particle_type.hpp

```cpp
#ifndef PARTICLE_HPP
#define PARTICLE_HPP

#include "particle_type.hpp"
#include "resonance_type.hpp"

#include <array>
#include <cmath>

class Particle {
 public:
  double get_px() const;
  double get_py() const;
  double get_pz() const;
  char   get_name() const;
  int    get_index() const;
  void   set_index(char name);
  double get_mass() const;
  int    get_charge() const;
  double get_energy() const;
  void   set_p(double px, double py, double pz);

  double invMass(const Particle& other) const;

  static void addParticleType(char name, double mass, int charge,
                              double width = 0);
  static void printParticleTypes();

  void printParticle() const;

  int decay2body(Particle& dau1, Particle& dau2) const;

  Particle();
  Particle(char name, double px = 0, double py = 0, double pz = 0);

 private:
  static const int max_n_particle_type_ = 7;
  static std::array<ParticleType*, max_n_particle_type_> particle_types_;
  static int                                             n_particle_type_;

  int index_;

  double px_;
  double py_;
  double pz_;

  void boost(double bx, double by, double bz);

  static int
      findParticle(char name); // trova il tipo di particella a partire dal suo
                               // nome (serve a settare correttamente l'indice)
  // puntatore perché può restituire un valore nullo se il nome non esiste
};

inline double Particle::get_px() const { return px_; }
inline double Particle::get_py() const { return py_; }
inline double Particle::get_pz() const { return pz_; }

inline char Particle::get_name() const {
  return particle_types_[index_]->get_name();
}

inline void Particle::set_p(double px, double py, double pz) {
  px_ = px;
  py_ = py;
  pz_ = pz;
}

inline int  Particle::get_index() const { return index_; }
inline void Particle::set_index(char name) {
  const int find_particle = findParticle(name);
  if (find_particle >= 0) { index_ = find_particle; }
}

inline double Particle::get_mass() const {
  return particle_types_[index_]->get_mass();
}

inline int Particle::get_charge() const {
  return particle_types_[index_]->get_charge();
}

inline double Particle::get_energy() const {
  double mass = get_mass();
  double p2   = px_ * px_ + py_ * py_ + pz_ * pz_;
  return std::sqrt(mass * mass + p2);
}

inline double Particle::invMass(const Particle& other) const {
  double other_px = other.get_px();
  double other_py = other.get_py();
  double other_pz = other.get_pz();

  double px2 = (px_ + other_px) * (px_ + other_px);
  double py2 = (py_ + other_py) * (py_ + other_py);
  double pz2 = (pz_ + other_pz) * (pz_ + other_pz);

  double e12 = get_energy() + other.get_energy();

  return std::sqrt(e12 * e12 - px2 - py2 - pz2);
}

#endif
```