Programowanie II R

Zadania – seria 4.

Klasy

1 Zadanie 1: Klasa `Vector2D`

Celem zadania jest napisanie klasy reprezentującej wektor w dwuwymiarowej przestrzeni.

1. Utwórz plik nagłówkowy `Vector2D.hpp`. Zdefiniuj w nim klasę `Vector2D`.
2. Klasa powinna zawierać:
   - Prywatne pola `double x, y` reprezentujące składowe wektora.
   - Konstruktor domyślny inicjalizujący x=0, y=0.
   - Konstruktor pozwalający inicjalizować inne wartości x, y.
   - Przeciążone operatory: `+` (suma wektorów), `-` (różnica wektorów) oraz `*` i `/` (mnożenie i dzielenie wektora przez skalar typu `double`).
   - Metody: `length()` zwracającą długość wektora oraz `print()` wyświetlającą współrzędne.
3. **Test poprawności:** Napisz program wykonujący powyższe operacje, sprawdź poprawność wyników.

Wskazówki:

- Metody i operatory, które nie modyfikują obiektu (czyli w naszym przypadku nie zmieniają wartości x i y), powinny być oznaczone wyrażeniem `const`, np. `Vector2D operator+(Vector2D v) const {...}`.
- W konstruktorze do inicjalizacji wartości pól najlepiej wykorzystać listę inicjalizacyjna: `Vector2D(double _x, double _y) : x(_x), y(_y){}`, a nie: `Vector2D(double _x, double _y) : {x=_x, y=_y}`. Lista inicjalizacyjna jest bardziej wydajna i pozwala inicjalizować pola typu const i referencje.

2 Zadanie 2: Symulacja Układu Słonecznego w 2D

1. W oddzielnym pliku nagłówkowym zdefiniuj klasę `Planeta`. Klasa powinna zawierać:
   - prywatne pola: nazwa, masa, trzy wektory dwuwymiarowe (`Vector2D`): położenie, prędkość, siła wypadkowa.
   - konstruktor inizjalizujący wartości pól.
   - metodę zerującą wektor siły wypadkowej.
   - metodę do obliczania siły oddziaływania między dwiema planetami i dodającą wynik do siły wypadkowej.
   - metodę, która oblicza przyspieszenie na podstawie siły wypadkowej, a następnie aktualizuje prędkość i położenie metodą Eulera z zadanym krokiem czasowym.

- metodę, która wypisuje nazwę planety jej aktualne położenie i odległość od Słońca.

2. Użyj klasy `Planeta` do napisania programu modelującego ewolucję Układu Słonecznego (Słońce oraz planety od Merkurego do Urana). Zainicjalizuj planety na osi X (koniunkcja) z prędkościami prostopadłymi (oś Y). Wykorzystaj dane początkowe z tabelki. Przeprowadź symulację ruchu planet przez co najmniej 100 lat. Sprawdź stabilność orbity Ziemi — czy po jednym pełnym roku wraca do punktu startowego, jak bardzo zmienia się odległość Ziemi od Słońca w trakcie roku.

Nazwa	**Masa** $[M_{Sun}]$	**Odległość** $[AU]$	**Prędkość** $[AU/rok]$
Słońce	1.0	0.0	0.0
Merkury	1.66e-7	0.387	10.10
Wenus	2.44e-6	0.723	7.39
Ziemia	3.00e-6	1.000	6.28
Mars	3.22e-7	1.524	5.08
Jowisz	9.54e-4	5.203	2.75
Saturn	2.85e-4	9.537	2.04
Uran	4.36e-5	19.191	1.43

Wskazówki:

- Nazwę planety można przechować w zmiennej typu `std::string`.
- Stałą grawitacji dobrze zdefiniować wewnątrz klasy jako prywatne pole statyczne (czyli dzielone między wszystkimi instancjami klasy) o wartości znanej w momencie kompilacji (`static constexpr double G = ...`).
- Wygodnie wykorzystać jednostki: AU (jednostka astronomiczna) dla położenia, rok dla czasu, AU/rok dla prędkości, masa Słońca dla masy. Stała grawitacji wynosi $G \approx 39.478 AU^3 rok^{-2} M_{slonce}^{-1}$.

Uwagi: Jeśli argument funkcji nie jest przez nią modyfikowany, ale tworzenie jego kopii jest kosztowne, często bardziej wydajnie jest przekazać argument przez stałą referencję (jak w metodzie liczącej siłę grawitacji).