



EEE123 COMPUTER PROGRAMMING

MINI PROJECT

“Tabular Image Data to Matrix Data Converter
(for OpenCV purposes)”

School of Electrical and Electronic Engineering,
Universiti Sains Malaysia

Group 23

NO	GROUP MEMBERS	MATRIC NO.
1	LIM WIKY	22305415
2	LIM XING	22301447
3	MOHAMAD AYDIN BIN MOHD GHAZALI	22300017
4	AEESHA ERINA BINTI DAUD	22301711
5	EDWARD EE JIN HAO	22301440
6	LEE WEI XIAN	22302757

Table of Contents

Abstract	5
1. Introduction	6
1.1 Problem Statement	6
1.2 Objectives	6
1.3 Scope	8
1.4 Requirements & Specifications	9
Project Specifications	9
2. Methodology	11
2.1.0 – Intro	11
2.1.1 – [Section 1 – INPUT]	11
2.1.2 – [Section 2 – SORT]	12
2.1.3 – [Section 3 – Convert]	13
2.1.4 - Continued [Section 2 – SORT]	15
2.1.5 - Continued [Section 3 – CONVERT]	15
2.1.6 – [Section 4 – OUTPUT]	15
2.2 Flow Chart	16
2.2.1 Project General Flowchart	16
2.2.2 Program General Flowchart	17
2.2.3 Program Flowchart by Section	19
2.2.3.1 Start of code	19
2.2.3.2 [Section 1 – INPUT]	20
2.2.3.3 [Section 2 – SORT]	21
2.2.3.4 [Section 3 – CONVERT]	22
2.2.3.5 [Section 4 – OUTPUT]	24
2.2.4 Functions	25
2.2.4.1 List of functions (14)	25
[Section 1 – INPUT]	25
[Section 2- SORT]	25
[Section 3 – CONVERT]	25
[Section 4 – OUTPUT]	25

2.2.4.2 – Flowchart for functions in [Section 1 – INPUT]	26
2.2.4.3 – Flowchart for functions in [Section 2 – SORT]	29
2.2.4.4 – Flowchart for functions in [Section 3 – CONVERT]	34
2.2.4.5 – Flowchart for functions in [Section 4 – OUTPUT]	35
3. Testing Procedures	Error! Bookmark not defined.
3.0 Intro	38
3.1 Generation of raw data	38
3.1.1 Requirements of data generation	38
3.1.2 Manual	40
3.1.3 Auto	43
3.2 Data Validation	46
4. Results & Discussions	48
4.1 Trial Runs	48
4.1.1 Trial Run #1	48
4.1.2 Trial Run #2	49
4.1.3 Trial Run #3	50
4.1.4 Trial Run #4	51
4.2 Clean Mode	52
4.3 Vector As Data Container	52
4.4 Invalid data / Data out of bounds	55
5. Conclusions	56
6. References	57

Appendix	61
- Task Distribution	61
- // main.cpp (version 4.52 Vectorstruct)	63
- // functions.hpp	75
- Input data (Image set)	81
- Input data (Raw data coordinates)	82
- Output data (Results in output_v4.txt)	83

Abstract

Given a scenario in robotics application, a robot is required to perform tasks based on given instructions. But in this case, the instructions are in a graphical form and no human intervention, where the tasks is represented by a table with different image sets in different cells inside the table. And the combination and permutations of the images inside the table varies from time to time. The robot is required to read it by capturing an image of the table, identify and process the image and then proceed with the instructions.

The objective of this project is to create an algorithm that can process the image data and turn it into an array form of data that can be read by computer/robot/system, while outputting the data into a matrix form for humans to read and evaluate. This project relates to machine vision, so OpenCV library is, while not directly, involved.

This project is in C++ language, it uses all basic functions of C++, and contains 30++ variables for processing the data while not including global variables. Throughout the project some common algorithms are used such as the Bubble-Sort, while the rest of the algorithms are written specifically to achieve the goals of the project. This project involve in analyzing the coordinates of the data inside the table, sorting the coordinates, copying the coordinate data into another variable for processing, appending the temporary data into the main data, and outputting processed data and final data both in terminal and in text file.

The project has proven to be a success and can convert a tabular image with sets of data images into readable computer data, but it is not thoroughly optimized, and a lot of sections are repetitive. To make the code more efficient and flexible, code maintenance and more time is required.

1. Introduction

1.1 Problem Statement

Given an image of a table containing different images within the cells:

Order Board

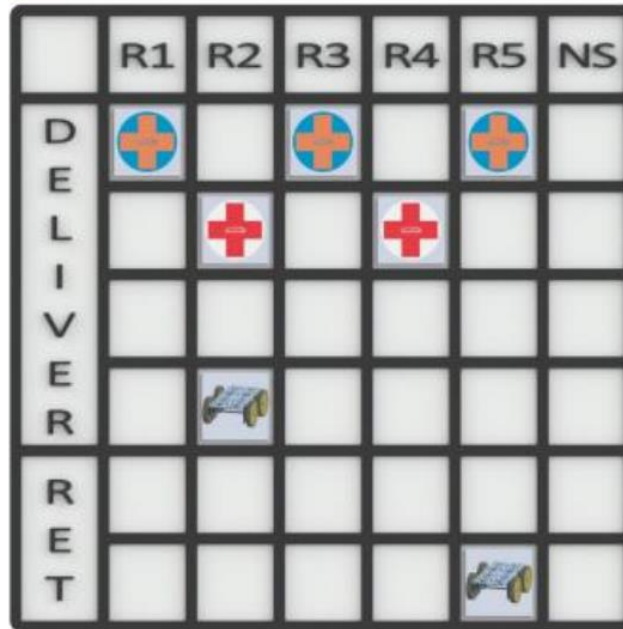


Figure 1.1-0-1 Tabular Image

certain images in certain cells means specific instructions for the robot to perform task. The arrangement/combination/permutation of the images may vary.

1.2 Objectives

The objective of the project is to generate a solution where the code can interpret the table and convert from image data into numeric data in the form of arrays and matrix. The project involves image processing, which uses OpenCV library. Since that a program that uses OpenCV to process the image and get the coordinates of the data existed, but the code is in python language, and the only output can be used are the raw unsorted coordinates generated from the program, so the goal of this project is to write the processing algorithm to process the raw unsorted data in C++, hence the indirect involvement of OpenCV.

1.3 Scope

This project's current scope is strictly limited to:

- Processing coordinate data generated from OpenCV library
- Able to only accept raw data in specific format
- Currently able to process a table format of 700x700px of table image, 600x600px of area where data exists.
- Results (matrix) are not to be used as calculations

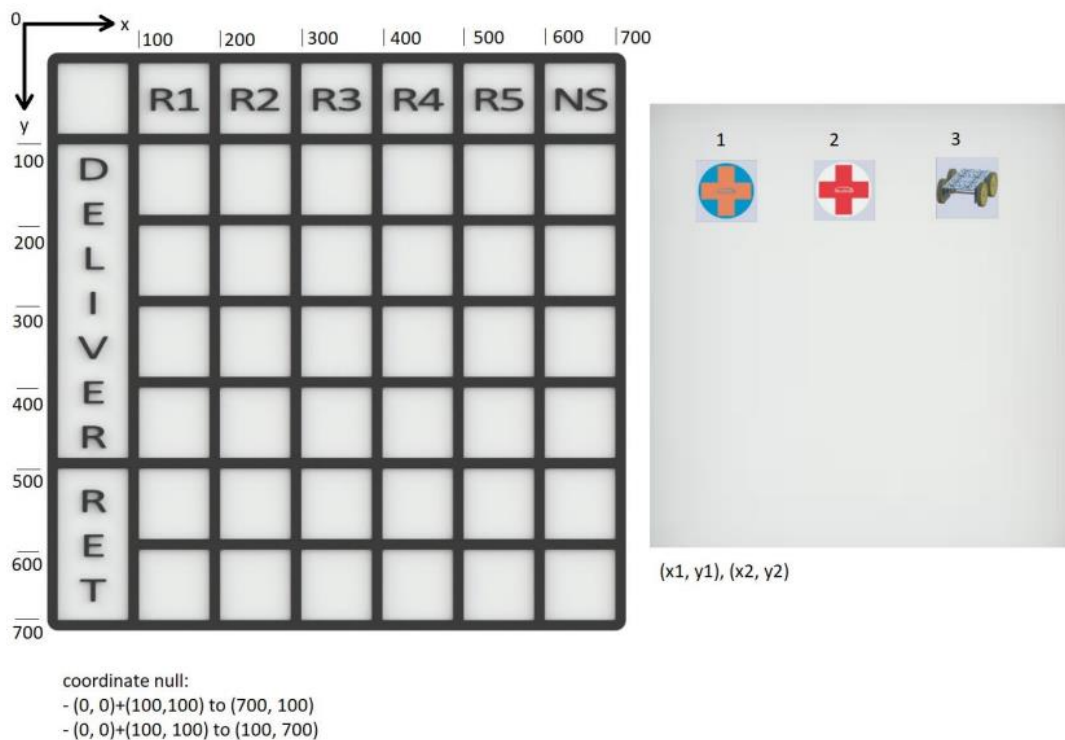


Figure 1.3-1 Table format 700x700 px

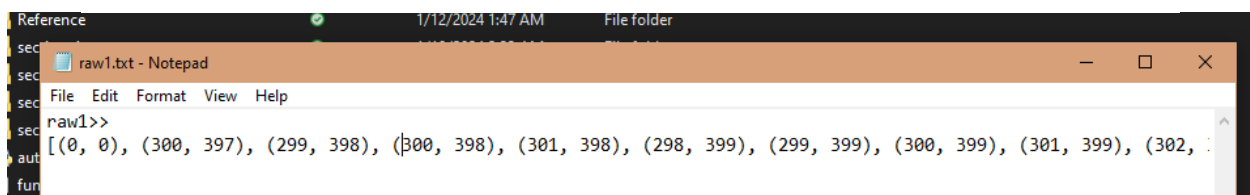


Figure 0-1.3-2 Specific raw data format

Users can also specify the location of the raw data to be processed, or choose local folder where the data is in the same folder as the program.

1.4 Requirements & Specifications

Project Specifications

- i. Version
v4.52 Vectorstruct
- ii. Functional Requirements:
 - Data converter
 - Converts tabular image data into readable numeric matrix data
 - Location of raw data (local or specified)
 - 14 Functions
- iii. User Interface (UI):
 - Terminal control/prompt
- iv. Data Requirements:
 - Raw input data raw1.txt, raw2.txt and raw3.txt
 - Raw data contains x-coordinates and y-coordinates
 - Raw data contains a null coordinate "(0, 0)" at start of file
 - Output data in the form of .txt (report.txt) in local folder (same folder as the code)
- v. System Architecture:
 - C++ environment
 - Any operating system
- vi. Testing Requirements:
 - Using data manually retrieved from main.py
 - Using data generated by automatic data generator *autoGenerator.py*

vii. Constraints & Assumptions:

- Specific raw data format:

"raw1>> [(x₁, y₁), (x₂, y₂), (x₃, y₃), (x_{n-1}, y_{n-1}), (x_n, y_n)] "

Starts with "raw(1/2/3)>>" , data contents surrounded by brackets.

- Any amount of data more than three (3) types may result in unexpected behavior
- Raw data coordinates are generated from OpenCV in Python language
- Table image processed by OpenCV is in the size of 700x700 px
- Data image contained inside the table image is in the size of 90x90 px
- Table image contains 6 rows and 6 columns where the data images will exist. (6x6 matrix)
- The result matrix is not to be used as calculations but for visualization and data locating purposes only

2. Methodology

2.1.0 – Intro

Due to time constraints and lack of knowledge in OpenCV among team members, the input data that we will be using as raw data will be generated through an existing program that uses OpenCV in Python language to process the image data. After the data is obtained, it will be then required to be packaged into a .txt file with a fixed format.

2.1.1 – [Section 1 – INPUT]

The data at this point will be our raw data to feed into our C++ program. The raw data should contain at least one (1) coordinate in the form of (x, y), where x and y are integers. The data naturally are scattered, and the goal of our C++ code is to sort and convert the scattered data coordinates.

The program will first prompt the user to input the location of the raw data to be processed. E.g.: “C\Users\user\Desktop\USM\ProgramData”, or the user can enter “Local” to search and use the raw data in the same folder as the program is running. If there are no raw data to be found, the program will not proceed and allows the user to reenter until the user enters a valid location where raw data can be found.

Continuing, the program reads the raw data file and calculates the size of the data (number of contiguous numbers present in the file), we will call the size of data as index of data. Since that the coordinates is in the form of (x, y), the index of the data will be:

$$dataIndex = \frac{\text{Number of contiguous numbers, } N}{2}$$

The dataIndex is stored and used to initialize a temporary raw data variable,

```
int raw_data[dataIndex][2];
```

Where the first container represents the number of coordinate sets, and the second container represents the x and y data. The reason for initializing the data later is due to the number of data will vary, the size of the variable must be accurate, no more, no less, otherwise will result in error later in the process.

After declaring the `raw_data`, the program will extract the contents of the file and write it into the `raw_data` variable. We also wrote the code so that it will display the size of the index. After extraction, we will repeat the variable initialization again:

```
int data[dataIndex][2].
```

This variable is used for processing and allows the `raw_data` variable to remain untouched. After initializing, the data are once again copied from `raw_data[][]` to `data[][]`. When copying the data, the null data (0, 0) will present at the beginning of the data, and will be filtered by skipping copying the first data, and then the size of index of the data `dataIndex` will be decreased by 1.

2.1.2 – [Section 2 – SORT]

After obtaining the data, we will then sort the data by the x-coordinates (first container of the data) in ascending order using bubble-sort. After sorting, the program will display the sorted data array for debugging and validation purposes.

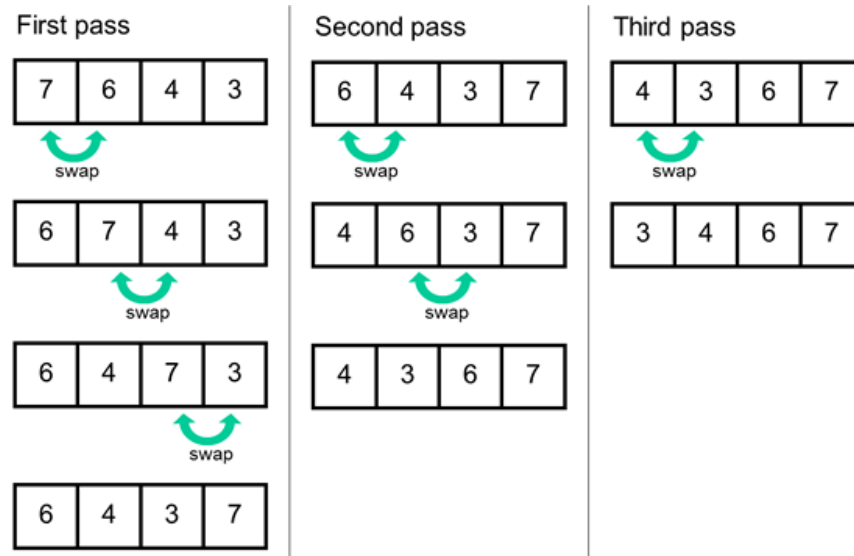


Figure 2.1-1 - Bubble Sort

2.1.3 – [Section 3 – Convert]

The nature of the data is that it will have repetitive data coordinates from the raw data, so now we will proceed by counter. The program will count the presence of data in each respective column in the table. The counter counts how many coordinates in the data exists in that particular column. The purpose of the counter is to visualize the data points in the table and most importantly, to calculate the size of the index for row data. First, we have a variable to store the amount of existence of data in different columns:

$$\text{int } xcoord[6] = \{0, 0, 0, 0, 0, 0\};$$

where the first index of $xcoord[]$ represents the first column, and subsequently. To increase the data point in the variable, a discrete number is required for finding the index of the variable to store the data in. The mathematical model for determining the y-location (row) of the x-location (column) of the data as the index is:

$$\text{int } index = \text{int}(\text{round}(\frac{\text{double}(data[])[0])}{100.0}) - 1)$$

Given an example, a data set of $(x, y) = (297, 499)$. We are certain that the data is located at cell $(300, 500)$, but how do we obtain the digit number 3 so that we can directly use it as indexing in for-loops? By first casting the integer into double, divide by 100 should result in 2.97. Then we round it up, so it becomes 3.00. Since we consider $x=300$ as the third column, and indexing works by counting from 0, we subtract the number by 1. Lastly to use it as an index for the data, we will recast the number again back into an integer. Then, we can do increment to increase the amount of that data exists in that particular column:

$$xcoord[index]++;$$

After obtaining the existence points of the data, we will create a series of row data variables:

$$\begin{aligned} &\text{int } x100[f_1]; \\ &\text{int } x200[f_2]; \\ &\dots \\ &\text{int } x500[f_5]; \\ &\text{int } x600[f_6]; \end{aligned}$$

Where f_n are the index sizes of the respective rows. A diagram shown for quick reference below:

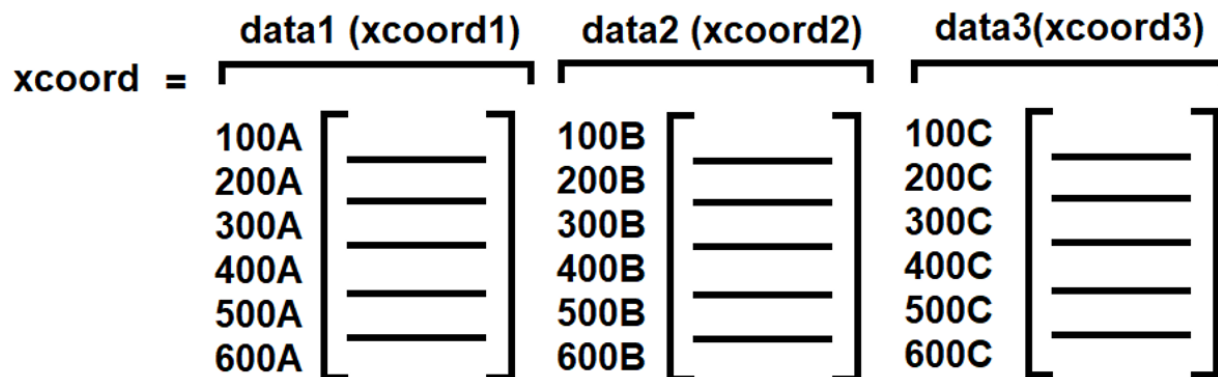


Figure 2.1-2 Representation of Multidimensional data container

Where x100 represents the columns coordinates, x100 for x=100, x200 for x=200, x300 for x=300, and so on. But declaring the variables one by one is not efficient. So we will declare the variable all at once in vector form:

```
vector<vector<vector<int>>> xcoord(3, vector<vector<int>>(6, vector<int>(1)));
```

The reason for not using a regular 3D array is due to the raw data varies and the index size of the data variables are dynamic, when using a regular 3D array we must have a predetermined size, after initializing, the size of array cannot be changed, otherwise will result in redefining variable error. As mentioned before, the size of the variable must be accurate, so by using a vector data, the variable can be more flexible, and the size of the container can be changed easily.

initializing 3D array using vector

```
vector<vector<vector<int>>> xcoord(3, vector<vector<int>>(6, vector<int>(1)));
```

equivalent to:

```
int xcoord[3][6][n];
```

where n is constant, in vector n can be resized by `xcoord[x][y].resize();`

After this stage, we have narrowed down the data by pinpointing the absolute x-coordinates for the data. Proceeding, the y-coordinates are then copied from `data[i][j]` into the multidimensional vector array `xcoord[i][j][k]` where i is the type of data, j is the column and k being the container to contain a list of y-coordinates.

2.1.4- Continued [Section 2 – SORT]

After segregating all the y-coordinate data into their respective x-coordinate containers, we will once again bubble-sort the data in ascending order.

2.1.5- Continued [Section 3 – CONVERT]

After sorting the y-coordinates, the existence of data in the form of y-coordinates is then again counted. This time, the target location for the data is the data matrix. A for-loop algorithm will count and append the amount of existence of that particular data in that particular cell in the table. For determining the absolute y-location for indexing and store it into the data matrix, we use the same formula as in 2.1.3 – [Section 3 – Convert]:

```
int indx = int((round(static_cast<double>(arr[i])/100.0)-1));  
if (indx>=0&&indx<6) { // Check if indx is within bounds  
    data[indx][xloc]++;}
```

Where xloc represents the column of the data. At this stage, the data matrix contains the number of existences of the data in each particular cell. The data (number of existence) is then converted into a discrete value. The program will first display a list of y-coordinates classified by their respective x-coordinates, and then display the appended matrix data.

Since there might be more than one data sets, all the different matrix data (data1.matrix, data2.matrix and data3.matrix) will then be added into one final matrix data and then be displayed to the user to show the calculated results.

2.1.6 – [Section 4 – OUTPUT]

The program will generate a text report containing all information such as time and date of execution, raw data fed, processed data and the results (data in matrix form).

