This code simulates an **Ant Colony Simulation**, which models the behavior and interactions of ants within an ant farm. The simulation includes multiple components, such as ant species, different types of ants, ant farms, and the overall meadow ecosystem. Below is a detailed explanation:

**Purpose and Features**

1. **Ant Colony Simulation:**
   * The program models multiple ant colonies, where each colony is represented as an **AntFarm**.
   * Each farm has a **queen**, different types of ants (drones, warriors), and **rooms** with specific functionalities (spawning, resting, storage, and battle).
2. **Species Diversity:**
   * Different ant species are created, each with specific bonuses (strength, efficiency, and harvest bonuses).
   * The Meadow singleton manages these species and ensures diversity across farms.
3. **Ant Behavior:**
   * Each ant type (DroneAnt, WarriorAnt, QueenAnt) has unique behavior determined by the act method:
     + Resting when tired.
     + Consuming food to sustain activity.
     + Working to perform tasks in the colony.
4. **Simulation of Time (Ticks):**
   * The program uses a **tick-based simulation**, where each "tick" represents a unit of time.
   * During each tick, all ants act according to their behavior, and food consumption and energy levels are updated.
   * The simulation ends when one or fewer active colonies remain or the tick count reaches a maximum limit.
5. **Colony Survival:**
   * Colonies are deactivated if they cannot sustain their ants due to insufficient food or other conditions.
   * The program tracks the status of each farm, ensuring realistic interactions between resources and ant behavior.
6. **Room Management:**
   * Ants are assigned to rooms, which have capacities and functions. These rooms simulate the organizational structure of an ant colony.
7. **Meadow Ecosystem:**
   * The Meadow class serves as a central management system for all ant farms.
   * It controls the simulation flow and ensures the correct initialization and progress of the simulation.

**Code Components**

**1. Enumerations:**

* **AntType**: Defines the type of ants (DRONE, WARRIOR, QUEEN).
* **RoomType**: Defines the type of rooms (SPAWNING, RESTING, STORAGE, BATTLE).

**2. Core Classes:**

* **Species**:
  + Represents different ant species, each with unique bonuses.
  + Encapsulates species attributes like strength, efficiency, and harvest bonuses.
* **Ant**:
  + Base class for ants, providing shared functionality like food consumption, energy management, and actions (rest, work).
  + Specialized into DroneAnt, WarriorAnt, and QueenAnt with specific behaviors.
* **Room**:
  + Represents different areas within an ant farm. Each room has a type, capacity, and a list of ants.
* **AntFarm**:
  + Represents an ant colony, holding a queen, various ants, rooms, and the colony's food supply.
  + Handles colony dynamics like adding ants, managing food consumption, and determining colony activity status.
* **Meadow**:
  + Singleton class that oversees the simulation, managing all ant farms and species.
  + Contains the main simulation loop, ensuring the progression of the simulation.

**Ant Colony Simulation**

**Description:**  
This project is an object-oriented simulation of ant colonies. It models the interactions and behavior of ants within different ant farms using concepts like inheritance, polymorphism, and encapsulation. The simulation is tick-based, evolving dynamically based on the food supply, energy levels, and actions of ants.

**Key Features:**

* Diverse species with unique attributes.
* Realistic ant behaviors (resting, working, consuming food).
* Ant farm management with queens, drones, and warriors.
* Room-based functionality for colony organization.
* Centralized ecosystem management through the Meadow singleton.

**Technologies:**

* **Language**: C++
* **Paradigms**: Object-Oriented Programming (OOP)

**Usage:**

1. Clone the repository and compile the code using a C++ compiler (e.g., g++, Clang).
2. Run the program to simulate ant colonies and observe their behaviors over time.

**Learning Objectives:**

* Understand object-oriented design principles.
* Learn about managing complex systems using classes and inheritance.
* Explore simulation techniques using tick-based updates.