Projekt: Otoczka wypukła

Skład zespołu nr 1

Agata Wąsik
Nr indeksu 116020

Magda Sarełło
Nr indeksu 112986

Maciej Nachtygal
Nr indeksu 101100

Matteo Dawide Catalani
Nr indeksu 106114

Lista wykonanych zadań w projekcie

Agata Wąsik
Nr indeksu 116020

- Dokumentacja
- Zarządzanie projektem

Magda Sarełło
Nr indeksu 112986

- Dokumentacja
- Testowanie
- Przeanalizowanie kodu

Maciej Nachtygal
Nr indeksu 101100

- Pisanie programu

Matteo Dawide Catalani
Nr indeksu 106114

- Dokumentacja
- Odszukanie informacji na temat projektu

Manual

Zadanie, które program ma realizować

Program wyznacza otoczkę wypukłą czterech punktów na płaszczyźnie:

- informuje, jakim zbiorem jest otoczka wypukła (czworokąt, trójkąt, odcinek, punkt),
- program wypisuje współrzędne kolejnych wierzchołków otoczki wypukłej.

Dane wejściowe: współrzędne czterech punktów na płaszczyźnie.

Ograniczenia

- Punkty muszą znajdować się w pierwszej ćwiartce układu współrzędnych
- Zakres wartości dla x to [0, 800], dla y to [0, 500]
- Maksymalna liczba punktów wynosi 5
- Prezentacja punktów w programie jest lustrzanym odbiciem względem osi OX

Lista opcji do wyboru z krótkim opisem każdej z nich

Po pierwszym uruchomieniu programu, w folderze, w którym jest umieszczony plik z kodem źródłowym tworzy się log.txt. Zawiera on chronologiczny zapis wykonywanych przez program działań.

Interfejs użytkownika

-h, --help	Pokaż ten komunikat pomocniczy i wyjdź
-r, --random	Losowy zestaw czterech punktów
-n NUMBERS, --numbers NUMBERS	Wpisz punkty, przykład: „(1,1), (2,2), (3,3)”

Niezgodności z założeniami przekazanymi w treści zadania

Niekompletność programu

- Dla czworokątów, przy czyszczeniu linii, które nie wchodzą w skład otoczki wypukłej, mogą pozostać linie sprawdzające położenia pomiędzy punktami.

Funkcje dodatkowe nie objęte w treści zadania

- Plik z logami pokazujący kolejne kroki działania programu
- Możliwość stworzenia punktów z losowymi współrzędnymi
- Wyznaczenie otoczki wypukłej dla ilości punktów od 1 do 5

Testowanie

Testy zostały przeprowadzone na konkretnych przypadkach testowych w związku z niewielkim formatem programu.

W każdym przypadku warunki wejściowe były takie same: program został uruchomiony w wierszu poleceń, w systemie Windows. Wersja Pythona obecna na komputerze to 3.11, a przed uruchomieniem programu został zainstalowany pakiet z pliku requirements.txt.

Nazwa	Opis
Sprawdzenie działania --r	Wprowadzono w wierszu poleceń komendę `python main.py -r` Powinny zostać wybrane 4 losowe punkty o współrzędnych zawartych w zakresie wartości i dla nich przeprowadzone działanie programu.
2/3/4 punkty o takich samych współrzędnych	Wprowadzono w wierszu poleceń komendę `python main.py -n (x1,y1),(x1,y1)` Program powinien narysować punkt, przeprowadzić algorytm i wyświetlić komunikat, że powstała figura to punkt.
2/3/4 punkty leżące na linii prostej równoległej do osi x lub y	Wprowadzono w wierszu poleceń komendę `python main.py -n (x1,y1),(x1,y2)` Program powinien narysować odcinek zwierający wszystkie punkty i poprawnie określić go jako odcinek.
2/3/4 punkty leżące na linii prostej, która nie jest równoległa ani do osi x ani do y	Wprowadzono w wierszu poleceń komendę `python main.py -n (x1,y1),(x2,y2)` Program powinien narysować odcinek zwierający wszystkie punkty i poprawnie określić go jako odcinek.
3 punkty tworzące trójkąt o boku równoległym do osi x lub y	Wprowadzono w wierszu poleceń komendę `python main.py -n (x1,y1),(x1,y2),(x3,y3)` Program powinien narysować otoczkę wypukłą między tymi trzema punktami i nazwać ją trójkątem.
3 punkty tworzące trójkąt o boku, który nie jest równoległym ani do osi x ani do y	Wprowadzono w wierszu poleceń komendę `python main.py -n (x1,y1),(x2,y2),(x3,y3)` Program powinien narysować otoczkę wypukłą między tymi trzema punktami i nazwać ją trójkątem.
4 punkty tworzące trójkąt	Wprowadzono w wierszu poleceń komendę `python main.py -n (x1,y1),(x2,y2),(x3,y3),(x4,y4)` Program powinien narysować otoczkę wypukłą między trzema z czterech punktów, tak aby czwarty znajdował się wewnątrz trójkąta lub na jego obwodzie. Figura musi zostać nazwana trójkątem.
4 punkty tworzące czworokąt	Wprowadzono w wierszu poleceń komendę `python main.py -n (x1,y1),(x2,y2),(x3,y3),(x4,y4)` Program powinien narysować otoczkę wypukłą między wszystkimi czterema punktami i określić figurę jako czworokąt.