2.2 Flow Chart

2.2.1 Project General Flowchart

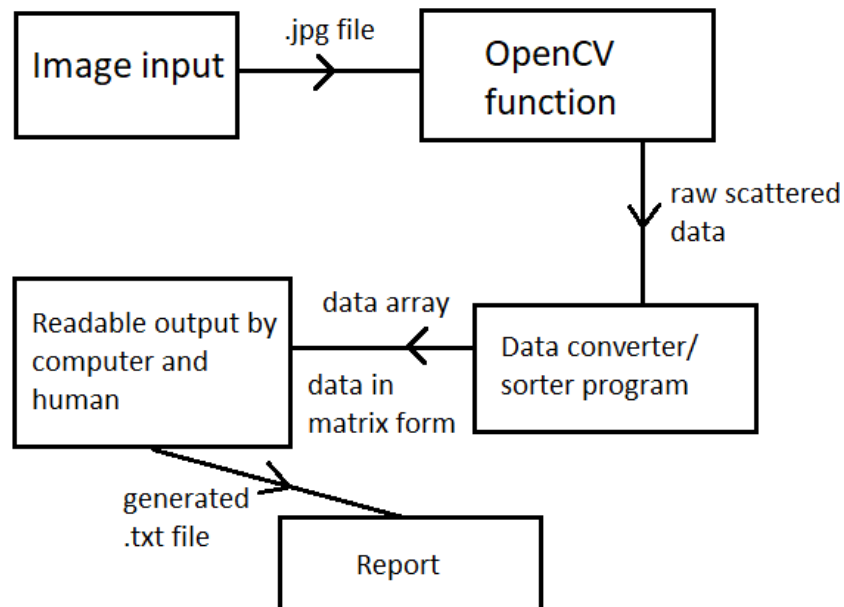


Figure 2.2-1 Flowchart 1

Since that we will not be using OpenCV function directly in C++, thus

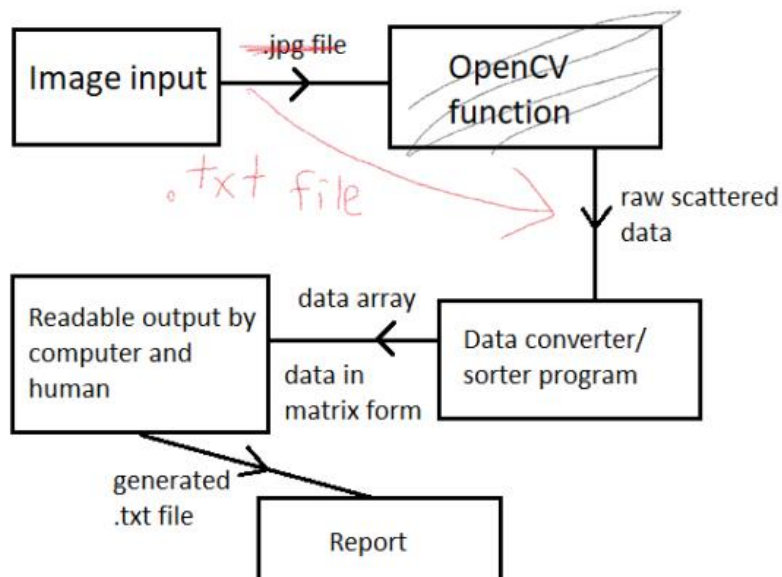


Figure 2.2-2 Actual flowchart

2.2.2 Program General Flowchart

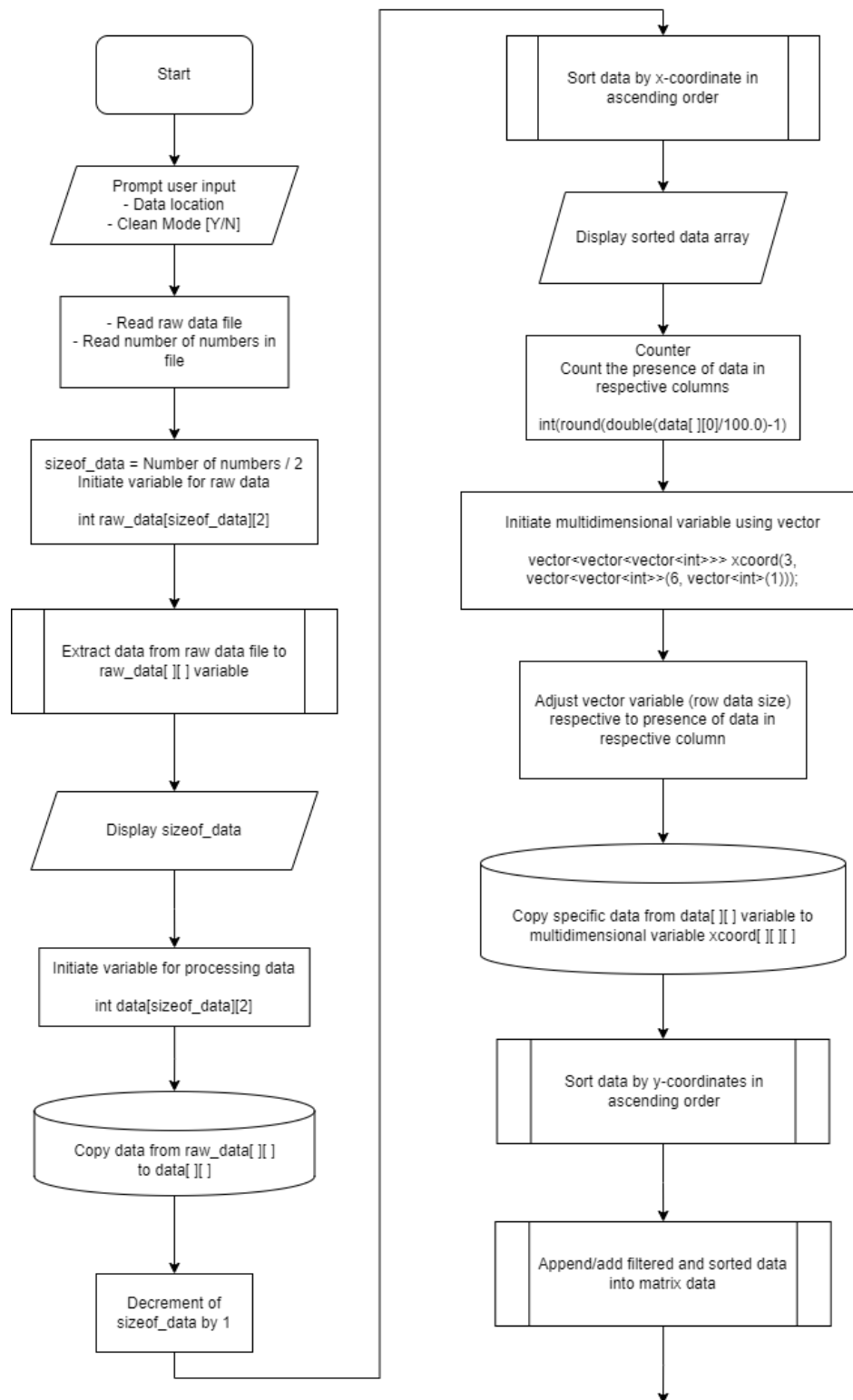


Figure 2.2-3 General flowchart

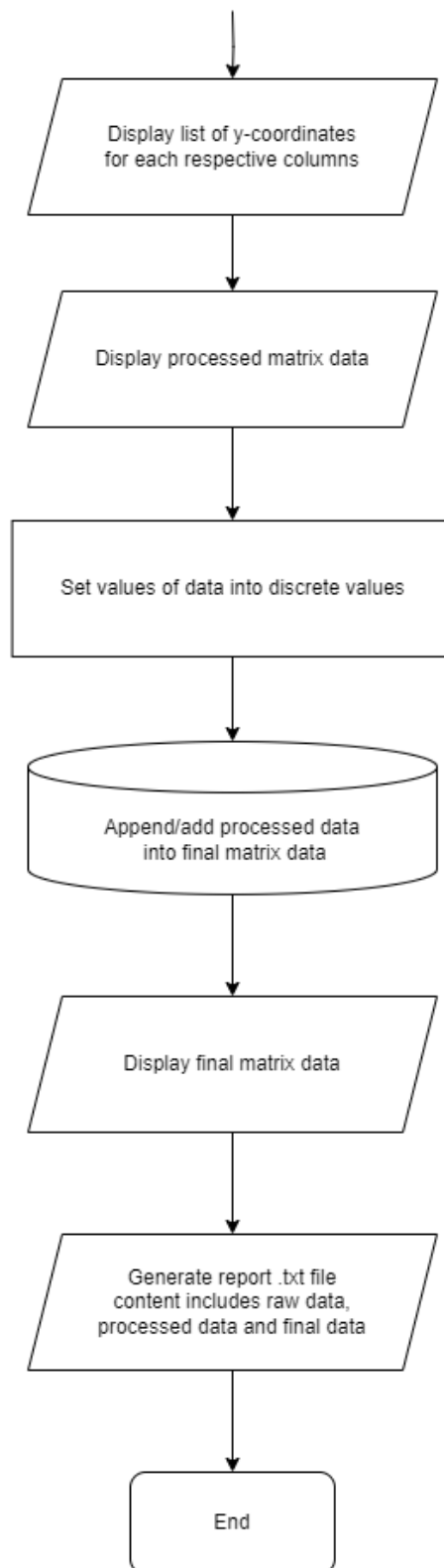


Figure 2.2-4 General Flowchart continued

2.2.3 Program Flowchart by Section

2.2.3.1 Start of code

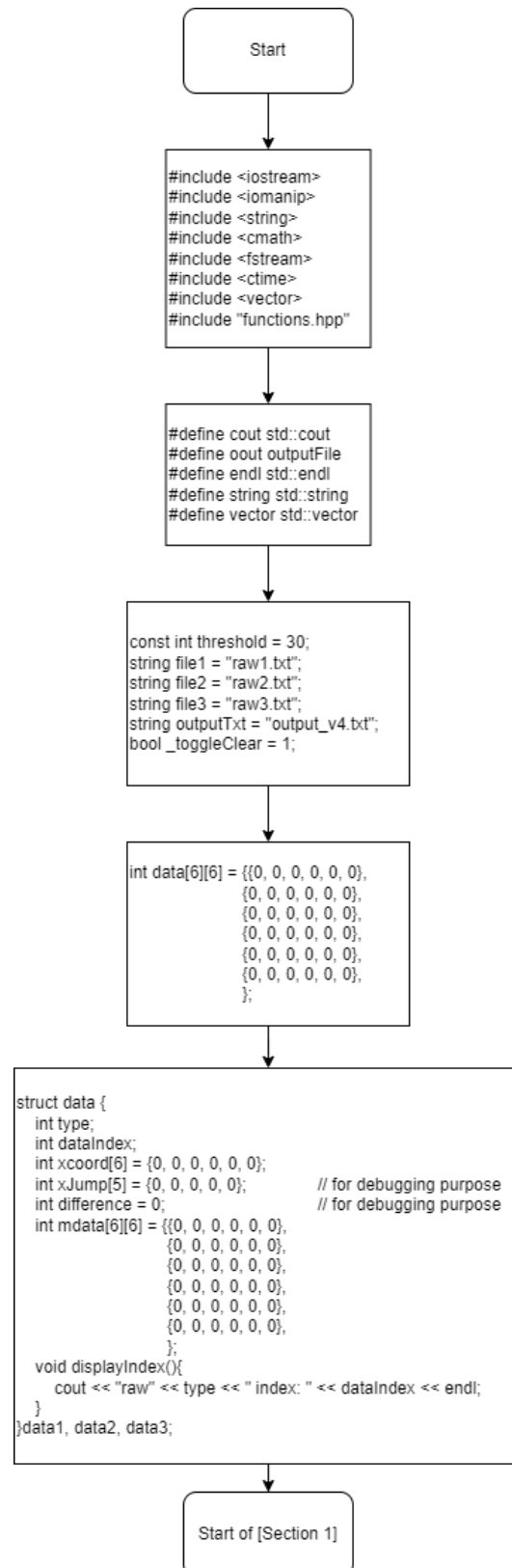


Figure 2.2.3-1 Start of code

2.2.3.2 [Section 1 – INPUT]

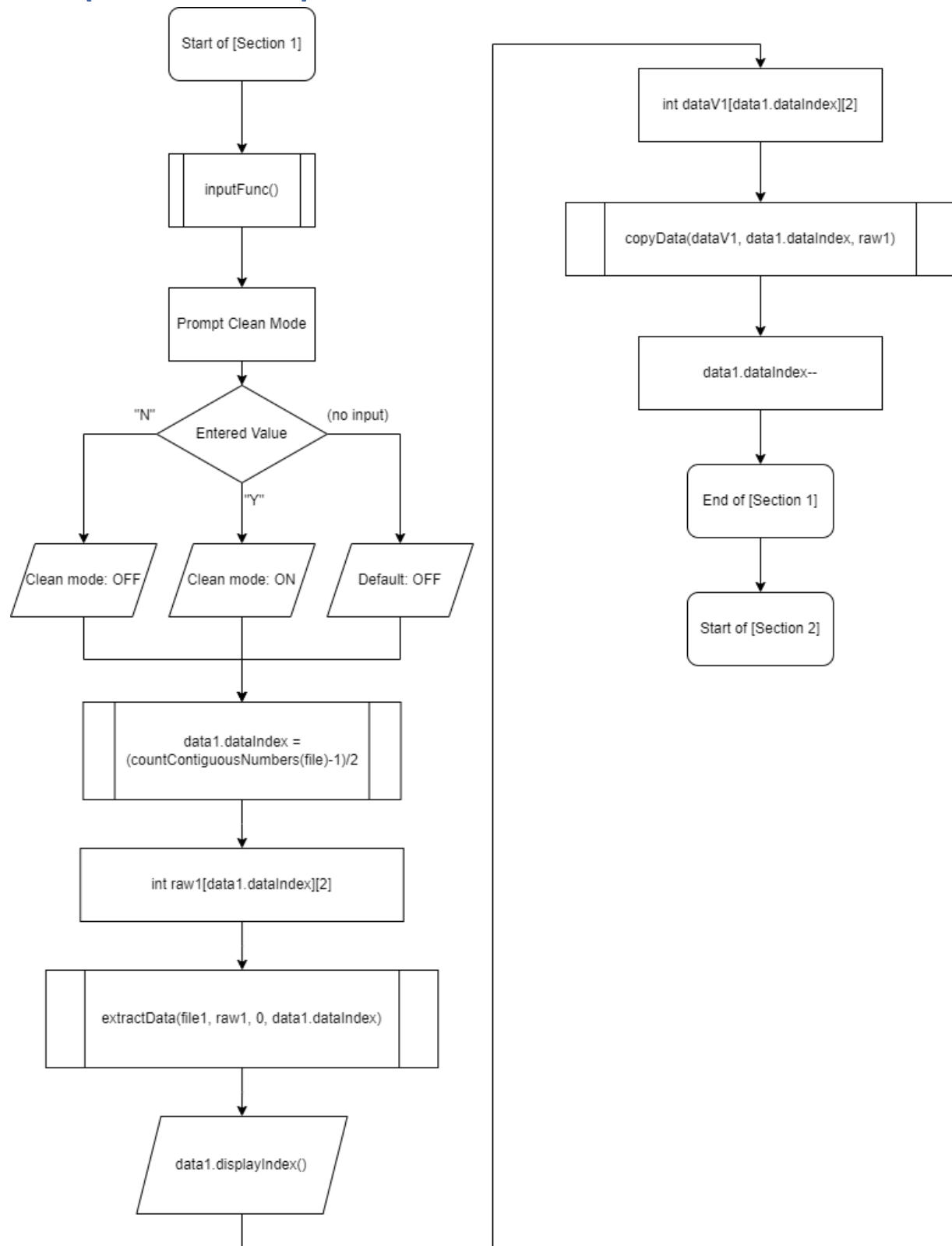


Figure 2.2.3-2 [Section 1 – INPUT]

2.2.3.3 [Section 2 – SORT]

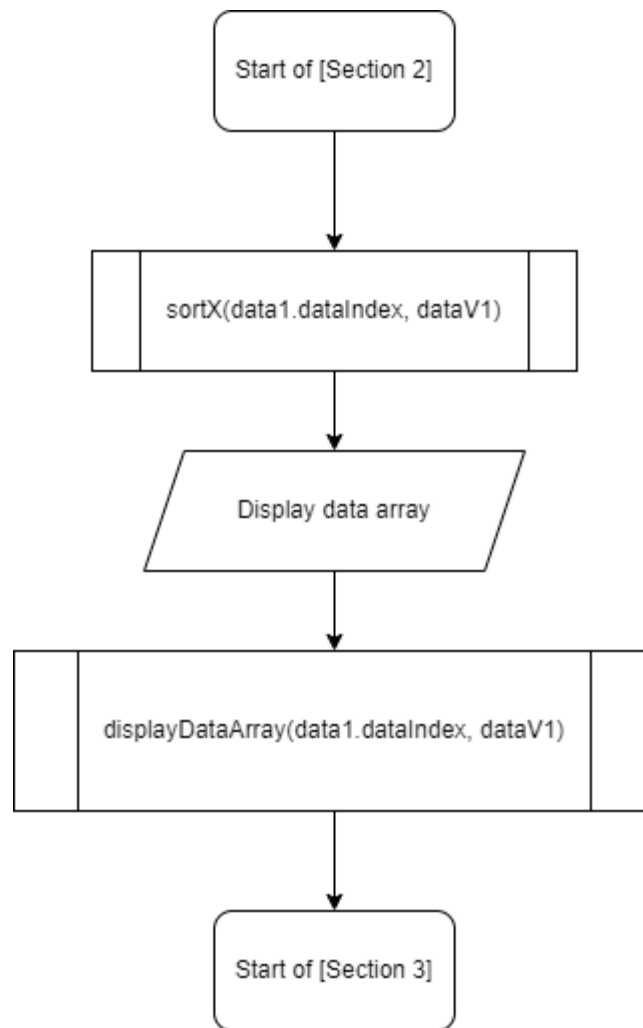


Figure 2.2.3-3 [Section 2 – SORT]

2.2.3.4 [Section 3 – CONVERT]

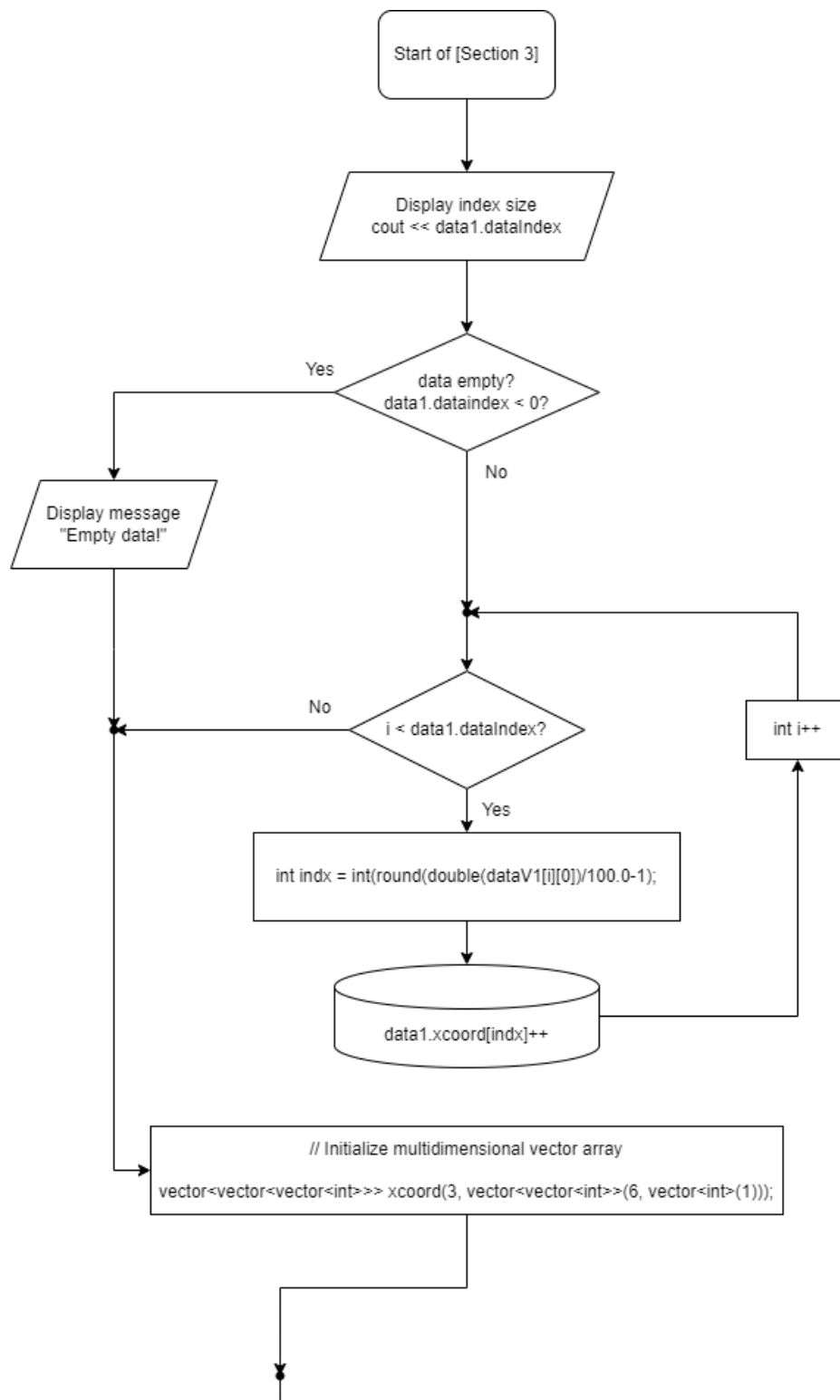


Figure 2.2.3-4 [Section 3 – CONVERT]

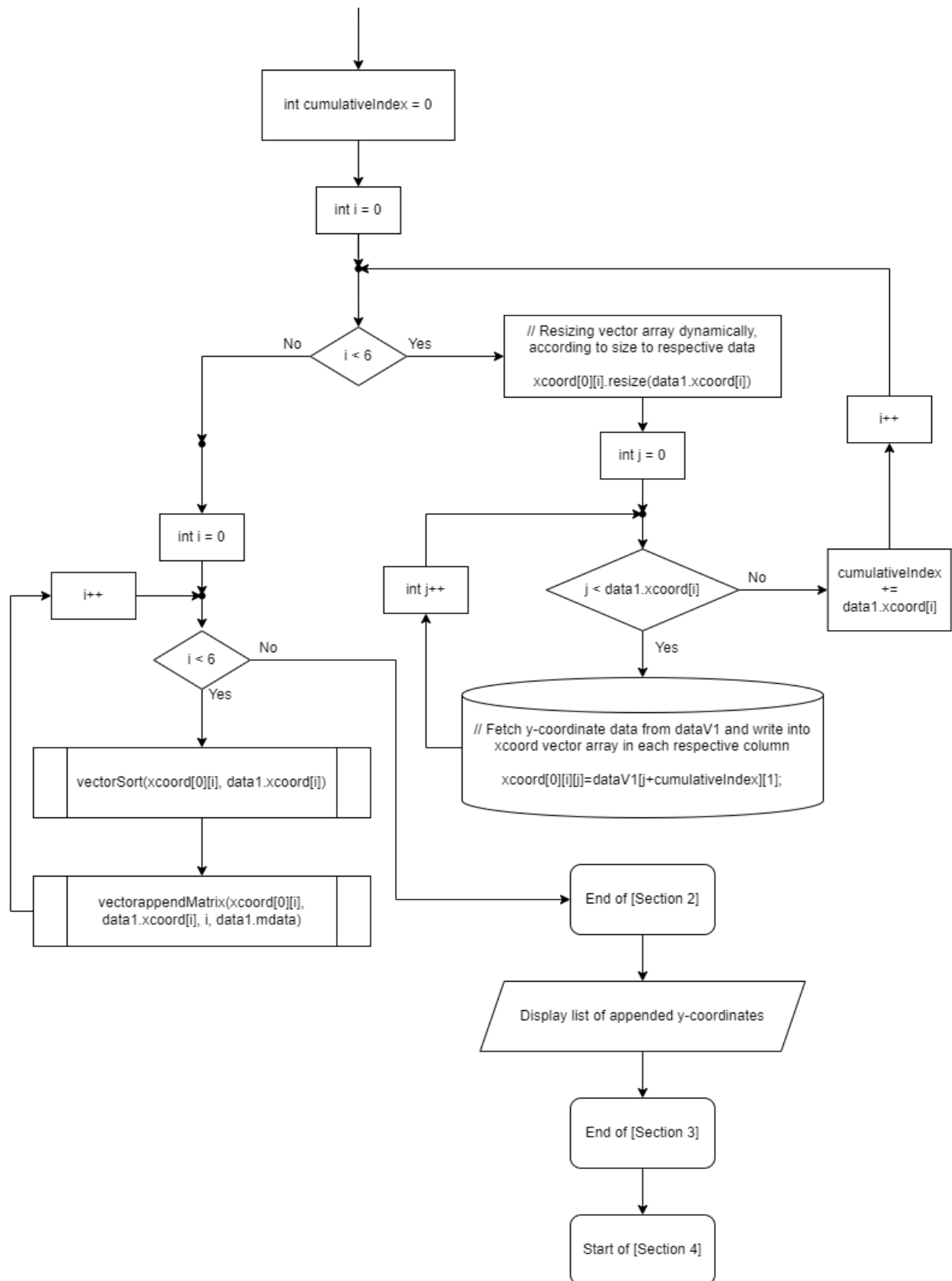


Figure 2.2.3-5 [Section 3 – CONVERT] continued

2.2.3.5 [Section 4 – OUTPUT]

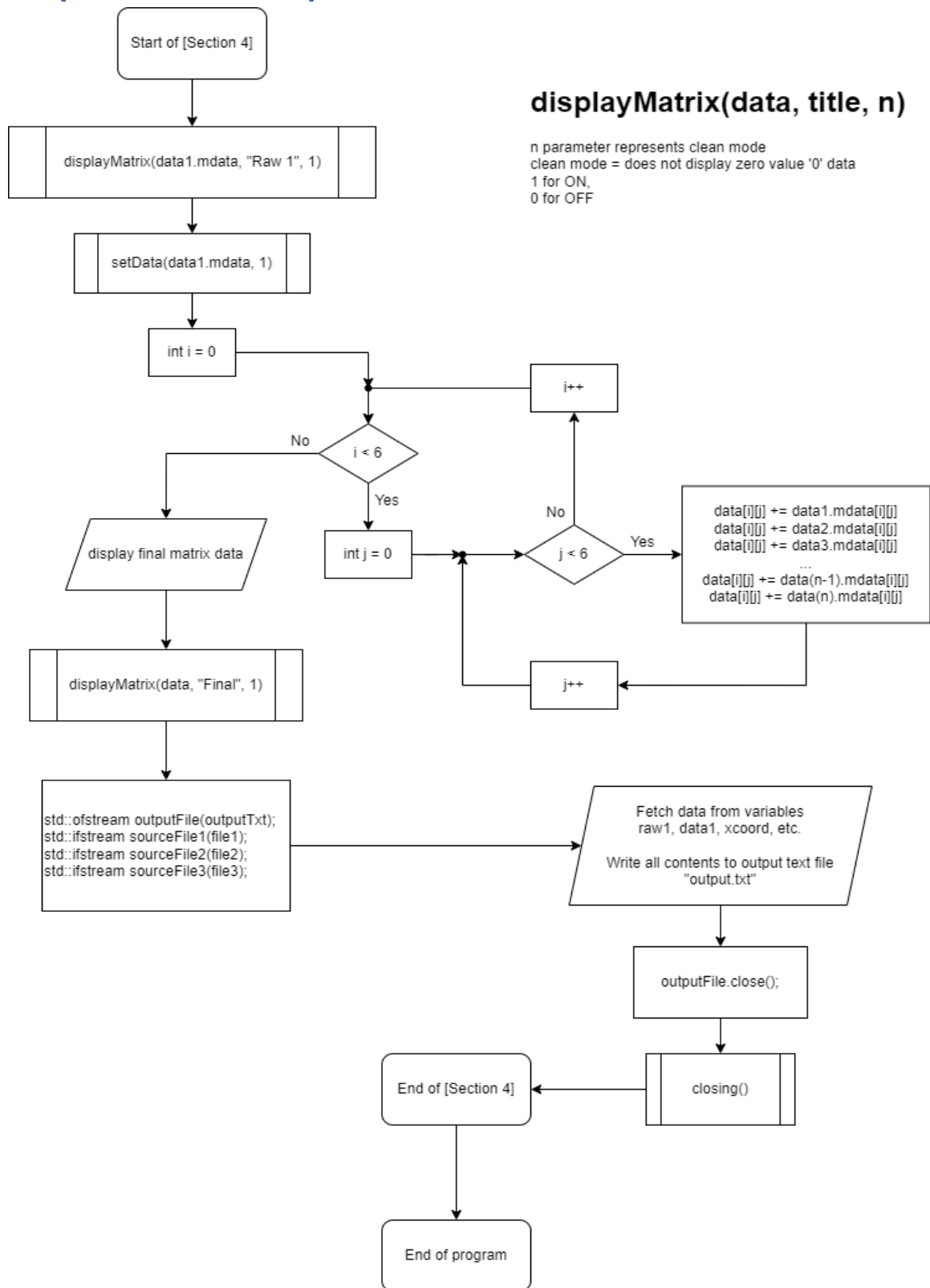


Figure 2.2.3-6 [Section 4 - OUTPUT]

2.2.4 Functions

2.2.4.1 List of functions (14)

*** indicates function is only used in the earlier versions of the code, latest version does not use (v4.52 Vectorstruct)*

- int getIndex()^{**} // fetch the index size of an array

[Section 1 – INPUT]

- void inputFunc() // service function that prompts user input
- int countContiguousNumbers() // fetch the number of contiguous numbers of the data
- void extractData() // extract raw data into local data

[Section 2- SORT]

- void copyData() // copies data from one variable to another
- void sortX() // Bubble sorting, sort 2D array, sort by x-coordinates
- void _sort()^{**} // Bubble sorting, sort 1D array, sort by y-coordinates
- void vectorSort() // Bubble sorting, sort vector multidimensional array, sort by y-coordinates

- void displayDataArray() // display data array

[Section 3 – CONVERT]

- void appendMatrix()^{**} // appends data into target 2D array
- void vectorappendMatrix() // appends vector data into target 2D array

[Section 4 – OUTPUT]

- void displayMatrix() // display array in the form of matrix
- void setData() // set data to discrete values
- void closing() // closing message

2.2.4.2 – Flowchart for functions in [Section 1 – INPUT]

- void inputFunc(string IN, string IN, string IN)

