Smart Pointers - Multi-Threading

Q1: Do you notice any problems in the sample code before?
Q2: If there is any problem, how can you detect it?
Q3: How can you fix it?

A1: this line in src/inheritance/inheritance_demo.cpp

```
Animal* a = new Animal;
```

```
$ valgrind ./src/inheritance_demo
...
==393388==
==393388== HEAP SUMMARY:
==393388==     in use at exit: 8 bytes in 1 blocks
==393388==   total heap usage: 3 allocs, 2 frees, 73,736 bytes allocated
==393388==
==393388== LEAK SUMMARY:
==393388==    definitely lost: 8 bytes in 1 blocks
==393388==    indirectly lost: 0 bytes in 0 blocks
==393388==      possibly lost: 0 bytes in 0 blocks
==393388==    still reachable: 0 bytes in 0 blocks
==393388==         suppressed: 0 bytes in 0 blocks
==393388== Rerun with --leak-check=full to see details of leaked memory
==393388==
==393388== For counts of detected and suppressed errors, rerun with: -v
==393388== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

A2: valgrind - link

A3: one possible fix is:

```
delete a;
```

```
==394745== Invalid free() / delete / delete[] / realloc()
==394745==    at 0x4C3323B: operator delete(void*) (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==394745==    by 0x108D17: main (inheritance_demo.cpp:88)
==394745==  Address 0x1ffefffff90 is on thread 1's stack
==394745==  in frame #1, created by main (inheritance_demo.cpp:59)
==394745==
==394745==
==394745== HEAP SUMMARY:
==394745==     in use at exit: 8 bytes in 1 blocks
==394745==   total heap usage: 3 allocs, 3 frees, 73,736 bytes allocated
==394745==
==394745== LEAK SUMMARY:
==394745==    definitely lost: 8 bytes in 1 blocks
==394745==    indirectly lost: 0 bytes in 0 blocks
==394745==      possibly lost: 0 bytes in 0 blocks
==394745==    still reachable: 0 bytes in 0 blocks
==394745==         suppressed: 0 bytes in 0 blocks
==394745== Rerun with --leak-check=full to see details of leaked memory
==394745==
==394745== For counts of detected and suppressed errors, rerun with: -v
==394745== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

You can see it still has a complaint. How can you achieve the following?

```
==395385==
==395385== HEAP SUMMARY:
==395385==     in use at exit: 0 bytes in 0 blocks
==395385==   total heap usage: 3 allocs, 3 frees, 73,736 bytes allocated
==395385==
==395385== All heap blocks were freed -- no leaks are possible
==395385==
==395385== For counts of detected and suppressed errors, rerun with: -v
==395385== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

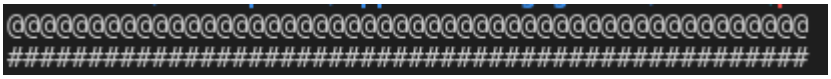
Smart Pointer

- A garbage collection mechanism by C++11
- A wrapper class over a pointer
    - Help programmers not to worry about any memory leaks
- See example in **src/smart_pointer/mem_leak_demo.cpp**
    - Motivation for using smart pointers
- See example in **src/smart_pointer/smart_pointer_demo.cpp**
    - Smart Pointer usage

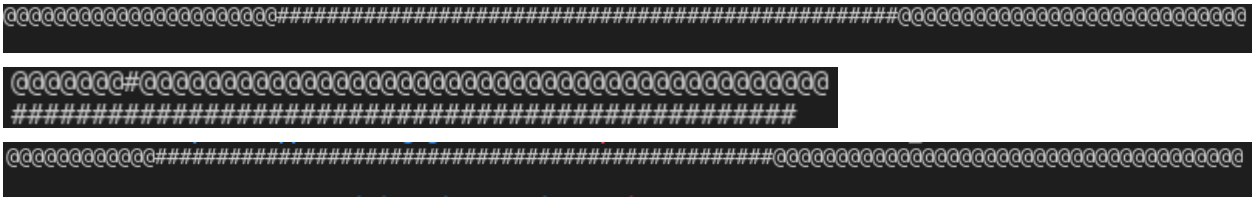
Multithreading and its mechanism

- See example in **src/multithread/thread_demo_with_problem.cpp**
    - Identify the problem and propose the solution

What we want (for ALL executions):



What might actually happen:

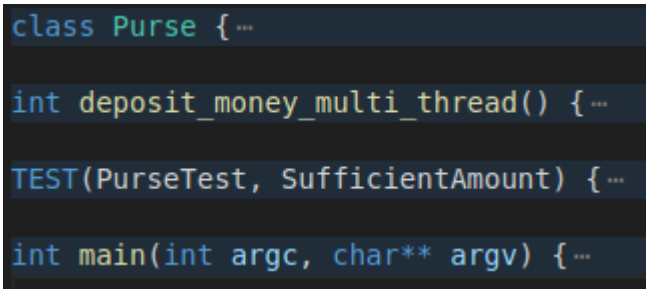


Task 1: Getting familiar with Smart Pointer (combined with Polymorphism)

This is a very *simple* task to help you get familiar with Smart Pointer usage.

- Write a program with a parent class (e.g., ParentClass)
    - The parent has a *pure virtual* member function (e.g., say_your_name())
    - Q: What is a virtual member function?
- Derive two classes from the parent class (e.g., Child1Class, Child2Class)
    - And define a function out of the parent's virtual function above.
    - Q: What does this action call?
    - Q: What will happen if we don't perform the above action?
- Create two smart pointers of parent class type to objects of these parent classes.
    - Use the pointers to invoke the proper polymorphic behavior.
- As always, write a unit test for the two child classes you just create
    - You need to design the test properly

Task 2: Multithreading with C++11



In this task, you need to utilize multithreading and synchronization to implement the following:

- Create two constants to represent:
  - Total number of threads
  - An amount of money
- A class **Purse** with the following:
  - A **constructor** that initialize its private member (money_) to 0
  - A **set**_ and **get**_ for that variable (use macro)
  - A **deposit_money()** that adds a proper amount to its private data member
    void deposit_money(int amount) {...}
    NOTE: for this function, don't use something like: x += amount;
    Use a for loop to add the proper amount
- Create a **deposit_money_multi_thread()** that does the following:
  - Create a Purse object
  - Create a vector of threads
  - For each thread in the vector (with the total number of threads declared above), make it:
    - Run the function deposit_money()
    - On the Purse object above
    - With the amount of money (constant declared above)
    - Then, push each thread to the vector
  - Wait for each thread to finish its execution
  - Return the amount of money from the purse
- Write the test to verify that:
  - **deposit_money_multi_thread()** runs correctly with the two constant you set
    - For ALL times
  - Also, another test to illustrate that if you don't apply proper synchronization mechanism, you will get incorrect result

```
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from PurseTest
[ RUN      ] PurseTest.SufficientAmount
/workspaces/cpp-training-github/src/multithread/thread_exercise.cpp:42: Failure
Expected equality of these values:
  5*100000
    Which is: 500000
  deposit_money_multi_thread()
    Which is: 311575
[  FAILED  ] PurseTest.SufficientAmount (10 ms)
[----------] 1 test from PurseTest (10 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (10 ms total)
[  PASSED  ] 0 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] PurseTest.SufficientAmount

 1 FAILED TEST
```

Incorrect

```
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from PurseTest
[ RUN      ] PurseTest.SufficientAmount
[       OK ] PurseTest.SufficientAmount (7 ms)
[----------] 1 test from PurseTest (7 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (7 ms total)
[  PASSED  ] 1 test.
```

Correct