Programowanie II R

Zadania – seria 5.

Dziedziczenie

Zadanie 1: Obiekty lekkie i masywne

Na podstawie klasy `Planeta` z poprzednich zadań utwórz klasę ObiektKosmiczny oraz dziedziczące po niej klasy ObiektMasywny (który przyciąga inne obiekty) i ObiektLekki (którego przyciąganie zaniedbujemy). Klasa `ObiektKosmiczny` powinna zawierać pola i metody wspólne dla obiektów lekkich i masywnych, oraz interfejs, czyli czysto wirtualne metody, których różne definicje zostaną zawarte w klasach ObiektLekki i ObiektMasywny.

Przykładowo, ObiektKosmiczny może zawierać:

1. Pola: nazwa, masa, położenie itd.
2. Metodę aktualizujStan.
3. Czysto wirtualną metodę `Przyciagaj`, która liczy siłę grawitacji tego obiektu działającą na inny obiekt (przekazany jako argument tej metody). Różne definicje tej metody znaleźć się powinny w klasach ObiektLekki i ObiektMasywny.

Poprawność działania klas sprawdź w układzie dwóch obiektów: jednego lekkiego (który powinien być przyciągany) i jednego masywnego (który nie powinien być przyciągany).

Wskazówki:

- Klasę pochodną definiujemy: `class Dervied : public Base {...};`
- Przykład metody czysto wirtualnej: `virtual void foo() = 0;`
- W klasie pochodnej, metody przesłaniające funkcję wirtualną dobrze jest oznaczyć słowem `override`, np. `virtual void foo() override{...}`
- W klasach bazowych dobrą praktyką jest definiowanie wirtualnego destruktora, np. wirtualny destruktor o domyślnej definicji: `virtual ~Base() = default;`
- Domyślnie, konstruktory nie są dziedziczone. Można je odziedziczyć pisząc w klasie pochodnej `using Base::Base;`

Zadanie 2: Uproszczona symulacja Układu Słonecznego

Zastosuj klasy ObiektLekki i ObiektMasywny w symulacji z poprzednich zajęć. Przyjmij, że jedynym obiektem masywnym jest Słońce. Obiekty przechowuj w kontenerze zawierającym wskaźniki na ObiektKosmiczny (np. `std::vector<ObiektKosmiczny*>`.

Wskazówki:

- Metody wirtualne klasy pochodnej (Derived) można wywołać poprzez wskaźnik do klasy bazowej (Base):

```
Base *pB = new Derived;
pB->foo();
```

Pozwala to na użycie jednego rodzaju wskaźnika do reprezentacji różnych klas (przy czym rodzaj klasy może być znany dopiero po uruchomieniu programu) - polimorfizm.

Zadanie 3: Symulacja Układu Słonecznego z mniejszymi obiektami

Celem jest rozszerzenie symulacji US o księżyce, asteroidy i kometę. Lekkie obiekty słabo oddziaływują grawitacyjnie, dlatego można zaniedbać liczenie ich przyciągania w celu skrócenia czasu symulacji. Plik `uklad_sloneczny_pelny.csv` zawiera informacje o obiektach, które powinny być modelowane. Do wczytania danych z pliku można użyć załączonej funkcji `wczytajUklad`.

1. Modeluj układ przez 75 lat. Czy orbity pozostają stabilne? Czy kometa Halleya wraca w okolice Słońca?
2. Sprawdź czas wykonywania programu, np. narzędziem `time` w Linuxie bądź używając biblioteki `<chrono>`.
3. Zmodyfikuj program tak, żeby liczone było przyciąganie między wszystkimi obiektami. Jak to wpływa na czas wykonania? Pomiń przyciąganie między obiektami, które oddalone są o mniej niż 1e-15 AU (ponieważ położenia i prędkości asteroid zostały wygenerowane losowo, a zderzenia między nimi nie są modelowane).

Wskazówki:

- Czas wykonania może znacznie skrócić optymalizacja podczas kompilacji, np.: `$g++ -O2 prog.cpp`.
- Jeśli do całkowania używasz jawnej metody Eulera, możesz łatwo poprawić stabilność orbit używając metody półjawnej (`https://en.wikipedia.org/wiki/Semi-implicit_Euler_method#The_method`)

Opracowanie: Piotr Dziekan.