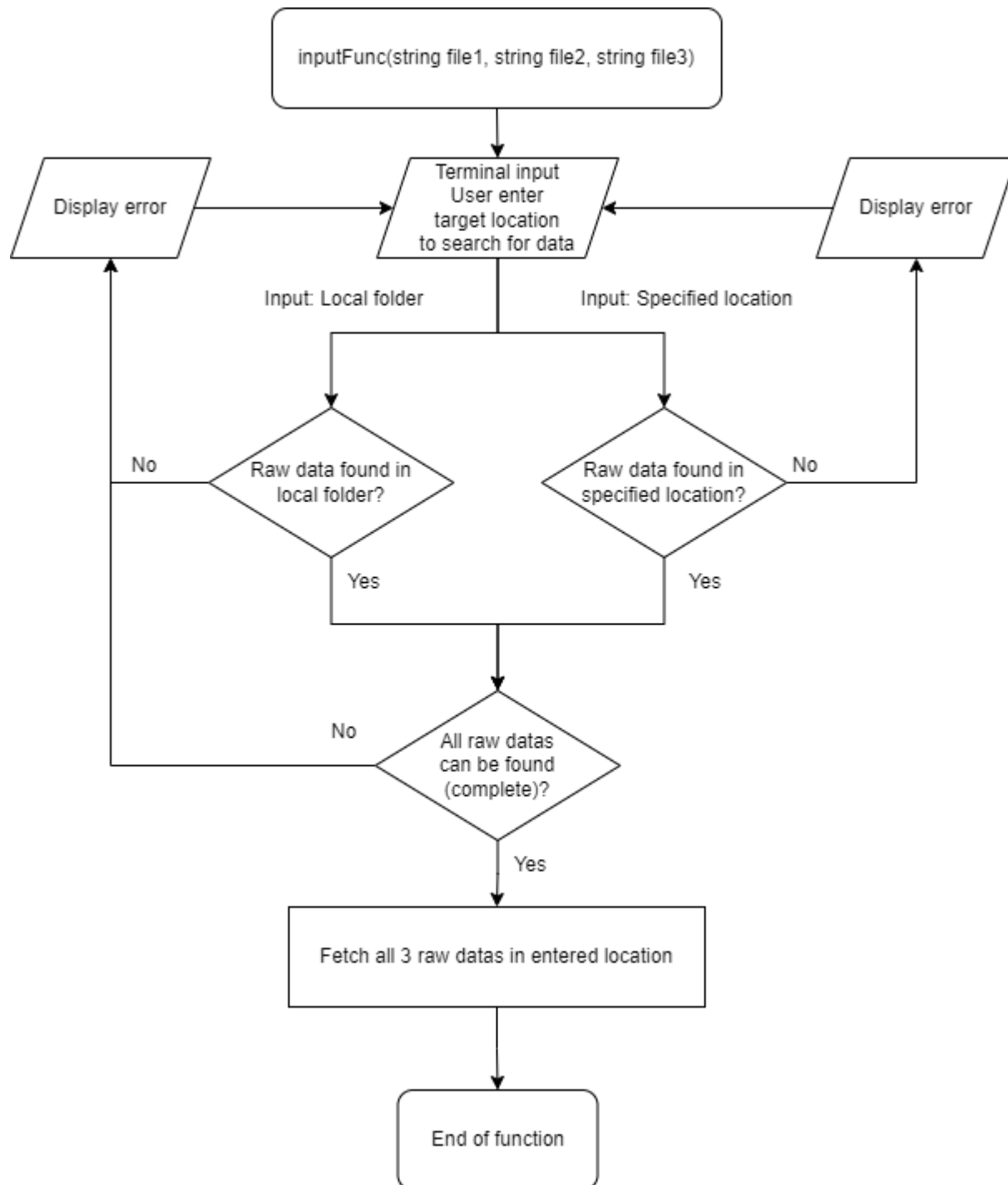


Figure 2.2.4-1 inputFunc()

- int countContiguousNumbers(string IN)

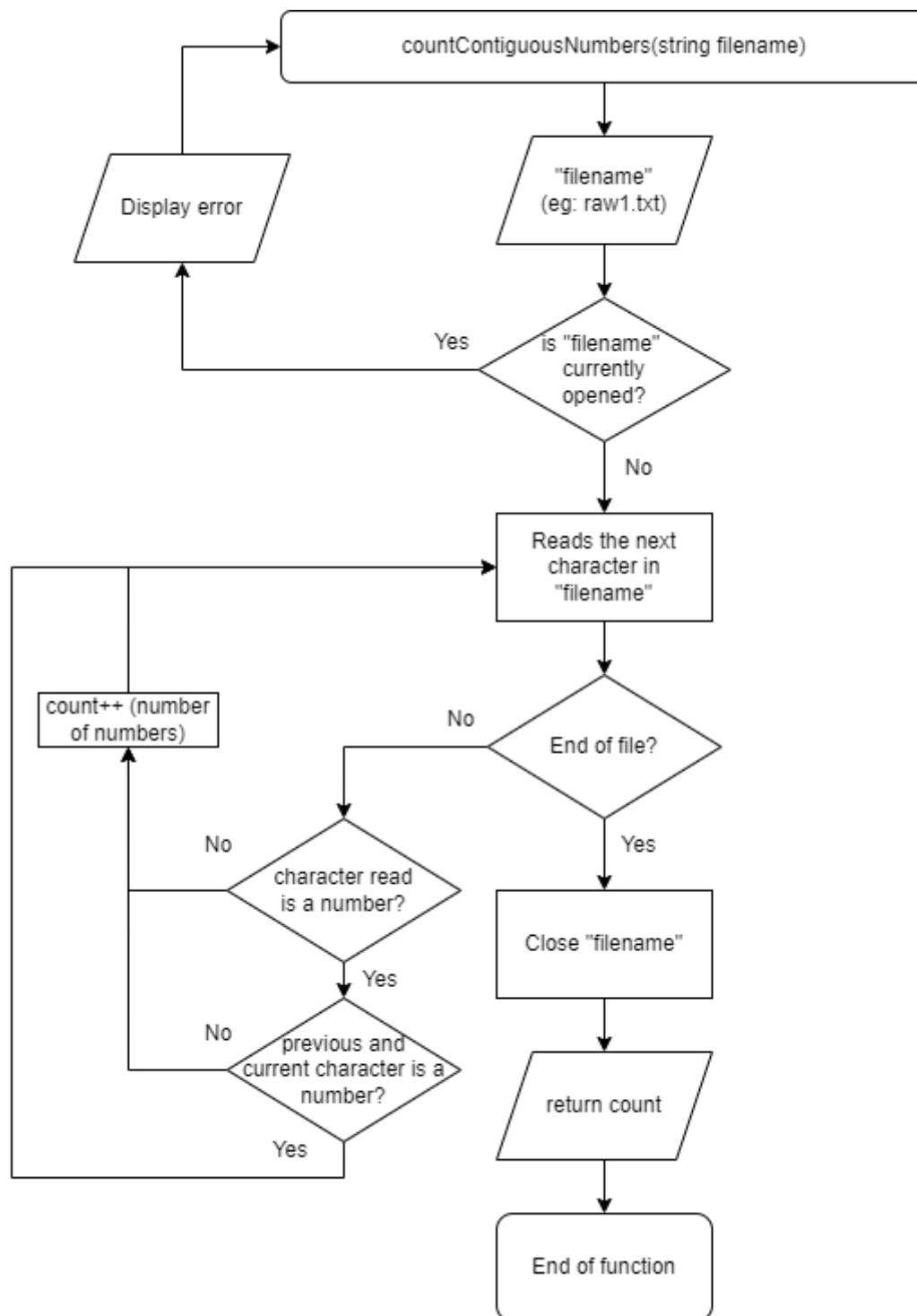


Figure 2.2.4-2 countContiguousNumbers

- void extractData(string IN, int OUT[][], int IN, int IN)

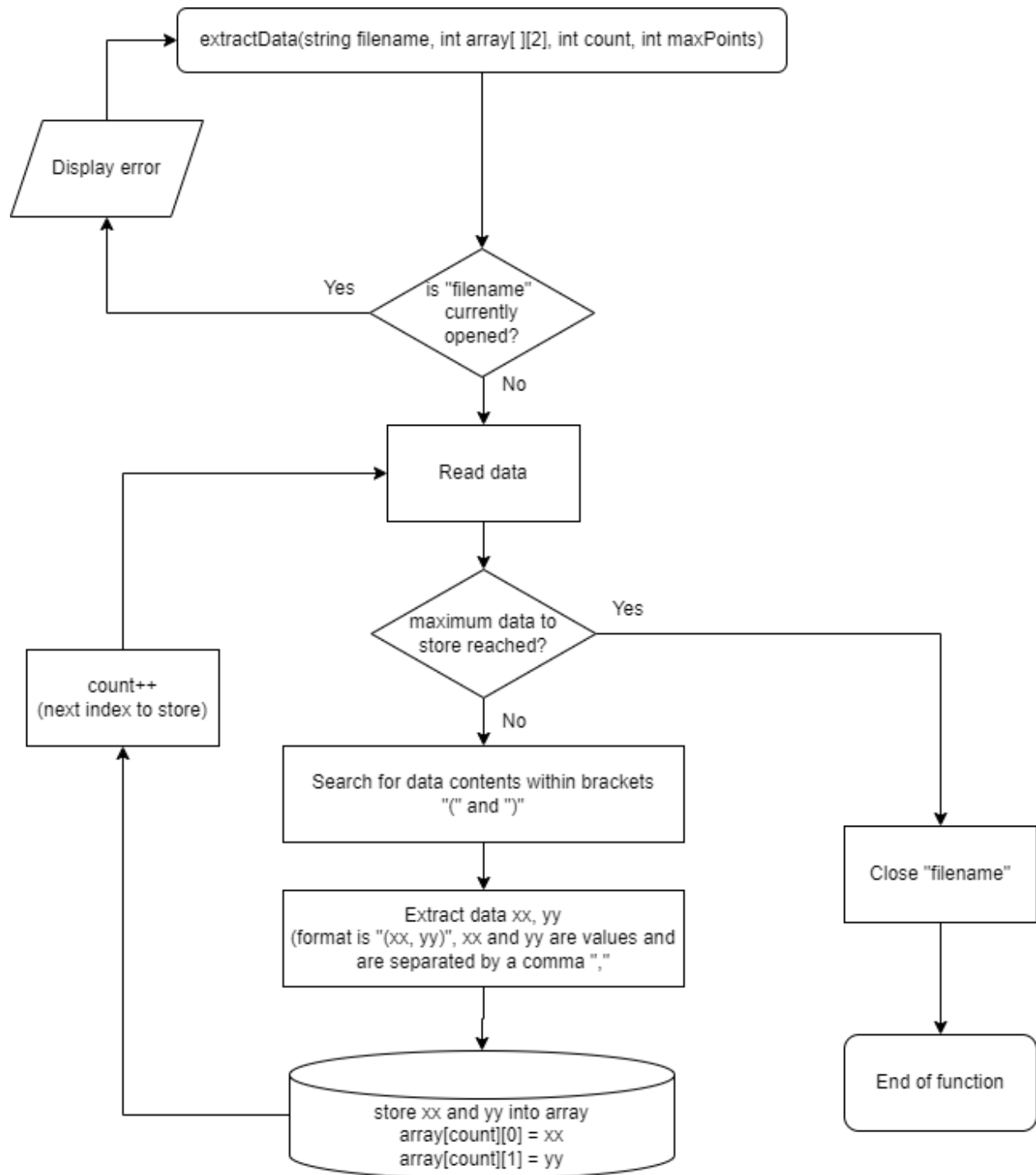


Figure 2.2.4-3 extractData()

2.2.4.3 – Flowchart for functions in [Section 2 – SORT]

- void copyData(int IN[][], int IN, int OUT[][])

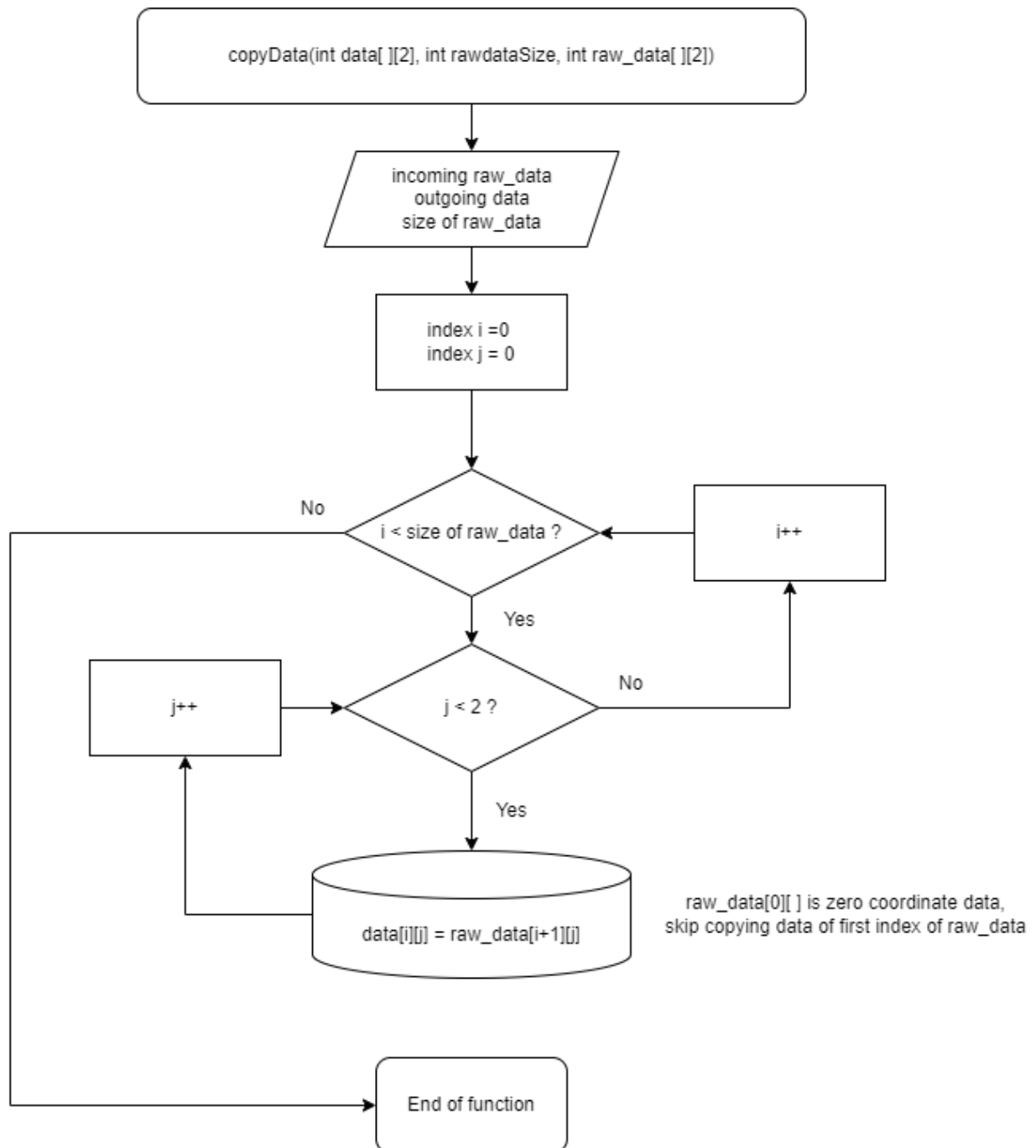


Figure 2.2.4-4 copyData()

- void sortX(int IN, int IN/OUT[][])

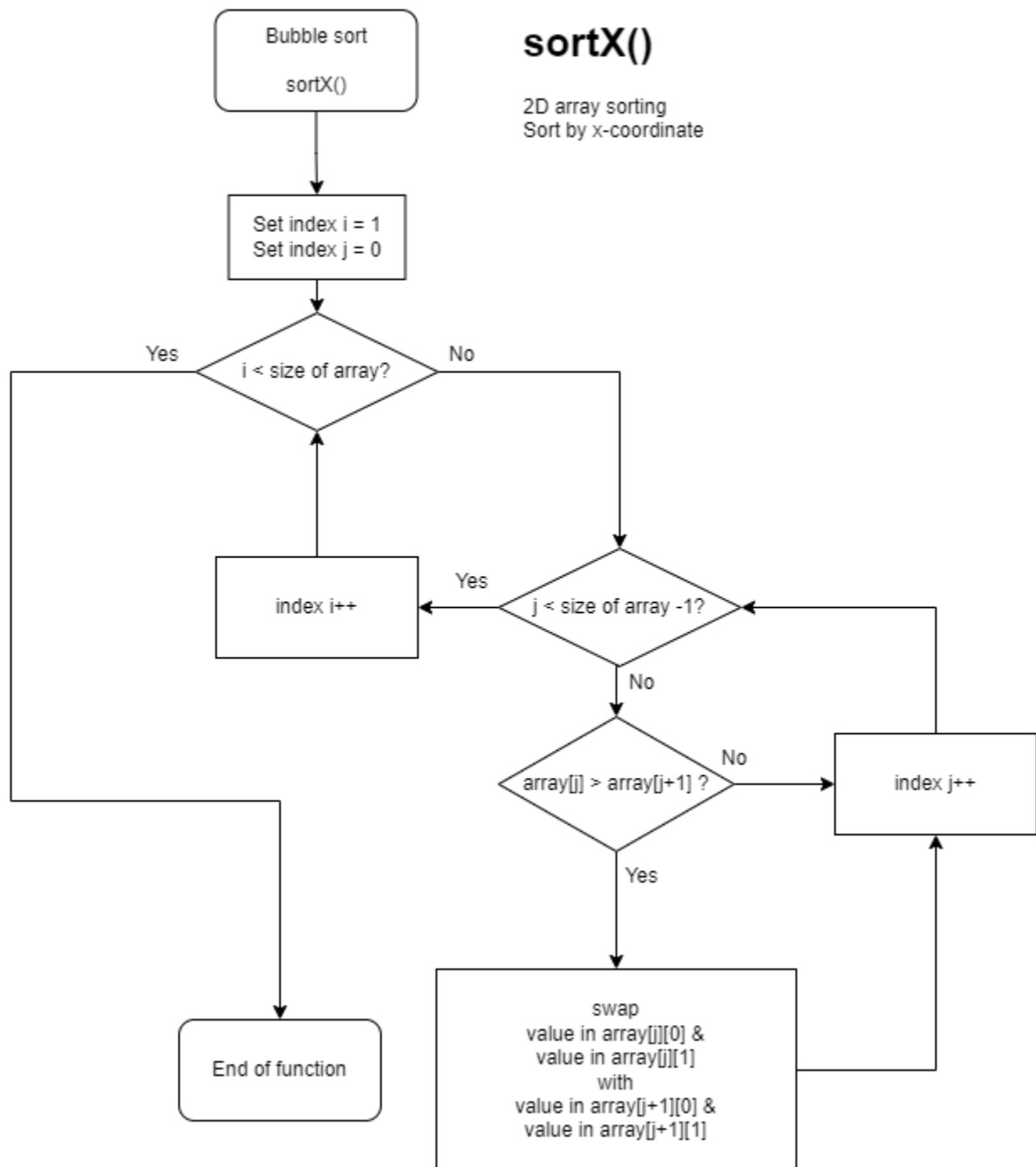


Figure 2.2.4-5 sortX()

- void _sort(int IN/OUT[][], int IN)

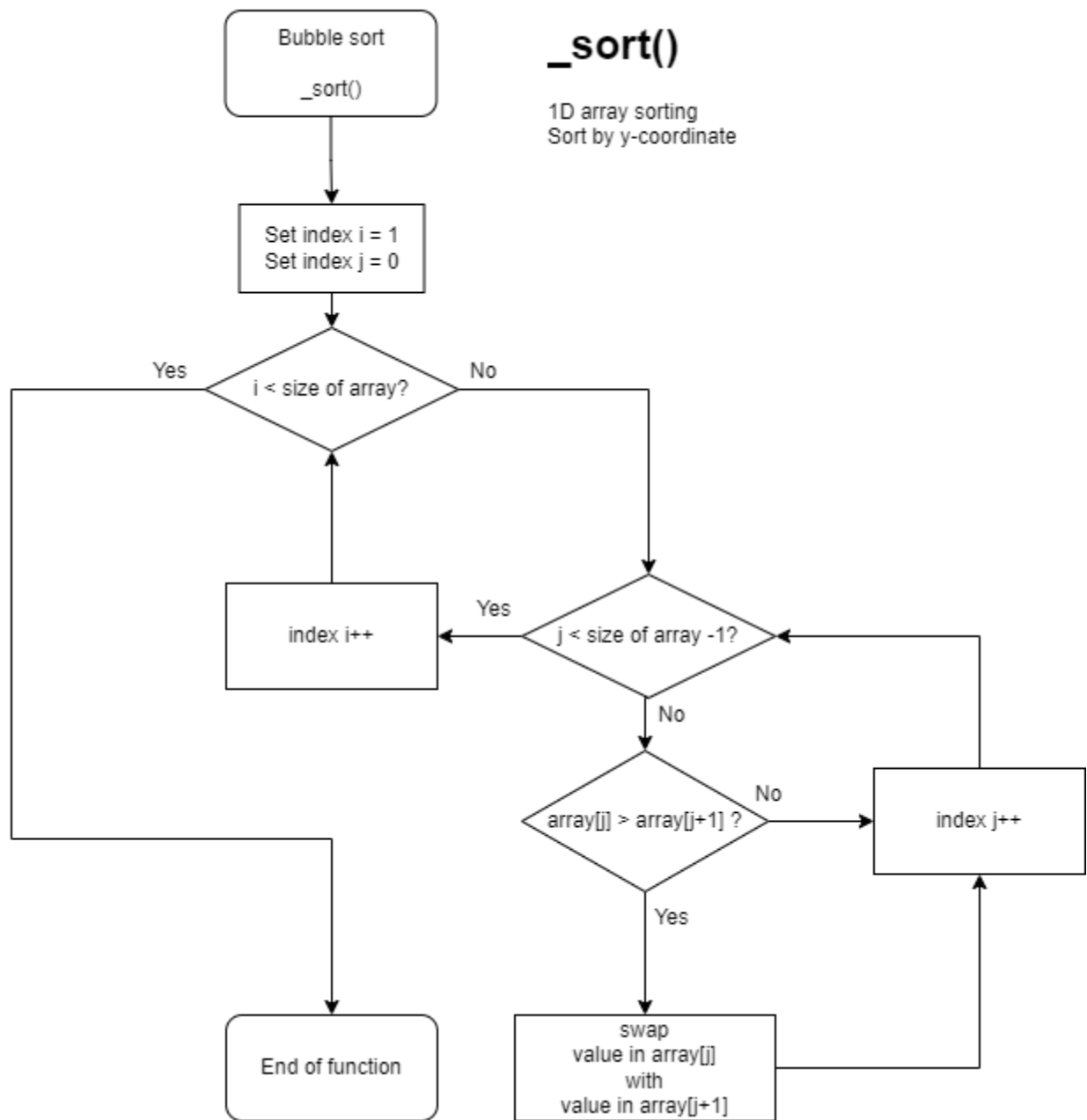


Figure 2.2.4-6 _sort()

- void vectorSort(vector<int> IN, int IN)

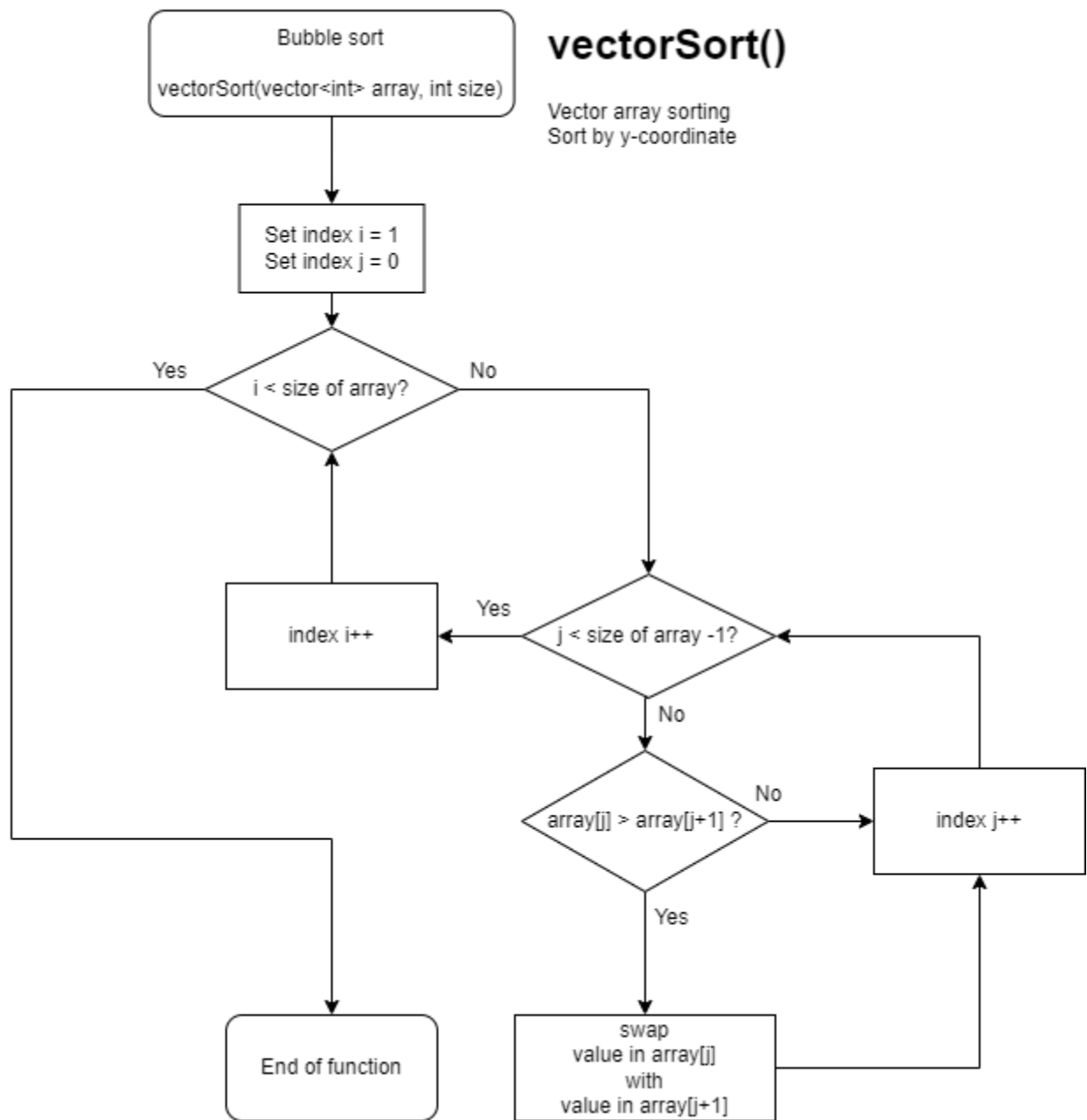


Figure 2.2.4-7 vectorSort()

- void displayDataArray(int IN, int IN[][])

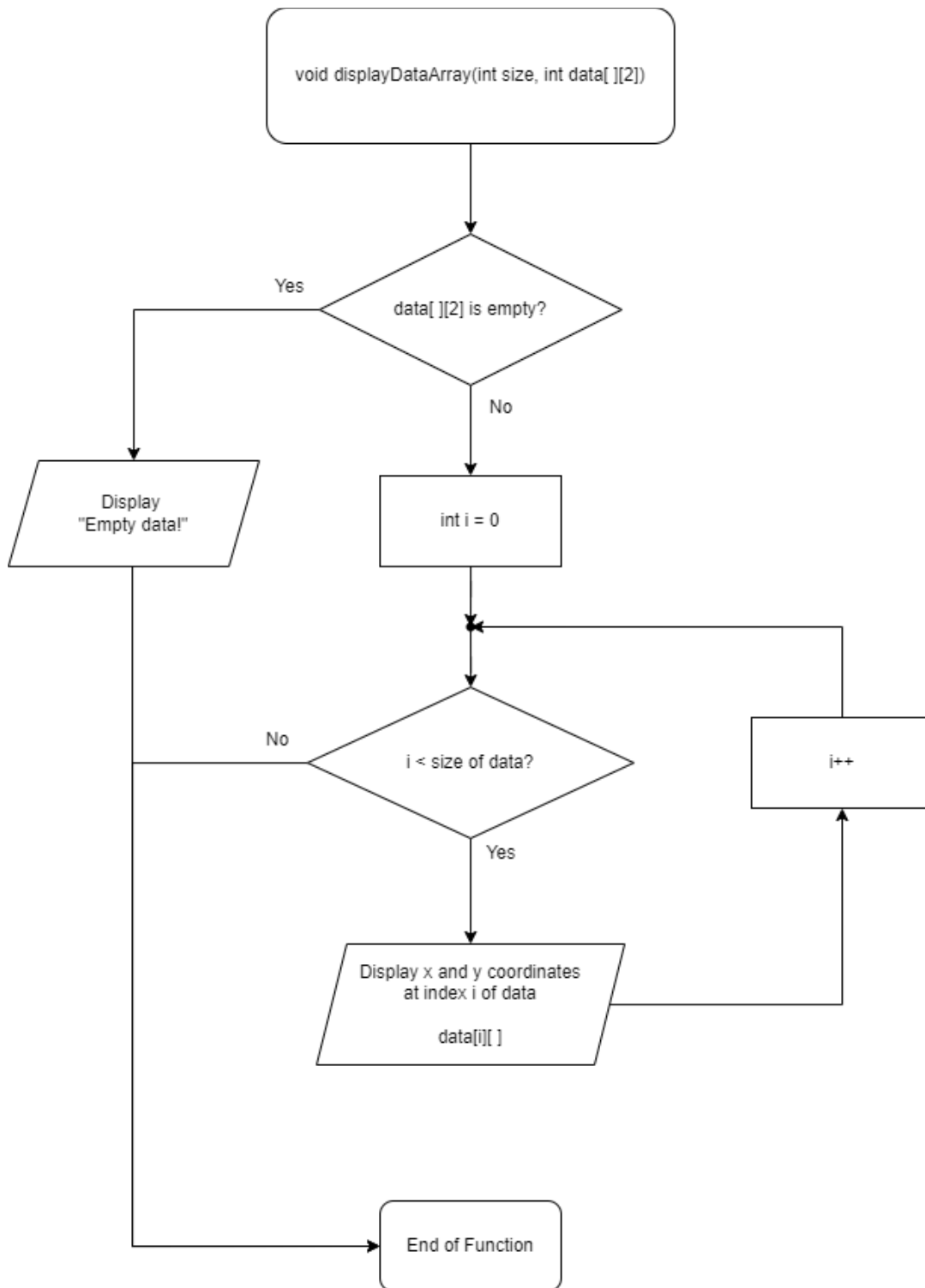


Figure 2.2.4-8 displayDataArray()

2.2.4.4 – Flowchart for functions in [Section 3 – CONVERT]

- void appendMatrix(int IN[], int IN, int IN, int OUT[6][6])
- void vectorappendMatrix(vector<int> IN, int IN, int IN, int OUT[6][6])

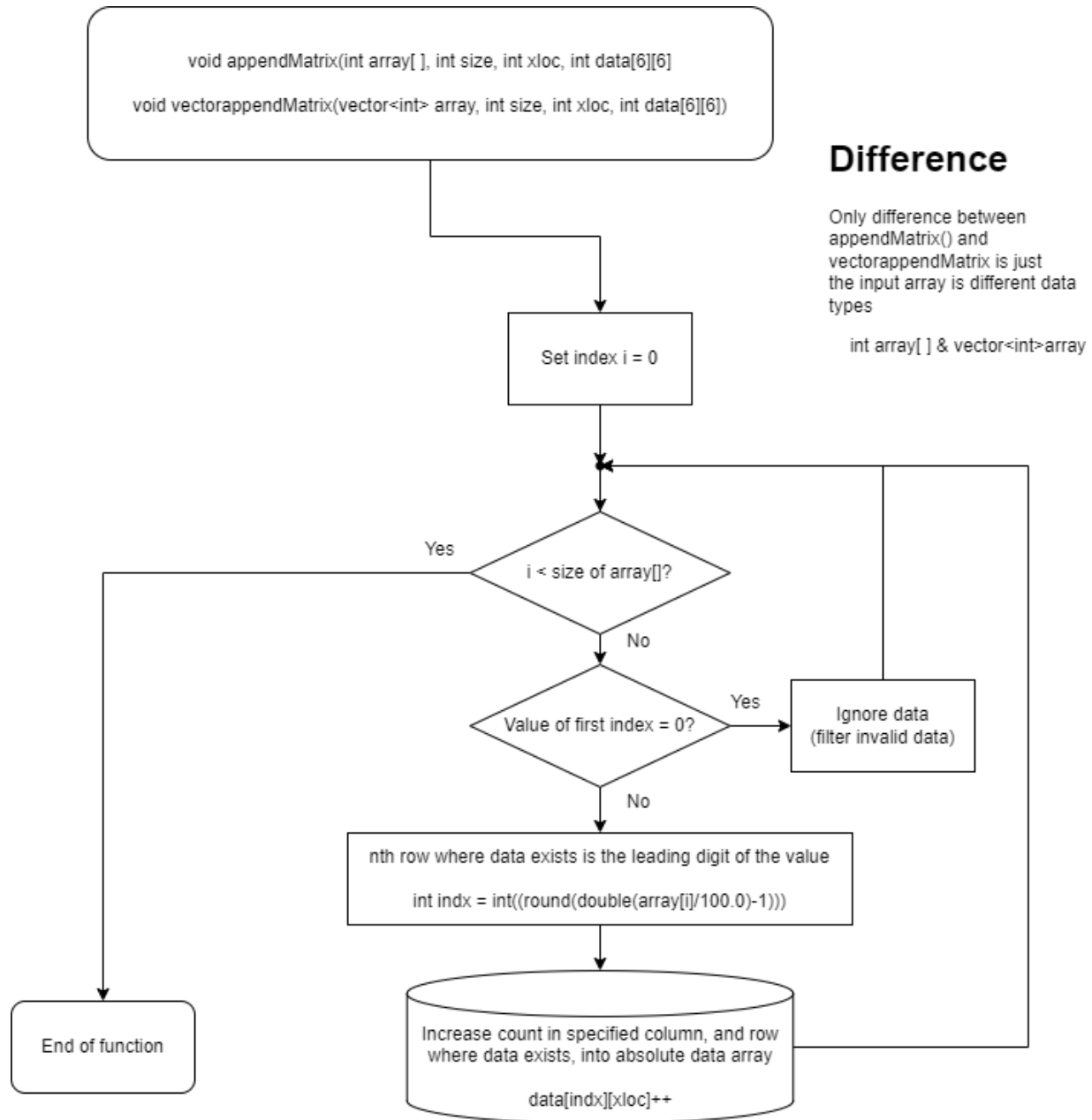


Figure 2.2.4-9 appendMatrix() and vectorappendMatrix()

2.2.4.5 – Flowchart for functions in [Section 4 – OUTPUT]

- void displayMatrix(int IN[6][6], string IN, bool IN)

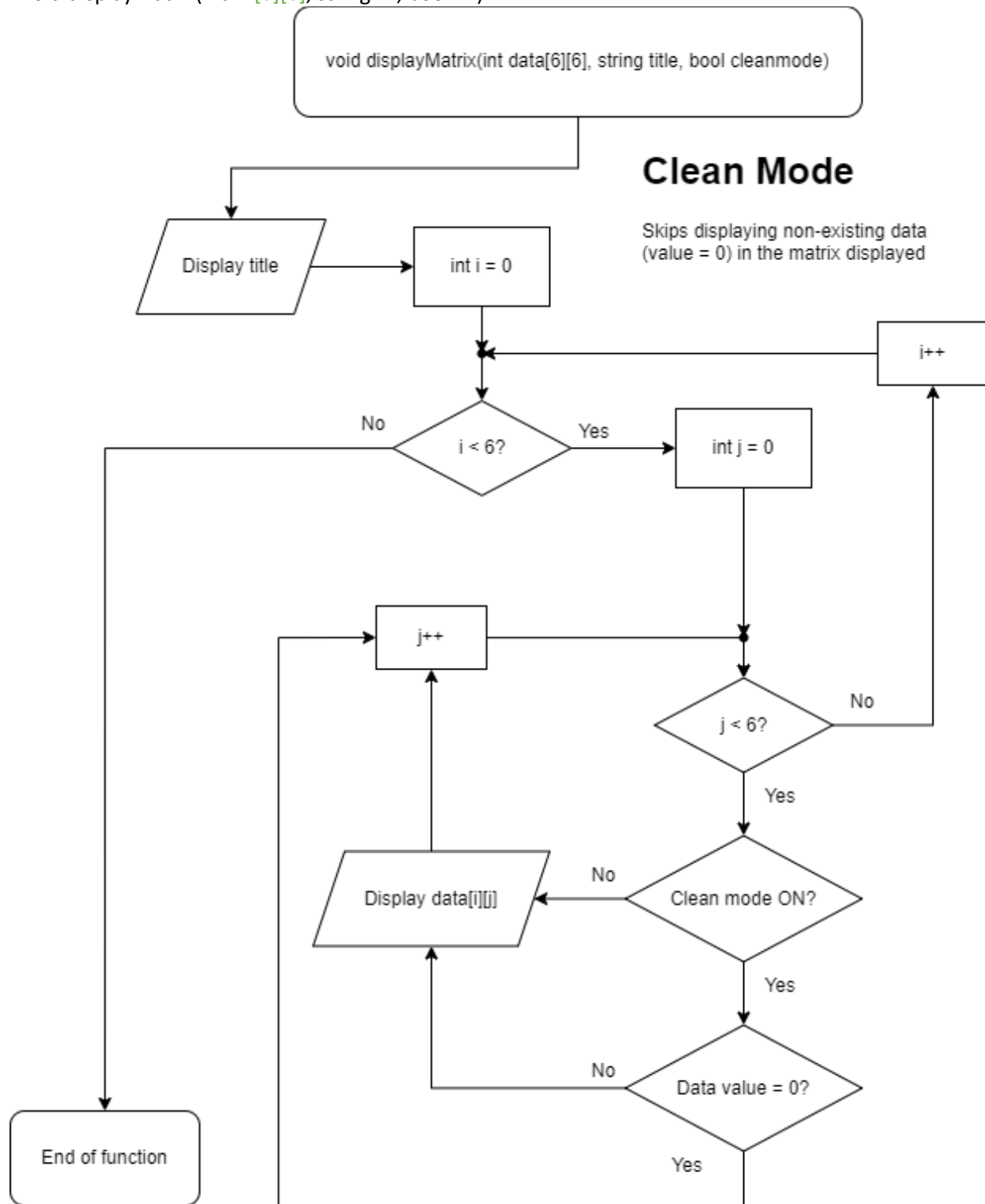


Figure 2.2.4-10 displayMatrix

- void setData(int IN/OUT[6][6], int IN)

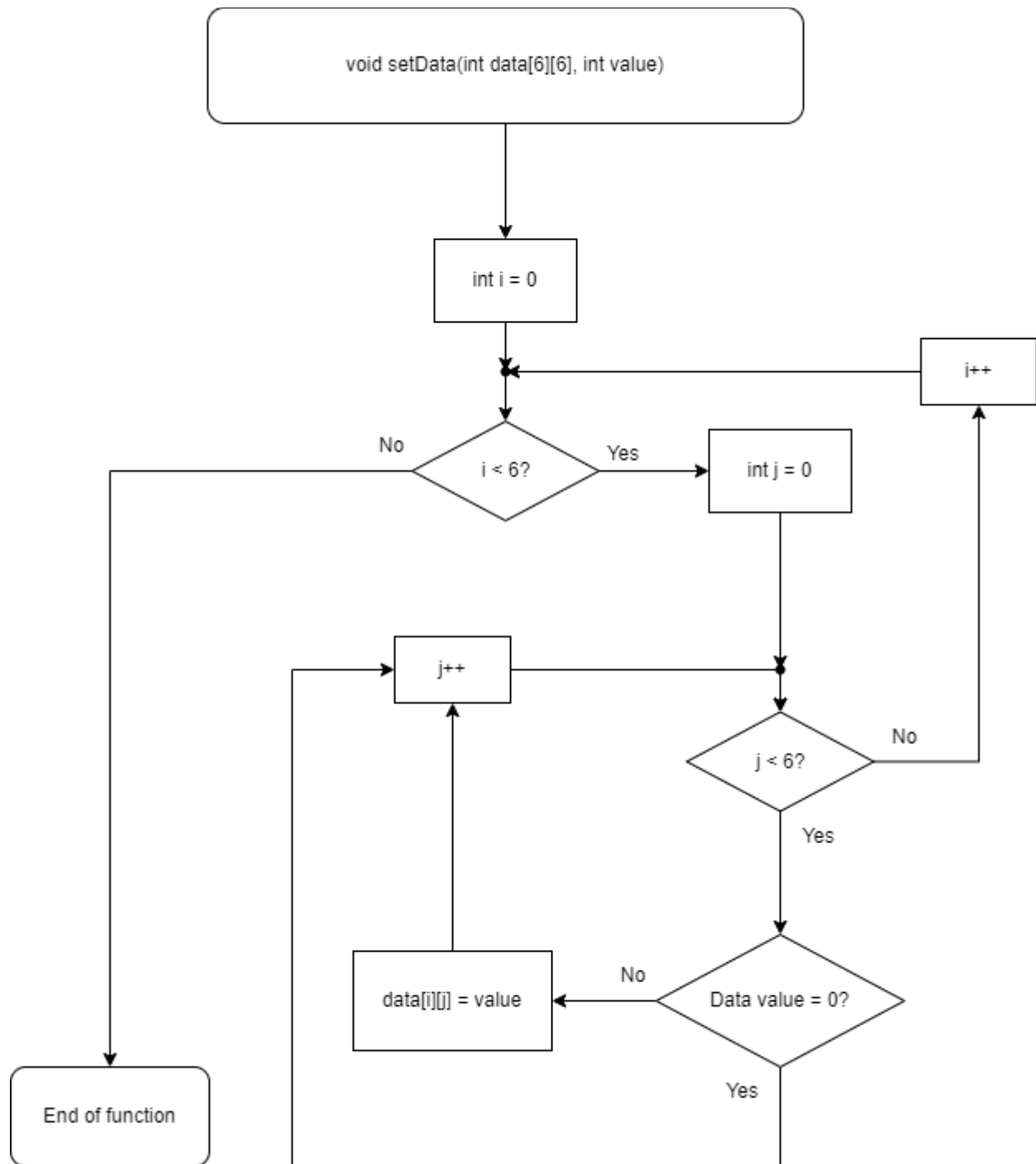


Figure 2.2.4-11 setData()

- void closing()

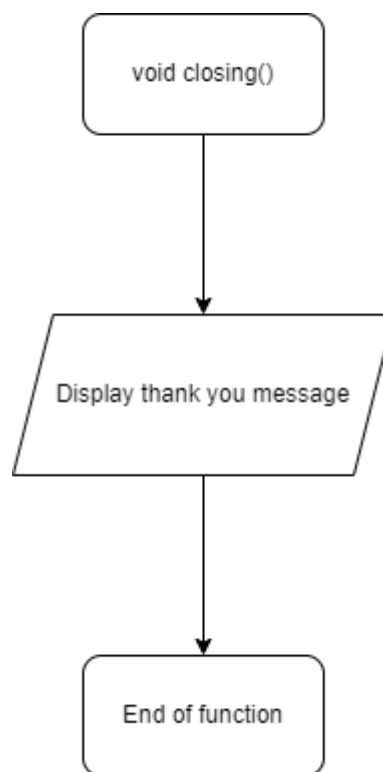


Figure 2.2.4-12 closing()

3. Testing Procedures

3.0 Intro

3.1 Generation of raw data

There are two (2) methods in generating raw data:

- Manual
- Auto

3.1.1 Requirements of data generation

Following list of items are required for generating the raw data files raw.txt:

- Table template template.png sizing 700x700 px
- Image data.png sizing 90x90 px
- main.py (Python program the code used to process image and generate data coordinates, uses OpenCV template matching)
- Python environment to run the Python program

	R1	R2	R3	R4	R5	NS
DELIVERY						
RET						

Figure 3.1.1-1 Empty table template template.png



Figure 3.1.1-2 Data images, A.png, B.png and C.png

```

1  import cv2 as cv
2  import numpy as np
3  import time
4  from matplotlib import pyplot as plt
5  import pandas
6
7  #----- absolute coordi
8
9  roomcoord = [100, 200, 300, 400, 500, 600]
10 delivery = [100, 200, 300, 400, 500, 600]
11 #retrieve = [245, 290]
12 thresholdpoint = 30
13
14 rooms = ['R1', 'R2', 'R3', 'R4', 'R5', 'NS']
15 colhead = ['D', '', '', '', 'R', '']
16
17 #-----
18
19 null = []
20
21
22 datared = [[0, 0, 0, 0, 0, 0],
23            [0, 0, 0, 0, 0, 0],
24            [0, 0, 0, 0, 0, 0],
25            [0, 0, 0, 0, 0, 0],
26            [0, 0, 0, 0, 0, 0],
27            [0, 0, 0, 0, 0, 0]]
28
29 datablu = [[0, 0, 0, 0, 0, 0],
30            [0, 0, 0, 0, 0, 0],
31            [0, 0, 0, 0, 0, 0],
32            [0, 0, 0, 0, 0, 0],
33            [0, 0, 0, 0, 0, 0],
34            [0, 0, 0, 0, 0, 0]]
35
36 datagur = [[0, 0, 0, 0, 0, 0],
37            [0, 0, 0, 0, 0, 0],
38            [0, 0, 0, 0, 0, 0],
39            [0, 0, 0, 0, 0, 0],
40            [0, 0, 0, 0, 0, 0],
41            [0, 0, 0, 0, 0, 0]]
42
43 datahaz = [[0, 0, 0, 0, 0, 0],
44            [0, 0, 0, 0, 0, 0],
45            [0, 0, 0, 0, 0, 0],
46            [0, 0, 0, 0, 0, 0],
47            [0, 0, 0, 0, 0, 0],
48            [0, 0, 0, 0, 0, 0]]
49
50 matrix = [[0, 0, 0, 0, 0, 0],
51            [0, 0, 0, 0, 0, 0],
52            [0, 0, 0, 0, 0, 0],
53            [0, 0, 0, 0, 0, 0],
54            [0, 0, 0, 0, 0, 0],
55            [0, 0, 0, 0, 0, 0]]
56
57 redcoord = [(0, 0)]
58 absred=[]
59 tempredcoord = [(0, 0)]
60 tempabsred=[]
61
101 sort(temp1, temp2, 1, 1)
102 leny = len(temp2)
103 datain.pop(0)
104 #print("")
105 #print("rows: ", leny)
106 #print("cols: ", lenx)
107 #print("")
108 for i in range(0, len(temp2)):
109     for j in range(0, len(delivery)):
110         if((temp2[i][1]+thresholdpoint) > d
111             #print("hit! row ", j)
112             for k in range(0, len(datain)):
113                 if((datain[k][1]+thresholdp
114                     #print("hit! index ", k)
115                     difference_array = np.ab
116                     index = difference_array
117                     #print("hit! room index
118                     if(matrixdata[j][index]
119                         matrixdata[j][index]
120
121
122 #print("[INFO] loading images...")
123 #image = cv.imread(targettable)
124 #template = cv.imread("1.png")
125 #cv.imshow(targettable, image)
126 #cv.imshow("1.png", template)
127
128 #----- Vision Manag
129
130 img_rgb = cv.imread(targettable)
131 img_gray = cv.cvtColor(img_rgb, cv.COLOR_BGR2GRA
132
133 template = cv.imread('A.png',0)
134 w, h = template.shape[::-1]
135
136 res = cv.matchTemplate(img_gray,template,cv.TM_C
137 threshold = 0.7
138 loc = np.where( res >= threshold)
139 for pt in zip(*loc[::-1]):
140     cv.rectangle(img_rgb, pt, (pt[0] + w, pt[1]
141         redcoord.append(pt)
142 cv.imwrite('res1.png',img_rgb)
143 #-----
144 img_rgb = cv.imread(targettable)
145 img_gray = cv.cvtColor(img_rgb, cv.COLOR_BGR2GRA
146
147 template = cv.imread('B.png',0)
148 w, h = template.shape[::-1]
149
150 res = cv.matchTemplate(img_gray,template,cv.TM_C
151 threshold = 0.8
152 loc = np.where( res >= threshold)
153 for pt in zip(*loc[::-1]):
154     cv.rectangle(img_rgb, pt, (pt[0] + w, pt[1]
155         blucoord.append(pt)
156 cv.imwrite('res2.png',img_rgb)
157 #-----
158 img_rgb = cv.imread(targettable)
159 img_gray = cv.cvtColor(img_rgb, cv.COLOR_BGR2GRA
160
161 template = cv.imread('C.png',0)

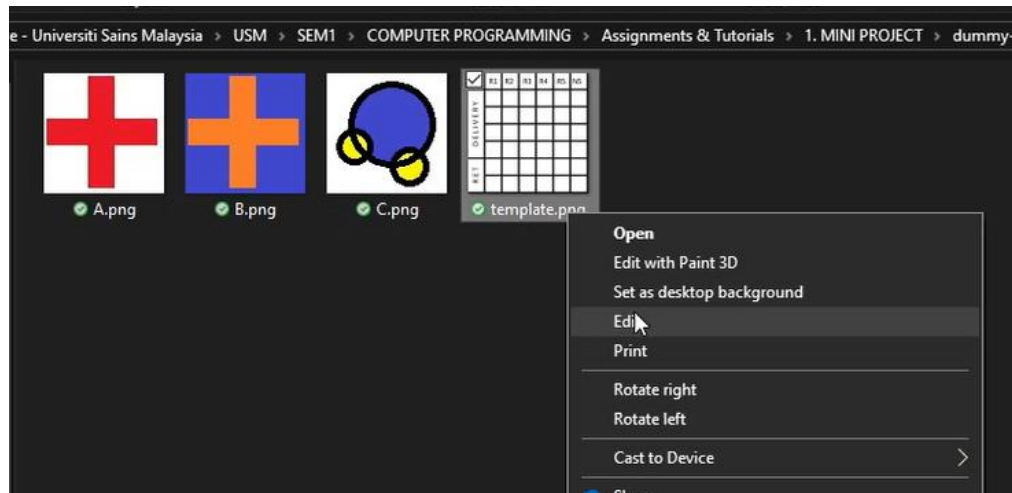
```

Figure 3.1.1-3 main.py containing OpenCV template matching function

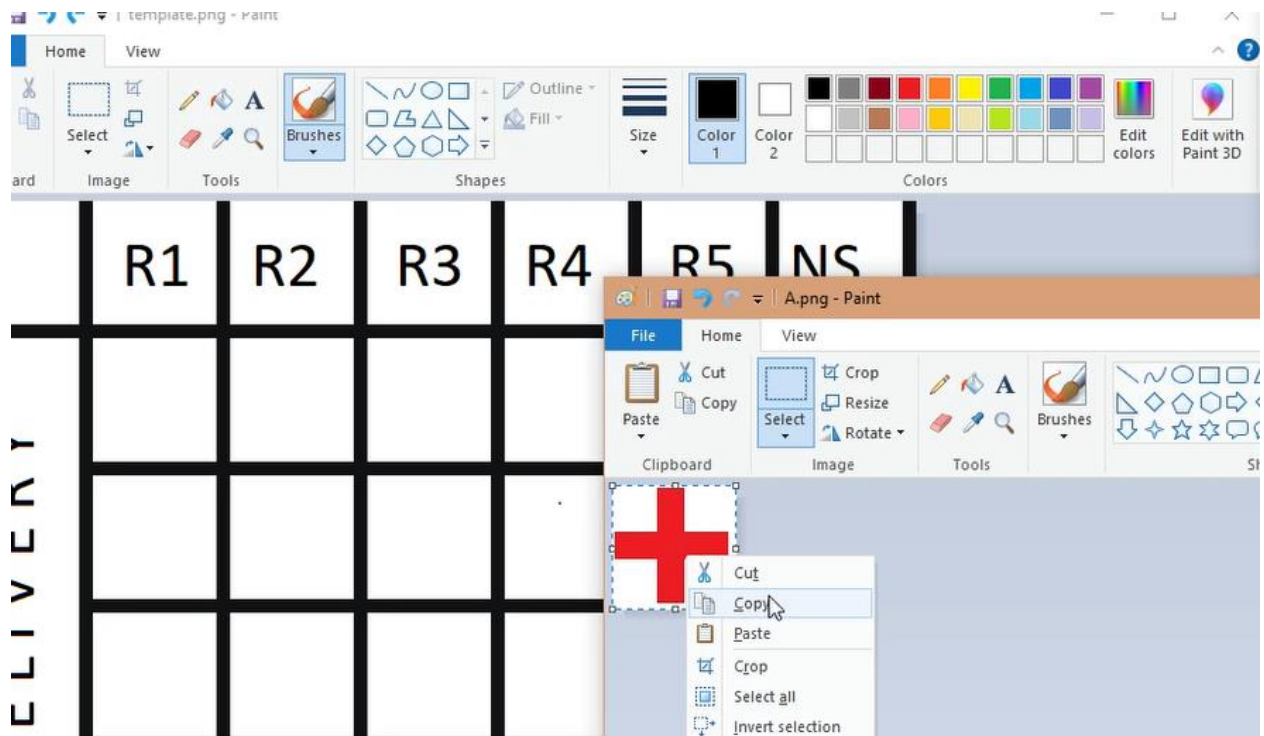
3.1.2 Manual

Manual method is the early stage of data generation for preparing preliminary data, and it involves using MS Paint to manually edit the table image by inserting image data into the table template to obtain a different combination of data sets.

