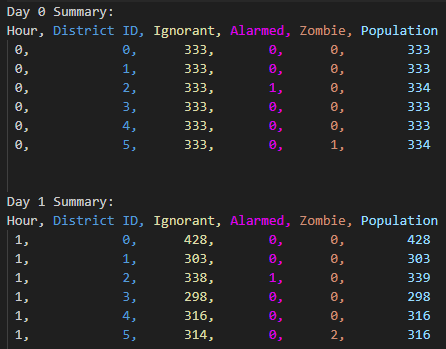
Our goal is to meet the requirement specifications for the Zombie Apocalypse and to push our ideas to what we want it to look like even with some visuals to make it more appealing. We started with creating a UML Diagram that will be the foundation in what we will need for our simulation to function. From our UML Diagram we built pseudo code header files as our layout (View Pseudo Code for base structure) then from here we began creating the cpp files.

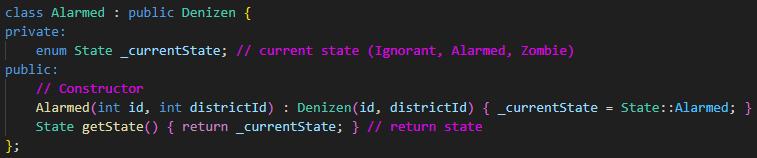
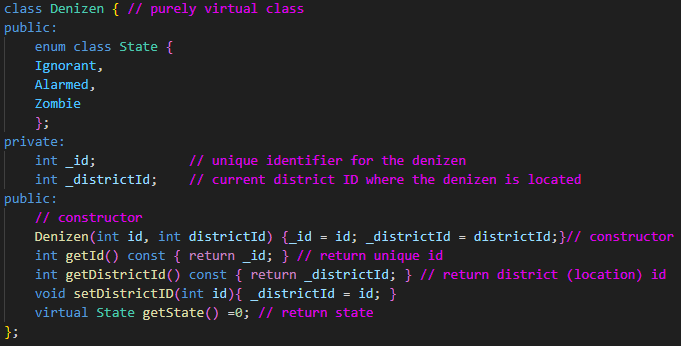
Our data that we created from the Simulation is then transferred over to a csv file where it is laid out as a table of.. Day (x) Summary: with Hour, DistrictID, Ignorant, Alarmed, Zombie and total Population shown below is an example.

Ex: 

**Requirement Specifications:**

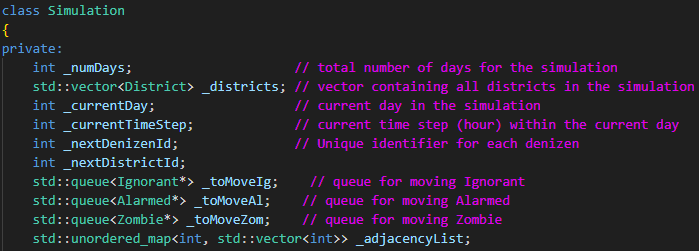
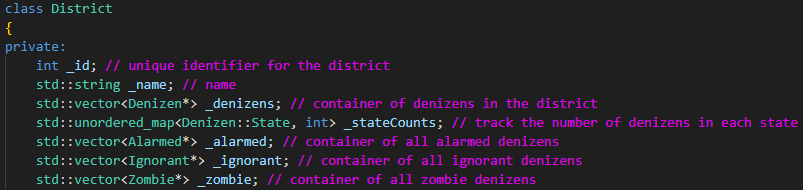
**Implementation:**

*Use of class inheritance and polymorphism*  
Our Alarmed, Ignorant, and Zombie classes inherit the vast majority of their functions and attributes from the Denizen class, which is a purely virtual class. With this polymorphic set up, Alarmed, Ignorant and Zombie objects can all exist in vectors at the district level for Denizen objects.

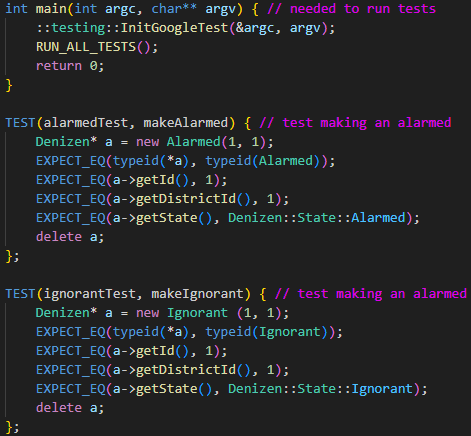
Ex: 

*Use of STL sequential data structures, associative data structures, and (BONUS) queues*

