CS415 Project 3

Kyle Aure

Version 1.0, 2018-11-14

Project Description

Implement a Tombstone class in C++ to be an intermediary between pointers and addresses. The Tombstone class will then allow easier garbage collection, while checking for dangling pointers, memory leaks, and deallocated memory access.

Course Details

- **Course** - CS415: Principles of Programming Languages
- **Instructor** - Daniel Nash

Project Goals

- Create a tombstones.h file that can be imported using the `#include "tombstones.h"` directive.
- Create a `Ptr` class which will be used to replace the generic `T*` pointers.
- Check for dangling pointers, memory leaks, and deallocated memory access errors.
- Use the included test classes to ensure the `Ptr` class is working as expected.

Running project

Create a local copy of this project by running the following command:

```
git clone git@github.com:KyleAure/WSURochester.git
```

Then navigate to this project directory:

```
cd WSURochester/CS415/Project3/src
```

Then run the following goals to build and run the test files :

```
g++ test1.cpp -o test1 && ./test1
```

Replace `test1` with whichever test you want to run.

Documentation

Output

Output from running the tests can be found below:

1. Test 1: Ensures assignment and re-assignment works correctly.

```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test1.cpp -o test1 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test1
foo1: OK

[Done] exited with code=0 in 0.63 seconds
```

2. Test 2: Ensures two pointers to the same tombstone cannot be released multiple times.

```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test2.cpp -o test2 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test2
Unaccessable pointer reference at tombstones.h line: 106

[Done] exited with code=1 in 0.721 seconds
```

3. Test 3: Ensure non-primitive times are assigned correctly and released correctly.

```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test3.cpp -o test3 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test3
foo3: OK

[Done] exited with code=0 in 0.775 seconds
```

4. Test 4: Ensures tombstones that do not actually point to any data cannot be released.

```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test4.cpp -o test4 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test4
Unaccessable pointer reference at tombstones.h line: 106

[Done] exited with code=1 in 0.556 seconds
```

5. Test 5: Ensures basic memory leak scenario is handled correctly.

```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test5.cpp -o test5 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test5
Memory leak at tombstones.h line: 82

[Done] exited with code=1 in 0.607 seconds
```

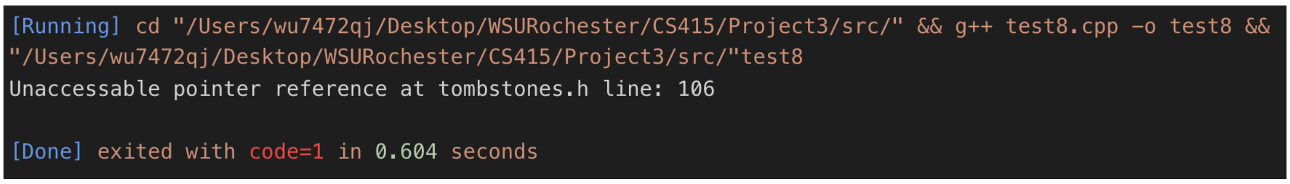
6. Test 6: Recursively define pointers and ensure that data integrity is maintained and check again for leaked memory.

```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test6.cpp -o test6 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test6
Memory leak at tombstones.h line: 119

[Done] exited with code=1 in 0.674 seconds
```

7. Test 7: Copy pointers repeatedly to check that they all point to the same place and ensure dangling references cannot be accessed.

```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test7.cpp -o test7 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test7
Unaccessable pointer reference at tombstones.h line: 106

[Done] exited with code=1 in 0.723 seconds
```

8. Test 8: Recursively redefine pointers to check that data is persevered and check dangling reference.



```
[Running] cd "/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/" && g++ test8.cpp -o test8 &&
"/Users/wu7472qj/Desktop/WSURochester/CS415/Project3/src/"test8
Unaccessable pointer reference at tombstones.h line: 106

[Done] exited with code=1 in 0.604 seconds
```