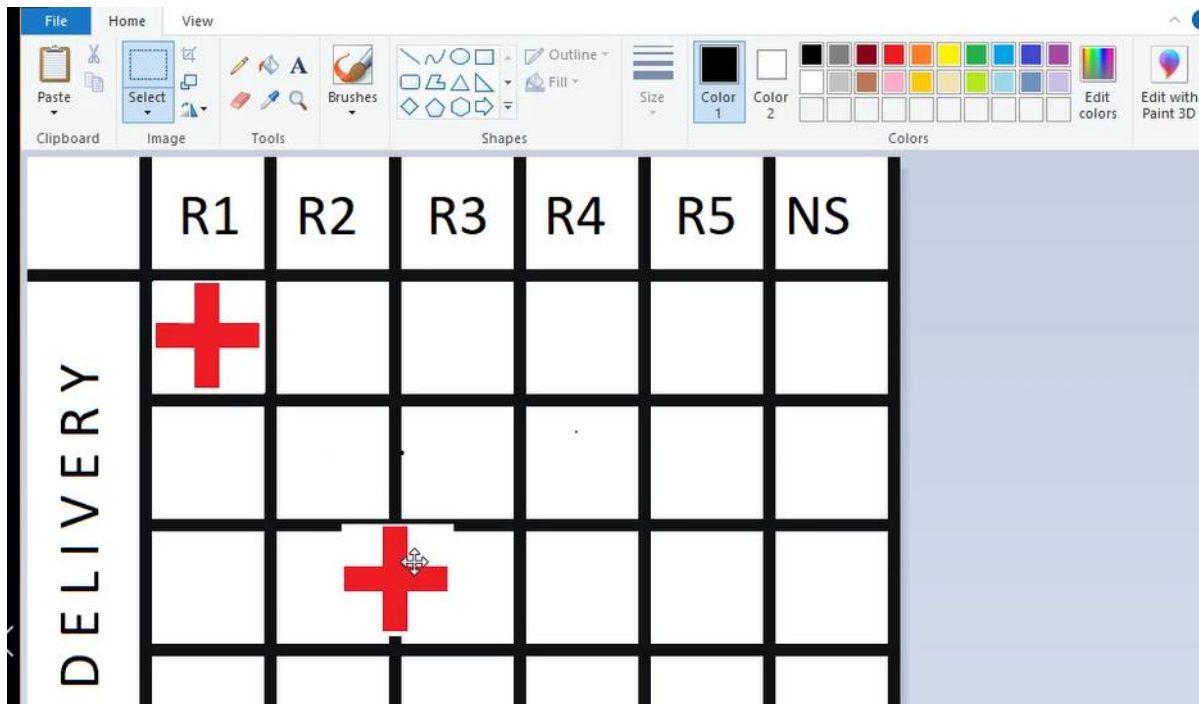
1. First, edit template (template.png) and data to use (A.png) using MS paint



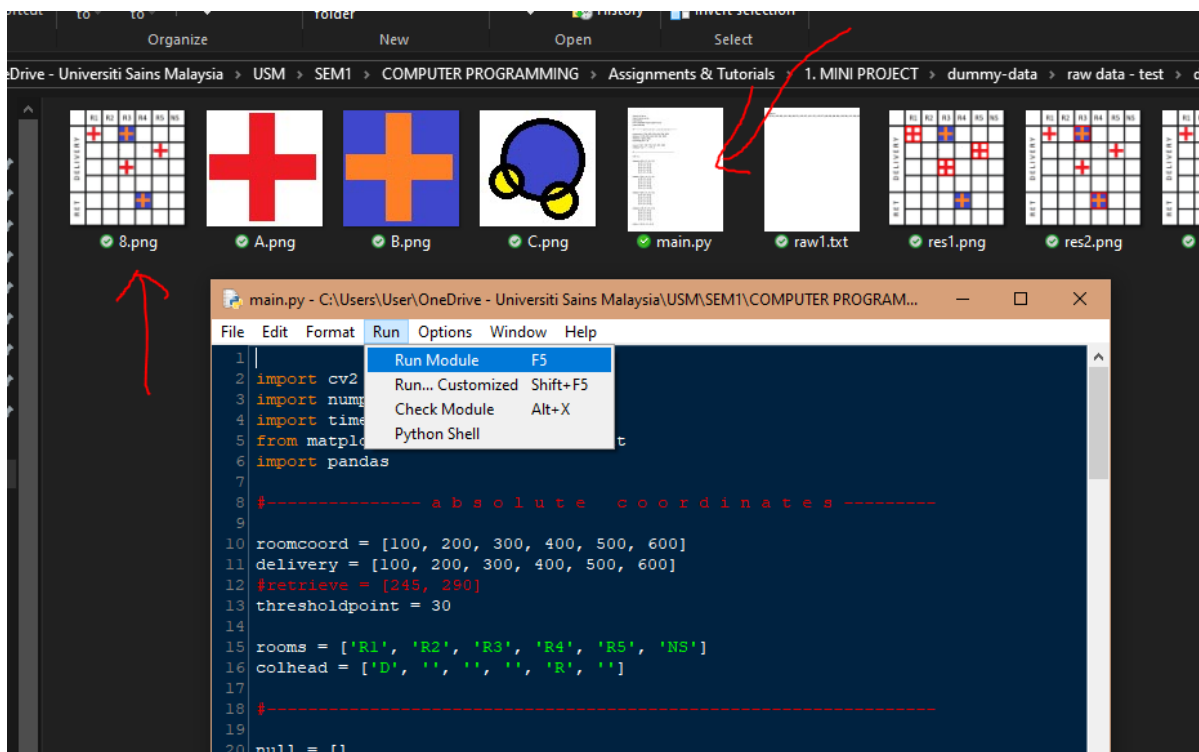
2. With both images opened in MS Paint, in the data image, CTRL+A to select the whole image of the data image, then copy the image



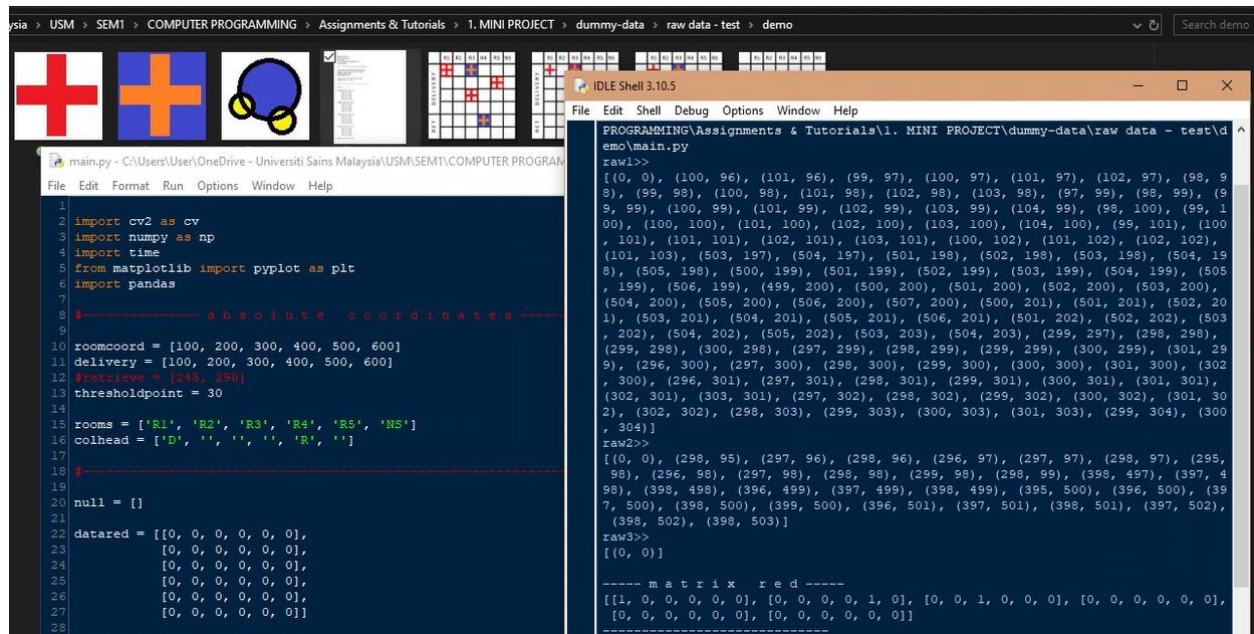
- Then in the template.png, CTRL+V paste the data image, and drag and move to position the data into desired cell. It can be repeated paste and can be done with multiple data images as well. After done with a different combination set, save the image to the same location as the main.py.



- Open the main.py, right click and select “Edit with IDLE”, in the same location as the data set, press F5 to run the program



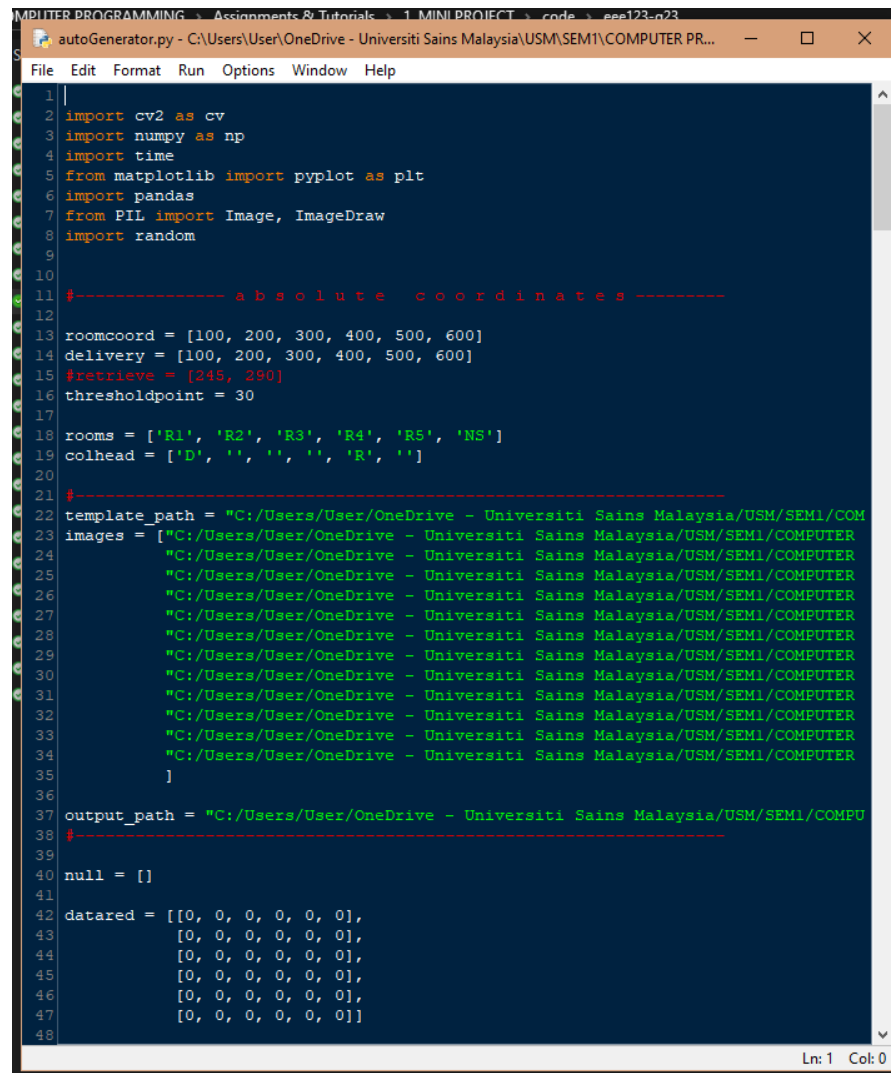
5. A terminal will pop up after executing the code. The results will be in the terminal (IDLE SHELL).



```
main.py - C:\Users\User\OneDrive - Universiti Sains Malaysia\USM\SEM1\COMPUTER PROGRAMMING\Assignments & Tutorials\1. MINI PROJECT\dummy-data\raw data - test\demo\main.py
File Edit Format Run Options Window Help
1
2 import cv2 as cv
3 import numpy as np
4 import time
5 from matplotlib import pyplot as plt
6 import pandas
7
8 ----- absolute coordinates -----
9
10 roomcoord = [100, 200, 300, 400, 500, 600]
11 delivery = [100, 200, 300, 400, 500, 600]
12 threshold = 1248, 340]
13 thresholdpoint = 30
14
15 rooms = ['R1', 'R2', 'R3', 'R4', 'R5', 'NS']
16 colhead = ['D', '', '', '', 'R', '']
17
18
19
20 null = []
21
22
23 datedred = [[0, 0, 0, 0, 0, 0],
24             [0, 0, 0, 0, 0, 0],
25             [0, 0, 0, 0, 0, 0],
26             [0, 0, 0, 0, 0, 0],
27             [0, 0, 0, 0, 0, 0],
28             [0, 0, 0, 0, 0, 0]]
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
259
```

3.1.3 Auto

Automatic data generation is introduced in later stages in the development of the C++ code for generating more diverse data sets rapidly. Automatic data generation involves in running a custom Python program that includes OpenCV function and uses the template.png and data images to generate random data sets on each execution, and outputs the random generated data set image, and three (3) full sets of raw data text files to a certain location. The location for the text file can be configured to output to a specified location, in our case, to the same location as the main.cpp file.

A screenshot of a Python IDE window titled 'autoGenerator.py - C:\Users\User\OneDrive - Universiti Sains Malaysia\USM\SEM1\COMPUTER PR...'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is as follows:

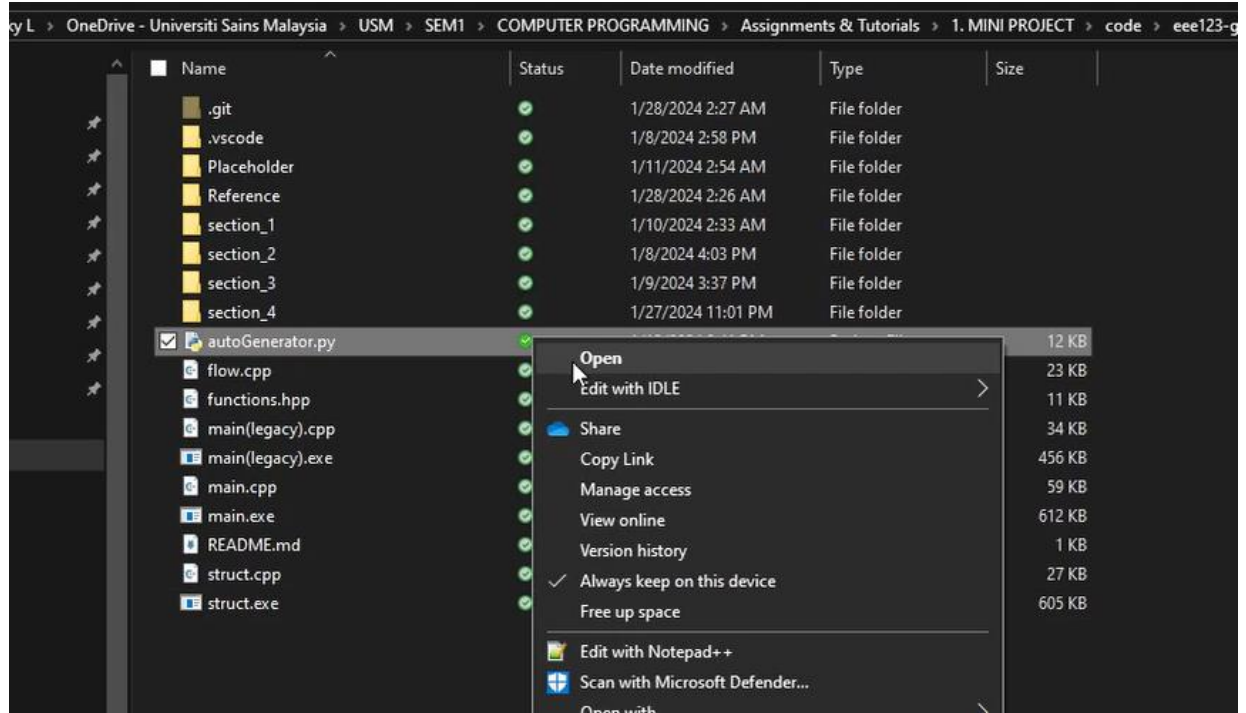
```
1 |
2 | import cv2 as cv
3 | import numpy as np
4 | import time
5 | from matplotlib import pyplot as plt
6 | import pandas
7 | from PIL import Image, ImageDraw
8 | import random
9 |
10 |
11 | ----- absolute coordinates -----
12 |
13 | roomcoord = [100, 200, 300, 400, 500, 600]
14 | delivery = [100, 200, 300, 400, 500, 600]
15 | $retrieve = [245, 290]
16 | thresholdpoint = 30
17 |
18 | rooms = ['R1', 'R2', 'R3', 'R4', 'R5', 'NS']
19 | colhead = ['D', '', '', '', 'R', '']
20 |
21 | -----
22 | template_path = "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COM
23 | images = ["C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
24 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
25 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
26 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
27 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
28 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
29 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
30 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
31 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
32 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
33 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
34 |           "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
35 |           ]
36 |
37 | output_path = "C:/Users/User/OneDrive - Universiti Sains Malaysia/USM/SEM1/COMPU
38 | -----
39 |
40 | null = []
41 |
42 | datared = [[0, 0, 0, 0, 0, 0],
43 |            [0, 0, 0, 0, 0, 0],
44 |            [0, 0, 0, 0, 0, 0],
45 |            [0, 0, 0, 0, 0, 0],
46 |            [0, 0, 0, 0, 0, 0],
47 |            [0, 0, 0, 0, 0, 0]]
48 |
```

Figure 3.1.3-1 Custom Python autoGenerator.py Program

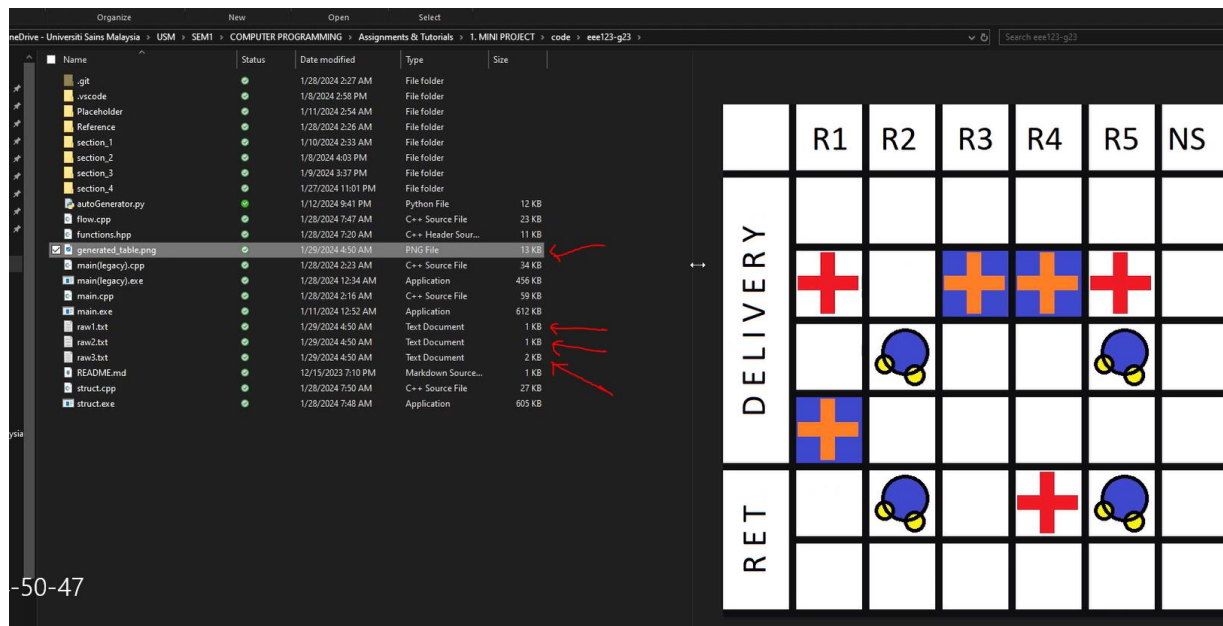
Auto generating raw data sets is very straightforward, we only need to execute the autoGenerator.py (given that the autoGenerator.py is configured beforehand), and we will obtain a random data set.

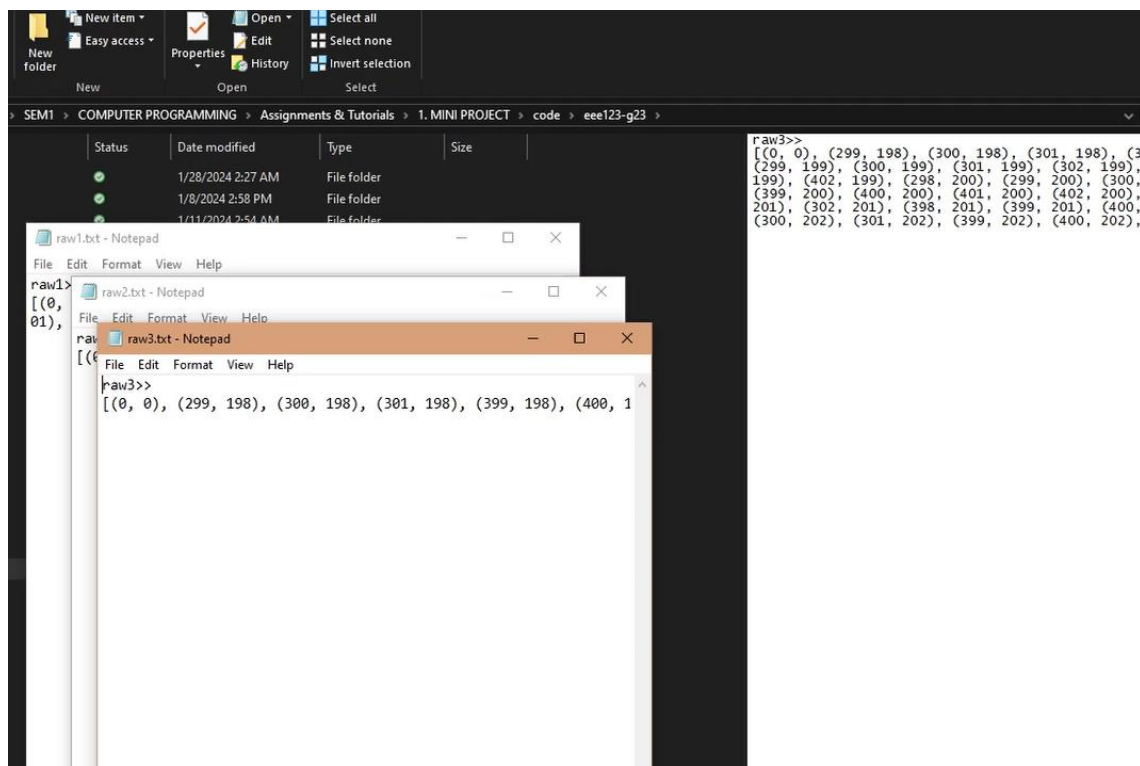
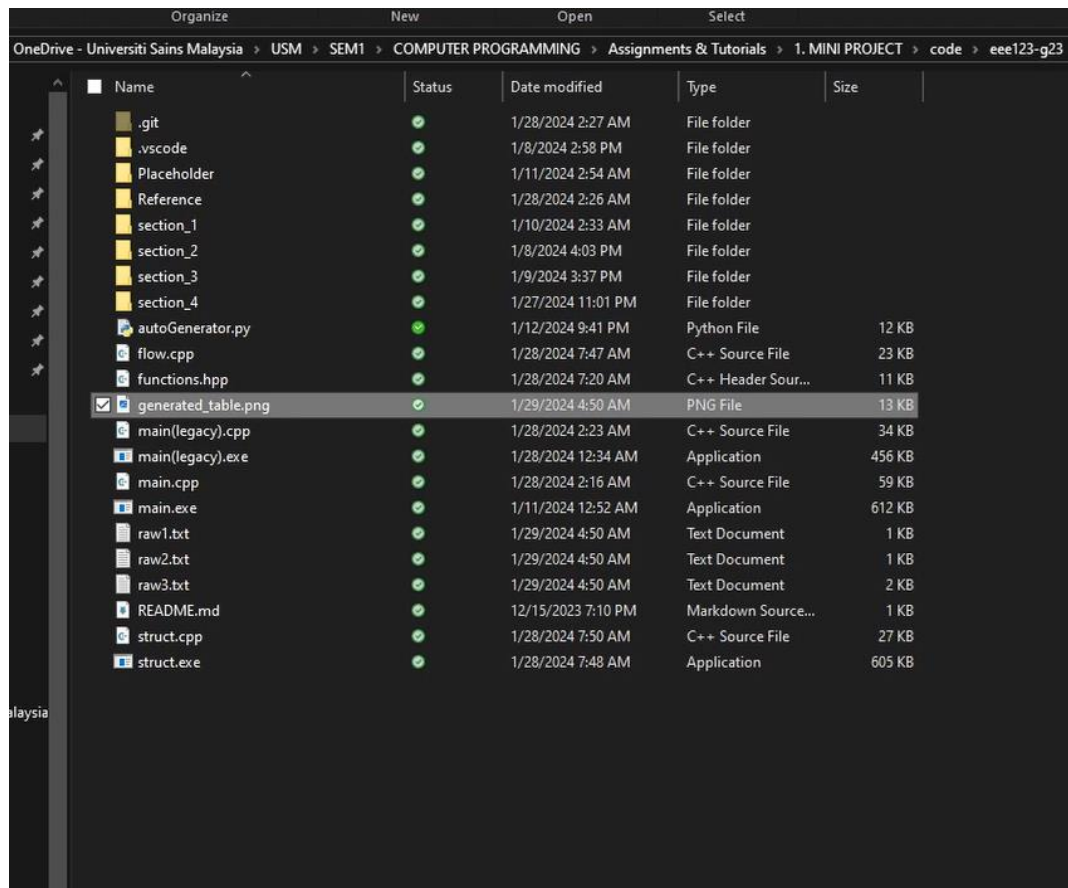
Thanks to ChatGPT, the auto data generator can be made within half an hour with limited knowledge in the Python language.

