C++ - Assignment 3

Practicing classes

Charanjit Kaur & Caterina Doglioni, PHYS30762 - OOP in C++ 2023

Part 1a. Write a C++ class for describing (some of the) leptons

For a crash course on all particles we know about, see here

You can take inspiration from the particle class in the pre-lecture

- A particle object should contain the following (private) data (0.5 *mark, 0.25 if not all implemented or if problems with input checking*):
  - **particle type**: string
    - A lepton can be: an electron or a muon
  - **rest mass** [unit: MeV]: check the particle data group to find the mass for this particle, approximate to the closest MeV
  - **charge**: +1 (particles) or -1 (antiparticles)
  - **velocity** [unit: m/s]: between 0 (at rest) and the speed of light, in [m/s]
  - **a "beta" value** (how fast it goes with respect to the speed of light) double beta = velocity / speed_of_light, in the range 0-1
    - Note: beta must never be > 1, and this should be checked when setting it or when setting the velocity
    - use speed of light as a constant in your class (or as a preprocessor directive)

```
// light_spd -- The speed of light in vacuum in meters per second
 const double light_spd = 2.99792458e8
```

- The class should also contain several member functions (see next slide)

Part 1b. Write a C++ class for describing (some of the) leptons

For a crash course on all particles we know about, see here

- A particle object should contain the following functions *(0.5 mark, 0.25 if not all there):*
    - A **default constructor**; a **parameterized constructor**; a **destructor**
    - **setters/getters** for all the private member data
    - a member function (with implementation outside the class) to **print out** all particle data
- **Challenge marks:**
    - (*0.3 mark*) can you think about how you could have a combination of data member and function to represent **antiparticles** without code duplication, given that particles and antiparticles have the same mass but opposite charges?
    - (*0.3 mark*) Classes (this and the next one) have separate interface and implementation - add the exact line you used to compile
    - *(0.4 mark)* Code is on git (try to practice good commits practices, even if they are not marked).

Part 2. Write a C++ class for a simple lepton detector

For a crash course on all particles we know about, see here

- A detector object should contain the following (private) data *(2 marks, 1 if not all there):*
    - **detector type**: string
        - A detector can be: a tracker, a calorimeter, a muon chamber
            - a tracker detects electrons and muons (and their antiparticles)
            - a calorimeter detects electrons (and antielectrons)
            - a muon chamber detects muons (and antimuons)
    - **status**: can be *on* or *off* (use a bool data member, and turn_on/turn_off functions)
- The class should also contain the following member functions *(2 mark, 1 if not all there):*
    - **constructor, parameterised constructor, destructor, printing function:** *(1 mark, 0.5 if not all there)*
    - a **function** that takes a particle class as input, checks if the detector is on, returns 0 if the detector is off OR the particle is not detected and returns 1 if the particle detector was on AND was detected *(1 mark, 0.5 if it doesn't do the right thing)*
        - The function should also prints "[particle type] was detected" on the screen
        - The class should contain data members that are used to how many particles passed through it

Part 3: use your classes in *main*

- The main program should demonstrate use of the class through declaring and using objects *(2.5 marks total)*
  - Instantiate the different types of particles and their antiparticles. Your “test vector” should be made of two electrons, four muons, one antielectron, one antimuon.
    - Put them in a vector so you can iterate on them *(1 mark, 0.5 if no vector, 0 if no/too few particles)*
  - Print all information about particles (from that vector) (1 *mark, 0.5 if info missing, 0 if printing not there)*
  - Instantiate and operate the three different kinds of detectors *(2 marks, 1 if it only partially does what it should or something is missing, 0 if it doesn’t work at all)*
    - Turn them on
    - Pass each of the particles through the detector
    - Turn them off when you’re done with the particles
    - (For fun, you can also try to pass a particle through the detector when the detector is off and see what happens)
- 1 mark will be deducted if your program produces any compilation warnings. **If your code does not compile, we will not debug/mark it and you will get zero marks.**

