W programie trzeba było wyznaczyć podzbiór punktów który tworzy tzw. otoczkę wypukłą, która otocza wszystkie inne punkty zbioru  
Do napisania programu użyłem algorytmu Grahama, który użyłem ze względu na jego złożoność obliczeniową wynoszącą O(nlog (n)).  
Polega on na znalezieniu spośród zbioru punktu o najniższej wartości y, a jeśli jest takich kilka to dodatkowo o najniższym x, używając tego punktu ,,głównego” jako początek sortujemy rosnąco inne punkty w zależności od kąta wektorem tych punktów oraz osią OX (jeżeli kąt jest taki sam dla różnych punktów, wtedy wybieramy ten, który jest bliżej punktu ,,głównego”). Ostatnim krokiem jest wyliczenie iloczynu wektorowego obecnego punktu oraz 2 następnych, jeśli wynik jest + wtedy zachowujemy środkowy wektor a w innym przypadku usuwamy go. Po powtórzeniu dla każdego punktu powinniśmy mieć odpowiedż na zadanie.

Do realizacji algorytmu Grahama użyłem :  
Struktury Point: przechowująca współrzędne x,y  
Funkcja pointAngle: funkcja oblicza kąt względem punktu odniesienia(orientationPoint) a osią OX wykorzystując atan2  
Funkcja pointLenght: funkcja obliczająca odległość punktów do punktu odniesienia aby uniknąć konfliktów dla tych samych kątów  
Funkcja pointCrossProduct: funkcja ta oblicza iloczyn wektorowy dla 3 punktów, dzięki której możemy sprawdzić czy dany punkt należy do szukanych punktów.  
Funkcja ConvexHull: główna funkcja programu, której zadaniem jest znaleźć oreintaionPoint(punkt o najniższym Y, a jeśli Y nie jest unikalne to dodatkowo punkt o mniejszym X), później punkty są sortowane względem kąta tworzonego z punktem odniesienia, oraz usuwamy punkty z vectora points punktów, które leżą wewnątrz otoczki  
Vector <Point> points: Tablica dynamiczna przechowywująca punkty zbioru, w programie używana jest tez komenda points.size()która przechowuje ilość punktów zbioru  
Vector <Point>outline: Tablica dynamiczna, której za zadanie jest ,,zbierać” punkty należące do otoczki wypukłej