Particle catalogue project

Goal: design and demonstrate the functionality of a catalogue of objects, based on a class hierarchy, for storing data of Standard Model particles (& particles beyond it, if you'd like)

Note: read the full script and slides with implementation instructions before starting to design your code, we have simplified certain things in the physics to make the assignment worth the points and time you will put into it.

Since this project is more challenging than the books/article catalogue project idea, the total number of marks available including challenge marks is 38 instead of 35 (you will get 100% with 35 marks and above). This also means that the additional 3 challenge marks (for the physically consistent four-vector) won't appear in the BB rubric but instead they will be assigned by the demonstrator in your final project mark.

Note 2: we will not be able to respond to project questions until the teaching break ends, but even if you have a question this shouldn't stop you from starting to design your classes and make a start on elements of the whole project(you also have a lot of lecture/pre-lecture material to go through, as well as assignment 5). We will go over all questions collected until then in the lecture on April 9th.

Rubric and marking: general principles

We require your code to have the following to obtain the core marks (approximately 75% of the total project marks):

- Several particle classes in a hierarchy, using abstract classes and polymorphism, and a separate 4-momentum class.
- Particle catalogue = a customised container for particles (here you can wrap your class around existing STL containers).
- Proper (fancy) printing functions for all particles in your catalogue.

Challenge marks (saturating to 100%, approximately 25% of the total project marks) given by:

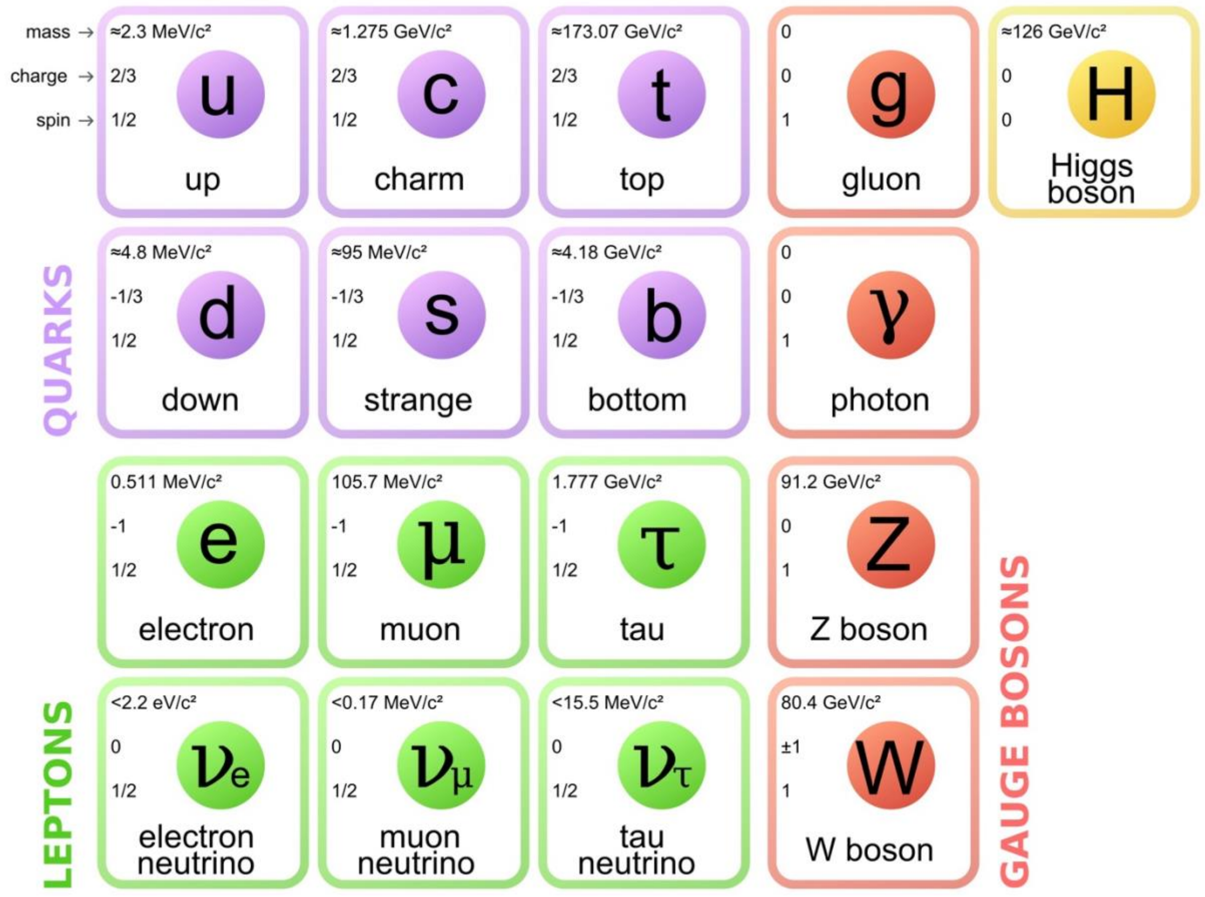
- Use of advanced C++ features in lectures 9 and 10.
- Physically consistent implementation of 4-vector.
- GitHub commit history over the course of multiple (>2) days.
- Splitting into interface and implementation, one class per file.
- Good and explanatory code comments.

