**Ogólny zamysł projektu:**  
Biblioteczka z użytecznymi strukturami i algorytmami do zadań geometrycznych z Competetive Programming.

**KLASY:**

* **Point**
  1. Klasa abstrakcyjna, tak jakby interfejs dla punktu
  2. Zawiera tylko czysto wirtualne metody
  3. Wymusza aby wszystkie klasy implementujące interfejs Point miały zdefiniowane:
* operatory matematyczne (+,-),
* operatory porównywania (<,>,<=,>=,==),
* length - funkcję zwracającą długość wektora
* Iloczyn skalarny
* Distance - funkcję biorącą jeden Punkt jako argument i zwracającą odległość euklidesową do tego punktu
  1. Z przyczyn technicznych nie udało mi się dołożyć tu funkcji norm i unrooted\_distance
* **Point2D**
  1. Klasa reprezentująca 2-wymiarowy punkt
  2. Pochodna klasy Point
  3. Zawiera pola opisujące współrzędne – x i y
  4. Współrzędne będą miały generyczny typ danych (int, long long,float,double,long double)
  5. Zawiera dodatkowe metody:
* Unrooted\_distance – to samo co distance tylko bez nałożonego pierwiastka
* Norm - norma
* Angle – zwraca kąt względem osi OX lub względem jednego lub dwóch punktów
* Triangle\_area – zwraca pole trójkąta wyznaczone przez siebie i dwa dodatkowe punkty
* Signed\_area\_parallelogram – zwraca skierowane pole równoległoboku wyznaczone przez siebie i dwa dodatkowe punkty
* Clockwise\_area – funkcja sprawdzająca czy this punkt1 i punkt 2 są w kolejności zgodnej z ruchem wskazówek zegara
* **Point3D**
  1. Klasa reprezentująca 3-wymiarowy punkt
  2. Pochodna klasy Point
  3. Zawiera pola opisujące współrzędne – x, y, z
  4. Współrzędne będą miały generyczny typ danych (int, long long,float,double,long double)
  5. Zawiera dodatkowe metody:
* Unrooted\_distance – to samo co distance tylko bez nałożonego pierwiastka
* Norm - norma
* **PointND**
  1. Klasa reprezentująca n-wymiarowy punkt
  2. Pochodna klasy Point
  3. Zawiera listę długości n, która trzyma kolejne współrzędne n wymiarowego punktu
  4. Współrzędne będą miały generyczny typ danych (int, long long,float,double,long double)
  5. Zawiera dodatkowe metody:
* Unrooted\_distance – to samo co distance tylko bez nałożonego pierwiastka
* Norm - norma
* **Line2D**
  1. Klasa reprezentująca prostą w 2D
  2. Prosta zapisana w postaci ogólnej
  3. Zawiera konstruktor przyjmujący :
* Współczynniki dla postaci kierunkowej
* Współczynniki dla postaci ogólnej
* Dwa punkty2D
  1. Zawiera metody:
* Parallel - Czy proste są równoległe
* Inside - Czy punkt należy do prostej
* Intersect – funkcja sprawdza czy punkt przecięcia istnieje
* Intersection - Punkt przecięcia dwóch prostych
* Angle - Kąt prostej względem osi OX lub punktu2D
* (prywatna) normalize – funkcja dla typów zmienno-przecinkowych dba by współczynniki prostej nie były za duże, co zwiększy precyzje przy obliczeniach. Dla typów całkowitych dzieli współczynniki przez gcd() współczynników
* **Ray2D**
  1. Klasa reprezentująca półprostą w 2D
  2. Pochodna klasy Line2D
  3. Zawiera dodatkowe pola utrzymujące:
* Początek półprostej
* Kierunek półprostej – bool który mówi czy punkty mają większe x od początku półprostej czy mniejsze lub czy jest półprostą pionową w górę
  1. Nadpisuje metodę inside z klasy bazowej
* **Segment2D**
  1. Klasa reprezentująca odcinek w 2D
  2. Pochodna klasy Line2D
  3. Zawiera dodatkowe pola utrzymujące:
* Początek i koniec odcinka
  1. Nadpisuje metodę inside z klasy bazowej
  2. Zawiera dodatkową metodę:
* Length – Długość
* **Set\_of\_Point2D**
  1. Klasa reprezentująca zbiór punktów 2D
  2. Zawiera listę punktów, w której kolejność nie ma znaczenia
  3. Zawiera różne warianty metod modyfikujących zbiór:
* Add\_point - Dodaj punkt do zbioru
* Erase\_point - Usuń punkt ze zbioru
  1. Zawiera dodatkowe metody:
* Lex\_sort – sortowanie po kolejnych współrzędnych(leksykograficzne)
* Angle\_sort - Sortowanie kątowe względem punktu (domyślnie (0,0))
* Convex\_hull – funckja zwracająca otoczkę wypukłą tego zbioru, będącą obiektem klasy Polygon
* **Polygon2D**
  1. Klasa reprezentująca wielokąt w 2D
  2. Zawiera listę punktów, w której kolejność ma znaczenia
  3. Zawiera metody:
* Area – Zwraca pole wielokąta
* Inside – czy punkt jest w środku wielokąta