Answer Sheet Session 5

To be given in the second last session. Students try out the problems first. Then they see this solution and compare it with theirs. Final submission to be done in the final session.

Reminder to TRY IT YOURSELF FIRST

Note that this answer sheet is just a guide to show how things can turn out. The real submission would be the pictures of the results viewed in CURA.

Also note that you should only see this sheet if you CANNOT come up with a code that works at all. Otherwise, your CURA images are enough for a submission.

However, this is the final session. This is going to be complicated and if you feel like you are getting stuck, talk to each other and discuss potential solutions. Write algorithms first and see if the code can work out in theory. Do the checking before you write too much code to avoid wasting too much time. As things get more complex, you need to be more careful with each step.

# Session 5: Abacus

We use the same housekeeping functions and declared variables as session 4.

## Rectangle Structure and functions

### Structure

struct rect

{

// variables

double c[2], l, b, pts[4][2], dis[4];

// c = bottom left corner lb length and breadth pts are the 4 vertices in order

// functions

void set\_vertices()

{ // start from 2nd point since c[] is already bottom left corner

pts[0][0] = c[0] + l;

pts[0][1] = c[1]; // point 1 (10 bottom right)

dis[0] = l;

pts[1][0] = c[0] + l;

pts[1][1] = c[1] + b; // point 2 (11 top right)

dis[1] = b;

pts[2][0] = c[0];

pts[2][1] = c[1] + b; // point 3 (01 top left)

dis[2] = l;

pts[3][0] = c[0];

pts[3][1] = c[1]; // point 4 (00 bottom left)

dis[3] = b;

// use these 4 points to make rectangle easily using loop

}

};

### Foundational function of making rectangle

This function will be called by all others to make the basic rectangular shape in different ways to eventually make an abacus.

double make\_rect(rect a, double e)

{ // function to make a rectangle from a rectangle structure variable

// need the initial extrusion position and the final extrusion position needs to be returned

double x, y, zraise; // xy coordinates are for writing it on

// zraise is the z height we raise to when doing the moves

// cout<<"\nMaking Rectangle\n";

fileMain.open(gcodeSes4, ios::app); // append to add

x = a.c[0];

y = a.c[1];

zraise = z + 2; // raise 2mm up from the current level

fileMain<<"\nG1 F1000 Z"<<zraise; // raise for move

fileMain<<"\nG1 F1500 X"<<x<<" Y"<<y<<" E"<<e; // go to the starting point with fast move

fileMain<<"\nG1 F1200 Z"<<z; // back to current level and ready for printing speed

// cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

a.set\_vertices(); // set vertices and distances to the 4 values

for(int i = 0; i<4; i++)

{

x = a.pts[i][0];

y = a.pts[i][1];

e = e + a.dis[i]\*er; // add the corresponding distance to it

fileMain<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e; // writing all the points

//cout<<"\nG1 X"<<x<<" Y"<<y<<" E"<<e;

}

fileMain<<endl; // leave gap after completion

fileMain.close();

return e; // return the e value

}

## Making Sticks

### Filled rectangle function

This function calls the make\_rect function repeatedly to make a completely filled rectangle. If we make many layers, we can make solid blocks.

double fill\_rect(rect orig, double w, double e) // w is the width used (typically 0.2mm)

{

rect a;

int num;

double ratio, d, xc, yc;

ratio = orig.b/(2\*w); // since along the shorter length, we can go 2 times

num = floor(ratio); // max number of loops

a = orig;

for(int i=0;i<num;i++)

{

e = make\_rect(a,e);

// cout<<"\nE value after "<<i<<"th rectangle: "<<e;

a.b = a.b - (2\*w);

a.l = a.l - (2\*w); // reduce length and breadth by 2 thicknesses

a.c[0] = a.c[0] + w;

a.c[1] = a.c[1] + w; // add w to both centers

// cout<<"\nLength and Breadth after "<<i<<"th rectangle: "<<a.l<<" and "<<a.b<<endl;

}

if(a.b>(w/2))

{

cout<<"\nMaking the inner point\n";

d = (a.b\*a.l)/w; // effective area

cout<<"\nEffective length is: "<<d<<endl;

e = e + d\*er; // amount to purge in the center

fileMain.open(gcodeSes4, ios::app); // append to add the center point

xc = a.c[0] + (orig.l-a.l)/2 + w; // left start point

yc = a.c[1] + orig.b/2; // mid area

fileMain<<"\nG1 X"<<xc<<" Y"<<yc; // go to left point

xc = xc + a.l - 2\*w; // end point

fileMain<<"\nG1 X"<<xc<<" Y"<<yc<<" E"<<e; // go to end point

fileMain<<endl; // leave gap after completion

fileMain.close();

