# Session 3: Write Simple Gcode using Functions

Copy paste original stuff from CURA and write that in the file using C++ program.

Make separate functions for adding CURA material at the beginning and end of the gcode.

Making functions in C++:

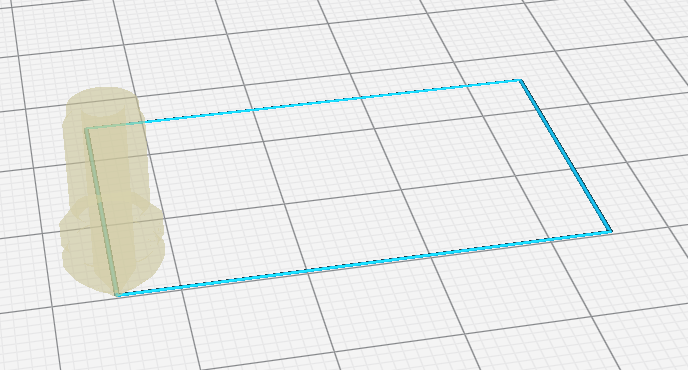
1. Refer to this video for functions: <https://www.youtube.com/watch?v=BGmDRQzY4CA>
2. Refer to the playlist for C++ learning in general: <https://www.youtube.com/watch?v=iBG0fN8lY8Y&list=PL43pGnjiVwgQHLPnuH9ch-LhZdwckM8Tq>
3. Refer to this video to learn how to use pointers to change values in a function: <https://www.youtube.com/watch?v=qhJLVxXMBE8>

## Making a Rectangle Parametrically

The idea is to make sure that the function created to make one shape can be used multiple times in loops. Thus, the function cannot directly append the gcode file. It has to:

1. Return the values of x, y, and e to get the 3 values to be used in the print.
2. Receive initial values of x, y, and e to know where to start from and how to calculate e.
3. Receive values of extrusion rate and side length to do the calculations.

Use this information to write a function to make squares of given side lengths.

The number of variables can become too much in the function in this case. So, instead of taking in x, y, and e, we can take in a structure object.

See this video to implement structure object declaration and returning a structure as a result.

<https://www.youtube.com/watch?v=wi4tjMZ65C4>

Generalize the square to a rectangle. Take length and breadth instead of just side.

Note that squares and rectangles have 4 sides so the array length for the coordinates (x,y) and the extrusion (e) is 4. But for other polygons it will be different. Thus, instead of having (x,y,e) as fixed arrays, we will have them dynamically declared in the function to size 4.

Here are some videos to explain dynamic declaration of variables and arrays:

1. Dynamic declaration of array: <https://www.youtube.com/watch?v=axsplPtoQF0&t=515s>
2. 2D array declaration: <https://www.youtube.com/watch?v=mGl9LO-je3o&t=807s>
3. Another dynamic array: <https://www.youtube.com/watch?v=MwwbgqG6bSk&t=406s>

The algorithm will be:

1. Declare a structure with \*x,\*y,\*e to store 4 values of coordinates and 4 extrusion values.
2. Make a function where x, y and e are dynamically declared as arrays of length 4.
3. Fill the values of the coordinates in x and y. Note that the extrusion is only done for the motion. So e[0] is for the movement from (x0,y0) (Point O) to (x[0],y[0]) (Point A) as (x0,y0) is the original point where the rectangle/square starts (and ends).

O, D

A

C

B

1

2

3

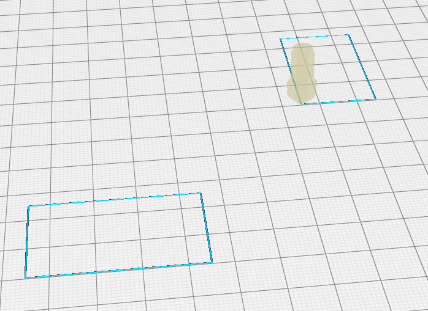
4

length

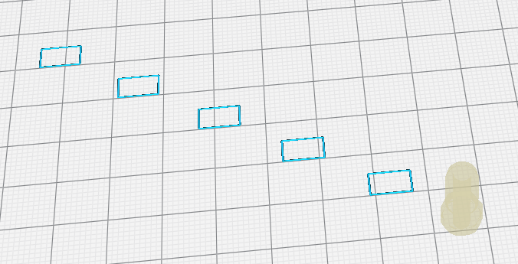
breadth

1. Return the structure object which has 4 values of (x,y,e)
2. Write on to the file in the main function using this structure object.

## Make an array of squares/ rectangles



1. Call the function twice in the same program to make different size rectangles/ squares.
2. Call the function in a loop to make a line of squares/ rectangles.

Enter x0: 40

Enter y0: 100

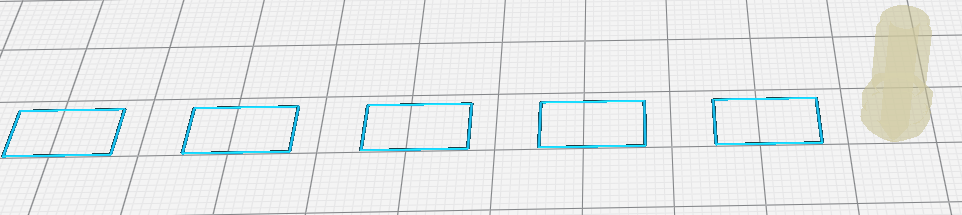
Enter Length: 8

Enter Breadth: 5

Enter x distance: 16

Enter y distance: -10

Enter number of rectangles: 5

Enter x0: 40

Enter y0: 100

Enter Length: 12

Enter Breadth: 8

Enter x distance: 20

Enter y distance: 0

Enter number of rectangles: 5

Remember to **lift the z axis in the travel move** when it is not extruding. That helps with maintaining print quality.

Additional things you can try:

1. Make a function for an array of rhombuses (square but tilted by 45 degrees).
2. Put fail-safes in the function to prevent:
   1. Print going out of bed area (put limits on x and y)
   2. Print overlap (ensure minimum horizontal, vertical, or radial distance)
3. Make arrays in horizontal, vertical and diagonal directions.
4. Make a loop where the user gives start point after each print to make a random pattern.

Assignment: Write a function for making an array of rectangles where the user inputs the number of iterations

In this session we use the structure approach since we have to return the values of the coordinates and extrusion values to the main() function. This is because we only call the file read and write functions in the main(). As we start using loops and make more complex shapes, this will become more difficult, so in later sessions, we would use file read and write within functions to declutter and simplify the main() function.