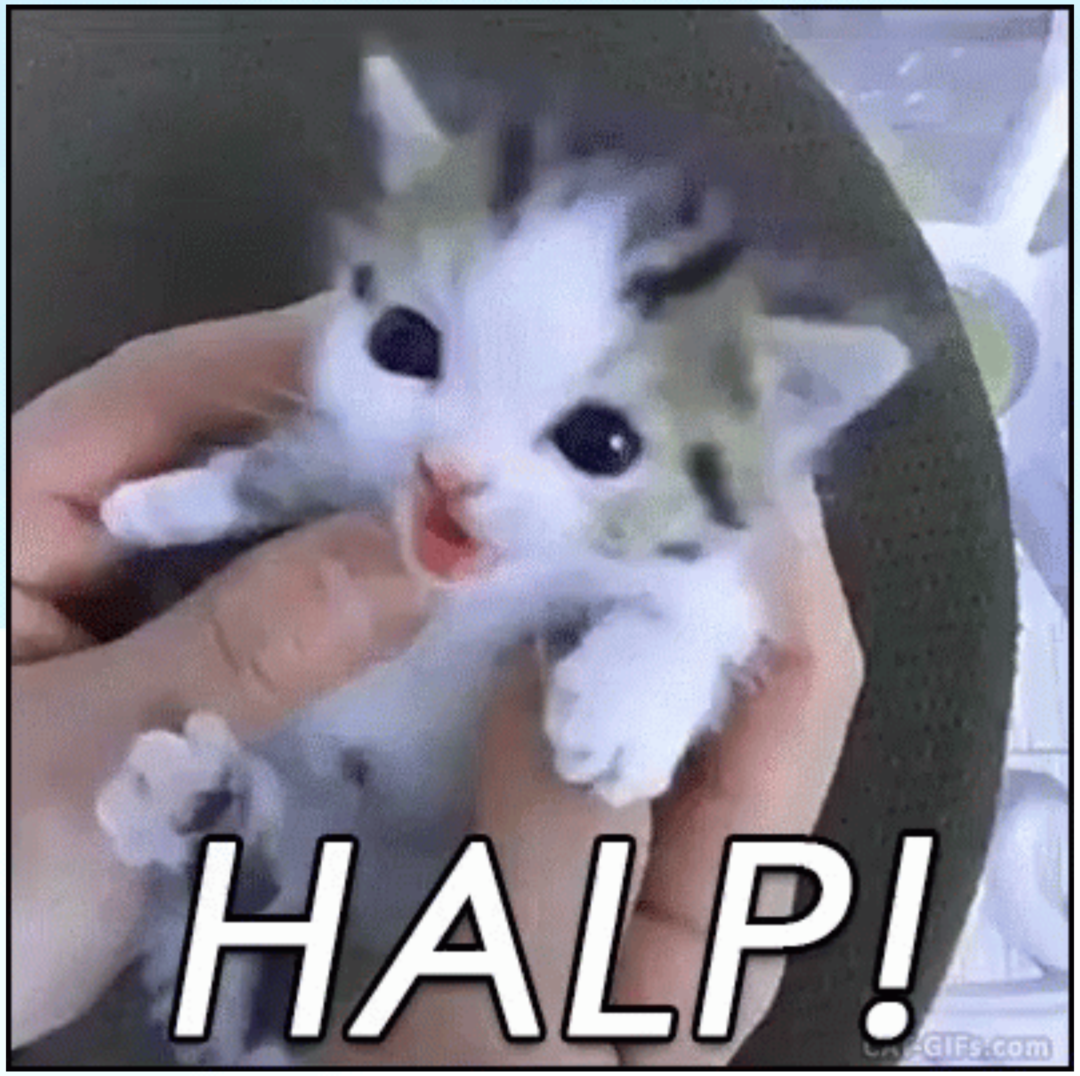
Suggestion on designing/writing code

- Suggestion: don't write all code at once, write one thing at a time
- Example:
  - start with making the particle class with only constructor and destructor, instantiate it in main()
  - compile (if it does, commit & push to git_
  - add the other data member and member function, test them in main()
  - compile (& commit)
  - split into interface and implementation
  - compile (& commit)
- This way if something doesn't compile by the submission deadline you still have something that compiles to submit!

Link to join the GitHub repository:

https://classroom.github.com/a/HsF5FN7D

Reminder about how not to ask for help to demonstrators/course leaders:



- I have an error in printing things out in my code, can you help?

