ДОДАТОК А ТЕКСТ ПРОГРАМИ

```
import math
from ortools.constraint_solver import routing_enums_pb2
from ortools.constraint_solver import pywrapcp
import numpy as np
import matplotlib.pyplot as plt

def create_data_model(): # функція створення моделі для пошуку маршруту
    """Зберігає дані"""
    data = {}
    # Розташування в блоках
    data["locations"] = [ # масив точок
      # початок точок АКТСІ-20-2
      (36, 454), (45, 430), (54, 399), (62, 379), (73, 347), (82, 316), (91, 288), (106, 255), (121, 275), (124, 286),
      (136, 338), (143, 340), (148, 360), (152, 379), (163, 403), (169, 425), (180, 454), (78, 381), (95, 381),
      (113, 381), (136, 381), (235, 253), (235, 286), (235, 353), (235, 400), (236, 427), (235, 454), (246, 364),
      (263, 349), (278, 325), (302, 299), (320, 279), (335, 262), (342, 253), (270, 340), (285, 360), (296, 377),
      (313, 401), (328, 421), (337, 438), (348, 454), (370, 262), (398, 262), (431, 262), (475, 262), (497, 262),
      (516, 262), (442, 454), (442, 417), (442, 375), (442, 327), (442, 290), (678, 436), (651, 449), (606, 445),
      (575, 429), (555, 399), (542, 369), (549, 318), (561, 281), (595, 266), (629, 259), (658, 262), (678, 266),
      (738, 253), (728, 294), (728, 358), (728, 395), (728, 427), (728, 454), (844, 371), (820, 373), (791, 371),
```

```
        (772, 373), (876, 281), (907, 268), (935, 266), (959, 277), (976, 310), (968, 344), (950, 369), (935, 388),
        (911, 410), (894, 425), (878, 443), (992, 445), (968, 445), (939, 445), (911, 443), (894, 443), (1140, 368),
        (1132, 327), (1127, 303), (1110, 273), (1084, 268), (1062, 273), (1042, 303), (1033, 336), (1029, 368),
        (1036, 405), (1055, 434), (1094, 447), (1118, 432), (1134, 405), (1140, 379), (1173, 371), (1197, 371),
        (1223, 373), (1243, 371), (1391, 443), (1363, 445), (1350, 443), (1310, 441), (1284, 443), (1267, 445),
        (1278, 283), (1302, 270), (1336, 264), (1369, 286), (1374, 310), (1367, 345), (1345, 377), (1315, 403),
        (1297, 423)
        # кінець точок АКТСІ-20-2
    ]
    data["num_vehicles"] = 1 # кількість проходів
    data["depot"] = 0 # з якої точки починаємо рух
    data2=np.array(data["locations"])
    x, y = data2.T
    plt.scatter(x, y)
    plt.gca().invert_yaxis()
    plt.show()
    return data

def compute_euclidean_distance_matrix(locations): # розрахунок евклідової відстані між точками
    """Створює функцію зворотнього виклику для повернення відстані між точками."""
    distances = {} # Створюємо пустий словник для збереження відстаней
    for from_counter, from_node in enumerate(locations):
```

```
        distances[from_counter] = {}
        for to_counter, to_node in enumerate(locations):
            if from_counter == to_counter:
                distances[from_counter][to_counter] = 0
            else:
                # Розраховуємо Евклідову відстань
                distances[from_counter][to_counter] = int(
                    math.hypot((from_node[0] - to_node[0]), (from_node[1] - to_node[1])) #статичний метод повертає квадратний корінь із суми квадратів своїх аргументів
                    )
        return distances


    def print_solution(manager, routing, solution): # функція виведення даних в консоль
        """Виведення рішення в консоль"""
        print(f"Відстань: {solution.ObjectiveValue()}") # Ціна проходження маршруту за обраним алгоритмом
        index = routing.Start(0) # початковий індекс переміщення
        plan_output = "Маршрут:\n"
        route_distance = 0

        while not routing.IsEnd(index): # цикл перебирання всіх точок переміщення і їх індексів до кінцевої (якщо не кінцевий індекс)
            plan_output += f" {manager.IndexToNode(index)} ->" # виведення одного індекса переміщення, а далі знак ->
            previous_index = index
            index = solution.Value(routing.NextVar(index))
            # Повертає вартість транзитної дуги між двома вузлами.
```

```
            # Вхід - змінні індекси вузла. Це повертає 0, якщо транспортний засіб < 0.
            route_distance += routing.GetArcCostForVehicle(previous_index, index, 0)
        plan_output += f" {manager.IndexToNode(index)}\n"
        print(plan_output)
        plan_output += f"Мета: {route_distance}m\n"

    def main():
        """Вхідна точка в програму"""
        # Визначаємо проблему з даними
        data = create_data_model()

        # Створюємо менеджер індексів маршруту (для керування маршрутом)
        manager = pywrapcp.RoutingIndexManager(
            len(data["locations"]), data["num_vehicles"], data["depot"]
        )

        # Створення моделі маршрутизації
        routing = pywrapcp.RoutingModel(manager)

        distance_matrix = compute_euclidean_distance_matrix(data["locations"]) # присвоюємо значення Евклідової відстані точок

        def distance_callback(from_index, to_index):
            """Повертає відстань між двома вузлами"""
            # Перетворення змінної маршрутизації Index на матрицю NodeIndex відстані
            from_node = manager.IndexToNode(from_index)
```

```
        to_node = manager.IndexToNode(to_index)
        return distance_matrix[from_node][to_node]

    transit_callback_index = routing.RegisterTransitCallback(distance_callback) # Зареєструвати зворотній виклик переміщення

    # Визначаємо вартість кожної дуги переміщення
    routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index)

    # Встановлення евристики першого рішення
    search_parameters = pywrapcp.DefaultRoutingSearchParameters()
    search_parameters.first_solution_strategy = (
        routing_enums_pb2.FirstSolutionStrategy.PATH_CHEAPEST_ARC # метод для визначення івристики
    )

    # Вирішення проблеми з параметрами пошуку за обраним методом
    solution = routing.SolveWithParameters(search_parameters)

    # Виведення результату в консоль
    if solution:
        print_solution(manager, routing, solution)


if __name__ == "__main__":
    main()
```

ДОДАТОК Б. ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

Позначення	Найменування	Дод. відомості
	Текстові документи	
1 ГЮИК. 505300.007ПЗ	Пояснювальна записка	66 арк., ф.
	Графічні документи	
2	Демонстраційні матеріали у вигляді	
	презентації	11 арк., ф.

Змін.	Арк.	Номер докум.	Підп.	Дата	ГЮИК. 505300.007ВД			
Розроб.		Ключник			*Розроблення програмного забезпечення для формування маршруту переміщення свердла для верстатів з числовим програмним керуванням* Відомість кваліфікаційної роботи	ЛІТ	Аркуш	Аркуші
Перевір.		Бронніков				Н	1	1
Т. контроль						ХНУРЕ кафедра КІТАР		
Н. контроль		Бронніков						
Затверд.		Невлюдов						