Laboratorio di Elettromagnetismo e Ottica (A.A. 2022-23)

Parte di Programmazione C++/ROOT

S. Arcelli

Prima Prova

Prima di iniziare, alcune raccomandazioni:

- Separare ciascuna classe (ParticleType, ResonanceType, Particle) in moduli indipendenti, e ciascuna in file di intestazione e implementazione. Nel file di intestazione inserire la include guard. Cercate di usare nomi in inglese, etc...(seguire Coding Conventions).

- Nella dichiarazione dei metodi, ricordarsi di dichiarare const i metodi che soddisfano a questo requisito

- N.B: Questa prima prova può essere svolta anche senza l'utilizzo di ROOT. Per compilare e "testare" le vostre classi, potete utilizzare anche solo il compilatore che avete a disposizione.

- In rosso i punti un po' più complessi: su questi come sugli altri, non esitate a chiedere chiarimenti. Vi può essere utile utilizzare come riferimento anche le slide della lezione 7

Punto 1)

Scrivere una classe ParticleType, descrittiva di alcune proprietà di base delle particelle elementari, che abbia:

• come attributi (privati):
  – Nome della particella (char* fName)
  – Massa (double fMass)
  – Carica (int fCharge)
• Come metodi (pubblici):
  – I "Getters" per gli attributi
  – Un metodo Print() che stampi tutte le proprietà (gli attributi) su standard output
  – Un costruttore parametrico

Si noti che gli attributi della classe (nome, Massa e Carica della particella), una volta creata l'istanza, non possono variare (devono essere dichiarati const).

Punto 2)

Scrivere una classe ResonanceType (descrittiva di alcune proprietà di base specifiche delle risonanze), che eredita da ParticleType e che abbia:

• come attributo aggiuntivo (privato):
  – Larghezza della Risonanza (double fWidth)
• Come metodi (pubblici):
  – Il "Getter" per l' attributo extra fWidth
  – Un metodo Print() che stampi tutte le proprietà su standard output. Ridefinite il metodo che avete implementato nella classe ParticleType, e fate uso del metodo della classe base.
  – Un costruttore parametrico

Anche in questo caso gli attributi della classe, una volta creata l'istanza, non possono variare e devono essere dichiarat const.

Arrivati a questo punto, scrivete un programma di prova che faccia un'istanza di ParticleType, una di ResonanceType, e verificate che i metodi che avete implementato facciano quanto richiesto.

Inoltre, all'interno del programma di test definite un array di puntatori a ParticleType (2 elementi), assegnate al primo elemento una ParticleType e al secondo una ResonanceType, e con un ciclo sugli elementi dell'array invocate il metodo di stampa degli attributi. Per l'elemento corrispondente a ParticleType devono essere stampati a schermo tre attributi (nome,massa,carica), per ResonanceType quattro attributi (nome,massa,carica,larghezza). Se ciò non succede, vi siete "dimenticati" di qualificare il metodo di stampa nella maniera che vi consente di avere questo comportamento (virtual).

Punto 3)
Si scriva una classe Particle, descrittiva sia delle proprietà di base definite nella classe ParticleType e nella sua derivata che delle proprietà cinematiche (le tre componenti dell'impulso) di una particella, che abbia:
• come attributi (privati):

– Un array di puntatori (comune a tutte le istanze, quindi static) di tipo ParticleType (fParticleType), che contenga l'informazione dei tipi di particella che saranno generati nel corso del programma di simulazione. Si faccia l'ipotesi di poter generare al massimo 10 tipi di particelle (dichiarare un membro static const int fMaxNumParticleType = 10;).
– Un contatore (static int fNParticleType, anche qui comune a tutte le istanze) dei tipi di particelle considerati,corrispondente al numero di elementi non nulli nell'array, che si assumono inseriti consecutivamente senza discontinuità (senza lasciare "buchi").
– Una variabile intera che esprima un codice connesso al tipo di particella, corrispondente all'indice dell'elemento nell'array di puntatori (fIndex)
– Le tre componenti dell'impulso (fPx, fPy, fPz)

Punto 4)
Sempre per quanto riguarda la classe Particle, implementare i seguenti metodi:

• Un costruttore parametrico con parametri di ingresso il nome della particella e le tre componenti dell'impulso (4 argomenti, con gli ultimi 3 settati di default =0).

• Nel corpo del costruttore controllare se il tipo di particella istanziato è già contemplato nell' array di puntatori a ParticleType, identificandone la posizione in tabella e conferendo all'attributo fIndex tale valore.Per fare ciò utilizzare nel costruttore un metodo (**privato**) int FindParticle(nomeParticella), che implementate contestualmente, e che scorra sull'array statico di ParticleType e trovi la corrispondenza fra il nome in input e il nome dell'oggetto puntato, ritornando l'indice di elemento per cui questo avviene. Tale indice sarà il valore da attribuire a fIndex. Segnalare con una stampa a schermo il caso in cui non ci sia corrispondenza.

Punto 5)
Implementare gli ulteriori metodi (pubblici):

– Il "Getter" per l'attributo corrispondente al codice della particella

– Un metodo **statico** (AddParticleType) che permetta di riempire/aggiornare il contenuto dell'array di puntatori di tipo ParticleType, con parametri di ingresso:

- nome della particella,
- massa,
- carica
- ed eventualmente larghezza (default=0).

Nel metodo controllare se il tipo di particella che si vuole aggiungere non sia già presente, e che non sia stato raggiunto il numero massimo di tipi di particella che possono essere generati. Usare anche in questo caso il metodo privato FindParticle che è stato già utilizzato nel costruttore. Per poterlo utilizzare, occorrerà dichiarare FindParticle ...?

– Un metodo (un "Setter") che permetta di settare esplicitamente l'attributo fIndex corrispondente al codice della particella, controllando se il tipo corrispondente è già contemplato nell'array di puntatori a ParticleType. Fornire due versioni in overload, uno con argomento un intero, e uno con argomento il nome nella particella. Usare anche in questo caso il metodo privato FindParticle.

Punto 6)
Implementare come altri metodi (pubblici):
– Un metodo che stampi a schermo l'intero contenuto dell'array di puntatori a ParticleType (può essere statico? Se si, fatelo statico).
– Un metodo che stampi a schermo indice del tipo di particella, nome della particella, le componenti dell'impulso
-I "getters" per le componenti dell'impuso (sono Getters e quindi potete dichiararli const)
-Un metodo che ritorni il valore della massa della particella (const)
-Un metodo che ritorni il valore dell'energia totale della particella (const):

$$E = \sqrt{(m^2 + |\vec{p}|^2}\ (c = 1)$$

-Un metodo che ritorni il valore della massa invariante fra la particella corrente e un'altra particella, tipo double InvMass(particle & p):

$$(M_{inv} = \sqrt{(E_1 + E_2)^2 - (\vec{p}_1 + \vec{p}_2)^2}\ )$$

-Un metodo che permetta di "settare" le tre componenti dell'impulso:
void SetP(double px,double py,double pz);