Opis kodu

Lista plików z kodem źródłowym wchodzących w skład programu

Do utworzenia programu wykorzystano język programowania Python w wersji 3.10.9.

Biblioteki

PyGame - Wykorzystano bibliotekę PyGame do wizualizacji okna, w którym pojawia się animacja algorytmu Jarvisa.

Logging – Wykorzystano bibliotekę Logging pozwalająca zapisać logi z informacjami o obecnie wykonywanej czynności.

Animacja polega na pojawiających i znikających odcinkach pomiędzy wierzkołkami. Efekt końcowy animacji wyświetla otoczkę wypukłą.

W momencie zakończenia programu wyświetlana jest informacja o kształcie otoczki wypukłej oraz które wierzchołki należą do otoczki.

Pliki

main.py – plik z kodem źródłowym programu
requirements.txt – wymagane paczki, które trzeba zainstalować dla prawidłowego działania programu

Schemat algorytmu albo pseudokod (z odniesieniami do kodu programu)

Algorytm jest napisany na podstawie pseudokodu z podanej dokumentacji na stronie Wrocławskiego Portalu Informatycznego (http://informatyka.wroc.pl/node/910?page=0,0):

```
JARVIS-HULL(P):
    punktOtoczki <- punkt ze zbioru P o największej współrzędnej x
    i = 1
    repeat
        CH[i] <- punktOtoczki
        nastepnyPunkt <- P[i]
        for j <- 1 to n:
            if nastepnyPunkt ≠ CH[i] and NA-LEWO(P[j], {CH[i], nastepnyPunkt}) > 0:
                nastepnyPunkt <- P[j]
            else if NA-LEWO(P[j], {CH[i], nastepnyPunkt}) < 0:
                if ODLEGLOSC(CH[i], nastepnyPunkt) < ODLEGLOSC(CH[i], P[j]):
                    nastepnyPunkt <- P[j]
        i = i + 1
        punktOtoczki <- nastepnyPunkt
    until nastepnyPunkt = CH[i]
    return CH
```