Preliminary explanations (physics background)

The Standard Model has 17 fundamental particles, which you can see in the image below. These particles are:

- The quarks (up, down, strange, charm, top, bottom)
- The leptons (electron, muon, tau)

- The lepton neutrinos (electron neutrino, muon neutrino, tau neutrino)
- The gauge bosons (photon, W boson, Z boson, gluon)
- The Higgs boson


mass →
charge →
spin →
≈2.3 MeV/c² 2/3 1/2 u up
≈1.275 GeV/c² 2/3 1/2 c charm
≈173.07 GeV/c² 2/3 1/2 t top
0 0 1 g gluon
≈126 GeV/c² 0 0 H Higgs boson
≈4.8 MeV/c² -1/3 1/2 d down
≈95 MeV/c² -1/3 1/2 s strange
≈4.18 GeV/c² -1/3 1/2 b bottom
0 0 1 γ photon
0.511 MeV/c² -1 1/2 e electron
105.7 MeV/c² -1 1/2 μ muon
1.777 GeV/c² -1 1/2 τ tau
91.2 GeV/c² 0 1 Z Z boson
<2.2 eV/c² 0 1/2 νe electron neutrino
<0.17 MeV/c² 0 1/2 νμ muon neutrino
<15.5 MeV/c² 0 1/2 ντ tau neutrino
80.4 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
Source: Wikipedia

You can find their electric charge, spin and rest masses in the picture above.

Each particle has an antiparticle, except for the Z and Higgs bosons, so in total your final catalogue should contain at least 15+15+2 separate particle/antiparticle instances, one per type.

All charges and lepton/baryon number (not spin or other properties) for antiparticles are opposite with respect to the corresponding particles.

In this exercise, we will consider most particles stable with some exceptions (e.g. we don't deal with top decays).

The particles that we are considering as unstable are:
- The tau lepton, which can decay to:
  - a lepton and an antineutrino of the same lepton flavour electrons or muons) and a tau antineutrino,
  - two quarks and a tau antineutrino.
- The Higgs boson, which can decay to (mainly):

    - Two Z bosons
    - Two W bosons of opposite signs (W boson and W antiboson)
    - Two photons
    - A b quark and a b antiquark
- The W boson, which can decay to:
    - Two quarks (or antiquarks) of different flavours with charges summing to +1 (or -1),
    - A lepton and a neutrino
- The Z boson, which can decay to:
    - A quark and an antiquark
    - A lepton and an antilepton

The general properties of these particles that we want to capture in the catalogue are:

- Four-momentum (you should derive the rest mass of the particle from the four-momentum). Use natural units where h=c=1 (so you don't have to worry about the factors of c when you print things out).
- Charge,
- Spin,