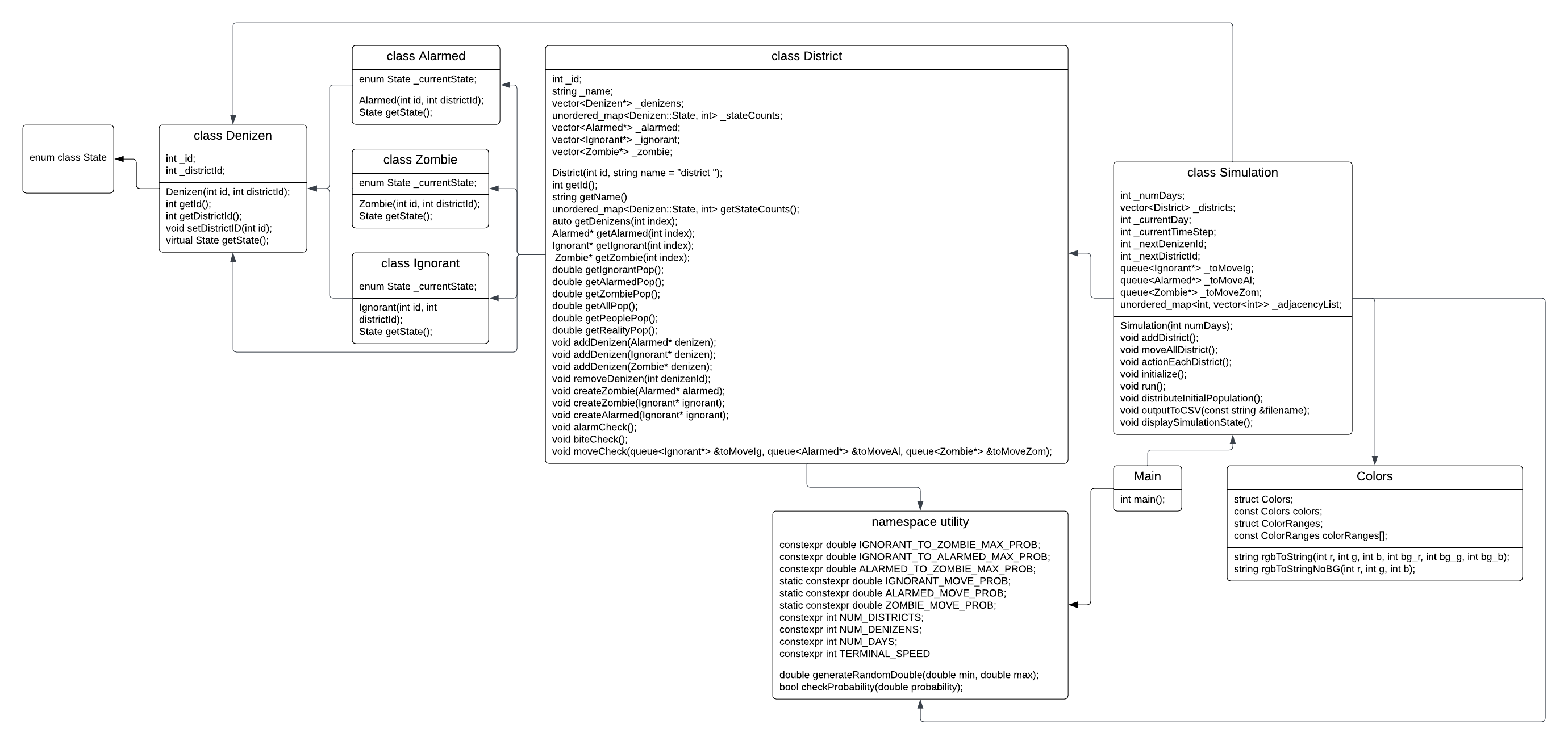
We utilized vectors of pointers to Denizen, Alarmed, Ignorant, Zombie, and District objects. We utilized unordered maps both to create our adjacencies for the districts at the simulation level and for the counts of our different states at the district level. We utilized queues both inside functions within district and for moving pointers for denizens between districts at the simulation level.

