Zaimplementowane funkcje programu:
-Interaktywne dodawanie odcinków
-Losowanie zbioru odcinków
-Sprawdzanie czy dowolne odcinki nie dotykają się końcami i czy nie są pionowe
-Funkcja zapisująca zbiór odcinków do pliku i wczytująca z pliku
-Funkcja sprawdzająca czy dwa odcinki się przecinają
-Funkcje obsługujące zdarzenia: początek odcinka, koniec odcinka, przecięcie
-Drzewo AVL jako struktura stanu miotły
-Kolejka priorytetowa (PriorityQueue z biblioteki queue) jako struktura zdarzeń
-Funkcja anyLinesIntersect(lines) sprawdzająca czy dowolne dwa odcinki w zbiorze się przecinają
-Funkcja sweepingAlg(lines) wyszukująca wszystkie przecięcia w zbiorze odcinków zwracają ich listę w postaci (punkt, (odcinek1, odcinek2))
-Funkcja sweepingAlgWithScenes(lines, scenes) realizująca sweepingAlg, dodatkowo zapisując w liście scenes sceny wizualizujące działanie algorytmu

Funkcja znajdująca jakiekolwiek przecięcie i funkcja znajdująca wszystkie przecięcia nie różnią się zasadniczym algorytmem, działają one bliźniaczo. Różnicą jest fakt, iż pierwszy z nich zatrzymuje się na pierwszym znalezionym przecięciu i je zwraca.

Struktura stanu w drugiej funkcji jest wzbogacona o słownik zawierający punkty przecięć, potrzebny jest on do sprawdzania czy znaleziony punkt nie został wcześniej zakwalifikowany jako przecięcie odcinków.

Struktura zdarzeń w obu przypadkach jest taka sama, różnią się tylko przechowywane zdarzenia, funkcja anyLinesIntersect() w swojej kolejce przechowuje tylko zdarzenia początek i koniec odcinka, sweepingAlg() dodatkowo w trakcie trwania algorytmu dodaje do kolejki zdarzenia punkt przecięcia. Można zauważyć, że w pierwszym przypadku do implementacji struktury zdarzeń wystarczyłaby posortowana lista zdarzeń, wyznaczona na początku. Nie zmieniłoby to jednak złożoności funkcji.