1. Locate the configured autoGenerator.py and run



2. After running, an image data set is randomly generated as well as raw text files.





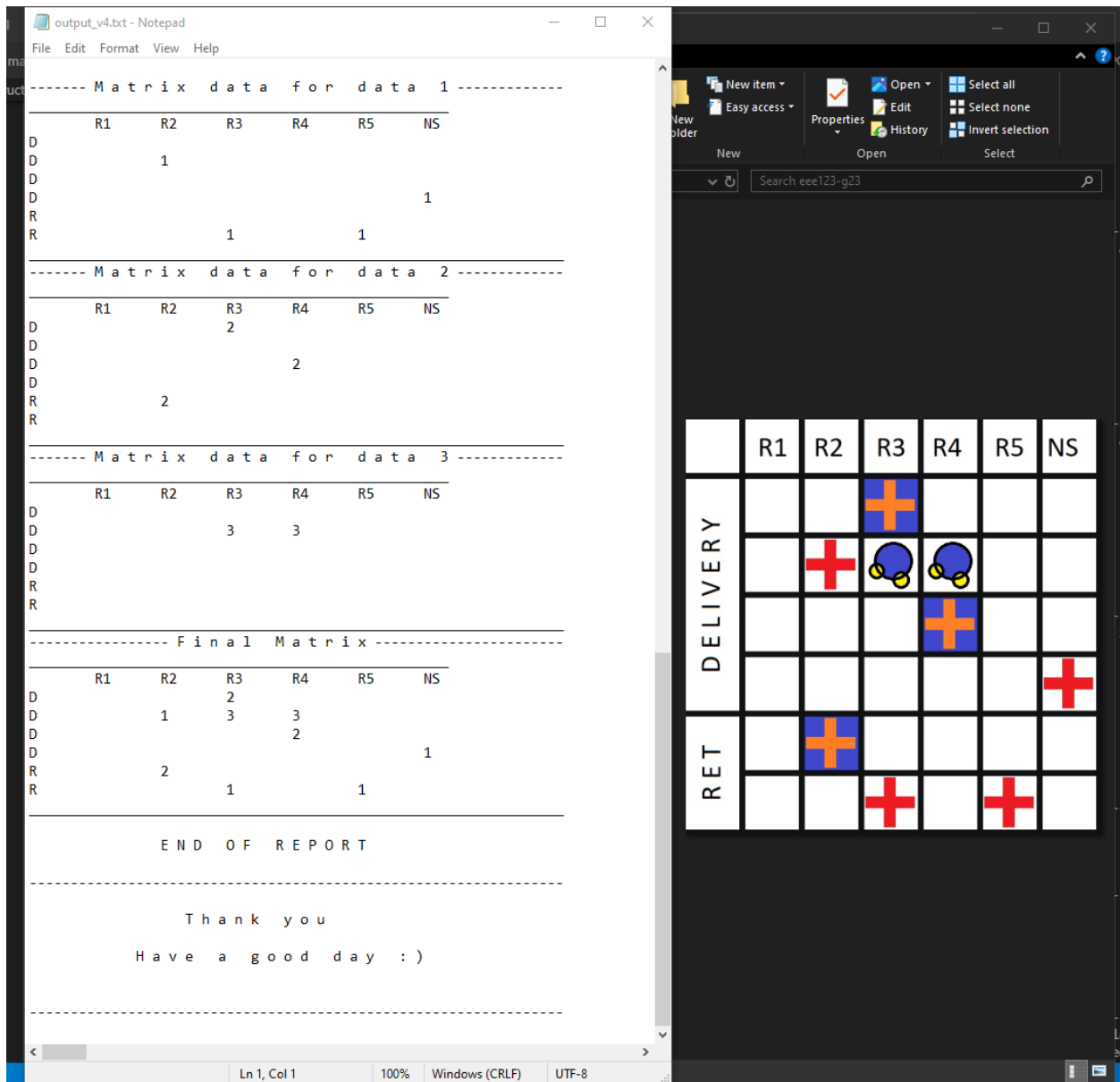


Figure 3.3-3 Comparing generated report (clean mode ON) with image set

The results... seems about right!

4. Results & Discussions

4.1 Trial Runs

4.1.1 Trial Run #1

Random generate image:

	R1	R2	R3	R4	R5	NS
DELIVERY				+	⦿	+
				+		
RET				⦿		+

Figure 4.1-1 Trial 1 image set

output_v4.txt results:

----- M a t r i x D i s p l a y -----						
Clean mode: ON						
----- M a t r i x d a t a f o r d a t a 1 -----						
	R1	R2	R3	R4	R5	NS
D				1		1
D						
D						
D						
R						1
R						
----- M a t r i x d a t a f o r d a t a 2 -----						
	R1	R2	R3	R4	R5	NS
D						
D						
D						
D				2		
R						
R						
----- M a t r i x d a t a f o r d a t a 3 -----						
	R1	R2	R3	R4	R5	NS
D					3	
D						
D						
D						
R				3		
R						
----- F i n a l M a t r i x -----						
	R1	R2	R3	R4	R5	NS
D				1	3	1
D						
D						
D				2		
R				3		1
R						
E N D O F R E P O R T						

Figure 4.1-2 Trial 1 results

4.1.2 Trial Run #2

Random generate image:









	R1	R2	R3	R4	R5	NS
DELIVERY						
						
						
RET						

Figure 4.1-3 Trial 2 image set

output_v4.txt results:

----- M a t r i x D i s p l a y -----						
Clean mode: ON						
----- M a t r i x d a t a f o r d a t a 1 -----						
	R1	R2	R3	R4	R5	NS
D						1
D						1
D						
D						
R	1					
R						
----- M a t r i x d a t a f o r d a t a 2 -----						
	R1	R2	R3	R4	R5	NS
D				2		
D						
D						
D						
R						2
R						
----- M a t r i x d a t a f o r d a t a 3 -----						
	R1	R2	R3	R4	R5	NS
D						
D	3					
D	3					3
D						
R						
R						
----- F i n a l M a t r i x -----						
	R1	R2	R3	R4	R5	NS
D				2		1
D	3					1
D	3					3
D						
R	1					2
R						
E N D O F R E P O R T						

Figure 4.1-4 Trial 2 results

4.1.3 Trial Run #3

Random generate image:





	R1	R2	R3	R4	R5	NS
DELIVERY						
			+			+
			+		+	
RET						
						

Figure 4.1-5 Trial 3 image set

output_v4.txt results:

----- Matrix Display -----						
Clean mode: ON						
----- Matrix data for data 1 -----						
	R1	R2	R3	R4	R5	NS
D			1			1
D			1		1	
D						
R						
R						
----- Matrix data for data 2 -----						
	R1	R2	R3	R4	R5	NS
D						
D						
D						
R						
R						
----- Matrix data for data 3 -----						
	R1	R2	R3	R4	R5	NS
D						
D						
D						3
R					3	
R				3	3	
----- Final Matrix -----						
	R1	R2	R3	R4	R5	NS
D			1			1
D			1		1	3
R					3	
R				3	3	
END OF REPORT						

Figure 4.1-6 Trial 3 results

4.1.4 Trial Run #4

Manual set image (Set 7) :

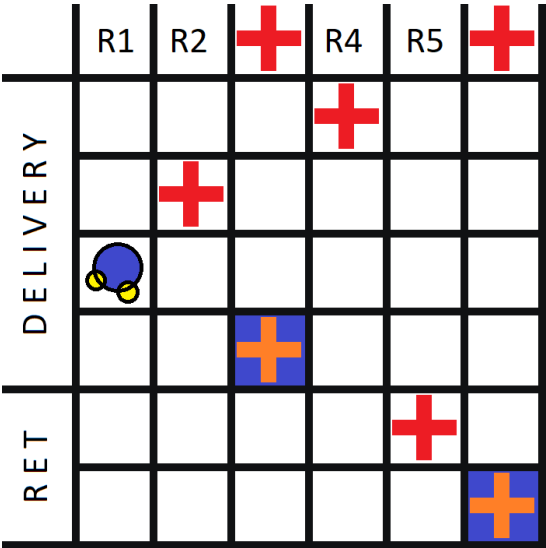


Figure 4.1-7 Trial 4 image set

output_v4.txt results:

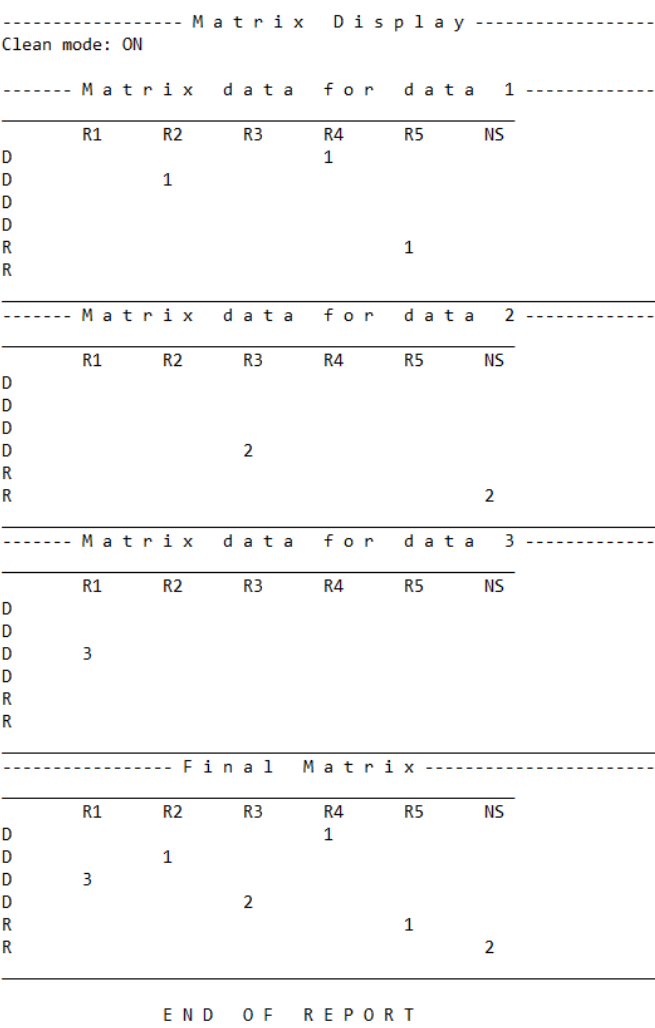


Figure 4.1-8 Trial 4 results

4.2 Clean Mode

The reason for having clean mode is so that the user can watch and validate the results clearer. Watching a bunch of zeros (0) is pretty eyesore to some people (at least to some among team members). To illustrate the difference between Clean Mode ON and Clean Mode OFF:

Clean mode: ON						
	R1	R2	R3	R4	R5	NS
D					1	
D				1		
D						
D	1					
R	1				1	
R						

Clean mode: OFF						
	R1	R2	R3	R4	R5	NS
D	0	0	0	0	1	0
D	0	0	0	1	0	0
D	0	0	0	0	0	0
D	1	0	0	0	0	0
R	1	0	0	0	1	0
R	0	0	0	0	0	0

Clean mode: ON						
	R1	R2	R3	R4	R5	NS
D						
D						
D						
D					3	
R	3					

Clean mode: OFF						
	R1	R2	R3	R4	R5	NS
D	0	0	0	0	0	0
D	0	0	0	0	0	0
D	0	0	0	0	0	0
D	0	0	0	0	3	0
R	0	0	0	0	0	0
R	3	0	0	0	0	0

Clean mode: ON						
	R1	R2	R3	R4	R5	NS
D					1	
D				1	3	
D						
D	1					
R	1				1	
R	3					

Clean mode: OFF						
	R1	R2	R3	R4	R5	NS
D	0	0	0	0	1	0
D	0	0	0	1	3	0
D	0	0	0	0	0	0
D	1	0	0	0	0	0
R	1	0	0	0	1	0
R	3	0	0	0	0	0

END OF REPORT

Thank you

Have a good day :)

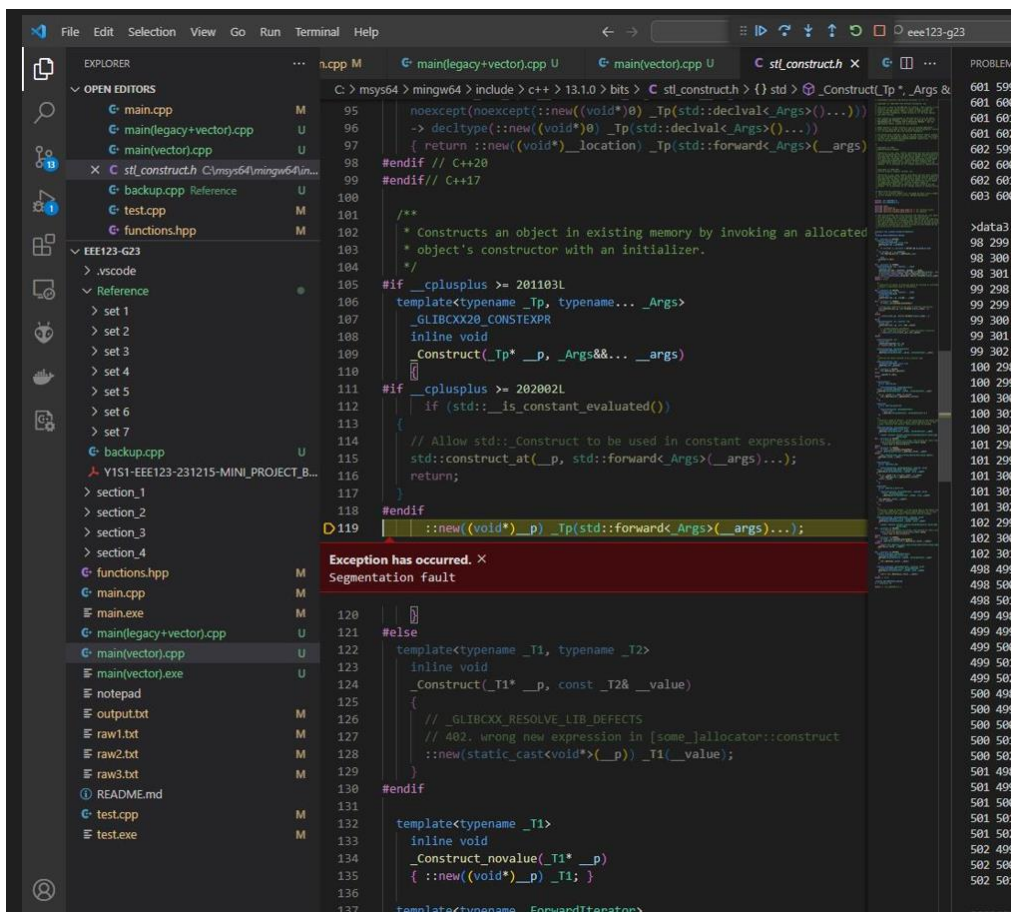
Figure 4.2-1 Results of Clean Mode ON and Clean Mode OFF, both share the same set of data

4.3 Vector As Data Container

The reason to use a vector as a multidimensional array variable instead of a regular one is due to the flexibility of the vector to adjust the size of its arrays. Because data sets are dynamic and will vary between iterations/executions, the amount of data presence in the raw data will differ as well. The way our code works can only fetch data given that the size of the index of the data obtained is accurate. The accuracy of the index size of the data is important as we need it to feed it into for-loops to transfer the data from one variable to another.

For example, given that raw1.txt contains 10 distinct coordinates. If we were to use a regular array variable, we must initiate and declare the size of the array beforehand, but the problem is we do not know the number of data (distinct coordinates) firsthand, which causes a contradicting situation. The only way for this to work is to declare the size of array by assumption, let's say, N. The problem arises when:

- if N is less than the actual number of data itself, we will miss out the rest of the data, thus obtaining an inaccurate result
- if N is more than the actual number of data itself, the container in the array that are unused will either hold a 0 or an undetermined value. The value 0 will give inaccurate results while undetermined value will result in segmentation error (since we are feeding the for-loop using the value as index, if "for(int i=0;i<size;i++)" and size suddenly jumps from a valid number to an undetermined number, for example, -32615)



```
File Edit Selection View Go Run Terminal Help
C:\msys64> mingw64 > include > c++ > 13.1.0 > bits > C:\stl_construct.h > {} std > _Construct_Tp*, _Args &...
main.cpp M 95 noexcept(noexcept(new((void*)0) _Tp(std::declval<_Args>()...)))
main(legacy+vector).cpp U 96 -> decltype(new((void*)0) _Tp(std::declval<_Args>()...))
main(vector).cpp U 97 { return new((void*)0) _Tp(std::forward<_Args>(_args))
stl_construct.h C:\msys64\mingw64\in... 98 #endif // C++20
backup.cpp Reference U 99 #endif // C++17
test.cpp M 100
functions.hpp M 101
102 /**
103  * Constructs an object in existing memory by invoking an allocated
104  * object's constructor with an initializer.
105  */
106 #if __cplusplus >= 201103L
107 template<typename _Tp, typename... _Args>
108 _GLIBCXX20_CONSTEXPR
109 inline void
110 Construct(_Tp* _p, _Args&&... _args)
111 {
112     if (std::is_constant_evaluated())
113     {
114         // Allow std::Construct to be used in constant expressions.
115         std::construct_at(_p, std::forward<_Args>(_args)...);
116         return;
117     }
118     #endif
119     ::new((void*)_p) _Tp(std::forward<_Args>(_args)...);
120 }
121 #else
122 template<typename _T1, typename _T2>
123 inline void
124 Construct(_T1* _p, const _T2& _value)
125 {
126     // _GLIBCXX_RESOLVE_LIB_DEFECTS
127     // 402. wrong new expression in [some_allocator::construct
128     ::new(static_cast<void*>(_p)) _T1(_value);
129 }
130 #endif
131
132 template<typename _T1>
133 inline void
134 Construct_novalue(_T1* _p)
135 { ::new((void*)_p) _T1; }
136
137 template<typename ForwardIterator>
```

Figure 4.3-1 Segmentation Error, condition: N > size of data

This results in variables cannot be declared/initiated before reading the raw data, and our code is written to declare variables sequentially along with the code. In earlier versions (*version 2 legacy.cpp*), a workaround is used before using vector as a multidimensional array. The current version (*version 4.52 Vectorstruct*) make use of struct and vectors just to optimize the code as per code maintenance.

```

// Legacy code (left pane)
316 //----- append y-coordinates i
317 int x100A[xcoord1[0]] = {0};
318 int x200A[xcoord1[1]] = {0};
319 int x300A[xcoord1[2]] = {0};
320 int x400A[xcoord1[3]] = {0};
321 int x500A[xcoord1[4]] = {0};
322 int x600A[xcoord1[5]] = {0};
323 if(data1_Index>0){
324     for(int i=0;i<xcoord1[0];i++){
325         x100A[i]=data1[i][1];
326     }
327     for(int i=0;i<xcoord1[1];i++){
328         x200A[i]=data1[i+xcoord1[0]][1];
329     }
330     for(int i=0;i<xcoord1[2];i++){
331         x300A[i]=data1[i+xcoord1[0]+xcoord1[1]][1];
332     }
333     for(int i=0;i<xcoord1[3];i++){
334         x400A[i]=data1[i+xcoord1[0]+xcoord1[1]+xcoord1[2]][1];
335     }
336     for(int i=0;i<xcoord1[4];i++){
337         x500A[i]=data1[i+xcoord1[0]+xcoord1[1]+xcoord1[2]+xcoord1[3]][1];
338     }
339     for(int i=0;i<xcoord1[5];i++){
340         x600A[i]=data1[i+xcoord1[0]+xcoord1[1]+xcoord1[2]+xcoord1[3]+xcoord1[4]][1];
341     }
342 }
343
344 int x100B[xcoord2[0]] = {0};
345 int x200B[xcoord2[1]] = {0};
346 int x300B[xcoord2[2]] = {0};
347 int x400B[xcoord2[3]] = {0};
348 int x500B[xcoord2[4]] = {0};
349 int x600B[xcoord2[5]] = {0};
350 if(data2_Index>0){
351     for(int i=0;i<xcoord2[0];i++){
352         x100B[i]=data2[i][1];
353     }
354     for(int i=0;i<xcoord2[1];i++){
355         x200B[i]=data2[i+xcoord2[0]][1];
356     }
357     for(int i=0;i<xcoord2[2];i++){
358         x300B[i]=data2[i+xcoord2[0]+xcoord2[1]][1];
359     }
360     for(int i=0;i<xcoord2[3];i++){
361         x400B[i]=data2[i+xcoord2[0]+xcoord2[1]+xcoord2[2]][1];
362     }
363     for(int i=0;i<xcoord2[4];i++){
364         x500B[i]=data2[i+xcoord2[0]+xcoord2[1]+xcoord2[2]+xcoord2[3]][1];
365     }
366 }

// Vectorstruct code (right pane)
297
298 //----- append y-coordinates i
299 vector<vector<vector<int>>> xcoord(3, vector<vector<int>>(6, vector<int>()));
300 int cumulativeIndex = 0;
301
302 for(int i=0;i<6;i++){
303     xcoord[0][i].resize(data1.xcoord[i]);
304     for(int j=0;j<data1.xcoord[i];j++){
305         xcoord[0][i][j]=dataV1[j+cumulativeIndex][1];
306     }
307     cumulativeIndex += data1.xcoord[i];
308 }
309
310 cumulativeIndex = 0;
311 for(int i=0;i<6;i++){
312     xcoord[1][i].resize(data2.xcoord[i]);
313     for(int j=0;j<data2.xcoord[i];j++){
314         xcoord[1][i][j]=dataV2[j+cumulativeIndex][1];
315     }
316     cumulativeIndex += data2.xcoord[i];
317 }
318
319 cumulativeIndex = 0;
320 for(int i=0;i<6;i++){
321     xcoord[2][i].resize(data3.xcoord[i]);
322     for(int j=0;j<data3.xcoord[i];j++){
323         xcoord[2][i][j]=dataV3[j+cumulativeIndex][1];
324     }
325     cumulativeIndex += data3.xcoord[i];
326 }
327
328 //----- sort by y-coord
329
330 if(data1.dataIndex>0){
331     for(int i=0;i<6;i++){
332         vectorSort(xcoord[0][i], data1.xcoord[i]);
333         vectorappendMatrix(xcoord[0][i], data1.xcoord[i], i, data1);
334     }
335 }
336
337 if(data2.dataIndex>0){
338     for(int i=0;i<6;i++){
339         vectorSort(xcoord[1][i], data2.xcoord[i]);
340         vectorappendMatrix(xcoord[1][i], data2.xcoord[i], i, data2);
341     }
342 }
343
344 if(data3.dataIndex>0){
345     for(int i=0;i<6;i++){
346         vectorSort(xcoord[2][i], data3.xcoord[i]);
347         vectorappendMatrix(xcoord[2][i], data3.xcoord[i], i, data3);
348     }
349 }
  
```

Figure 4.3-2 Comparison before and after optimization, v2 legacy(left) vs v4.52 Vectorstruct(right), arrow indicates equivalency

Note that the code (variable declarations and functions) in left tab (legacy) is repetitive while in the right tab (Vectorstruct) is greatly compressed. Both are equivalent in terms of operation and functionality.

The left tab uses 129 worth of lines of code, the right tab uses 53 lines of code, a reduction in 73 lines of code, while both deliver the same results, resulting around ~60% optimization in the same part of the code.

4.4 Invalid data / Data out of bounds

Referring back to 4.1.4 Trial Run #4,

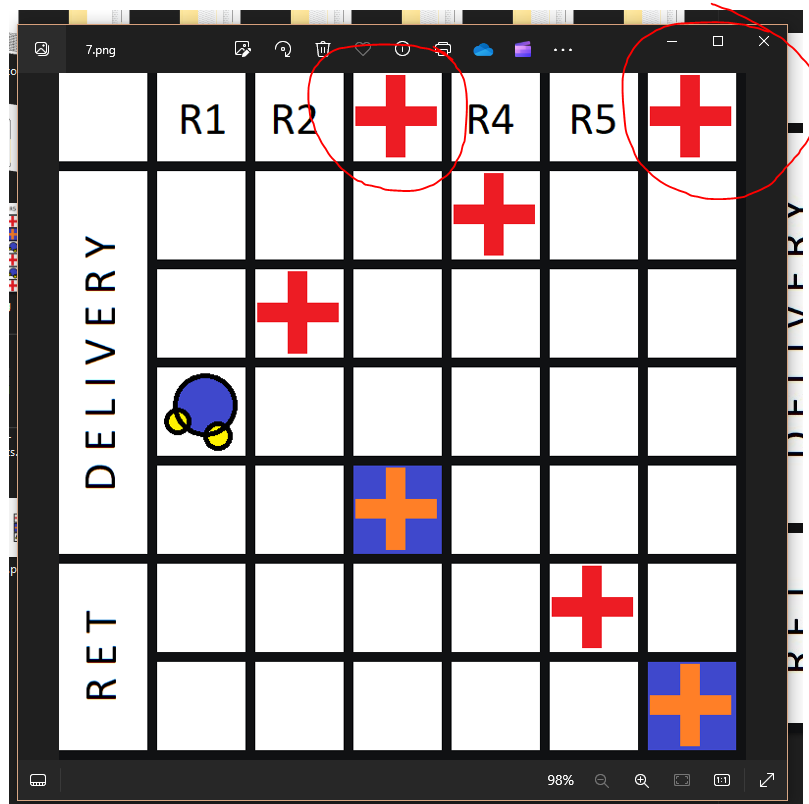


Figure 4.4-1 Trial Run #4 image set

The data seems to appear in places where it should not be (null coordinates). But there is no extra line of codes purposely written to check for the validity, and the results are still accurate and did not pick up the null coordinates, why?

When checking back our code, turns out it lies on the mathematical model itself (refer 2.1.3):

$$\text{int index} = \text{int}(\text{round}\left(\frac{\text{double}(\text{data}[i][0])}{100.0}\right) - 1)$$

The reason is that we consider the index to place the data to follow the for-loop indexing (starts from 0), thus the subtraction of 1 in the formula. And if any x or y coordinates starts below 100, it will be filtered out, because any coordinate number below 50 will result in -1, registering the data into `outputData[-1]` where `outputData[-1]` is nonexistent, thus the algorithm does not register the data into the matrix variables.

The assumption is that this will be a problem if the table has a size bigger than 6x6 and more image data is added (currently 3) to process, and might produce unexpected behavior or even errors, thus becoming the limitation of the code to cater for 3 types of data images and a working table size of 6x6 only.

5. Conclusions

The idea of the title is borrowed from the competition WorldSkills 2022 Malaysia Category Mobile Robotics, where our leader has participated in the past, and decided to use it as our topic. The reason is that our leader only written it in Python language (thus the existing Python program containing OpenCV template matching code to process and generate the raw data). The solution code is in Python language and is imperfect and buggy and we decided to work together and come up with a better solution while referencing some from the Python code.

The application for this code is limited to applications where it requires tabular data processing and data array sorting. Through this project, we have learnt a lot of extra C++ knowledge, a better understanding in solution-generating in programming, and data handling. All team members also learnt cross-roles collaboration and learnt to work and communicate together to tackle problems, thanks to guidance from our leader.

While the current version (v4.52 Vectorstruct) is stable, there are still more work required to do to further perfect the algorithm and to prevent errors and unexpected results as per code maintenance. Considering time and knowledge is limited to all team members, the result of the project is better than initially expected in the early stages.

Nonetheless, the goal of this project is achieved, concluding the project a success.

