**Dokument opisujący rozwiązanie zadania rekrutacyjnego nr 2**

W pierwszej kolejności rozwiązywanie tego zadania zacząłem od stworzenia klasy **Point**,która pomoże nam w przechowywaniu współrzędnych punktów, a także zwiększy czytelność kodu. Znajduje się w niej konstruktor oraz metody pomocnicze, w tym gettery i to\_string wraz z argumentem, pozwalającym na wyświetlenie odpowiedniej ilości cyfr po przecinku. Settery nie były potrzebne, ponieważ nie musimy zmieniać współrzędnych.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Działanie programu natomiast zaczynamy od pobrania nazwy pliku oraz uruchomienia metody loadPoints, która przyjmuje nazwę pliku i zwraca vector z obiektami Point, przechowującymi wszystkie współrzędne podane w pliku. Metoda rozpoczyna od stworzenia strumienia, który będzie używany do odczytania danych z pliku, a następnie sprawdza czy na pewno został poprawnie otwarty. Kolejnym krokiem jest zainicjowanie wcześniej wspomnianego wektora, pobranie liczby współrzędnych z pliku i sprawdzenie, czy jest ich wystraczająca ilość, aby stworzyć otoczkę. Na sam koniec zostaje nam jedynie pobranie pozostałych danych, zamknięcie strumienia/pliku oraz zwrócenie vectora.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

W osobnym pliku nazwanym Algorithm znajduje się część programu odpowiedzialna za znalezienie otoczki. Jest on oparty na algorytmie Jarvisa.

W pierwszej kolejności znajdujemy pierwszy punkt otoczki, który wyznaczamy poprzez wybranie punktu z najmniejszą współrzędną y, a jeśli jest kilka takich punktów, to wybieramy ten z najmniejszą współrzędną x. Metoda za to odpowiedzialna zwraca indeks takiego punktu.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

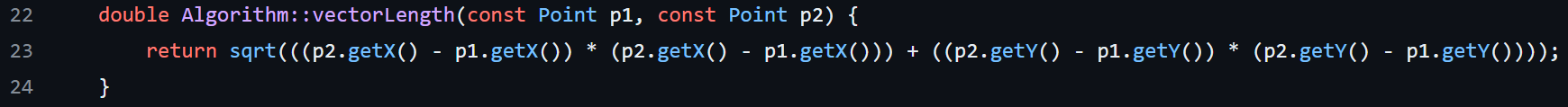
Następnie szukamy kolejnych punktów poprzez znalezienie największego kąta pomiędzy trzema wybranymi punktami. Możemy to zobaczyć na rysunku poniżej.

Obraz zawierający diagram, krąg, linia, tekst

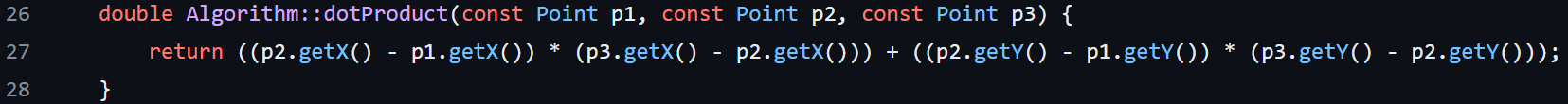
Opis wygenerowany automatycznie

Za znalezienie kąta pomiędzy tymi punktami odpowiedzialne są u mnie trzy metody. Ich główna idea to obliczenie kąta między wektorami. Odpowiedzialne są one kolejno za:

Obliczenie długości wektora między dwoma punktami.

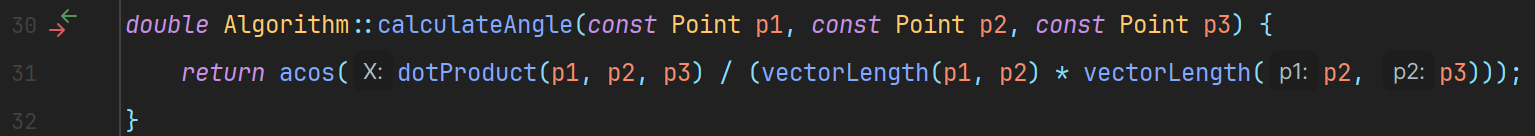


Obliczanie iloczynu skalarnego dwóch wektorów.



Obliczanie wartości kąta w radianach, ze wzoru:

Obraz zawierający Czcionka, tekst, pismo odręczne, numer

Opis wygenerowany automatycznie

W głównej metodzie nazwanej findVertices szukamy drugiego punktu poprzez implementacje pętli przechodzącej po wszystkich punktach. Dwa pierwsze argumenty funkcji calculateAngle to kolejno pierwszy znaleziony punkt z pomniejszoną współrzędną x o 1, a drugi to pierwszy znaleziony punkt. Dzięki temu możemy otrzymać na początek wektor poziomy, który pomaga nam w znalezieniu 2 punktu otoczki.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Oczywiście wartość zmiennej najmniejszego kąta na początku to 3.15, ponieważ wartość liczby pi (3.14…) to 180 stopni czyli największa wartość kąta w której możemy znaleźć kolejny punkt. Następnie mamy pętlę while, która kończy się w momencie, gdy następny znaleziony punkt otoczki okazuje się być tym pierwszym. W niej znajduje się dokładnie ta sama pętla for, która różni się jedynie argumentami metody calculateAngle i korzysta już z wcześniej znalezionych punktów.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Na koniec metody zwracamy vector zawierający wszystkie punkty otoczki. A następnie już w main’ie wypisujemy te punkty metodą to\_string do terminala.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie