






Important: Add meme
ntut-xuan authored 2 weeks ago

Name	Last commit	Last update
 README.md	Important: Add meme	2 weeks ago
 image_test.png	Fix the old image	2 weeks ago
 meme.jpg	Important: Add meme	2 weeks ago

 [README.md](#)

Homework 07

This homework was created by 黃漢軒 (109590031), please feel free to ask me if you have any questions.

Email: t10950031@ntut.org.tw / MS Teams 黃漢軒

⚠ Due: 11:59 p.m., 29 / 12 / 2022 ⚠

Goal

This homework has these goals:

- Know the concept of static, static member.
- Know the concept of template.
- Know the concept of `to_string()`.

Folder Structure Tree

- You should finish the unit test written by you.
- You can split the unit test into multiple files, just remember to include all of it into `ut_main.cpp` (see course repo).

While your project has been built by `makefile`, the structure tree should be the same as the following section.

```
bin/  
├─ ut_all  
src/  
├─ cake_showcase.h  
├─ cake.h  
├─ chocolate_cake.h  
├─ geode_cake.h  
├─ honey_cake.h  
test/  
├─ <some test file>  
├─ ut_main.cpp  
makefile
```

Problem Content

Since Uriah is a *Tainanese*, Uriah love to eat cake. (Actually, who hate?)

Cake have a lot of type, but Uriah only like eats three type of cake: Chocolate Cake, Honey Cake, and Geode Cake.

Uriah have it owns cake showcase, which can show specific cake, and it can return some statistics value or show all the cake with string.

In this task, you should complete two requirement: [Showcase Implement](#) and [Cake Count Record](#).

Showcase Implement

In this task, you should help Uriah to build his cake showcase, the cake showcase need to satisfy these requirement:

- The cake showcase can show the specific type of cake.
- The cake showcase can calculate the cake total price and total sweet.

- The cake showcase can print the current storage with string.

Cake Count Record

Uriah also want to know how many of cake with specific type has been made.

- The cake with specific type should record how many specific type of cake exists.
- The cake class should record how many cake exists.

For example, if we constructor 3 Chocolate Cake and 3 Geode Cake,

the total of Cake should be 6, the count of Chocolate Cake should be 3, and the count of Geode Cake should be 3.

If we call the destructor to destruct a Chocolate Cake and a Geode Cake,

the total of Cake should be 4, the count of Chocolate Cake should be 2, and the count of Geode Cake should be 2.

You should satisfy the requirement above.

When we call the static function of Cake, Chocolate Cake, Honey Cake and Geode Cake, it should return the static member of count or total.

Task

In this task, you should create 5 class: `ChocolateCake`, `HoneyCake`, `GeodeCake` inheritance `Cake`, and `CakeShowcase`.

The public member listed in below.

- `class Cake`
 - `Cake(int price, int sweet)`
 - The constructor to initialize Cake.
 - `int get_price()`
 - Return the Cake price.
 - `int get_sweet()`
 - Return the Cake sweet.
 - `static int get_total()`
 - Return the total of cake.
 - `virtual std::string to_string() = 0`
 - The pure virtual function to get the string with derived class string.
- `class ChocolateCake, inheritance Cake`
 - `ChocolateCake(int price, int sweet)`
 - The constructor to initialize Chocolate Cake.
 - `std::string to_string()`
 - The function to return the string description of Chocolate cake.
 - It should use the following format:

	ChocolateCake		55		40	
--	---------------	--	----	--	----	--

Where 55 is price and 40 is sweet. You can use `sprintf` function to help you get the formatting function.

- `static int get_count()`
 - Return the count of Chocolate Cake.
- `class HoneyCake, inheritance Cake`
 - `HoneyCake(int price, int sweet)`
 - The constructor to initialize Honey Cake.
 - `std::string to_string()`
 - The function to return the string description of Honey Cake.

- It should use the following format:

HoneyCake		55		40	
-----------	--	----	--	----	--

Where 55 is price and 40 is sweet. You can use `printf` function to help you get the formatting function.

- `static int get_count()`
 - Return the count of Honey Cake.
- `class GeodeCake, inheritance Cake`
 - `GeodeCake(int price, int sweet)`
 - The constructor to initialize Geode Cake.
 - `std::string to_string()`
 - The function to return the string description of Geode Cake.
 - It should use the following format:

GeodeCake		55		40	
-----------	--	----	--	----	--

Where 55 is price and 40 is sweet. You can use `printf` function to help you get the formatting function.

- `static int get_count()`
 - Return the count of Geode Cake.
- `class CakeShowcase, should use template (CakeShowcase<T>)`
 - `CakeShowcase(int length, T** array)`
 - Initialize array and length of the showcase.
 - `int calc_total_price()`
 - Calculate the total price in cake showcase.
 - `int calc_total_sweet()`
 - Calculate the total sweet in cake showcase.
 - `std::string to_string()`
 - The function to return the string description of Cake Showcase.
 - For example, if we have 3 Chocolate cake in Chocolate Cake Showcase, it should use the following format:

CakeName		Price		Sweet	
-----		-----		-----	
ChocolateCake		45		15	
ChocolateCake		45		15	
ChocolateCake		45		15	
-----		-----		-----	

Where 55 is price and 40 is sweet. You can use `printf` function to help you get the formatted string.

- `T* operator[](int index)`
 - The operator overloading to get the pointer of showcase element.
 - `throw std::out_of_range` if index is out of range of the array.

You should **delete the default constructor** and **implement destructor** in every class.

For every cake class, if a cake has been constructed, you should increase the total of the cake and the count of the specific cake.

Also, if a cake has been destructed, you should decrease the total of the cake and the count of the specific cake.

Test

In this homework, you should use `gcovr` tool to make sure your *code coverage* in `/src` is all above 90%\$.

- If your lines of *code coverage* are below 90%\$, you will receive `FAILURE` in the HW Job.

File	Lines			Functions		Branches	
cake.h	<div><div></div></div>	100.0%	13 / 13	100.0%	5 / 5	-%	0 / 0
cake_showcase.h	<div><div></div></div>	100.0%	26 / 26	100.0%	5 / 5	100.0%	18 / 18
chocolate_cake.h	<div><div></div></div>	100.0%	11 / 11	100.0%	4 / 4	100.0%	1 / 1
geode_cake.h	<div><div></div></div>	100.0%	11 / 11	100.0%	4 / 4	100.0%	1 / 1
honey_cake.h	<div><div></div></div>	100.0%	11 / 11	100.0%	4 / 4	100.0%	1 / 1

Generated by: [GCOVR \(Version 5.2\)](#)

****You will get the 35% score if HW Job passed, otherwise, you will lose the 35% score if HW Job failed. ****

See the course slide ([OOP_gcovr.pptx](#)) to know how to install and how use it.

Notice

- Use [nullptr](#) if you want to have a null pointer, which is a special pointer that doesn't point to anything.
- Use `ASSERT_EQ` to test integers, `ASSERT_NEAR` to test floating-point numbers, and `ASSERT_THROW` to test exceptions.
- You should neither add a bin folder to your git nor add a file with the name '.gitignore' in the bin folder (see our class repo).
- In some situations you will lose score:

- **You lose 5 points for each test that has a memory leaks. You can check memory leak with `valgrind` cmd.**

```
valgrind --track-origins=yes --leak-check=all <executable_file>
```

- **You will lose 10% if your bin folder contains compiled ut_all in the git repo.**

Meme

