# 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.

# 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**: bool | 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 | |

# Function Declarations

FontData **storeFontData**(const char filename)

* Parameters:
  + filename - path to SingleStrokeFont.txt
* Return value: returns a pointer to the 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 the 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 the word was 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 a line break is possible

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 the G-code was sent successfully

# Testing Information

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Test Case** | **Test Data** | **Expected Output** |
| main | Successful font data loading | SingleStrokeFont.txt | FontData structure loaded correctly |
|  | Failed font data loading | A file that doesn’t run | Error message displayed |
|  | Valid text height input | 8 | Text height set to 8mm |
|  | Invalid text height input | -5 | Error message displayed, height set to default |
|  | Successful text file opening | example.txt | Text drawn correctly on CNC plotter |
|  | Failed text file opening | non-existent.txt | Error message displayed |
|  | Word fits on current line | "hello" | G-code generated and sent to CNC plotter |
|  | Word exceeds line width | "supercalifragilisticexpialidocious" | Line break created, word drawn on new line |
| calculateWordWidth | Valid word and font data | "example", fontData, 8 | Calculated width returned |
|  | Invalid word or font data | NULL, fontData, 8 | Error value returned |
| processWord | Valid word, font, coordinates, height | "word", fontData, &x, &y, 8 | true returned, coordinates updated |
|  | Invalid parameters | NULL, fontData, &x, &y, 8 | false returned |
| handleLineBreak | Valid coordinates | &x, &y | true returned, coordinates updated |
|  | Invalid coordinates | NULL, NULL | false returned |
| generateGCode | Valid movement data and pen state | movementData, true | true returned, G-code sent |
|  | Invalid movement data | NULL, true | false returned, no G-code sent |

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

# Flowchart(s)

May be included as separate pdf