Zadanie 2 17-20 marca 2025 r.

kurs języka C++

punkty i koła na płaszczyźnie

Instytut Informatyki
Uniwersytetu Wrocławskiego Paweł Rzechonek

Prolog

Koło to zbiór wszystkich punktów płaszczyzny, których odległość od ustalonego punktu na tej płaszczyźnie (nazywanego środkiem koła) jest mniejsza lub równa od pewnej ustalonej długości (zwanej promieniem koła).

Przekształcenie izometryczne płaszczyzny zachowuje odległości dowolnej pary punktów — jeśli jakąś figurę przekształcimy za pomocą izometrii, to nie zmieni ona ani kształtu ani rozmiaru. W geometrii euklidesowej istnieją cztery zasadnicze przekształcenia płaszczyzny:

1. przesunięcie/translacja – przesunięcie wszystkich punktów płaszczyzny/figury o zadany wektor;
2. obrót – obrót wszystkich punktów płaszczyzny/figury wokół ustalonego punktu o zadany kąt;
3. symetria środkowa – odbicie wszystkich punktów płaszczyzny/figury względem ustalonego punktu;
4. symetria osiowa – odbicie wszystkich punktów płaszczyzny/figury względem ustalonej osi.

Zadanie

Zdefiniuj klasy `punkt` i `kolo`, które będą reprezentowały odpowiednio punkt i koło na płaszczyźnie euklidesowej. Klasa `punkt` powinna zawierać dwa pola `x` i `y` typu `double` do pamiętania współrzędnych punktu. Klasa `kolo` ma reprezentować koło na płaszczyźnie z określonym środkiem typu `punkt` i promieniem typu `double`. Pamiętaj o hermetyzacji, aby ukryć stan każdego obiektu.

W wymienionych klasach zdefiniuj konstruktory (w tym konstruktor bezargumentowy) oraz funkcje składowe do wykonywania przekształceń izometrycznych (przesunięcia, obroty oraz symetrie środkowe i osiowe). W konstruktorze klasy `kolo` zgłoś wyjątek, gdy zadany promień będzie niedodatni. Ponadto obydwu klasach zdefiniuj gettery (do odczytu parametrów figur) i settery (do zmiany parametrów figur).

Dla klasy `punkt` zdefiniuj funkcją globalną, która będzie wyznaczać odległość pomiędzy parą punktów.

W klasie `kolo` zdefiniuj funkcje składowe obliczające obwód koła i pole powierzchni koła oraz funkcję składową sprawdzającą, czy zadany punkt należy do koła. Dodatkowo zdefiniuj dwie

funkcje globalne – pierwsza funkcja ma sprawdzać czy jedno koło zawiera się w drugim, druga funkcja ma sprawdzać czy dwa koła są rozłączne.

Na koniec napisz program rzetelnie testujący działanie obiektów tych klas. W programie testującym uruchom wszystkie funkcje globalne oraz wszystkie publiczne funkcje składowe. Rezultaty wywołanych funkcji zaprezentuj na standardowym wyjściu. Wszystkie obiekty w tym programie powinny być utworzone na stosie.

Uwaga

Podziel program na pliki nagłówkowe i źródłowe. Funkcję `main()` z programem testującym umieść w osobnym pliku źródłowym.

Wskazówka

Do wykonania translacji będziesz potrzebować wektora — zdefiniuj go w postaci odrębnej klasy typu `wektor` (wektor pamiętaj w postaci kierunkowej dx i dy).

Do wykonania symetrii osiowej będziesz potrzebować prostej — zdefiniuj ją w postaci odrębnej klasy typu `prosta` (prostą pamiętaj w postaci równania ogólnego Ax + By + C = 0, przy czym A i B nie mogą być jednocześnie równe 0).

Zalecenie

Zawsze, gdy napotykamy w programie jakieś błędy, niejednoznaczności czy sprzeczności należy to sygnalizować na pomocą wyjątków. Sytuacje wyjątkowe zgłaszamy instrukcją `throw`. Na przykład w konstruktorze klasy reprezentującej koło należy zgłosić wyjątek, gdy promień będzie niedodatni. Niech wyjątkami będą obiekty typu `invalid_argument` (deklaracje tej klasy wyjątku znajduje się w pliku nagłówkowym `<stdexcept>`).

Ważne elementy programu

- Definiowanie klas w pliku nagłówkowym i funkcji składowych w pliku źródłowym.
- Ukrywanie stanu w definicjach klas i upublicznienie funkcjonalności (hermetyzacja).
- Definicja konstruktorów, w tym konstruktora bezargumentowego.
- Wykorzystanie klasy `punkt` w definicji klasy `kolo` (kompozycja).
- Implementacja algorytmów geometrii analitycznej.
- Zgłaszanie wyjątków.
- Program testujący w funkcji `main()`.