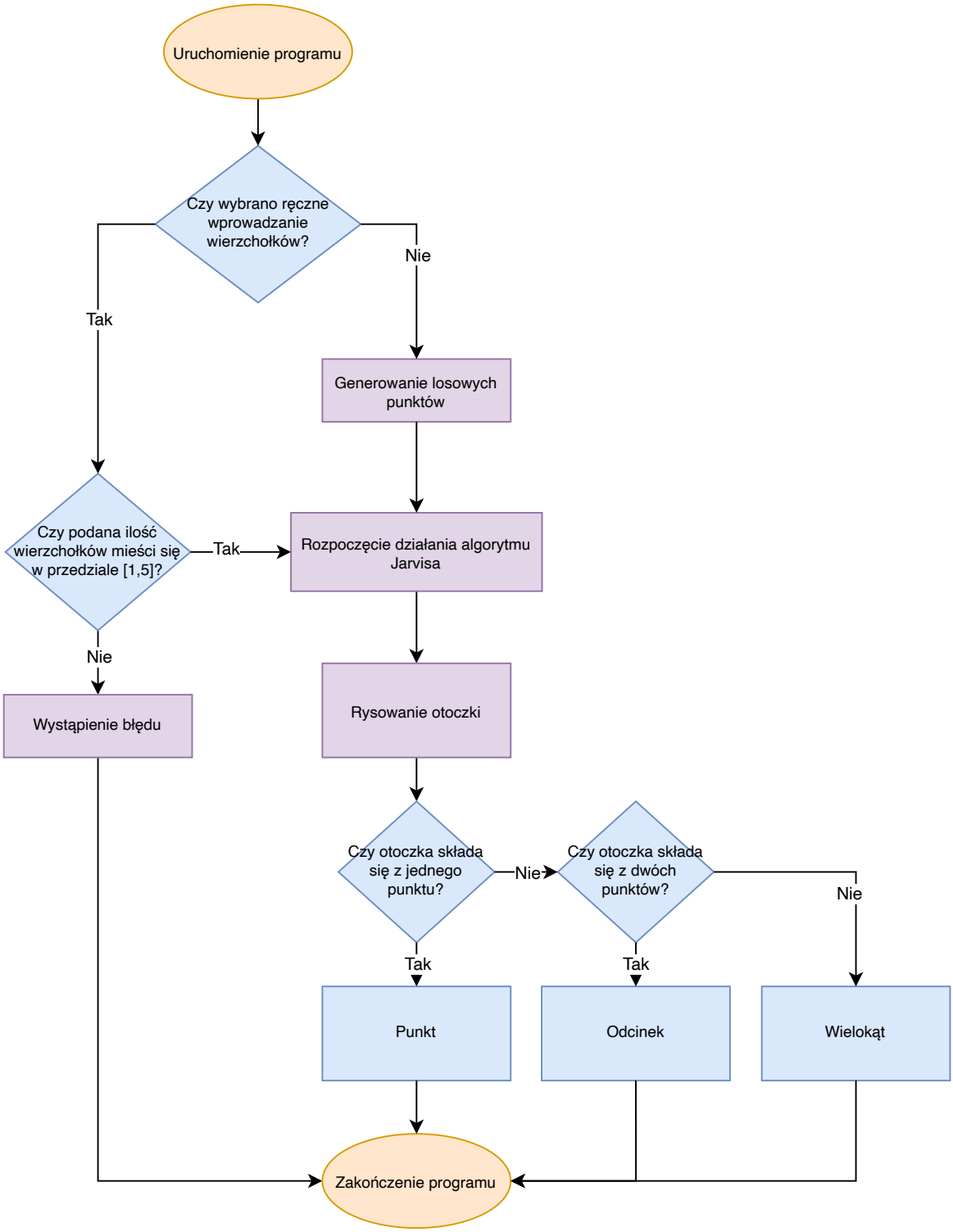
Oto znaczenie zmiennych i struktur użytych w procedurze JARVIS-HULL:

- $P[1 \dots n]$ – tablica punktów płaszczyzny podanych na wejściu
- $CH[1 \dots n]$ – tablica, w której przechowywane są kolejne punkty należące do otoczki wypukłej. Wielkość tej tablicy to n, ponieważ może się zdarzyć, że każdy punkt z P należy do otoczki (w większości przypadków będzie ich znacznie mniej).
- $punktOtoczki$ – jest to zmienna przechowująca punkt, który na pewno znajdzie się w otoczce.
- $nastepnyPunkt$ – zmienna pomocnicza, która przechowuje kandydatów na kolejny punkt otoczki
- $NA - LEWO$ – funkcja pomocnicza. Definicja funkcji to:

$$NA - LEWO(a, \{b, c\}) = (c_x - b_x)(a_y - b_y) - (a_x - b_x)(c_y - b_y)$$

- $ODLEGLOSC$ – funkcja pomocnicza. Definicja funkcji to:

$$ODGLEGLOSC(p_1, p_2) = \sqrt{(p_{1x} - p_{2x})^2 + (p_{1y} - p_{2y})^2}$$


Uruchomienie programu
Czy wybrano ręczne wprowadzanie wierzchołków?
Tak
Nie
Generowanie losowych punktów
Czy podana ilość wierzchołków mieści się w przedziale [1,5]?
Tak
Rozpoczęcie działania algorytmu Jarvisa
Nie
Wystąpienie błędu
Rysowanie otoczki
Czy otoczka składa się z jednego punktu?
Nie
Czy otoczka składa się z dwóch punktów?
Nie
Tak
Tak
Punkt
Odcinek
Wielokąt
Zakończenie programu

Zmiany wprowadzone po prezentacji

Poniższe zmiany naprawiają błąd związany z niekompletnością programu opisane w rozdziale „Niezgodności z założeniami przekazanymi w treści zadania”:

- Naprawiono błędy w funkcji określającej jaki wielokąt tworzy otoczka wypukła

Dodano nowe funkcje:

- Po zakończeniu działania programu można wcisnąć:
  - Klawisz ENTER, aby ponownie uruchomić program
  - Klawisz ESC lub Q, aby zakończyć działanie programu
  - Klawisz K lub L, aby zwolnić lub przyśpieszyć prędkość animacji