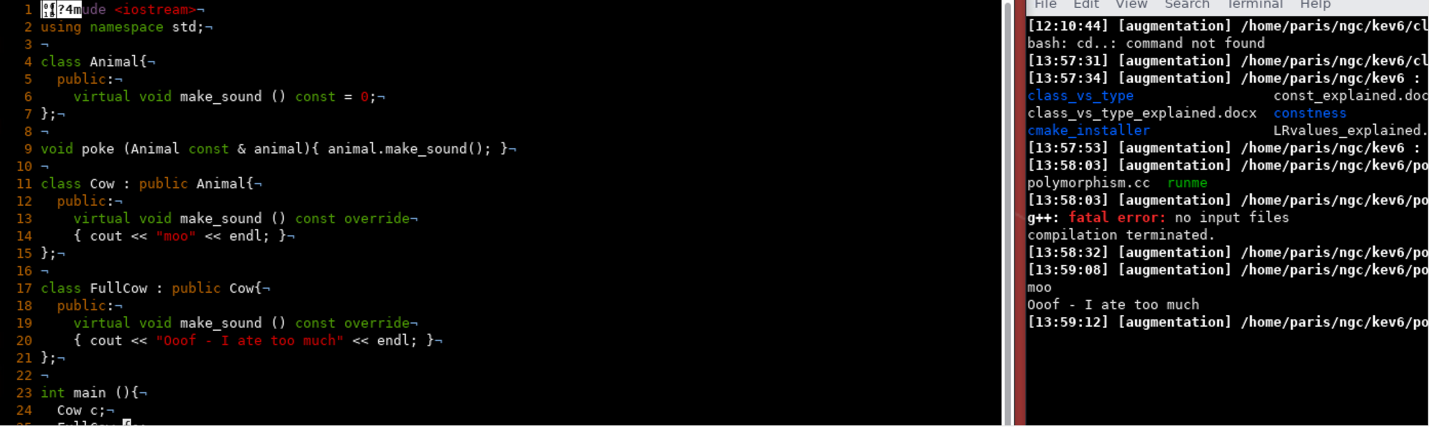
*Exercise 4: Basic Polymorphism & Virtual*

*Functions (polymorphism.cc)*

***Explained***

* **This exercise has us…?**
* **The initial output of this code displays the following:**

******

* + In the code there is a parent class called “Animal” which has the pure virtual function “make\_sound()”.
    - By definition, a pure virtual function is essentially a placeholder function set in the parent class that **MUST** be redefined in each of the derived child classes.
  + Next there is a function “poke()” that requires a constant reference to an animal object. This function simply calls the make\_sound() function and the “sound” made will be indicated by the nature of the object being passed.
  + Next there is the “Cow” Class which is derived from the “Animal” class. This means that the “Cow” class must define its own version of the “make\_sound()” function and it does so by printing the string “moo”.
  + Finally, there is the “FullCow” Class which is a child of the “Cow” class (which also means it is technically a child of the “Animal” class). Since it is also technially child of “Animal”, “FullCow” must also define its own version of the “make\_sound()” function and it does so by printing the string “oof - I ate too much”.
  + The Main() function simply creates an object of both “Cow c;” and “FullCow fc;”. It then proceeds to call the “poke()” function twice, each time with a different version of c/fc as the object being passed
    - The behavior is as expected. Both functions respectively print:
      * poke(c) = “moo”
      * poke(fc) = “oof - I ate too much”
* ***After removing the “virtual” keyword from the “make\_sound()” function in both “Cow” and “FullCow”, surprisingly the program still works.***

Text

Description automatically generated

* + This apparently means that, though pure virtual functions still must be redfined in each child class, the literal “virtual” keyword is not necessary for successful compilation. Apparently the compiler is smart enough to recognize this function as an implementation of the pure virtual function without having to explicitly state that it is each time.
  + Removing the “override” keyword has the same results, ***but removing the “const” keyword changes the output!***
* ***Placing a trailing “final” keyword in the “Cow”’s “make\_sound()” function actually causes the program to crash!!?!?:***
  + I, apparently, had a misunderstanding about the “final” keyword. I thought it worked the same was as “const”, but not exactly.
    - “Final” is typically used in the context of inheritance to indicate that a virtual function or class cannot be further modified or extended, while “const” is used to indicate that a function or variable can’t modify the state of an **object**.
  + Here, marking the “make\_sound()” function as “final” means that any class derived from the current class cannot override this function. Since “FullCow” is derived from “Cow” and not “Animal”, it prevents “FullCow” from overriding the virtual function “make\_sound()”. Because overriding “make\_sound()” is required yet cannot be done, this causes the program to crash.