UDACITY

‹ Return to "C++" in the classroom

Memory Management Chatbot

REVIEW

CODE REVIEW 2

HISTORY

Meets Specifications

Hi Udacious

Congratulations 🎉 🎉

Thank you for this wonderful project, I enjoyed reviewing this project, please continue with the good work. So far this one of the best submissions I have reviewed so far. You have brilliantly addressed all the rubrics required by this project. I wish you the best in the rest of your program. Stay smart.

More materials

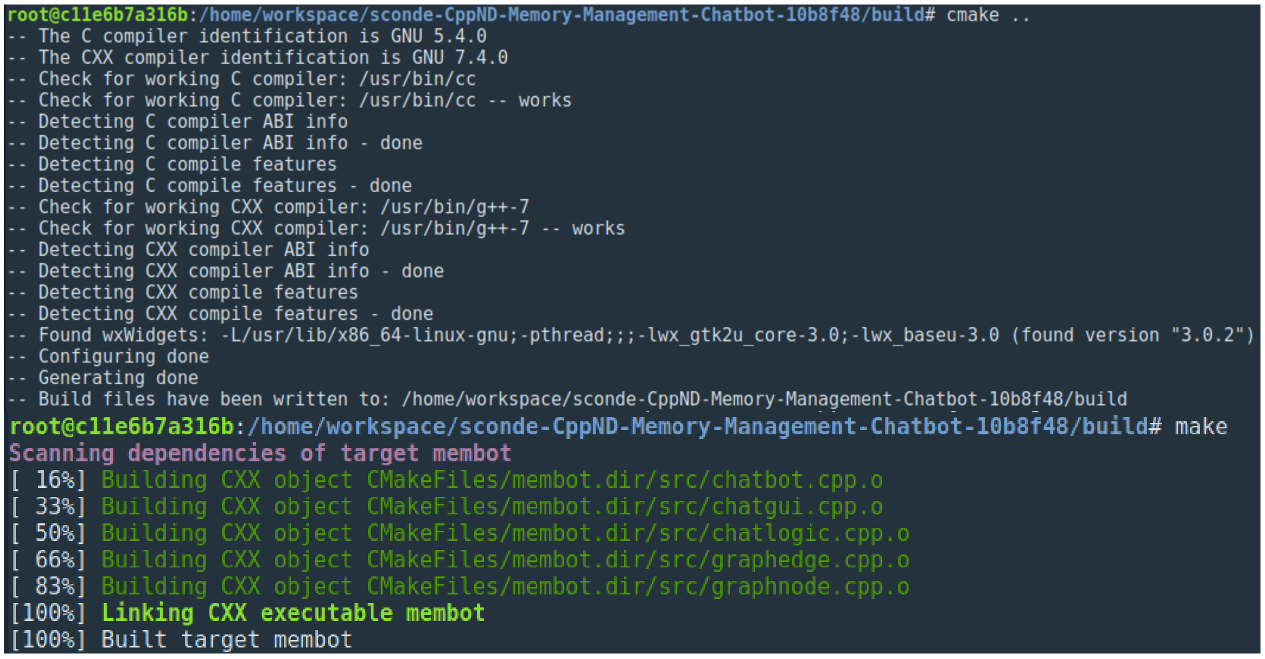
Below are some materials that could help you write more organized code and increase your knowledge of pointers.

- 10 Tips for C and C++ Performance Improvement Code Optimization
- More on pointers in C++ for beginners

Quality of Code

The code compiles and runs with `cmake` and `make`.

Awesome! Your code runs with `cmake` and `make` perfectly 👍



```
root@c11e6b7a316b:/home/workspace/sconde-CppND-Memory-Management-Chatbot-10b8f48/build# cmake ..
-- The C compiler identification is GNU 5.4.0
-- The CXX compiler identification is GNU 7.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/g++-7
-- Check for working CXX compiler: /usr/bin/g++-7 -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found wxWidgets: -L/usr/lib/x86_64-linux-gnu;-pthread;;;-lwx_gtk2u_core-3.0;-lwx_baseu-3.0 (found version "3.0.2")
-- Configuring done
-- Generating done
-- Build files have been written to: /home/workspace/sconde-CppND-Memory-Management-Chatbot-10b8f48/build
root@c11e6b7a316b:/home/workspace/sconde-CppND-Memory-Management-Chatbot-10b8f48/build# make
Scanning dependencies of target membot
[ 16%] Building CXX object CMakeFiles/membot.dir/src/chatbot.cpp.o
[ 33%] Building CXX object CMakeFiles/membot.dir/src/chatgui.cpp.o
[ 50%] Building CXX object CMakeFiles/membot.dir/src/chatlogic.cpp.o
[ 66%] Building CXX object CMakeFiles/membot.dir/src/graphedge.cpp.o
[ 83%] Building CXX object CMakeFiles/membot.dir/src/graphnode.cpp.o
[100%] Linking CXX executable membot
[100%] Built target membot
```

Task 1: Exclusive Ownership 1

In file `chatgui.h` / `chatgui.cpp`, `_chatLogic` is made an exclusive resource to class `ChatbotPanelDialog` using an appropriate smart pointer. Where required, changes are made to the code such that data structures and function parameters reflect the new structure.

