**1 Libraries**

#include <iostream>

#include <vector>

#include <algorithm>

#include <cstdlib>

#include <ctime>

#include <chrono>

#include <thread>

**2. Constants**

The code includes necessary libraries for input/output, data structures, random number generation, time, and threading.

const int SCREEN\_WIDTH = 800;

const int SCREEN\_HEIGHT = 600;

**3. Cat Class**

These constants define the dimensions of the game screen.

class Cat { // here are attributes of class cat

public:

int x;

int y;

double happiness;

double hydration;

double energy;

Cat(int x, int y) : x(x), y(y), happiness(100), hydration(100), energy(100) {}

// Methods that is for move the cat, update its cores, and check if it's depleted

};

The **Cat** class **represents a cat object in the game**. It has member variables for its position (**x** and **y**) and three core attributes: happiness, hydration, and energy. The constructor initializes the cat's position and sets the core attributes to their maximum values. The class also includes methods to move the cat randomly, update its core attributes over time, and check if any of the cores are fully depleted.

**4. Item Class**

class Item {

public:

std::string name;

double price;

double happinessBoost;

double hydrationBoost;

double energyBoost;

Item(std::string name, double price, double happinessBoost, double hydrationBoost, double energyBoost)

: name(name), price(price), happinessBoost(happinessBoost), hydrationBoost(hydrationBoost),

energyBoost(energyBoost) {}

};

The **Item** class represents an item that the player can buy in the game. It has member variables for the item's name, price, and core boosts (happiness, hydration, and energy). The constructor initializes these member variables.

**5. Main Function**

int main() {

// Initialize random seed

// Create vectors to store cats and items

// Initialize player's money and its increasing rate

// Game loop

// Add a new cat

// Update the cats

// Check if any cat's core is fully depleted

// Display the number of cats on the screen

// Display the player's money

// Display the cat cores for each cat on the screen

// Display the available items for purchase

// Player buying an item

// Increase the player's money

// Wait for a moment

// End of the game loop

return 0;

}

The **main()** function is the entry point of the program. It initializes the random seed, creates vectors to store cats and items, and initializes the player's money and its increasing rate.

The **main part of the function is the game loop**, which runs indefinitely until the program is terminated. Inside the loop, it performs the following actions:

* Adds a new cat to the screen with a certain probability.
* Updates the positions and core attributes of the cats.
* Checks if any cat's core is fully depleted and removes it from the vector.
* Displays the number of cats on the screen and the player's current money.
* Displays the core attributes for each cat on the screen.
* Displays the available items for purchase.
* Simulates the player buying an item with a certain probability.
* Increases the player's money.
* Waits for a moment before the next iteration of the loop.

**6. Additional Functionality**

The additional functionality added for the game is implemented within the game loop.

* Randomly adding cats to the screen: In each iteration of the game loop, there is a certain probability to add a new cat to the **cats** vector.
* Updating cat cores: The core attributes of each cat in the **cats** vector are continuously decremented over time.
* Checking and removing depleted cats: After updating the cat cores, the program checks if any cat's core is fully depleted. If so, the cat is removed from the **cats** vector.
* Displaying cat cores: The program displays the core attributes (happiness, hydration, and energy) for each cat on the screen.
* Displaying available items for purchase: The program displays the available items and their prices for the player to buy.
* Player buying items: With a certain probability, the program simulates the player buying an item. If the player has enough money to buy the selected item, the item's price is deducted from the player's money, and the core attributes of all cats are increased based on the item's boosts. The purchased item is then removed from the available items vector.
* Increasing player's money: The player's money is increased continuously at a predefined rate.
* Waiting for a moment: The program waits for a short duration (3 second) before the next iteration of the game loop to control the speed of the game.