Since we want you to use polymorphism and abstract classes, we want you to implement specific properties for each particle category and type. These are:

Leptons: lepton number (+1 for particles, -1 for antiparticles)
Quarks: baryon number (+1/3 for particle, -1 for antiparticles)

Electrons: energy deposited in four calorimeter layers (layer_1, layer_2, layer_3, layer_4)
Muons: isolation variable (true/false)
Neutrinos: variable determining the interaction with the detector (true/false)
Quarks: colour charge (can be red, blue, green)
Taus, W, Z, Higgs: vector of particles it decays into.

Detailed marking rubric

The slides/video on Blackboard have more explanation and hints, this file only contains the summary of the rubric.

Minimum hierarchy and design specification rubric (7 marks)

- *Abstract base class: [**2** marks]* if abstract base class is present.
- *Complete set of derived classes with deep copy functionalities: [**2** marks]* if all 17 particles with correct antiparticles are present,
    - functionality adds points (see later)
    - lack of particles reduces points proportionally to the number of missing particles.
- *Levels of hierarchy: [**2** marks]* for 3+ levels of hierarchy, 1 mark for 1 or 2 levels, 0 marks for no hierarchy.

- *Use of virtual functions: [**1** mark]* for proper use of virtual functions

Minimum particle attributes and functionality (7 marks)

General:

- *[**1** mark]* Charge and spin, and respective getters
- Four-momentum class (see below for marks)
- *[**1** mark]* A class that prints out nicely all the properties of a particle (including specialised ones)

Leptons and quarks:

- *[**0.5** mark]* Lepton and quark number, and respective getters

Specific properties for each particle:

- *[**0.5** mark]*: calorimeter layers for electrons, with consistency check,
- *[**0.5** marks]*: isolation and interaction variable for muons and neutrinos respectively
- *[**0.5** marks]*: color charge for quarks and pair of color charges for gluons
- *[**3** marks]:* vectors of (polymorphic) pointers to particle objects the original particle decayed to for taus, Higgs, W and Z bosons, with setters (as there are more options) and getters, with consistency checks.

Four-momentum attributes and functionality (7 marks)

Minimum four-momentum attributes and functionality:

- *[**1** mark]* std::vector for elements, plus setters and getters with consistency check (E>0 at minimum unless you want extra marks)
- *[**1** mark]* overloaded dot product, addition + and subtraction – operators
- *[**1** mark]* function returning the invariant mass (no physical checks needed)

Advanced (challenge) marks for four-momentum functionality

- *[**3+3 challenge** marks]* four-momentum making physical sense, including invariant mass corresponding to that of the particle in the table above

Minimum particle container (catalogue) functionality (5 marks)

- *[**0.5** mark]*: presence of a container-based class implementation (more marks can be obtained via STL containers, see below)
- *[**4.5** marks]*: user-implemented methods for total number of particles, number of particles per type, summing four-momentum of all particles, getting a sub-container of pointers to particles of the same kind with deep copies, printing information about one or more particles.

Main code (4 marks)

- *[**1.5** mark]* container with all 32 particles, one per type

- *[**1** mark]* full report on particle and particle containers displayed on screen
- *[**0.5** mark]* test of input checking with one "badly inputted" particles
- *[**1** mark]* streamlined code with function calls

Challenge marks (common to all projects) (5 marks)

- *[**2** marks]* templates
- *[**0.5** marks]* lambda functions or static variables
- *[**1** mark]* exceptions
- *[**1** mark]* STL containers/algorithms beyond vector/string/pair
- *[**1** mark]* commit history beyond single upload
- *[**1** mark]* code comments
- *[**0.5** mark]* split into interface and implementation

Negative marks (common to all projects)

- *[**0** marks overall]* code does not compile
- *[**-2** marks overall]* code is not easy to read (e.g. non-informative class/function/variable names) or does not respect the house style