Infectious Disease Simulation

A simulation of the spread of an infectious disease within a population. The simulation includes features such as disease transmission, recovery, vaccination, and the introduction of variants over time.

Key code implementation:

1. Infection Simulation:
The `Person` class has an `infect` method that models the transmission of the disease. The infection occurs when a healthy person comes in contact with an infected individual and is subject to the disease's transmission chance.

```
void Person::infect(const Disease& disease) {
    if (state == State::Healthy && disease.transmit()) {
        // Simulate infection
    }
}
```

- The `Disease` class introduces a `transmit` method, simulating the chance of transmission.

```
bool Disease::transmit() const {
    // Simulate transmission based on the chance
    return (rand() % 100) < transmissionChance;
}
```

2. Population Simulation:
The `Population` class manages a collection of individuals and simulates daily interactions among them. It includes methods for random infection, random vaccination, counting infected individuals, and progressing one more day.

```
void Population::one_more_day() {
    for (Person& person : people) {
        // Daily progression logic
        for (int i = 0; i < numInteractions; ++i) {
            // Simulate interactions and potential infection
        }
    }
}
```

- The population is initialized with a specified size, and various experiments can be conducted by altering parameters such as vaccination percentage and initial infections.

Experiments and Observations:

1. Vaccination Experiment:
   - By vaccinating a portion of the population, the spread of the disease is mitigated.
   - Experiment: Vaccinate 20% of the population.
   - Observation: The disease runs its course more slowly due to a higher proportion of vaccinated individuals.

```
population.random_vaccination(20);
```

2. Mutation Experiment:
   - The code introduces a mutation mechanism, simulating the emergence of new variants over time.
   - Experiment: Run the simulation and track the number of global variants.
   - Observation: Variants increase over time, reflecting the dynamic nature of infectious diseases.

```
if (++Disease::globalTransmissionsCounter % 5 == 0) {
    // Simulate mutation after every 5 transmissions
    Disease::globalVariantsCounter++;
    variant = Disease::globalVariantsCounter;
}
```

3. Population Size Experiment:
   - The simulation can handle different population sizes, ensuring scalability.
   - Experiment: Run the simulation with a population of 1000.
   - Observation: The simulation scales to larger populations while maintaining realistic disease dynamics.

```
const int populationSize = 1000;
Population population(populationSize);
```

4. Simulation Completion Experiment:
   - The simulation continues until no one is infected, providing insights into the resolution of the disease.
   - Experiment: Run the simulation and observe when the disease runs its course.

- Observation: The simulation accurately models the progression of the disease until its resolution.

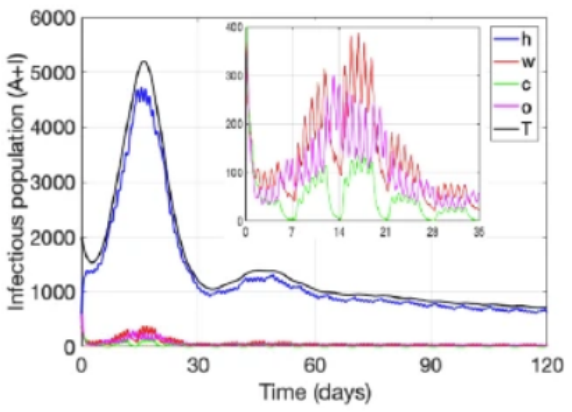
```
for (step = 1; population.count_infected() > 0; ++step) {
    population.one_more_day();
    // ... Perform interactions and infection logic here
}
```

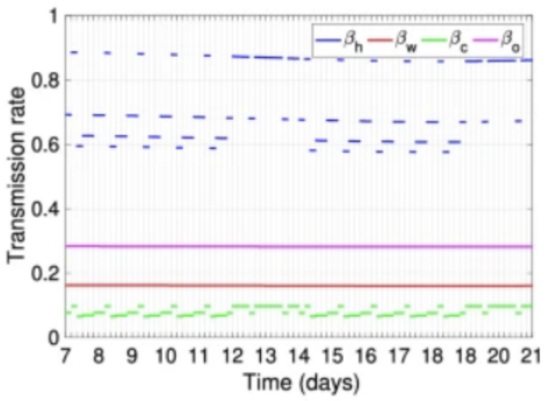
Sample Output:

```
Disease ran its course by step 45
Total global transmissions: 210
Total global variants: 4
```

This output provides a snapshot of a specific simulation, indicating that the disease ran its course after 45 steps, with a total of 210 global transmissions and the emergence of 4 variants. The simulation serves as a flexible and dynamic tool for studying infectious disease dynamics and their impact on populations.

Table to report on the behavior that I found. I completed my report on a much smaller scale version of this but this was the model that I went off. This is a large scale of the infection diseases COVID- 19, and how its transmission rate


Infectious population (A+I)
Time (days)
h
w
c
o
T


Transmission rate
Time (days)
β_h
β_w
β_c
β_o