# Programowanie II R

Zadania – seria 7.

Polimorfizm - rozpad bozonu Higgsa

Celem ćwiczeń jest uproszczone modelowanie eksperymentów prowadzących do wykrycia bozonu Higgsa. Do reprezentacji cząstek potrzebne są klasy przygotowane podczas poprzednich zajęć oraz dwie nowe klasy reprezentujące bozon W i bozon Higgsa. Przyjmujemy następujące, uproszczone kanały rozpadu cząstek niestabilnych oraz ich prawdopodobieństwa:

- **Bozon Higgsa ($H$):**
  - $H \to t\bar{t}$ (kwark i anty-kwark górny) – 75%
  - $H \to WW$ (bozon i anty-bozon W) – 22%
  - $H \to ZZ$ (dwa bozony Z) – 3%
- **Bozon Z ($Z$):**
  - $Z \to t\bar{t}$ (kwark i anty-kwark górny) – 70%
  - $Z \to \nu_e + \nu_e$ (neutrino i anty-neutrino) – 20%
  - $Z \to e^+e^-$ (elektron i pozyton) – 10%
- **Bozon W ($W$):**
  - $W \to t\bar{t}$ (kwark i anty-kwark górny) – 67%
  - $W \to e\nu_e$ (elektron/pozyton i anty-neutrino/neutrino) – 33%

Odpowiednie klasy można dodać do kodu z poprzednich zajęć (usuwając wypisywanie informacji w konstruktorach i destruktorach), bądź posłużyć się udostępnionym plikiem `model_standardowy.hpp`.

## Zadanie 1: "Złoty kanał" rozpadu bozonu Higgsa

Wykonaj symulację, w której na początku tworzonych jest co najmniej $10^5$ bozonów Higgsa, a następnie modelowany jest rozpad cząstek do momentu, aż zostaną tylko cząstki stabilne. Na podstawie produktów rozpadu policz, ile razy nastąpił tzw. "złoty kanał" rozpadu: $H \to ZZ \to e^+e^-e^+e^-$. Taki rozpad, ze względu na łatwe do zmierzenia produkty końcowe, posłużył do udowodnienia istnienia bozonu Higgsa w LHC.

- Do przechowywania cząsteczek wykorzystaj `std::vector<Czastka*>`.
- Pamiętaj o czyszczeniu pamięci (`delete`) alokowanej przy użyciu `new`.
- Klasy zdefiniowane w pliku `model_standardowy.hpp` używają funkcji `rand()` do generowania liczb pseudolosowych. Generator należy zainicjalizować funkcją `srand(unsigned int seed)`. Jako seed mozna użyc aktualnego czasu (funkcja `time(nullptr)` z nagłówka `<ctime>`).

- Do sprawdzenia rodzaju cząstki należy użyć `dynamic_cast`. Ten operator służy do zmiany typu obiektu między dziedziczącymi klasami. Przykładowo, `dynamic_cast<Derived*>(B)` zwróci obiekt typu `Derived*` jeśli `B` to wskaźnik na instancję klasy, po której dziedziczy `Derived`. W przeciwnym razie zwrócony zostanie `nullptr`.
- W pliku `model_standardowy.hpp` klasy należą do przestrzeni nazw `ModelStandardowy`. Spoza tej przestrzeni nazw można ich użyć za pomocą (przykładowo) `ModelStandardowy::Czastka`. Można też zadeklarować `using namespace ModelStandardowy;`.

# Zadanie 2: Refaktoryzacja kodu z zadania 1: inteligentne wskaźniki

Często unika się bezpośredniego korzystania z `new` i `delete`. Ręczne zarządzanie pamięcią jest bardzo podatne na błędy i prowadzi do trudnych w debugowaniu wycieków. Standard nowoczesnego C++ rozwiązuje ten problem poprzez inteligentne wskaźniki - "wrappery" dla zwykłych wskaźników, które same czyszczą pamięć, gdy są niszczone. W tym zadaniu wykorzystany będzie `std::unique_ptr` - inteligentny wskaźnik do obiektu, do którego istnieje tylko jeden wskaźnik (w przypadku wielu wskaźników do jednego obiektu używa się `std::shared_ptr`).

## Cel zadania

Zmodyfikuj kod z Zadania 1 tak, aby całkowicie pozbyć się z niego słów kluczowych `new` i `delete`.

## Wytyczne do modyfikacji

1. **Zmiana typu wskaźników:** Zastąp wszystkie surowe wskaźniki `Czastka*` inteligentnym wskaźnikiem `std::unique_ptr<Czastka>` (plik nagłówkowy `memory`).
2. **Tworzenie obiektów:** Zamiast używać `new`, do tworzenia nowych cząstek (np. w metodzie `rozpad()`) wykorzystaj funkcję `std::make_unique<TypCzastki>(...)`.
3. **Transfer własności:** Ponieważ `unique_ptr` zapewnia wyłączną własność nad obiektem, nie można go skopiować do drugiego wektora. Podczas przepisywania cząstek stabilnych oraz dopisywania nowych produktów do wektora `nastepna_generacja`, musisz użyć funkcji `std::move()`, aby przenieść zasób (np. `vec.push_back(std::move(c))`).
4. **Rzutowanie polimorficzne:** `dynamic_cast` działa tylko na "surowych" wskaźnikach lub referencjach. Wykorzystaj metodę `get()` na swoim inteligentnym wskaźniku, aby uzyskać surowy wskaźnik.

*Opracowanie: Piotr Dziekan*