Programowanie II R

Zadania – seria 6.

Dziedziczenie - rozpad bozonu Z

Celem ćwiczeń jest modelowanie rozpadu bozonu Z przy użyciu klas reprezentujących wybrane cząstki Modelu Standardowego.

Zadanie 1: Klasa Cząstka

Utwórz klasę `Czastka`, z której będą dziedziczyć wszystkie obiekty w naszej symulacji.

Wymagania:

1. Chronione (`protected`), stałe (`const`) pola przechowujące właściwości cząstki: `nazwa` (string), `ladunek_elektryczny` (double), `spin` (double), `czyStabilna` (bool) oraz `czyAntyczastka` (bool).
2. Zaimplementuj konstruktor klasy `Czastka`. Ponieważ pola są stałe, musisz użyć **listy inicjalizacyjnej** (przed klamrami konstruktora), aby przypisać im wartości. Dodaj wewnątrz konstruktora wypisywanie na konsolę komunikatu z nazwą klasy i wartością pola `nazwa`.
3. Stwórz **wirtualny destruktor**, który wypisze w konsoli informację o usunięciu cząstki (analogicznie do konstruktora).
4. Napisz publiczne „gettery" dla każdego z pól. Getter nazwy powinien dodawać przedrostek „Anty-", jeśli flaga `czyAntyczastka` ma wartość `true`.
5. Utwórz czysto wirtualną (abstrakcyjną) metodę `rozpad()`, która zwraca `std::vector<Czastka*>`.

Zadanie 2: Hierarchia cząstek Modelu Standardowego

1. **Klasy drugiego rzędu (Spin):** Utwórz klasy `Fermion` oraz `Bozon` dziedziczące publicznie po `Czastka`.
   - W konstruktorze `Fermion` sprawdź, czy przekazany spin jest liczbą połówkową (np. 0.5, 1.5). Jeśli nie, rzuć wyjątek `std::runtime_error`.
   - W konstruktorze `Bozon` analogicznie sprawdź, czy spin jest liczbą całkowitą (np. 0.0, 1.0).
2. **Klasy trzeciego rzędu (Ładunek):** Utwórz klasy `Lepton` i `Kwark` dziedziczące po `Fermion`.
   - W konstruktorze `Lepton` upewnij się, że ładunek jest całkowity (-1, 0, 1).
   - W konstruktorze `Kwark` upewnij się, że ładunek jest wynosi $\pm 1/3$ lub $\pm 2/3$.
3. **Klasy cząstek konkretnych:**
   - `Elektron` (lepton, stabilny, ładunek -1 lub 1, spin 1/2)
   - `Neutrino` (lepton, stabilny, ładunek 0, spin 1/2)

- `KwarkGorny` (kwark, stabilny, ładunek 2/3, lub -2/3 dla anty, spin 1/2)
- `BozonZ` (bozon, niestabilny, ładunek 0, spin 1)

4. *Uwaga diagnostyczna:* Każdy konstruktor i destruktor na każdym poziomie dziedziczenia powinien wypisywać nazwę klasy i pole `nazwa` na konsolę.

Wskazówki:

- Konstruktor klasy pochodnej musi wywoływać konstruktor klasy bazowej. Domyślnie wywoływany jest konstruktor klasy bazowej, który nie przyjmuje żadnych argumentów (o ile taki istnieje). Inny konstruktor można wywołać w liście inicjalizacji konstruktora klasy pochodnej.
- Do liczenia reszty z dzielenia (razem z częścią ułamkową) służy funkcja `std::fmod`.
- "Rzucanie" błędu: `throw std::runtime_error("OPIS BLEDU");`.

Zadanie 3: Uproszczony rozpad bozonu Z

W klasach cząstek konkretnych zaimplementuj metodę `rozpad()`. Dla cząstek stabilnych zwraca ona pusty wektor, a dla klasy `BozonZ` zwraca wektor wskaźników na nowo utworzone produkty rozpadu. Prawdopodobieństwa rozpadu:

- **70%:** Para `KwarkGorny` i anty-`KwarkGorny`.
- **20%:** Para `Neutrino` i anty-`Neutrino`.
- **10%:** Para `Elektron` i anty-`Elektron`.

Przeprowadź symulację rozpadu bozonu Z. Zadbaj o wyczyszczenie pamięci alokowanej za pomocą `new`. Zwróć uwagę na kolejność wywołania konstruktorów i destruktorów klas.

Wskazówki:

- Do generowania liczb losowych można użyć funkcji `rand`. Generator inicjalizuje się funkcją `srand`. Tak generowane liczby pseudolosowe są kiepskiej jakości. Funkcje dające lepszy rozkład liczb pseudolosowych poznamy na kolejnych zajęciach.

Opracowanie: Piotr Dziekan