Smart Pointers & C++ STL

Lab 12

TA : Changmin Jeon, Minji Kim, Hyunwoo Jung,
Hyunseok Oh, Jingyu Lee, Seungwoo Jo

SEOUL NATIONAL UNIVERSITY

Objectives

- Learn concepts of smart pointers
- Practice C++ STL

Smart Pointers

- In large programs with many programmers, it is hard to track all the pointers.
- Failing to handle pointers can lead to memory leak. Sometimes it causes fatal problems.

Smart Pointers

- C++ introduced **smart pointers** to avoid memory leak problems.
- Smart pointers are used to make sure that an object is deleted if it is no longer referenced. Programmers don’t have to care about deleting memories manually.
- There are three kinds of smart pointers; `unique_ptr`, `shared_ptr`, and `weak_ptr`
- You may get detailed information here: https://en.cppreference.com/book/intro/smart_pointers

Unique Pointers

- A `unique_ptr` can be owned by only one owner.
- Cannot be copied or shared.

```
#include <iostream>
#include <memory>

using std::unique_ptr; using std::make_unique;
int main() {
    unique_ptr<Test> test_unique1(new Test(1));
    unique_ptr<Test> test_unique2 = std::make_unique<Test>(2);
    //unique_ptr<test> test_unique3 = test_unique2; // this is not allowed
    std::cout << "id : " << test_unique1->test_id << std::endl;
    std::cout << "id : " << test_unique2->test_id << std::endl;
}
```

Shared Pointers

- A `shared_ptr` can be owned by multiple owners.
- When no owner is using the object, it is destructed.

```
using std::shared_ptr; using std::make_shared;
shared_ptr<Test> test_shared() {
    shared_ptr<Test> test_shared1(new Test(1));
    shared_ptr<Test> test_shared2 = make_shared<Test>(2);
    shared_ptr<Test> test_shared3 = test_shared2;
    std::cout << "id : " << test_shared1->test_id << std::endl;
    std::cout << "id : " << test_shared2->test_id << std::endl;
    return test_shared3;
}
```

Shared Pointers (continued)

```
int main() {
    shared_ptr<Test> ptr = test_shared();
    std::cout << "id : " << ptr->test_id <<
std::endl;
    return 0;
}
```

Output

```
constructed
constructed
id : 1
id : 2
destructed
id : 2
destructed
```

Weak Pointers

- If two shared pointers point to each other, they are never released.
- `weak_ptr` pointing to a resource doesn’t affect the resource's reference count.
- When the last `shared_ptr` pointing the resource is destroyed, the resource will be freed, even if there are `weak_ptr` objects pointing to that resource.

```
int main() {
    shared_ptr<Test> test_shared1(new Test(1));
    shared_ptr<Test> test_shared2 = test_shared1;
    std::cout << "use count before : " << test_shared1.use_count() << std::endl;
    weak_ptr<Test> test_weak = test_shared1;
    std::cout << "id : " << test_weak.lock()->test_id << std::endl;
    std::cout << "use count after : " << test_weak.use_count() << std::endl;
    return 0;
}
```

Standard Template Library

- STL provides container, iterator, and algorithm.
- Usage of STL is somehow similar to java Collections.
- In this practice, we will mainly learn how to use vectors and algorithms.
- You may find these links helpful:

  https://en.cppreference.com/w/cpp/container/vector

  https://en.cppreference.com/w/cpp/algorithm

Practice 1 - Restaurant Rating System (1)

- Write a C++ Program that contains names of restaurants and their rating data list. There are no duplicates in input restaurant names.
- Your program should get console inputs repeatedly and do the appropriate action each time.
- You don't have to consider wrong input types. Assume that <name> contains only alphanumeric characters, and X is always integer.

Input Format	Desired Action
RATE <name> X	Add rating data to the restaurant <name> by int X. If the data doesn't exist, add restaurant <name> to your data.
LIST	Print all the restaurants you have in one line.
SHOW <name>	Print all the ratings of the restaurant <name>. The printed ratings should be sorted in ascending order. If there is no restaurant with name <name>, print "<name> does not exist".
(none above, or wrong input format)	Do not print anything.

Output Example

Console Inputs

```
SHOW McDonalds
RATE McDonalds 10
RATE McDonalds 50
RATE McDonalds 35
RATE BurgerKing 20
LIST
SHOW McDonalds
RATE BurgerKing 15
SHOW Lotteria
SHOW BurgerKing
RATE Lotteria 30
SHOW Lotteria
```

Console Outputs

```
McDonalds does not exist.




McDonalds BurgerKing
10 35 50

Lotteria does not exist.
15 20

30
```

Practice 2 - Restaurant Rating System (2)

- The clients want to see statistics of the restaurant ratings. Add AVE, MED, MIN, MAX functions to your program.
- You don't have to consider wrong input types. Assume that <name> contains only alphanumeric characters, and X is always integer.

Input Format	Desired Action
AVE <name>	Print the average value of the ratings of restaurant <name>. Print 2 digits digits after the decimal point.
MED <name>	Print the median value of the ratings of restaurant <name>. If there are even number of ratings, print the average of two values in the middle. Print 2 digits after the decimal point.
MIN <name>	Print the smallest value of the ratings of restaurant <name>.
MAX <name>	Print the biggest value of the ratings of restaurant <name>.
	For all 4 operations, If the restaurant doesn't have rates, print "0".

Output Example

Console Inputs

```
RATE McDonalds 10
RATE McDonalds 50
RATE McDonalds 35
SHOW McDonalds
AVE McDonalds
MAX McDonalds
RATE McDonalds 15
RATE McDonalds 5
RATE McDonalds 70
MED McDonalds
RATE McDonalds 25
AVE McDonalds
MIN McDonalds
SHOW McDonalds
```

Console Outputs

```



10 35 50
31.67
50



25.00      // average of 15, 35

30.00
5
5 10 15 25 35 50 70
```

Practice 3 - Restaurant Rating System (3)

- Some clients now want to cheat the restaurant ratings. Add DEL, CHEAT functions to your program.
- You don’t have to consider wrong input types. Assume that <name> contains only alphanumeric characters, and X is always integer.

Input Format	Desired Action
DEL <name> X	Delete all the rating values of restaurant <name> that are same with int X.
CHEAT <name> X	Delete all the rating values of restaurant <name> that are less than int X.

Output Example

Console Inputs

```
RATE McDonalds 10
RATE McDonalds 50
RATE McDonalds 35
RATE McDonalds 15
RATE McDonalds 5
RATE McDonalds 70
RATE McDonalds 25
SHOW McDonalds
DEL McDonalds 5
SHOW McDonalds
DEL McDonalds 20
SHOW McDonalds
CHEAT McDonalds 30
SHOW McDonalds
```

Console Outputs

```







5 10 15 25 35 50 70

10 15 25 35 50 70

10 15 25 35 50 70

35 50 70
```

Submission

- Compress your final project directory into a zip file.
- Rename your zip file as 20XX-XXXXX_{name}.zip - for example, 2020-12345_KimMinji.zip
- Upload it to eTL - Lab 12 assignment.
- You don’t have to consider about wrong input types not in this assignment.
- Today, we provide you the solution. You may just choose to follow the solution.