Name & Student ID: Dominic Harvey 20165856

Title: Software Project System Manual

Software Description

This software is designed to take an unchanging shape data file of coordinates, read it into memory and then take user input for a sequence of said shapes to draw on a grid. The software will start with opening a 'ShapeStrokeData.txt' file to read the total number of shapes – after this, it initialises a array of structs equal to this containing variables for the shape name, number of instructions and x, y and z values (where z represents pen position). It then initialises a struct called the command array, which contains variables for the number of shapes to draw, a flag for drawing the grid, x and y grid values, a 'shape marker' number, and the name of the shape (used to determine the shape marker needed).

After these have been initialised, the total number of shapes and a pointer to the array of structs are handed to a function that reads in the shape data. As this file has an established format, it uses the SHAPE_NAME N format (where N is the number of instructions for that shape) to fill a member of the struct array with the name and number of instructions, followed by for loop to fill the x y and z values. After this, it moves to the next struct member, and repeats this a number of times equal to the total shapes and closes the file.

Assuming this was successful, the main function then calls a function for defining the command array, passing it the total number of shapes and then the command array struct. It first prompts the user to the drawing sequence file they would like to open (and allows them to rerun the code with different files without having to recompile). It then gains the size of this file, and if necessary, it give the draw grid flag a high value. After this, it uses a for loop to fill the command array with x and y grid locations and the name of the shape going there. It then uses the name of the shape to assign a shape marker value for easier directing later. If a shape present in the data file but not known by name currently is detected, it will give it a new marker so that it can still be drawn. At the end of this function, the file is closed, and the command array is populated.

Assuming this was successful, the main function then calls the G-Code generation function, being handed the buffer, the command array, and the array of structs containing the shape data. This function initialises the pen properties and sends them to the printer. If the grid draw flag was high, it then draws the grid. After this, it runs through a for loop of each instruction in the command array, calls the correct shape from the array of structs based on the shape marker, and then runs through a for loop of sending the scaled and offset G-Code to the machine one at a time. It repeats this for each instruction, then zeroes the pen and the code ends and is ready for being reused. There are optional flags to switch from sending the G-Code one line at a time to generate a whole text file of the instructions – this is useful for diagnostic purposes, but should not be used when sending to the robot, as it causes an overflow that hampers it's ability to draw the correct sequence of moves.

Each function contains error trapping to indicate when a failed input has gone through.

Project Files

- System Manual (this document)
- main.c – the main code
- ShapeStrokeData.txt – the shape data provided
- DrawShapes.txt, TestShapes1.txt, TestShapes2.txt – examples of sequences to draw
- rs232.c/rs232.h and serial.c/serial.h – port information for sending to robot
- gpl-3.0.txt – licencing info
- RobotWriter4.0.code-workspace
- RS-232 for Linux and Windows pdf - port info for linux
- GCode.txt – example file for writing G-Code for based on diagnostic capabilities

Functions

Example (remove before submission)

Short description (one or two lines)

int TemperatureConversion(int InputTemp, float OutputTemp)*

Parameters:

InputTemp – input temperature in degrees C

OutputTemp – pointer to return output temperature in degrees F

Return value – returns 1 if successful, 0 if failed

A function for reading in the Shape Stroke Data into memory

- int ReadShapeDataFunction(int totalShapes, struct ShapeCoordSystem *);
    - totalShapes – the total number of shapes in the stroke data file
    - ShapeCoordSystem * - pointer to the array of structs relating to shape stroke data
    - Return value – 1 if successful, 2+ for failure cases

A function for reading in the Drawing instructions

- int DrawShapesFunction(int totalShapes, struct CommandArray *);
    - totalShapes – the total number of shapes in the stroke data file
    - CommandArray * - pointer to the command array struct
    - Return value – 1 if successful, 2+ for failure cases

A function for generating and sending the G-Code

- int GCodeGenerationFunction(char buffer[100], struct CommandArray *, struct ShapeCoordSystem *);
    - buffer – the immediate info being sent to the robot
    - CommandArray * - pointer to the command array
    - ShapeCoordSystem * - pointer to the array of structs relating to shape stroke data
    - Return value – 1 if successful, 2+ if failure cases