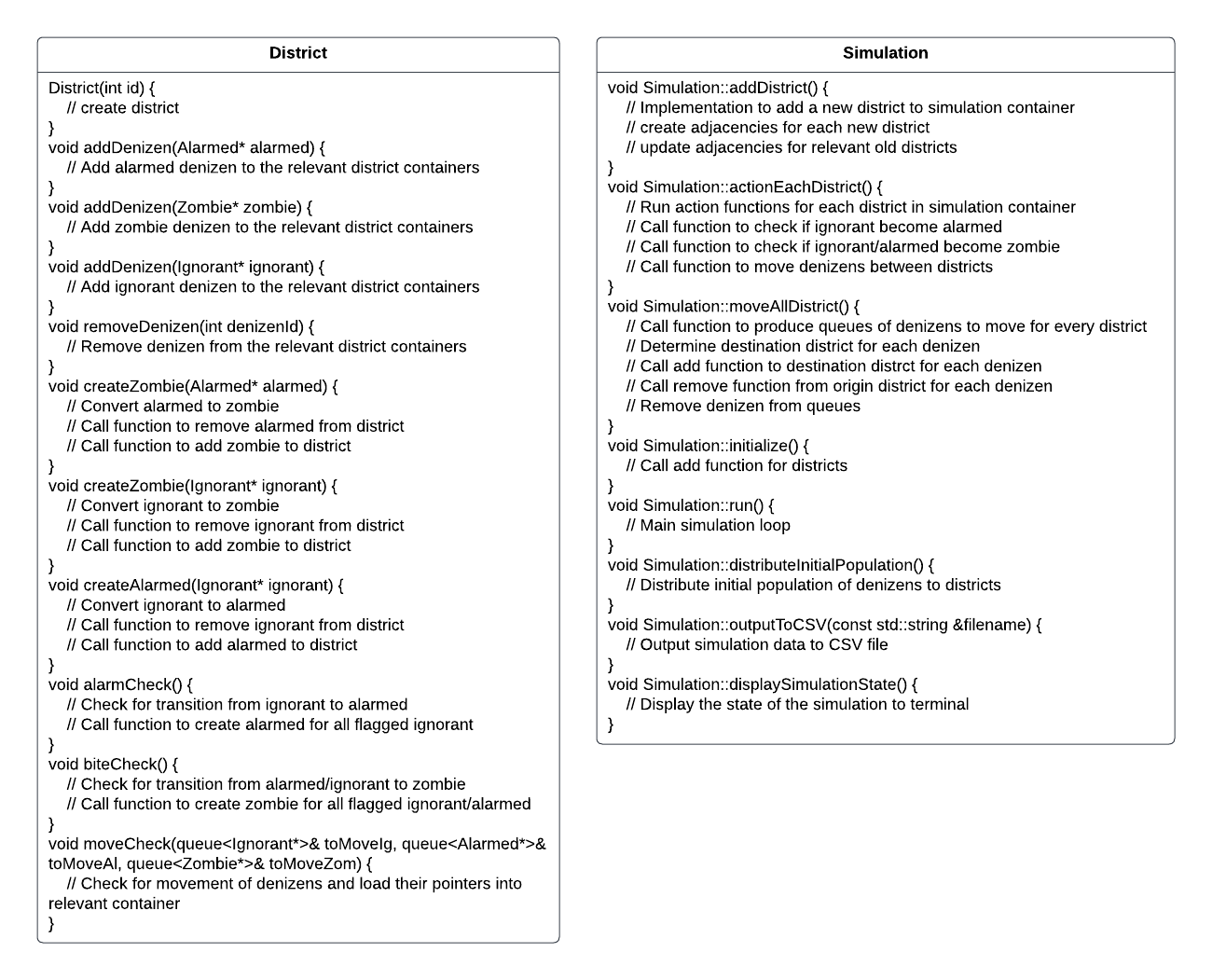
Ex: 

*Development of unit tests for at least the major units*  
We created unit tests for creating all the different class objects we’d be using, to test the functions that added/removed Alarmed, Ignorant, and Zombie objects from the District containers, and to test the functions that would copy information of one Denizen subclass in a new object of a different subclass and delete the new object.

Ex: 

**The Final state of our design:**  
*UML Diagram*

  
*Psuedo Code*



The initial state of our design:

A diagram of a data flow

Description automatically generated with medium confidence

We used some pseudo code to figure out our layout for each header file trying to figure what is needed for a functional header file. Down below are some of our pseudo code we used to build our project

INCLUDE <vector>

IFNDEF DENIZEN\_H

#define DENIZEN\_H

CLASS Denizen: // purely virtual class

PUBLIC:

ENUM CLASS State:

Ignorant,

Alarmed,

Zombie

PRIVATE:

INT \_id // unique identifier for the denizen

INT \_districtId // current district ID where the denizen is located

PUBLIC:

// constructor

METHOD Denizen(int id, int districtId):

\_id = id

\_districtId = districtId

// return unique id

METHOD INT getId() CONST:

RETURN \_id

// return district (location) id

METHOD INT getDistrictId() CONST:

RETURN \_districtId

METHOD VOID setDistrictID(int id):

\_districtId = id

// return state

METHOD ABSTRACT State getState():

ENDIF

INCLUDE "denizen.h"

IFNDEF ALARMED\_H

#define ALARMED\_H

CLASS Alarmed INHERITS Denizen:

PRIVATE:

ENUM State \_currentState // current state (Ignorant, Alarmed, Zombie)

PUBLIC:

// Constructor

METHOD Alarmed(int id, int districtId):

Denizen(id, districtId)

\_currentState = State::Alarmed

// return state

METHOD State getState():

RETURN \_currentState

ENDIF