CPP Problem Design Example

Subject: Observation Diary

Contributor: 陳宥潤,鄭永泰,范茗翔

Main testing concept: Class Design and Operator Overloading.

Basics

□ C++ BASICS
□ FLOW OF CONTROL
□ FUNCTION BASICS
□ PARAMETERS AND OVERLOADING
□ ARRAYS
■ STRUCTURES AND CLASSES
■ CONSTRUCTORS AND OTHER TOOLS
■ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES
□ STRINGS
□ POINTERS AND DYNAMIC ARRAYS

Functions

□ SEPARATE COMPILATION AND NAMESPACES
□ STREAMS AND FILE I/O
□ RECURSION
□ INHERITANCE
□ POLYMORPHISM AND VIRTUAL FUNCTIONS
□ TEMPLATES
□ LINKED DATA STRUCTURES
□ EXCEPTION HANDLING
□ STANDARD TEMPLATE LIBRARY
□ PATTERNS AND UML

Description:

You are experimenting on several unknown creatures to observe their evolution.
Design class “Creature” to store the status of appendages (body parts) for each creature.
Design class “Diary” to store individual logs kept for each creature.
You may modify everything except the provided “main.cpp” .

Input:

The main function will be replaced for each test case.
There are several available instructions, while the first line must be NewDay():

```
Diary::NewDay("Date???"); // Change the day of diary

Creature creatureA("CreatureA"); // A new creature named "CreatureA"

Creature creatureB("CreatureB", creatureA); // A clone (body parts) of creatureA
named "CreatureB"

creatureA["BodyPartA"] = integerX; // Set the number of "BodyPartA" of "CreatureA"

creatureA["BodyPartA"] += integerY; // Add the number of "BodyPartA" of "CreatureA"

creatureA["BodyPartA"] -= integerZ; // Subtract the number of "BodyPartA" of
"CreatureA"

creatureA.PrintStatus(); // Print the current status of creatureA

creatureA.PrintLog(); // Print the log of creatureA
```

Output:

1. Format of PrintStatus(): Look at sample 1.
   Print the name and number of existing appendages (number > 0) **sorted by name (string) in ascending order.**
2. Format of PrintLog(): Look at sample 2.
   A diary (or log) starts from the target’ s creation and is not copied during cloning.

Log day information when a creature is created and when NewDay() is called.
Log the change and values when the number of any appendage changes.
(appeared (0 -> X) / disappeared (X -> 0) / increased (X -> X + Y) /
decreased (X + Y -> X))

Output a new line after PrintStatus() and PrintLog().

Sample Input / Output：

Main	Sample Output

Diary::NewDay("-4500m");
Creature dog("Dog");
dog["tail"] = 1;
dog["leg"] += 4;
dog["antenna"] = 0;
dog["head"] = 3;
dog.PrintStatus();
	Dog’s status:
	head * 3
	leg * 4
	tail * 1

Diary::NewDay("00");
Diary::NewDay("01");
Creature fox("Fox");
fox["tail"] += 1;
fox["tail"] -= -8;
fox["tail"] = 9;
Diary::NewDay("10");
fox["tail"] += -8;
fox["tail"] = 0;
Diary::NewDay("11");
fox.PrintLog();
	Fox’s log:
	Day 01
	Fox’s tail appeared (0 -> 1).
	Fox’s tail increased (1 -> 9).
	Day 10
	Fox’s tail decreased (9 -> 1).
	Fox’s tail disappeared (1 -> 0).
	Day 11

Diary::NewDay("0000");
Creature unknownA("UA");
unknownA["leg"] = 16;
 	
Diary::NewDay("0102");
Creature unknownB("UB",
unknownA);
unknownB["leg"] += 26;
unknownA.PrintLog();
 	
Diary::NewDay("0227");
unknownA["leg"] = 0;
unknownA.PrintStatus();
unknownB.PrintLog();
 	
Diary::NewDay("0353");
unknownA["leg"] += 6;
unknownA["wing"] += 4;
unknownA.PrintLog();
	UA’s log:
	Day 0000
	UA’s leg appeared (0 -> 16).
	Day 0102
	
	UA’s status:
	
	UB’s log:
	Day 0102
	UB’s leg increased (16 -> 42).
	Day 0227
	
	UA’s log:
	Day 0000
	UA’s leg appeared (0 -> 16).
	Day 0102
	Day 0227
	UA’s leg disappeared (16 -> 0).
	Day 0353
	UA’s leg appeared (0 -> 6).
	UA’s wing appeared (0 -> 4).

□ Eazy,Only basic programming syntax and structure are required.

■ Medium,Multiple programming grammars and structures are required.

□ Hard,Need to use multiple program structures or more complex data types.

Expected solving time:

50 minutes

Other notes:

Any appendages numbers don’t less than 0.

When you clone some creature,the content of log won’t copy but the appendages and number will copy.

Don’t change the names of the class "Creature" and "Diary".