# Mathis Bellino – 20342807

# CNC Plotter Coursework

# Outline of the Problem to be Solved

**The program must:**

1. Open, and store the font data from 'SingleStrokeFont.txt':

i) Check for successful/failed opening of file.  
ii) If successful continue onto iv)

ii) Failed opening of file return error.

iv) Define a structure for each drawing instruction (X Y P).

v) Define a structure to hold the ASCII code for each character, number of movements (N), and an array of Movement instructions.  
vi) Use an array to store all font data.

1. Obtain desired text height (4-10mm) inputted by the user (via keyboard input).
   1. **printf** and **scanf** to obtain the valid height value.
   2. Scale the X and Y movements by dividing the height by 18 following the required scale factor.
2. Read text to be drawn from a file (filename via keyboard input)
   1. Request and read the name of the file using printf and scanf.
   2. Open text file
   3. Provide error message for failure to open file.
   4. Only singular words can be processed at once.
   5. Words have to be drawn instantly after being process, hence no storage of G-code.
   6. The width of words is determined after having read a full word letter by letter until a space appears.
   7. Check if width of scaled x position value is greater than 100mm.
   8. This width is then compared with the space already taken up by previous words one the same line, max line width 100mm.
   9. Check if adding new word exceeds limit.
   10. If limit exceeded create line break by resetting x and y positional values to (0,n-5), n being the current line y value.
   11. If the new word fits on the line, or the word is inputted successfully on a new line, send the word to generateGCode function.
3. Generating G-code for each word and sending instructions to the robot.
   1. Scale x and y movements for each character in current word.
   2. Generate G-code based on scaled x and y movements, making sure pen state is
   3. Send the G-code and reset datum point.
   4. Track x position for the start of the next word.   
        
      curret

# Key Data Items

# **Core Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data type** | | **Rationale** |
| storeFontData | **Input**: const char\*  (font name) **Output**: FontData\*  Structure? why have you chosen data type and what it depends on | | Reads entire SingleStrokeFont.txt file into memory. Creates indexed structure containing all character stroke data. Returns NULL if file read fails |
| calculateWordWidth | **Input**: const char\*  FontData\* float height **Output**: Float | | Determines if word fits on current line  Returns calculated width |
|  |  | |  |
|  |  | |  |
| processWord | **Input:** const char\*  FontData\* float current x  float current y  float height **Output**: Float | Handles individual word processing     Returns ending x position | |
| handleLineBreak | **Input**: float\*  float\*  **Output**: bool | Manages new line transitions Updates X and Y coordinates Returns true if line break possible Returns false if at page limit | |
| generateGCode | **Input**: const char\* **Output**: FontData\* | Creates and sends GCode commands | |
| initialiseGCode |  |  | |

Each letter has local coordinates  
square of 18x18, scaled determined on the height of the letter  
So the robot will have to start the plotting (when a letter has been plotted before) at the last point (bottom right) of the previously plotted letter  
hence could store

Extend table as required

# Function Declarations

**FontData\* storeFontData(const char\* filename)**

Parameters:

- filename - path to SingleStrokeFont.txt

Return value: returns pointer to complete font library structure containing all characters

**float calculateWordWidth(const char\* word, FontData\* font, float height)**

Parameters:

- word - string to calculate width for

- font - pointer to font data

- height - text height for scaling

Return value: returns calculated width in mm

**bool processWord(const char\* word, FontData\* font, float\* currentX, float\* currentY, float height)**

Parameters:

- word - string to process

- font - pointer to font data

- currentX - pointer to current X position

- currentY - pointer to current Y position

- height - text height

Return value: returns true if word processed successfully

**bool handleLineBreak(float\* currentX, float\* currentY)**

Parameters:

- currentX - pointer to current X position

- currentY - pointer to current Y position

Return value: returns true if line break possible

**bool scaleMovements(MovementData\* move, float height)**

Parameters:

- move - pointer to movement data to scale

- height - desired text height

Return value: returns true if scaling successful

**bool generateGCode(MovementData\* move, bool penState)**

Parameters:

- move - pointer to movement data

- penState - true for pen down, false for pen up

Return value: returns true if G-code sent successfully

Each function includes:

- Clear parameter list with types and purposes

- Return value explanation

- Uses consistent pointer pattern for outputs

- Follows error handling pattern from skeleton code

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Test Case | Test Data | Expected Output |
|  |  |  |  |
|  |  |  |  |

*Extend table as required. Note that ‘Function’ includes main()*

# Flowchart(s)

May be included as separate pdf