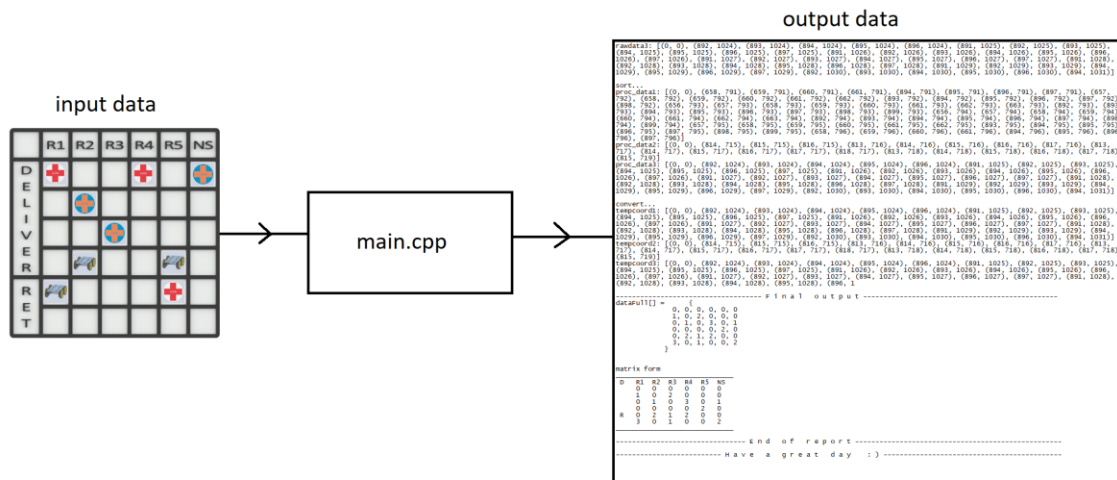


Figure 5.1 Goal achieved (refer Figure 1.2-1)

6. References

- Boost Library Documentation
<https://www.boost.org/doc/>
- Multidimensional Arrays in C++
<http://www.cplusplus.com/doc/tutorial/arrays/#multidimensional-arrays>
- User Input in C++
http://www.cplusplus.com/doc/tutorial/basic_io/
- File Input and Output in C++
<http://www.cplusplus.com/doc/tutorial/files/>
- STL (Standard Template Library)
<http://www.cplusplus.com/reference/stl/>
- 2D array with examples
<https://beginnersbook.com/2014/01/2d-arrays-in-c-example/>
- input/output with files
<https://cplusplus.com/doc/tutorial/files/>
- reading array from file
<https://cplusplus.com/forum/beginner/78150/>
- Coding Knowledge about Matrix Display
<https://cplusplus.com/forum/beginner/9126/>
- Headers of Matrix
<https://cplusplus.com/forum/general/254153/>
- OpenCV template matching
https://docs.opencv.org/3.4/de/da9/tutorial_template_matching.html
- OpenCV Documentation
<https://docs.opencv.org/master/>
- Eigen C++ Library Documentation
<https://eigen.tuxfamily.org/dox/>
- setw() function
<https://favtutor.com/blogs/setw-cpp>
- read file line by line
[https://medium.com/@teamcode20233/how-to-read-file-line-by-line-in-c-a1d829f697c0#:~:text=Using%20getline\(\)%20function%20to%20read%20file%20line%20by%20line%20in%20C%2B%2B&text=The%20getline\(\)%20function%20reads,handle%20different%20types%20of%20input.](https://medium.com/@teamcode20233/how-to-read-file-line-by-line-in-c-a1d829f697c0#:~:text=Using%20getline()%20function%20to%20read%20file%20line%20by%20line%20in%20C%2B%2B&text=The%20getline()%20function%20reads,handle%20different%20types%20of%20input.)

- out of bounds array
<https://stackoverflow.com/questions/15646973/how-dangerous-is-it-to-access-an-array-out-of-bounds>
- paused on exception
<https://stackoverflow.com/questions/39063281/typescript-promise-rejecting-and-vscode-debugger-behavior>
- read array from txt file
<https://stackoverflow.com/questions/40834066/how-do-i-read-text-from-txt-file-into-an-array>
- appending array size
<https://stackoverflow.com/questions/57023251/adding-to-an-array-in-c-c-with-no-size>
- passing 2d array to function
<https://stackoverflow.com/questions/8767166/passing-a-2d-array-to-a-c-function>
- Function of Curly Braces in C++
<https://www.arduino.cc/reference/en/language/structure/further-syntax/curlybraces/>
- C++ Arrays and Pointers:
<https://www.geeksforgeeks.org/arrays-and-pointers-in-c-cpp/>
- Sorting of Matrix
<https://www.geeksforgeeks.org/cpp-program-to-sort-the-matrix-row-wise-and-column-wise/>
- type of data type
<https://www.geeksforgeeks.org/derived-data-types-in-c/>
- Reading and Writing Files in C++
<https://www.geeksforgeeks.org/file-handling-c-classes/>
- Sorting Algorithms
<https://www.geeksforgeeks.org/introduction-to-algorithms/>
- Map of Tuples in C++ with Examples
https://www.geeksforgeeks.org/map-of-tuples-in-c-with-examples/?ref=ml_lbp
- size_t data type
https://www.geeksforgeeks.org/size_t-data-type-c-language/
- sizeof
https://www.geeksforgeeks.org/size_t-data-type-c-language/
- Understanding 3D Arrays in C++
<https://www.geeksforgeeks.org/sizeof-operator-c/>
- C++ Tuple Tutorial
<https://www.geeksforgeeks.org/three-dimensional-3d-array-in-c/>

- cpp tuple data structure
<https://www.geeksforgeeks.org/tuple-in-c/>
- Vectors in C++ language
<https://www.geeksforgeeks.org/tuples-in-c/>
- CPP vectors
<https://www.geeksforgeeks.org/vector-in-cpp-stl/>
- const &auto for loop
[https://www.google.com/search?q=for+\(const+auto%26+element+%3A+container\)+%7B&sourceid=chrome&ie=UTF-8](https://www.google.com/search?q=for+(const+auto%26+element+%3A+container)+%7B&sourceid=chrome&ie=UTF-8)
- sorting array in ascending order
<https://www.includehelp.com/cpp-programs/sort-an-array-in-ascending-order.aspx>
- for loop
<https://www.programiz.com/c-programming/c-for-loop>
- if loop
<https://www.programiz.com/c-programming/c-if-else-statement>
- incomplete type error
https://www.reddit.com/r/cpp_questions/comments/lwopto/c_incomplete_type_error/
- Ifstream explanation
<https://www.simplilearn.com/tutorials/cpp-tutorial/ifstream-in-cpp#:~:text=ofstream%2D%20This%20class%20describes%20an,from%20files%20and%20displays%20it.>
- Vectors as 3D arrays
<https://www.sololearn.com/en/Discuss/2189596/vectorintint-a-vectorpairintint-a-vectorvectorint-a-is-same-or-diff->
- C++ File Handling
<https://www.studytonight.com/cpp/file-handling-in-cpp.php>
- Introduction to tuples
<https://www.techtarget.com/whatis/definition/tuple#:~:text=In%20mathematics%2C%20a%20tuple%20is,square%20brackets%20or%20angle%20brackets.>
- file streams
https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm
- C++ Passing Arrays to Functions
https://www.tutorialspoint.com/cplusplus/cpp_passing_arrays_to_functions.htm
- array and loops
https://www.w3schools.com/cpp/cpp_arrays_loop.asp

- multidimensional array
https://www.w3schools.com/cpp/cpp_arrays_multi.asp
- array explanation
https://www.w3schools.com/cpp/cpp_arrays.asp
- C++ files
https://www.w3schools.com/cpp/cpp_files.asp
- for loop
https://www.w3schools.com/cpp/cpp_for_loop.asp
- how to read file with comma separated
<https://www.youtube.com/watch?v=lzYGiuX8QM>
- C++ full course
<https://www.youtube.com/watch?v=-TkoO8Z07hl&t=9694s>
- Quick sort (pivot principle) (algo)
<https://www.youtube.com/watch?v=Hoixgm4-P4M>
- output file streams
<https://www.youtube.com/watch?v=MMp4zV05R5k&t=552s>
- input file streams
<https://www.youtube.com/watch?v=QnCVoYnLIg8&t=61s>
- 2D array explanation
<https://www.youtube.com/watch?v=Vh4krbTnTAA>
- Bubble sort (algo)
https://www.youtube.com/watch?v=xli_FI7CuzA
- How to write to an output file in C++
<https://www.shcodes.io/athena/40458-how-to-write-to-an-output-file-in-c#:~:text=We%20create%20an%20object%20of,write%20data%20to%20the%20file.>
- Writing all program output to a txt file in C++
<https://stackoverflow.com/questions/574543/writing-all-program-output-to-a-txt-file-in-c>
- ChatGPT Prompt 1 History
<https://chat.openai.com/share/edf9d4cf-076a-45e5-83ab-d9564f2ce0f3>
- ChatGPT Prompt 2 History
<https://chat.openai.com/share/e4d4788f-2a07-415b-93f9-6d6a9409da71>
- ChatGPT Prompt 3 History
<https://chat.openai.com/share/9a96d3cd-4087-4552-ad5f-62be07e3b79d>
- WorldSkills Mobile Robotics
<https://worldskills.org/skills/id/9/>

- Task Distribution

177	LIM WIKY (LEADER)
178	LIM XING
179	MOHAMAD AYDIN BIN MOHD GHAZALI
180	28 AEESHA ERINA BINTI DAUD
181	EDWARD EE JIN HAO
182	LEE WEIXIAN

	Name	Matrix No.	Role	email	Contact
1	LIM WIKY	22305415	1, 2, 6	wikylim@student.usm.my	018 262 4913
2	LIM XING	22301447	5	limxing30@student.usm.my	017-4545260
3	MOHAMAD AYDIN BIN MOHD GHAZALI	22300017	2,3	aydin.ghaz@student.usm.my	
4	AEESHA ERINA BINTI DAUD	22301711	5	aeeshaerina@student.usm.my	010-6688767
5	EDWARD EE JIN HAO	22301440	3, 4	edwardeejinhao@student.usm.my	012-4447831
6	LEE WEI XIAN	22302757	4	leeweixian5561@student.usm.my	017-4252634

	Roles	Type	Description
1	Problem & solution seeker	Overlooker	Generate problem statement, propose method flow, prepare preliminary data
2	[Section 1] Input data	Algorithm	Propose solution and write code for inputting/sourcing data from .txt file data sets
3	[Section 2] Sorter	Algorithm	Propose solution and write code for sorting the (x, y) coordinate in array
4	[Section 3] Converter	Algorithm	Propose solution and write code for converting sorted data into matrix form
5	[Section 4] Output	Algorithm	Propose solution and write code for displaying raw data, processed data and final data in terminal (final data in matrix form), output .txt file
6	Juice mixer	Algorithm	Adjust, fix and combine all the codes (Section 1 to 4) into one main code
7	[Section 5] OpenCV (optional)	Algorithm	Write code for openCV template matching
8	Tester	Tester	Code inspection, test and validate code functionality

- // main.cpp (version 4.52 Vectorstruct)

/* EEE123 COMPUTER PROGRAMMING MINI PROJECT | SEMESTER 1 | SESSION 23/24

GROUP: 23

TITLE: Tabular Image Data to Matrix Data Converter (for OpenCV purposes)

VERSION: (4) Vectorstruct

REVISION: 4.52

MEMBERS DETAIL	MATRIC	ROLES	EMAIL	CONTACT
1 LIM WIKY	22305415	1, 2, 6	wikylim@student.usm.my	018 262 4913
2 LIM XING	22301447	5	limxing30@student.usm.my	017-4545260
3 MOHAMAD AYDIN BIN MOHD GHAZALI	22300017	2,3	aydin.ghaz@student.usm.my	
4 AEESHA ERINA BINTI DAUD	22301711	5	aeeshaerina@student.usm.my	010-6688767
5 EDWARD EE JIN HAO	22301440	3, 4	edwardeejinhao@student.usm.my	012-4447831
6 LEE WEI XIAN	22302757	4	leeweixian5561@student.usm.my	017-4252634

ROLES	TYPE	DESCRIPTION
1 Problem & solution seeker	Overlooker	Generate problem statement, propose method flow, prepare preliminary data
2 [Section 1] Input data	Algorithm	Propose solution and write code for inputting/sourcing data from .txt file data sets
3 [Section 2] Sorter	Algorithm	Propose solution and write code for sorting the (x, y) coordinate in array
4 [Section 3] Converter	Algorithm	Propose solution and write code for converting sorted data into matrix form
5 [Section 4] Output	Algorithm	Propose solution and write code for displaying raw data, processed data and final data in terminal (final data in matrix form), output .txt file
6 Juice mixer	Algorithm	Adjust, fix and combine all the codes (Section 1 to 4) into one main code
7 [Section 5] OpenCV (optional)	Algorithm	Write code for openCV template matching
8 Tester	Tester	Code inspection, test and validate code functionality

limitations of code:

- using file contents with the format " raw1>> [{0, 0}, {0, 0}, {0, 0}, ... {0, 0}]", any changes of the format of the input files will result in error of code
- works within the coordinates (x, y) = (100, 100) to (600, 600)
- size of data index must be according to raw data and accurate

*/

```

#ifdef __linux__ //DO NOT EDIT OR REMOVE
#define CATCH_CONFIG_RUNNER //DO NOT EDIT OR REMOVE
#include "catch.hpp" //DO NOT EDIT OR REMOVE
int runCatchTests(int argc, char* const argv[]){ //DO NOT EDIT OR REMOVE
    return Catch::Session().run(argc, argv);} //DO NOT EDIT OR REMOVE
#endif //DO NOT EDIT OR REMOVE

#include <iomanip>
#include <iostream>
#include <string>
#include <cmath>

```

```

#include <fstream>
#include <ctime>
#include "functions.hpp"

#include <vector>

#define cout std::cout
#define oout outputFile
#define endl std::endl
#define string std::string
#define vector std::vector

//----- V a r i a b l e s -----
const int threshold = 30;
string file1 = "raw1.txt";
string file2 = "raw2.txt";
string file3 = "raw3.txt";
string outputTxt = "output_v4.txt";

// for output
bool _toggleClear = 1;      // clean mode

int data[6][6] = {{0, 0, 0, 0, 0, 0},
                 {0, 0, 0, 0, 0, 0},
                 {0, 0, 0, 0, 0, 0},
                 {0, 0, 0, 0, 0, 0},
                 {0, 0, 0, 0, 0, 0},
                 {0, 0, 0, 0, 0, 0},
                 };

struct data {
    int type;
    int dataIndex;

    int xcoord[6] = {0, 0, 0, 0, 0, 0};
    int xJump[5] = {0, 0, 0, 0, 0};      // for debugging purpose
    int difference = 0;                  // for debugging purpose

    int mdata[6][6] = { {0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0},
                        {0, 0, 0, 0, 0, 0},
                        };

    void displayIndex(){
        cout << "raw" << type << " index: " << dataIndex << endl;
    }
}data1, data2, data3;

//-----

int main(int argc, char* const argv[]){      //DO NOT EDIT OR REMOVE
    #if __linux__                            //DO NOT EDIT OR REMOVE

```



```

    return runCatchTests(argc, argv);           //DO NOT EDIT OR REMOVE
#endif                                         //DO NOT EDIT OR REMOVE

//start here...

data1.type = 1;
data2.type = 2;
data3.type = 3;

//----- fetch raw data files -----
inputFunc(file1, file2, file3);

do{
    string yn;
    cout << "\nClean Mode? [Y/N]] > ";
    std::getline(std::cin, yn);
    if(yn=="Y" || yn=="y"){
        _toggleClear = 1;
        cout << "\nClean mode: ON" << endl;
        break;
    }
    else if(yn=="N" || yn=="n"){
        _toggleClear = 0;
        cout << "\nClean mode: OFF" << endl;
        break;
    }
    else{
        cout << "\nNo input given, clean mode off (default)" << endl;
        break;
    }
}
while(true);

data1.dataIndex = (countContiguousNumbers(file1)-1)/2;
int raw1[data1.dataIndex][2];
// Extract data from raw1.txt
extractData(file1, raw1, 0, data1.dataIndex);

data2.dataIndex = (countContiguousNumbers(file2)-1)/2;
int raw2[data2.dataIndex][2];
// Extract data from raw2.txt
extractData(file2, raw2, 0, data2.dataIndex);

data3.dataIndex = (countContiguousNumbers(file3)-1)/2;
int raw3[data3.dataIndex][2];
// Extract data from raw3.txt
extractData(file3, raw3, 0, data3.dataIndex);
//-----

//----- Process & append raw data -----

data1.displayIndex();
int dataV1[data1.dataIndex][2];           // declaring data1
copyData(dataV1, data1.dataIndex, raw1);
data1.dataIndex--;                       // after removing (0, 0), decrease the size of array by 1

```

```

data2.displayIndex();
int dataV2[data2.dataIndex][2];           // declaring data2
copyData(dataV2, data2.dataIndex, raw2);
data2.dataIndex--;                        // after removing (0, 0), decrease the size of array by 1

data3.displayIndex();
int dataV3[data3.dataIndex][2];           // declaring data3
copyData(dataV3, data3.dataIndex, raw3);
data3.dataIndex--;                        // after removing (0, 0), decrease the size of array by 1

//-----

//----- sort by x-coordinate in ascending order -----

sortX(data1.dataIndex, dataV1);
sortX(data2.dataIndex, dataV2);
sortX(data3.dataIndex, dataV3);

cout << "\n";
//-----

//----- Display data array -----
cout << "\n>data1" << endl;
displayDataArray(data1.dataIndex, dataV1);

cout << "\n>data2" << endl;
displayDataArray(data2.dataIndex, dataV2);

cout << "\n>data3" << endl;
displayDataArray(data3.dataIndex, dataV3);

cout << "\n";
//-----

//----- Cell element counter -----

cout << "----- R E P O R T 1 -----" << endl;
cout << "number of indexes found: " << data1.dataIndex << endl;
if(data1.dataIndex>0){                    // not empty set
    for(int i=0;i<data1.dataIndex;i++){
        int indx = int(round(static_cast<double>(dataV1[i][0])/100.0)-1);
        data1.xcoord[indx]++;
        if(i!=data1.dataIndex-1 && dataV1[i+1][0]-dataV1[i][0]>threshold){    // distinct value
            data1.difference++;
        }
    }
}

```

```

        cout << "found 1 difference! current difference: " << data1.difference << ", current index: " << i << ", index jump: " << i+1
<< endl;
        data1.xJump[data1.difference-1] = i+1;
    }
}
cout << "\nJump Index > { " ;
for(int i=0;i<6;i++){
    cout << data1.xJump[i] << " ";
}cout << "}" << endl;

cout << "\n{ " ;
for(int i=0;i<6;i++){
    cout << data1.xcoord[i] << "\t";
}cout << "}" << endl;
cout << "\n" << data1.difference << " distinct x-coordinates found!" << endl;

}
else{cout << "Empty data! No coordinates found!" << endl;}
cout << "-----\n" << endl;

cout << "----- R E P O R T 2 -----" << endl;
cout << "number of indexes found: " << data2.dataIndex << endl;
if(data2.dataIndex>0){ // not empty set
    for(int i=0;i<data2.dataIndex;i++){
        int indx = int(round(static_cast<double>(dataV2[i][0])/100.0)-1);
        data2.xcoord[indx]++;
        if(i!=data2.dataIndex-1 && dataV2[i+1][0]-dataV2[i][0]>threshold){ // distinct value
            data2.difference++;
            cout << "found 1 difference! current difference: " << data2.difference << ", current index: " << i << ", index jump: " << i+1
<< endl;
            data2.xJump[data2.difference-1] = i+1;
        }
    }
    cout << "\nJump Index > { " ;
    for(int i=0;i<6;i++){
        cout << data2.xJump[i] << " ";
    }cout << "}" << endl;

    cout << "\n{ " ;
    for(int i=0;i<6;i++){
        cout << data2.xcoord[i] << "\t";
    }cout << "}" << endl;
    cout << "\n" << data2.difference << " distinct x-coordinates found!" << endl;

}
else{cout << "Empty data! No coordinates found!" << endl;}
cout << "-----\n" << endl;

cout << "----- R E P O R T 3 -----" << endl;
cout << "number of indexes found: " << data3.dataIndex << endl;
if(data3.dataIndex>0){ // not empty set
    for(int i=0;i<data3.dataIndex;i++){
        int indx = int(round(static_cast<double>(dataV3[i][0])/100.0)-1);
        if(indx < 0){cout << "error!!!" << endl;}
        data3.xcoord[indx]++;
        if(i!=data3.dataIndex-1 && dataV3[i+1][0]-dataV3[i][0]>threshold){ // distinct value
            data3.difference++;

```

```

        cout << "found 1 difference! current difference: " << data3.difference << ", current index: " << i << ", index jump: " << i+1
<< endl;
        data3.xJump[data3.difference-1] = i+1;
    }
}
cout << "\nJump Index > { " ;
for(int i=0;i<6;i++){
    cout << data3.xJump[i] << " ";
}cout << "}" << endl;

cout << "\n{ " ;
for(int i=0;i<6;i++){
    cout << data3.xcoord[i] << "\t";
}cout << "}" << endl;
cout << "\n" << data3.difference << " distinct x-coordinates found!" << endl;

}
else{cout << "Empty data! No coordinates found!" << endl;}
cout << "-----\n" << endl;
//-----

//----- append y-coordinates into each respective column -----
vector<vector<vector<int>>> xcoord(3, vector<vector<int>>(6, vector<int>(1)));
int cumulativeIndex = 0;

for(int i=0;i<6;i++){
    xcoord[0][i].resize(data1.xcoord[i]);
    for(int j=0;j<data1.xcoord[i];j++){
        xcoord[0][i][j]=dataV1[j+cumulativeIndex][1];
    }
    cumulativeIndex += data1.xcoord[i];
}

cumulativeIndex = 0;
for(int i=0;i<6;i++){
    xcoord[1][i].resize(data2.xcoord[i]);
    for(int j=0;j<data2.xcoord[i];j++){
        xcoord[1][i][j]=dataV2[j+cumulativeIndex][1];
    }
    cumulativeIndex += data2.xcoord[i];
}

cumulativeIndex = 0;
for(int i=0;i<6;i++){
    xcoord[2][i].resize(data3.xcoord[i]);
    for(int j=0;j<data3.xcoord[i];j++){
        xcoord[2][i][j]=dataV3[j+cumulativeIndex][1];
    }
    cumulativeIndex += data3.xcoord[i];
}

//-----

```

```

//----- sort by y-coordinate in ascending order -----
if(data1.dataIndex>0){
    for(int i=0;i<6;i++){
        vectorSort(xcoord[0][i], data1.xcoord[i]);
        vectorappendMatrix(xcoord[0][i], data1.xcoord[i], i, data1.mdata);
    }
}
if(data2.dataIndex>0){
    for(int i=0;i<6;i++){
        vectorSort(xcoord[1][i], data2.xcoord[i]);
        vectorappendMatrix(xcoord[1][i], data2.xcoord[i], i, data2.mdata);
    }
}
if(data3.dataIndex>0){
    for(int i=0;i<6;i++){
        vectorSort(xcoord[2][i], data3.xcoord[i]);
        vectorappendMatrix(xcoord[2][i], data3.xcoord[i], i, data3.mdata);
    }
}

//-----

//----- Display appended y-coordinates -----
cout << "\n-----" << endl;
cout << " DISPLAYING LIST OF Y-COORDINATES FOR EACH RESPECTIVE X-AXIS " << endl;
cout << "-----";

cout << "\ndata1 y-coord: ";
for(int i=0;i<6;i++){
    cout << "\nx" << i+1 << "00A: ";
    for(int j=0;j<data1.xcoord[i];j++){
        cout << xcoord[0][i][j] << " ";
        // get xcoord y-coordinate for data1
        // x location = 100
    }
}

cout << "\ndata2 y-coord: ";
for(int i=0;i<6;i++){
    cout << "\nx" << i+1 << "00B: ";
    for(int j=0;j<data2.xcoord[i];j++){
        cout << xcoord[1][i][j] << " ";
        // get xcoord y-coordinate for data2
        // x location = 100
    }
}

cout << "\ndata3 y-coord: ";
for(int i=0;i<6;i++){
    cout << "\nx" << i+1 << "00C: ";
    for(int j=0;j<data3.xcoord[i];j++){
        cout << xcoord[2][i][j] << " ";
        // get xcoord y-coordinate for data2
        // x location = 100
    }
}

```

```

//-----
//----- Display matrix data in terminal-----
cout << "\n\n";

displayMatrix(data1.mdata, "R a w  1", 1);
setData(data1.mdata, 1);          // set data1 data into 1

displayMatrix(data2.mdata, "R a w  2", 1);
setData(data2.mdata, 2);          // set data2 data into 2

displayMatrix(data3.mdata, "R a w  3", 1);
setData(data3.mdata, 3);          // set data3 data into 3

// Add data1.mdata, data2.mdata, data3.mdata into Matrix
for(int i=0;i<6;i++){
    for(int j=0;j<6;j++){
        data[i][j] += data1.mdata[i][j];
        data[i][j] += data2.mdata[i][j];
        data[i][j] += data3.mdata[i][j];
    }
}
displayMatrix(data, "F i n a l", 1);

//-----
//----- Output file report generator-----
std::ofstream outputFile(outputTxt);
std::ifstream sourceFile1(file1);
std::ifstream sourceFile2(file2);
std::ifstream sourceFile3(file3);

// Vanity code, totally unnecessary
oout << "_____ " << endl;
oout<<"\n    T A B L E  T O  M A T R I X  R E P O R T" << endl;
oout << "_____ " << endl;
std::time_t currentTime = std::time(0);
std::tm* localTime = std::localtime(&currentTime);
const char* daysOfWeek[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
const char* monthAbbreviations[] = {
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
};
outputFile << "Time of report generation:\t"
    << std::setfill('0') << localTime->tm_mday
    << "-" << std::setfill('0') << monthAbbreviations[localTime->tm_mon]
    << "-" << std::setfill('0') << localTime->tm_year + 1900
    << " "
    << " " << std::setfill('0') << std::setw(2) << localTime->tm_hour
    << ":" << std::setfill('0') << std::setw(2) << localTime->tm_min
    << ":" << std::setfill('0') << std::setw(2) << localTime->tm_sec
    << " "
    << "(" << daysOfWeek[localTime->tm_wday] << ")" << endl;

outputFile << "Location of data:\t\t\t" << location << endl;
oout << "_____ \n" << endl;
// Write the array contents to the file