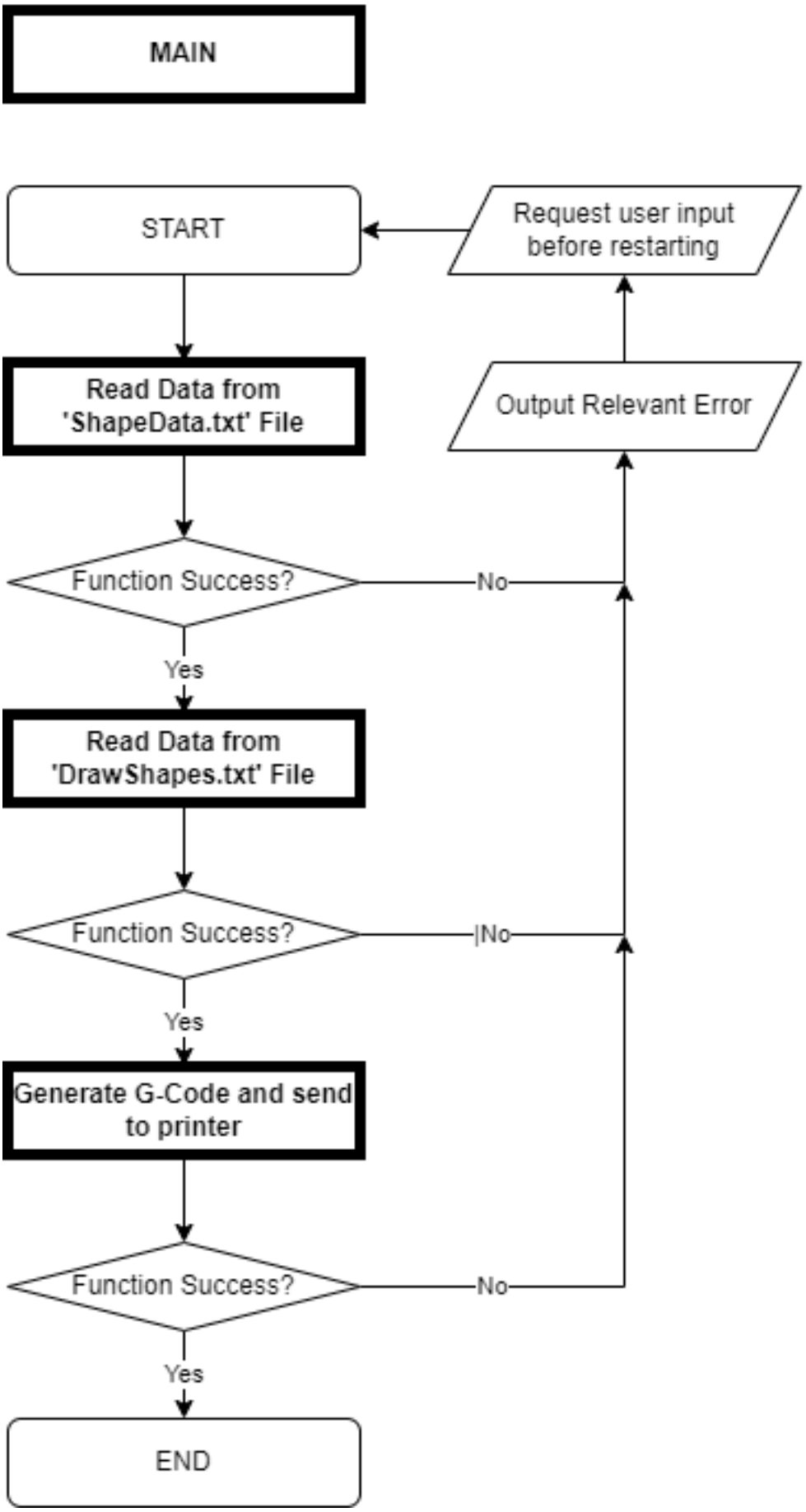
Key Data Items

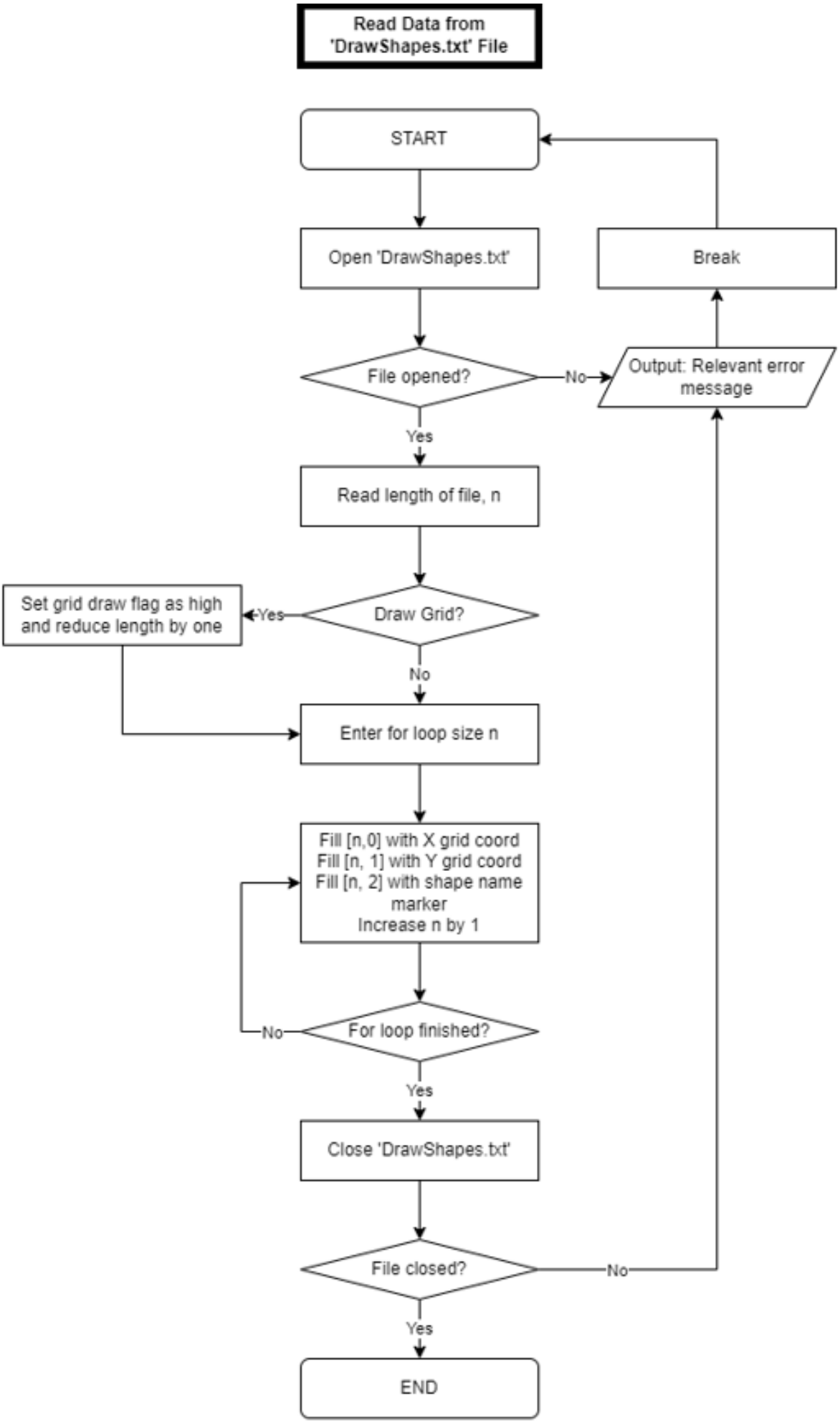
Name	Data type	Rationale
ShapeCoordSystem	Struct (array) char name int instructions int x, y, z	This contains is how shape data is saved to memory – it takes their x and y instructions, the state of the pen and its name to hold the information needed for the command array to generate into G-Code
CommandArray	Struct int numberOfCommands int drawFlag int xGrid, yGrid, shapeMarker char shape	This contains similar info to the ShapeCoordSystem structs, but relates to directing movement of the pen to the correct location on the grid, as well as holding a pointer to the correct shape data struct. It also contains info about enabling grid drawing.
totalShapes	int	While this is information that can technically be held in one of the structs, its more helpful to keep it outside given how often is called when the whole struct isn't needed. It is the total number of shapes read in from the shape stroke data file.
Function returns	int	Used in a switch case to know if the function succeeded or failed, and the cause of failure.