Great Job, you used the appropriate smart pointer. 👏

Task 2: The Rule of Five

In file `chatbot.h` / `chatbot.cpp`, changes are made to the class `ChatBot` such that it complies with the Rule of Five. Memory resources are properly allocated / deallocated on the heap and member data is copied where it makes sense. In each of the methods (e.g. the copy constructor), a string of the type "ChatBot Copy Constructor" is printed to the console so that it is possible to see which method is called in later examples.

The move constructor, copy constructor, copy assignment operator, and move assignment operator are now very well implemented (along with the existing destructor).

Task 3: Exclusive Ownership 2

In file `chatlogic.h` / `chatlogic.cpp`, the vector `_nodes` are adapted in a way that the instances of `GraphNodes` to which the vector elements refer are exclusively owned by the class `ChatLogic`. An appropriate type of smart pointer is used to achieve this.

You did well here.

Pro Tips

How to use smart Pointers in C++
Sorting a vector of custom objects using C++ STL

When passing the `GraphNode` instances to functions, ownership is not transferred.

Fantastic, the ownership is not transferred as required by the rubric.

Task 4: Moving Smart Pointers

In files `chatlogic.h` / `chatlogic.cpp` and `graphnodes.h` / `graphnodes.cpp` all instances of `GraphEdge` are changed in a way such that each instance of `GraphNode` exclusively owns the outgoing `GraphEdges` and holds non-owning references to incoming `GraphEdges`. Appropriate smart pointers are used to do this. Where required, changes are made to the code such that data structures and function parameters reflect the changes.

Nicely done, keep up 👍

In files `chatlogic.h` / `chatlogic.cpp` and `graphnodes.h` / `graphnodes.cpp`, move semantics are used when transferring ownership from class `ChatLogic`, where all instances of `GraphEdge` are created, into instances of `GraphNode`.

Your implementation is great.

Task 5: Moving the ChatBot

In file `chatlogic.cpp`, a local `ChatBot` instance is created on the stack at the bottom of function `LoadAnswerGraphFromFile` and move semantics are used to pass the `ChatBot` instance into the root node.

Nicely done

Pro tips

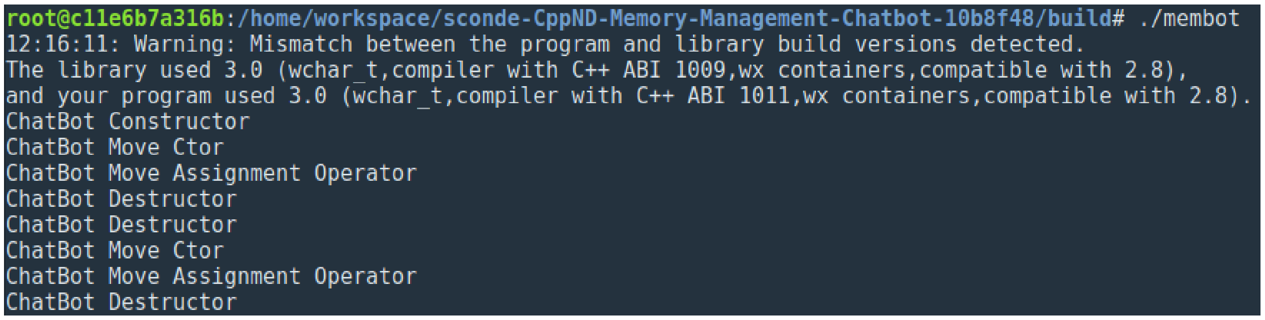
Move Constructors and Move Assignment Operators (C++)

`ChatLogic` has no ownership relation to the `ChatBot` instance and thus is no longer responsible for memory allocation and deallocation.

Perfectly done ✓

When the program is executed, messages are printed to the console indicating which Rule of Five component of `ChatBot` is being called.

Good, you really did well here, I am impressed.



```
root@c11e6b7a316b:/home/workspace/sconde-CppND-Memory-Management-Chatbot-10b8f48/build# ./membot
12:16:11: Warning: Mismatch between the program and library build versions detected.
The library used 3.0 (wchar_t,compiler with C++ ABI 1009,wx containers,compatible with 2.8),
and your program used 3.0 (wchar_t,compiler with C++ ABI 1011,wx containers,compatible with 2.8).
ChatBot Constructor
ChatBot Move Ctor
ChatBot Move Assignment Operator
ChatBot Destructor
ChatBot Destructor
ChatBot Move Ctor
ChatBot Move Assignment Operator
ChatBot Destructor
```

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ❯

RETURN TO PATH