This only works if there is a demonstrator sitting near you, and even then it's not ideal

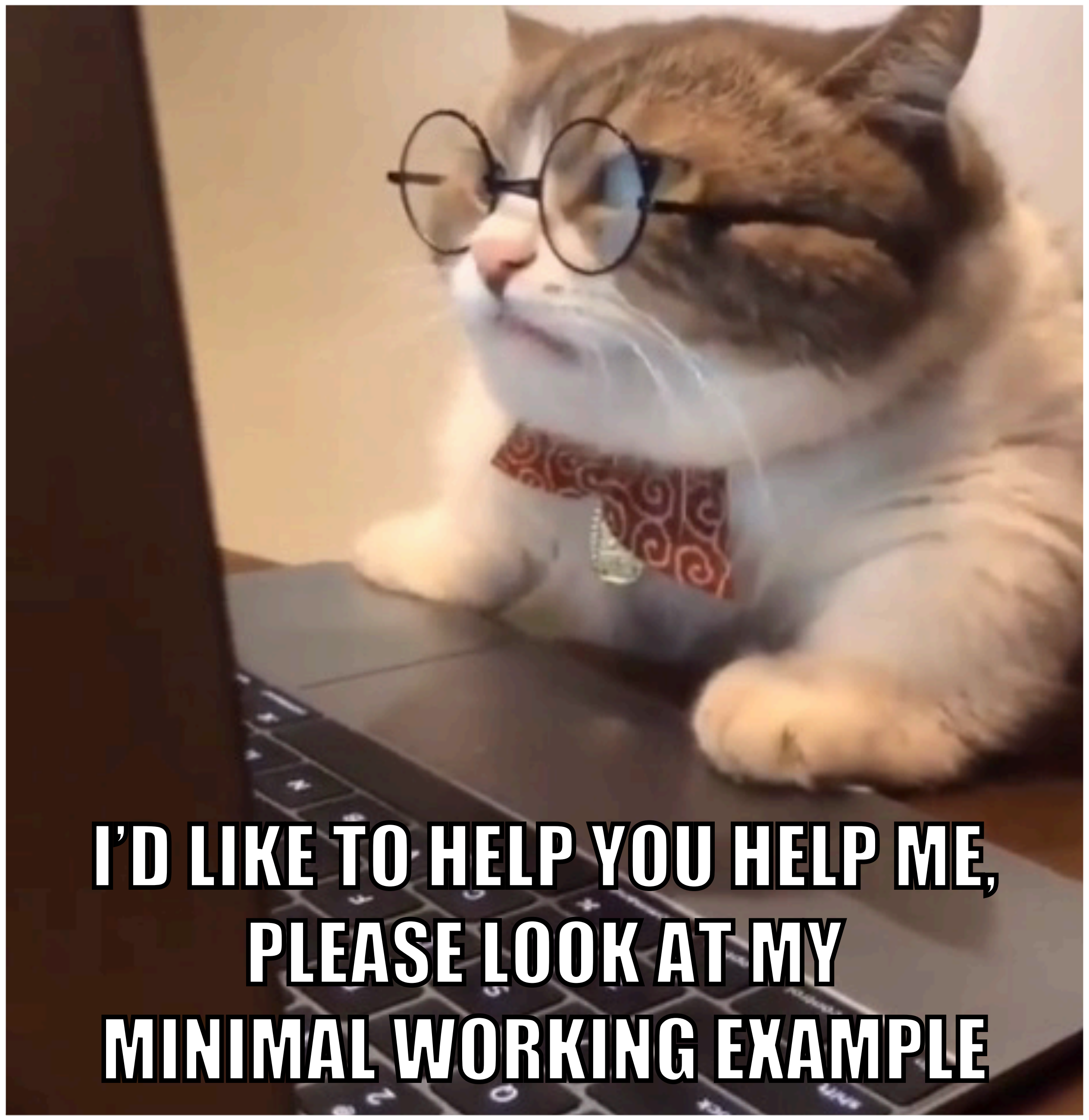
MANCHESTER 1824 The University of Manchester

Reminder about how not to ask for help to demonstrators/course leaders:

- I have an error in printing things out in my code
- I am using the printf function like this: printf(current_year, “%s");
- It does not compile
- Can you help?

This only works if your demonstrator remembers the syntax of printf

Reminder about how to ask for help to demonstrators/course leaders:



- I have an issue with the printf function
- This is a standalone C++ code where I have isolated the issue

```
int main (){
   int current_year = 2023;
   printf(current_year, “%s");
   return 0;
}
```

- You can reproduce my error if you “Build task” in Visual Studio Code with g++11 on a mac with Big Sur OS and an Intel chip

This way a demonstrator can just copy paste the code,
look at your error with the compiler,
and advise on what’s going wrong!

https://stackoverflow.com/help/minimal-reproducible-example