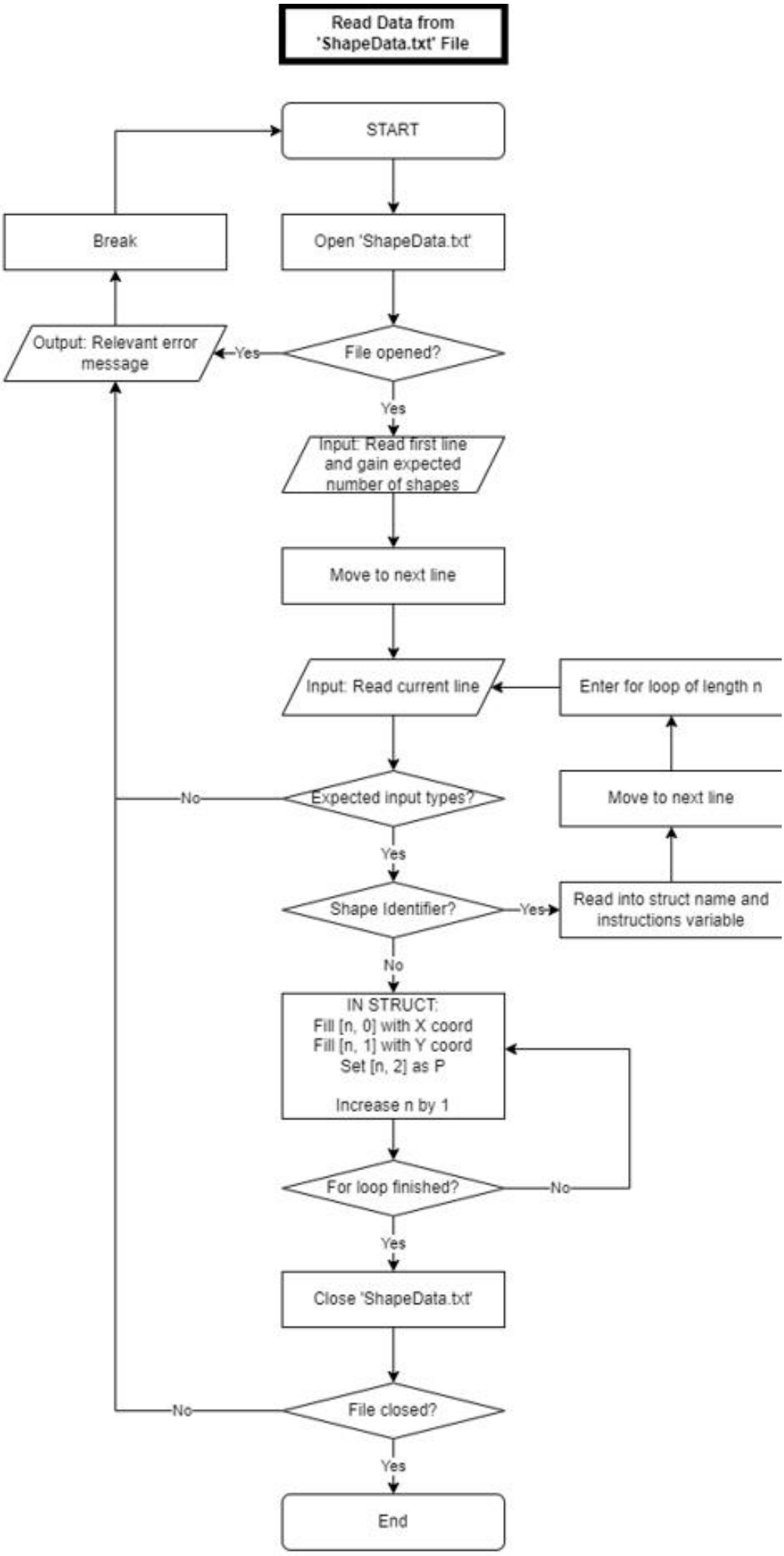
Testing Information

Function	Test Case	Test Data	Expected Output
Main	Function failure	Monitor Return values	Error message to user, highlighting which function failed
ReadShapeDataFunction/ DrawShapesFunction	Wrong file is given/file doesn't open	Returns 2 during file check	Error message to user, break from function
ReadShapeDataFunction/ DrawShapesFunction	File contains wrong data	Check if files contain characters and numbers, returns integer if poor formatting	Error message to user, break from function
ReadShapeDataFunction/ DrawShapesFunction	File does not close	Return 3 for function	Display error message to user, break from function, close all files

Flowchart(s)


MAIN
START
Request user input before restarting
Read Data from 'ShapeData.txt' File
Output Relevant Error
Function Success?
No
Yes
Read Data from 'DrawShapes.txt' File
Function Success?
No
Yes
Generate G-Code and send to printer
Function Success?
No
Yes
END


Read Data from 'DrawShapes.txt' File
START
Open 'DrawShapes.txt'
File opened?
No
Output: Relevant error message
Break
Yes
Read length of file, n
Draw Grid?
Yes
Set grid draw flag as high and reduce length by one
No
Enter for loop size n
Fill [n,0] with X grid coord
Fill [n, 1] with Y grid coord
Fill [n, 2] with shape name marker
Increase n by 1
For loop finished?
No
Yes
Close 'DrawShapes.txt'
File closed?
No
Yes
END


Read Data from 'ShapeData.txt' File
START
Open 'ShapeData.txt'
File opened?
Yes
Output: Relevant error message
Break
Input: Read first line and gain expected number of shapes
Move to next line
Input: Read current line
Enter for loop of length n
Expected input types?
No
Move to next line
Yes
Shape Identifier?
Yes
Read into struct name and instructions variable
No
IN STRUCT:
Fill [n, 0] with X coord
Fill [n, 1] with Y coord
Set [n, 2] as P
Increase n by 1
For loop finished?
No
Yes
Close 'ShapeData.txt'
File closed?
No
Yes
End


Generate G-Code and send to printer
START
Draw Grid and zero pen
Input: Command array and shape data
Get location from command array
Get shape array pointer from command array
Use shape array pointer to get relevant shape data struct member
Move to start location based on shape and offset
Enter for loop of length equal to number of rows
Output: To Printer
[n, 0] as X coord
[n, 1] as Y coord
[n,2] as G value
multiplied by scalar
Final Move?
No
move to next row
Yes
Final Shape?
No
Move to next row of command array
Yes
END