```

```

// Display raw data fed
string _line;
oout << "Input ";
while(std::getline(sourceFile1, _line)) {
    oout << _line << endl;
}
oout << "Input ";
while(std::getline(sourceFile2, _line)) {
    oout << _line << endl;
}
oout << "Input ";
while(std::getline(sourceFile3, _line)) {
    oout << _line << endl;
}
oout << "\n\n";
sourceFile1.close();
sourceFile2.close();
sourceFile3.close();

// Display raw data indices
oout << "raw1 Index: " << data1.dataIndex << endl;
oout << "raw2 Index: " << data2.dataIndex << endl;
oout << "raw3 Index: " << data3.dataIndex << endl;
oout << "\n\n";

// Display copied and sorted data (sort by x-coordinate)
oout << ">data1 (sorted by x-axis in ascending order)" << endl;
if(data1.dataIndex>0){
    for(int i=0;i<data1.dataIndex;i++){
        for(int j=0;j<2;j++){
            oout << dataV1[i][j] << " ";
        }
        oout << " | "; //"\n";
    }
}
else{oout << "Empty data!" << endl;}
oout << "\n\n";
oout << ">data2 (sorted by x-axis in ascending order)" << endl;
if(data2.dataIndex>0){
    for(int i=0;i<data2.dataIndex;i++){
        for(int j=0;j<2;j++){
            oout << dataV2[i][j] << " ";
        }
        oout << " | "; //"\n";
    }
}
else{oout << "Empty data!" << endl;}
oout << "\n\n";
oout << ">data3 (sorted by x-axis in ascending order)" << endl;
if(data3.dataIndex>0){
    for(int i=0;i<data3.dataIndex;i++){
        for(int j=0;j<2;j++){
            oout << dataV3[i][j] << " ";
        }
        oout << " | "; //"\n";
    }
}
}

```

```

else{oout << "Empty data!" << endl;}
oout << "\n\n";

// Display preliminary report
oout << "-----" << endl;
oout << "----- P R E L I M I N A R Y   R E P O R T -----" << endl;
if(data1.dataIndex>0){
oout << "----- R E P O R T   1 -----" << endl;
oout << "number of indexes found: " << data1.dataIndex << endl;
oout << "\t";
for(int i=0;i<6;i++){
oout << data1.xcoord[i] << "\t";
}oout << "]" << endl;
oout << " " << data1.difference << " distinct x-coordinates found!" << endl;
}
else{oout << "Empty data! No coordinates found!" << endl;}
oout << "-----\n" << endl;

if(data2.dataIndex>0){
oout << "----- R E P O R T   2 -----" << endl;
oout << "number of indexes found: " << data2.dataIndex << endl;
oout << "\t";
for(int i=0;i<6;i++){
oout << data2.xcoord[i] << "\t";
}oout << "]" << endl;
oout << " " << data2.difference << " distinct x-coordinates found!" << endl;
}
else{oout << "Empty data! No coordinates found!" << endl;}
oout << "-----\n" << endl;

if(data3.dataIndex>0){
oout << "----- R E P O R T   3 -----" << endl;
oout << "number of indexes found: " << data3.dataIndex << endl;
oout << "\t";
for(int i=0;i<6;i++){
oout << data3.xcoord[i] << "\t";
}oout << "]" << endl;
oout << " " << data3.difference << " distinct x-coordinates found!" << endl;
}
else{oout << "Empty data! No coordinates found!" << endl;}
oout << "-----\n" << endl;

// Displaying y-coordinates
oout << "\n-----" << endl;
oout << " DISPLAYING LIST OF Y-COORDINATES FOR EACH RESPECTIVE X-AXIS " << endl;
oout << "-----";
oout << "\ndata1 y-coord: ";
for(int i=0;i<6;i++){
oout << "\nx"<< i+1 << "00A: ";
for(int j=0;j<data1.xcoord[i];j++){ // get xcoord y-coordinate for data1
oout << xcoord[0][i][j] << " "; // x location = 100
}
}
oout << "\ndata2 y-coord: ";
for(int i=0;i<6;i++){
oout << "\nx"<< i+1 << "00B: ";
for(int j=0;j<data2.xcoord[i];j++){ // get xcoord y-coordinate for data2
oout << xcoord[1][i][j] << " "; // x location = 100
}
}

```



```

    }
}
oout << "\ndata3 y-coord: ";
for(int i=0;i<6;i++){
    oout << "\nx" << i+1 << "00C: ";
    for(int j=0;j<data3.xcoord[i];j++){
        oout << xcoord[2][i][j] << " "; // get xcoord y-coordinate for data2
        // x location = 100
    }
}
oout << "\n-----" << endl;
oout << "\n\n";

// Display for Matrix form
oout << "----- M a t r i x   D i s p l a y -----" << endl;
oout << "Clean mode: ";
if(_toggleClear){oout << "ON";}else{oout << "OFF";}oout << "\n\n";
oout << "----- M a t r i x   d a t a   f o r   d a t a   1 ----- \n";
oout << "
" << endl;
oout << "\tR1\tR2\tR3\tR4\tR5\tNS" << endl;
for(int j=0;j<6;j++){
    if(j<4){
        oout << "D\t";
    }
    else{oout << "R\t";}
    for(int i=0;i<6;i++){
        if(data1.mdata[j][i]==0 && _toggleClear){oout << "\t";}
        else{oout << data1.mdata[j][i] << "\t";}
    }
    oout << "\n";
} oout << "
" << endl;
//oout << "\n----- \n";

oout << "----- M a t r i x   d a t a   f o r   d a t a   2 ----- \n";
oout << "
" << endl;
oout << "\tR1\tR2\tR3\tR4\tR5\tNS" << endl;
for(int j=0;j<6;j++){
    if(j<4){
        oout << "D\t";
    }
    else{oout << "R\t";}
    for(int i=0;i<6;i++){
        if(data2.mdata[j][i]==0 && _toggleClear){oout << "\t";}
        else{oout << data2.mdata[j][i] << "\t";}
    }
    oout << "\n";
} oout << "
" << endl;
//oout << "\n----- \n";

oout << "----- M a t r i x   d a t a   f o r   d a t a   3 ----- \n";
oout << "
" << endl;
oout << "\tR1\tR2\tR3\tR4\tR5\tNS" << endl;
for(int j=0;j<6;j++){
    if(j<4){
        oout << "D\t";
    }
    else{oout << "R\t";}
    for(int i=0;i<6;i++){
        if(data3.mdata[j][i]==0 && _toggleClear){oout << "\t";}

```

```

        else{oout << data3.mdata[j][i] << "\t";}
    }
    oout << "\n";
} oout << "
//oout << "\n-----\n";

oout << "----- F i n a l   M a t r i x ----- \n";
oout << "
oout << "\tR1\tR2\tR3\tR4\tR5\tNS" << endl;
for(int j=0;j<6;j++){
    if(j<4){
        oout << "D\t";
    }
    else{oout << "R\t";}
    for(int i=0;i<6;i++){
        if(data[j][i]==0 && _toggleClear){oout << "\t";}
        else{oout << data[j][i] << "\t";}
    }
    oout << "\n";
} oout << "
//oout << "\n-----\n";
oout << "\n";

// bye
oout << "\t\tE N D   O F   R E P O R T" << endl;
oout << "\n-----\n";
oout << "\n";
oout << "          T h a n k   y o u" << endl;
oout << "\n          H a v e   a   g o o d   d a y   :) \n" << endl;
oout << "\n-----\n";

// Close the file
outputFile.close();

cout << "\nData has been written to output.txt\nReport generated" << endl;

closing();

return 0;

}

```

```

- // functions.hpp
// welcome...
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cctype>

#define cout std::cout
#define endl std::endl
#define string std::string
#define vector std::vector

string location;
string version = "v4.52 Vectorstruct";
template <std::size_t Rows, std::size_t Cols>
int getIndex(int(&array)[Rows][Cols]){
    return Rows;
}

//-----

void inputFunc(string& _file1, string& _file2, string& _file3){
    string filePath;
    cout << "\n-----\n";
    cout << "\n";
    cout << "          W E L C O M E\n" << endl;
    cout << " T a b u l a r d a t a t o m a t r i x c o n v e r t e r\n" << endl;
    cout << "\nVersion: " << version;
    cout << "\n-----\n";
    cout << "\nPlease enter reference folder location.\n\nEg : \"C:\\This PC\\EEE\\Reference\\Set 5\"";
    do {
        _file1 = "raw1.txt";
        _file2 = "raw2.txt";
        _file3 = "raw3.txt";
        cout << "\nFile location : ";
        std::getline(std::cin, filePath);
        bool local = (filePath!="LOCAL") && (filePath!="local") && (filePath!="Local");
        if(local){
            _file1 = filePath + "\\raw1.txt";
            _file2 = filePath + "\\raw2.txt";
            _file3 = filePath + "\\raw3.txt";
        }
        else{cout << "\nSearching local folder..." << endl;}

        std::ifstream input_file1(_file1);
        std::ifstream input_file2(_file2);

```

```

std::ifstream input_file3(_file3);
//to check either file location is valid or not
if(input_file1.is_open() && input_file2.is_open() && input_file3.is_open()){
    location = filePath;
    break;
}
if(!input_file1.is_open()){
    cout << " Missing raw1.txt!" << endl;
}
if(!input_file2.is_open()){
    cout << " Missing raw2.txt!" << endl;
}
if(!input_file3.is_open()){
    cout << " Missing raw3.txt!" << endl;
}
if(!local){
    std::cerr << "\nUnable to find data in local folder\nincomplete/missing data, try again.\n";
}
else{
    std::cerr << "\nUnable to find data in specified folder - '" << filePath << "'\nincomplete/missing
data, try again.\n";
}

}
while (true);
}

int countContiguousNumbers(const string& filename) {
    std::ifstream inputFile(filename);

    if (!inputFile.is_open()) {
        std::cerr << "Error opening file: " << filename << endl;
        return -1;
    }

    char currentChar;
    bool inNumber = false;
    int count = 0;

    while (inputFile.get(currentChar)) {
        if (std::isdigit(currentChar)) {
            // If the character is a digit, and we are not already in a number, start counting a new number
            if (!inNumber) {
                inNumber = true;
                count++;
            }
        } else {
            // If the character is not a digit, mark the end of the current number

```

```

        inNumber = false;
    }
}

inputFile.close();
return count;
}

void extractData(const string& fileName, int array[][2], int count, int maxPoints) {
    // Open the file
    std::ifstream inputFile(fileName);

    // Check if the file is open
    if (!inputFile.is_open()) {
        std::cerr << "Error opening the file: " << fileName << endl;
        return;
    }

    // Read the file content
    string line;

    while (std::getline(inputFile, line)) {

        size_t start = line.find("(");
        size_t end = line.find(")");

        while (start != string::npos && end != string::npos) {
            // Extracting the numbers between parentheses
            string point = line.substr(start + 1, end - start - 1);

            // Extracting x and y coordinates
            int x, y;
            sscanf(point.c_str(), "%d, %d", &x, &y);

            // Storing the values in the array
            if (count < maxPoints) {
                array[count][0] = x;
                array[count][1] = y;
                count++;
            }

            // Move to the next point
            start = line.find("(", end);
            end = line.find(")", start);
        }
    }
}

```

```

    // Close the file
    inputFile.close();
}

//-----

void copyData(int arr1[][2], int rowSize, int rawArr[][2]){
    for(int i=0;i<rowSize;i++){
        for(int j=0;j<2;j++){
            arr1[i][j] = rawArr[i+1][j];    // skip copying (0, 0)
        }
    }
}

void sortX(int size, int _data[][2]){
    for(int i=1;i<size;i++){
        for(int j=0;j<size-1;j++){
            if(_data[j+1][0]<_data[j][0]){
                int tempdata[2];
                tempdata[0]=_data[j][0];
                tempdata[1]=_data[j][1];

                _data[j][0]=_data[j+1][0];
                _data[j][1]=_data[j+1][1];

                _data[j+1][0]=tempdata[0];
                _data[j+1][1]=tempdata[1];
            }
        }
    }
}

void _sort(int arr[], int size){
    for(int i=1;i<size;i++){
        for(int j=0;j<size-1;j++){
            if(arr[j]>arr[j+1]){
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

void vectorSort(vector<int>& arr, int size) {
    for(int i = 1;i<size;i++) {
        for (int j=0;j<size-1;j++) {
            if (arr[j]>arr[j+1]) {

```

```

        std::swap(arr[j],arr[j+1]);
    }
}
}

void displayDataArray(int size, int _data[][2]){
    if(size>0){
        for(int i=0;i<size;i++){
            for(int j=0;j<2;j++){
                cout << _data[i][j] << " ";
            }
            cout << "\n";
        }
    }
    else{cout << "Empty data!" << endl;}
}

//-----

void appendMatrix(int arr[], int size, int xloc, int _data[6][6]){    // (input data, input data, input data,
output data)
//x100A
//int ycoord1[6] = {0, 0, 0, 0, 0, 0};
//int ydifference = 1;
for(int i=0;i<size;i++){
    //ycoord1[indx]++;
    if(arr[i]==0){                //filter out (0, 0)
        cout << "EMPTY!!" << endl;
    }
    else{
        int indx = int((round(static_cast<double>(arr[i])/100.0)-1));
        _data[indx][xloc]++;
    }
}
}

void vectorappendMatrix(const vector<int>& arr, int size, int xloc, int _data[6][6]) {
    for (int i=0;i<size;i++) {
        if (arr[i] == 0) {
            cout << "EMPTY!!" << endl;
        } else {
            int indx = int((round(static_cast<double>(arr[i])/100.0)-1));
            if (indx>=0&&indx<6) { // Check if indx is within bounds
                _data[indx][xloc]++;
            } else {
                cout << "Index out of bounds: " << indx << endl;
            }
        }
    }
}

```

```

    }
}
}

//-----

void displayMatrix(int _data[6][6], string _title, bool toggleClear){
    cout << "----- M a t r i x   d a t a   f o r   " << _title << " -----\\n" << endl;
    cout << "\\tR1\\tR2\\tR3\\tR4\\tR5\\tNS" << endl;
    for(int j=0;j<6;j++){
        if(j<4){
            cout << "D\\t";
        }
        else{cout << "R\\t";}
        for(int i=0;i<6;i++){
            if(_data[j][i]==0 && toggleClear){cout << "\\t";}
            else{cout << _data[j][i] << "\\t";}
        }
        cout << "\\n";
    }
    cout << "\\n-----\\n";

}










void setData(int _data[6][6], int target){
    for(int i=0;i<6;i++){
        for(int j=0;j<6;j++){
            if(_data[i][j]!=0){
                _data[i][j]=target;
            }
        }
    }
}

void closing(){
    cout << "\\n-----\\n";
    cout << "\\n";
    cout << "          T h a n k   y o u" << endl;
    cout << "\\n          H a v e   a   g o o d   d a y   : ) \\n" << endl;
    cout << "\\n-----\\n";
}

//-----

```


- Input data (Image set)

	R1	R2	R3	R4	R5	NS
DELIVERY						
						
						
						
RET						
						

- Input data (Raw data coordinates)

raw1>>

```
[(0, 0), (500, 97), (499, 98), (500, 98), (501, 98), (498, 99), (499, 99), (500, 99), (501, 99), (502, 99), (497, 100), (498, 100), (499, 100), (500, 100), (501, 100), (502, 100), (503, 100), (498, 101), (499, 101), (500, 101), (501, 101), (502, 101), (499, 102), (500, 102), (501, 102), (500, 103), (400, 397), (399, 398), (400, 398), (401, 398), (398, 399), (399, 399), (400, 399), (401, 399), (402, 399), (397, 400), (398, 400), (399, 400), (400, 400), (401, 400), (402, 400), (403, 400), (398, 401), (399, 401), (400, 401), (401, 401), (402, 401), (399, 402), (400, 402), (401, 402), (400, 403), (100, 497), (99, 498), (100, 498), (101, 498), (98, 499), (99, 499), (100, 499), (101, 499), (102, 499), (97, 500), (98, 500), (99, 500), (100, 500), (101, 500), (102, 500), (103, 500), (98, 501), (99, 501), (100, 501), (101, 501), (102, 501), (99, 502), (100, 502), (101, 502), (100, 503)]
```

raw2>>

```
[(0, 0), (400, 597), (600, 597), (399, 598), (400, 598), (401, 598), (599, 598), (600, 598), (601, 598), (398, 599), (399, 599), (400, 599), (401, 599), (402, 599), (598, 599), (599, 599), (600, 599), (601, 599), (602, 599), (397, 600), (398, 600), (399, 600), (400, 600), (401, 600), (402, 600), (403, 600), (597, 600), (598, 600), (599, 600), (600, 600), (601, 600), (602, 600), (603, 600), (398, 601), (399, 601), (400, 601), (401, 601), (402, 601), (598, 601), (599, 601), (600, 601), (601, 601), (602, 601), (399, 602), (400, 602), (401, 602), (599, 602), (600, 602), (601, 602), (400, 603), (600, 603)]
```

```
raw3>> [(0, 0), (399, 98), (400, 98), (401, 98), (398, 99), (399, 99), (400, 99), (401, 99), (402, 99), (398, 100), (399, 100), (400, 100), (401, 100), (402, 100), (398, 101), (399, 101), (400, 101), (401, 101), (402, 101), (399, 102), (400, 102), (401, 102), (499, 198), (500, 198), (501, 198), (498, 199), (499, 199), (500, 199), (501, 199), (502, 199), (498, 200), (499, 200), (500, 200), (501, 200), (502, 200), (498, 201), (499, 201), (500, 201), (501, 201), (502, 201), (499, 202), (500, 202), (501, 202), (399, 298), (400, 298), (401, 298), (398, 299), (399, 299), (400, 299), (401, 299), (402, 299), (398, 300), (399, 300), (400, 300), (401, 300), (402, 300), (398, 301), (399, 301), (400, 301), (401, 301), (402, 301), (399, 302), (400, 302), (401, 302), (599, 398), (600, 398), (601, 398), (598, 399), (599, 399), (600, 399), (601, 399), (602, 399), (598, 400), (599, 400), (600, 400), (601, 400), (602, 400), (598, 401), (599, 401), (600, 401), (601, 401), (602, 401), (599, 402), (600, 402), (601, 402)]
```

- Output data (Results in output_v4.txt)

TABLE TO MATRIX REPORT

Time of report generation: 29-Jan-2024 06:35:51 (Monday)

Location of data: local

Input raw1>>

[(0, 0), (500, 97), (499, 98), (500, 98), (501, 98), (498, 99), (499, 99), (500, 99), (501, 99), (502, 99), (497, 100), (498, 100), (499, 100), (500, 100), (501, 100), (502, 100), (503, 100), (498, 101), (499, 101), (500, 101), (501, 101), (502, 101), (499, 102), (500, 102), (501, 102), (500, 103), (400, 397), (399, 398), (400, 398), (401, 398), (398, 399), (399, 399), (400, 399), (401, 399), (402, 399), (397, 400), (398, 400), (399, 400), (400, 400), (401, 400), (402, 400), (403, 400), (398, 401), (399, 401), (400, 401), (401, 401), (402, 401), (399, 402), (400, 402), (401, 402), (400, 403), (100, 497), (99, 498), (100, 498), (101, 498), (98, 499), (99, 499), (100, 499), (101, 499), (102, 499), (97, 500), (98, 500), (99, 500), (100, 500), (101, 500), (102, 500), (103, 500), (98, 501), (99, 501), (100, 501), (101, 501), (102, 501), (99, 502), (100, 502), (101, 502), (100, 503)]

Input raw2>>

[(0, 0), (400, 597), (600, 597), (399, 598), (400, 598), (401, 598), (599, 598), (600, 598), (601, 598), (398, 599), (399, 599), (400, 599), (401, 599), (402, 599), (598, 599), (599, 599), (600, 599), (601, 599), (602, 599), (397, 600), (398, 600), (399, 600), (400, 600), (401, 600), (402, 600), (403, 600), (597, 600), (598, 600), (599, 600), (600, 600), (601, 600), (602, 600), (603, 600), (398, 601), (399, 601), (400, 601), (401, 601), (402, 601), (598, 601), (599, 601), (600, 601), (601, 601), (602, 601), (399, 602), (400, 602), (401, 602), (599, 602), (600, 602), (601, 602), (400, 603), (600, 603)]

Input raw3>>

[(0, 0), (399, 98), (400, 98), (401, 98), (398, 99), (399, 99), (400, 99), (401, 99), (402, 99), (398, 100), (399, 100), (400, 100), (401, 100), (402, 100), (398, 101), (399, 101), (400, 101), (401, 101), (402, 101), (399, 102), (400, 102), (401, 102), (499, 198), (500, 198), (501, 198), (498, 199), (499, 199), (500, 199), (501, 199), (502, 199), (498, 200), (499, 200), (500, 200), (501, 200), (502, 200), (498, 201), (499, 201), (500, 201), (501, 201), (502, 201), (499, 202), (500, 202), (501, 202), (399, 298), (400, 298), (401, 298), (398, 299), (399, 299), (400, 299), (401, 299), (402, 299), (398, 300), (399, 300), (400, 300), (401, 300), (402, 300), (398, 301), (399, 301), (400, 301), (401, 301), (402, 301), (399, 302), (400, 302), (401, 302), (599, 398), (600, 398), (601, 398), (598, 399), (599, 399), (600, 399), (601, 399), (602, 399), (598, 400), (599, 400), (600, 400), (601, 400), (602, 400), (598, 401), (599, 401), (600, 401), (601, 401), (602, 401), (599, 402), (600, 402), (601, 402)]

raw1 Index: 75

raw2 Index: 50

raw3 Index: 84

>data1 (sorted by x-axis in ascending order)

97 500 | 98 499 | 98 500 | 98 501 | 99 498 | 99 499 | 99 500 | 99 501 | 99 502 | 100 497 | 100 498 | 100 499 | 100 500
| 100 501 | 100 502 | 100 503 | 101 498 | 101 499 | 101 500 | 101 501 | 101 502 | 102 499 | 102 500 | 102 501 | 103
500 | 397 400 | 398 399 | 398 400 | 398 401 | 399 398 | 399 399 | 399 400 | 399 401 | 399 402 | 400 397 | 400 398 |
400 399 | 400 400 | 400 401 | 400 402 | 400 403 | 401 398 | 401 399 | 401 400 | 401 401 | 401 402 | 402 399 | 402 400
| 402 401 | 403 400 | 497 100 | 498 99 | 498 100 | 498 101 | 499 98 | 499 99 | 499 100 | 499 101 | 499 102 | 500 97 |
500 98 | 500 99 | 500 100 | 500 101 | 500 102 | 500 103 | 501 98 | 501 99 | 501 100 | 501 101 | 501 102 | 502 99 | 502
100 | 502 101 | 503 100 |

>data2 (sorted by x-axis in ascending order)

397 600 | 398 599 | 398 600 | 398 601 | 399 598 | 399 599 | 399 600 | 399 601 | 399 602 | 400 597 | 400 598 | 400 599
| 400 600 | 400 601 | 400 602 | 400 603 | 401 598 | 401 599 | 401 600 | 401 601 | 401 602 | 402 599 | 402 600 | 402
601 | 403 600 | 597 600 | 598 599 | 598 600 | 598 601 | 599 598 | 599 599 | 599 600 | 599 601 | 599 602 | 600 597 |
600 598 | 600 599 | 600 600 | 600 601 | 600 602 | 600 603 | 601 598 | 601 599 | 601 600 | 601 601 | 601 602 | 602 599
| 602 600 | 602 601 | 603 600 |

>data3 (sorted by x-axis in ascending order)

398 99 | 398 100 | 398 101 | 398 299 | 398 300 | 398 301 | 399 98 | 399 99 | 399 100 | 399 101 | 399 102 | 399 298 |
399 299 | 399 300 | 399 301 | 399 302 | 400 98 | 400 99 | 400 100 | 400 101 | 400 102 | 400 298 | 400 299 | 400 300 |

400 301 | 400 302 | 401 98 | 401 99 | 401 100 | 401 101 | 401 102 | 401 298 | 401 299 | 401 300 | 401 301 | 401 302 |
 402 99 | 402 100 | 402 101 | 402 299 | 402 300 | 402 301 | 498 199 | 498 200 | 498 201 | 499 198 | 499 199 | 499 200
 | 499 201 | 499 202 | 500 198 | 500 199 | 500 200 | 500 201 | 500 202 | 501 198 | 501 199 | 501 200 | 501 201 | 501
 202 | 502 199 | 502 200 | 502 201 | 598 399 | 598 400 | 598 401 | 599 398 | 599 399 | 599 400 | 599 401 | 599 402 |
 600 398 | 600 399 | 600 400 | 600 401 | 600 402 | 601 398 | 601 399 | 601 400 | 601 401 | 601 402 | 602 399 | 602 400
 | 602 401 |

 ----- PRELIMINARY REPORT -----

----- REPORT 1 -----

number of indexes found: 75

[25 0 0 25 25 0]

2 distinct x-coordinates found!

----- REPORT 2 -----

number of indexes found: 50

[0 0 0 25 0 25]

1 distinct x-coordinates found!

----- REPORT 3 -----

number of indexes found: 84

[0 0 0 42 21 21]

2 distinct x-coordinates found!

 ----- DISPLAYING LIST OF Y-COORDINATES FOR EACH RESPECTIVE X-AXIS -----

data1 y-coord:

x100A: 497 498 498 498 499 499 499 499 500 500 500 500 500 500 501 501 501 501 502 502 502 503

x200A:

x300A:

x400A: 397 398 398 398 399 399 399 399 400 400 400 400 400 400 401 401 401 401 401 402 402 402 403

x500A: 97 98 98 98 99 99 99 99 100 100 100 100 100 100 101 101 101 101 101 102 102 102 103

x600A:

data2 y-coord:

x100B:

x200B:

x300B:

x400B: 597 598 598 598 599 599 599 599 600 600 600 600 600 600 601 601 601 601 601 602 602 602 603

x500B:

x600B: 597 598 598 598 599 599 599 599 600 600 600 600 600 600 601 601 601 601 601 602 602 602 603

data3 y-coord:

x100C:

x200C:

x300C:

x400C: 98 98 98 99 99 99 99 100 100 100 100 100 101 101 101 101 101 102 102 102 298 298 298 299 299 299 299 300

300 300 300 300 301 301 301 301 301 302 302 302

x500C: 198 198 198 199 199 199 199 200 200 200 200 200 201 201 201 201 201 202 202 202

x600C: 398 398 398 399 399 399 399 400 400 400 400 400 401 401 401 401 401 402 402 402

----- Matrix Display -----

Clean mode: ON

----- Matrix data for data 1 -----

	R1	R2	R3	R4	R5	NS
D					1	
D						
D						
D				1		
R	1					
R						

----- Matrix data for data 2 -----

	R1	R2	R3	R4	R5	NS
D						
D						
D						
D						
R						
R				2		2

----- Matrix data for data 3 -----

	R1	R2	R3	R4	R5	NS
D				3		
D					3	
D				3		
D						3
R						
R						

----- Final Matrix -----

	R1	R2	R3	R4	R5	NS
D				3	1	
D					3	
D				3		
D				1		3
R	1					
R				2		2

END OF REPORT

Thank you

Have a good day :)

End of report...