}

return e; // return the e value

}

### Function to make sticks using data from txt file

It takes too many variables to make the sticks. Typically, we have an interface. But if not, it is still easier to edit parameters in a txt file rather than input them during the program execution.

double make\_stick\_row(double e) // make a row of n sticks

{

int n, i, j, layers;

double yd, h, w;

string name1 = "stick\_row\_info.txt";

cout<<"\nTaking information from text file "<<name1<<endl;

fileRead.open(name1, ios::in);

getline(fileRead,line); // first line removed

fileRead>>n;

cout<<"\nThe number of sticks is: "<<n;

rect \*a = new rect[n];

fileRead>>a[0].l>>a[0].b>>a[0].c[0]>>a[0].c[1]>>yd>>h>>w;

fileRead.close();

for(i=1;i<n;i++)

{

a[i].c[0] = a[i-1].c[0];

a[i].c[1] = a[i-1].c[1] + yd;

a[i].b = a[i-1].b;

a[i].l = a[i-1].l;

}

// cout<<"\nEnter the stick height and layer height: ";

// cin>>h>>w;

layers = floor(h/w); // number of layers is height divided by layer height (or print width) w

for(i=0;i<layers;i++)

{

z = i\*w + w; // current layer height starting from w

cout<<"\nZ value: "<<z;

for(j=0;j<n;j++)

{

e = fill\_rect(a[j],w,e);

cout<<"\nRectangle number "<<(j+1)<<" in layer "<<(i+1);

cout<<"\nCoordinates: ("<<a[j].c[0]<<","<<a[j].c[1]<<")";

}

}

delete [] a; // ensure to delete a every time to use different values

return e;

}

## Make Base

To make the foundation/base for the abacus, we need to have holes for the sticks to go in.

### Make holes

First, we need a new function that makes rectangles between an outer and inner boundary.

double mk\_hole(rect id, rect od, double w, double e) // w is the width used (typically 0.2mm)

{ // id = inner dimensions //od = outer dimensions

rect a;

int i,j,num;

double ratio, dif;

dif = od.l-id.l;

ratio = dif/(2\*w);

num = floor(ratio); // number of loops to be made

a = od;

for(i=0;i<num;i++)

{

e = make\_rect(a,e);

a.l = a.l -(2\*w);

a.b = a.b -(2\*w);

a.c[0] = a.c[0] + w;

a.c[1] = a.c[1] + w;

}

return e;

}

### Building the base

Just like the row of sticks, we use a text file to input data.

double make\_base(double e) // make the abacus foundation

{

int n, i, j, layers;

double w, h; // layer height(w), height of the base (h)

string name1 = "base\_info.txt";

cout<<"\nTaking information from text file "<<name1<<endl;

fileRead.open(name1, ios::in);

getline(fileRead,line); // first line removed

fileRead>>n;

cout<<"\nThe number of pegs is: "<<n;

rect \*id = new rect[n];

rect \*od = new rect[n];

fileRead>>id[0].l>>id[0].b>>od[0].l>>od[0].c[0]>>od[0].c[1]>>w>>h;

layers = floor(h/w);

cout<<"\nNumber of layers is: "<<layers;

fileRead.close();

od[0].b = id[0].b + (od[0].l-id[0].l); // same dimensions added to both l n b each time

id[0].c[0] = od[0].c[0] + (od[0].l-id[0].l)/2;

id[0].c[1] = od[0].c[1] + (od[0].b-id[0].b)/2; // hole will be half of the length differences away

cout<<"\nSet id, od number 0";

for(i=1;i<n;i++)

{

od[i] = od[i-1];

id[i] = id[i-1];

// length l and breadth b copied exactly for all

od[i].c[0] = od[i-1].c[0] + od[i-1].l + w;

id[i].c[0] = od[i].c[0] + (od[i].l-id[i].l)/2; // x dimension of hole changing

// id[i].c[1] = od[i].c[1] + (od[i].b-id[i].b)/2; // y dimension remains the same

cout<<"\nSet id,od number "<<i;

}

for(i=0;i<layers;i++)

{

z = (i+1)\*w;

cout<<"\n\nLayer "<<i<<endl;

for(j=0;j<n;j++)

{

e = mk\_hole(id[j],od[j],w,e);

cout<<"\nHole "<<j;

}

}

return e;

}

## Making the Beads

Although, very similar to base, we have 2 new things here:

1. Making a 2D array to fit more beads on the build plate.
2. Making the edges curved.

double make\_pegs(double e) // make a 2d array of pegs

{

int n, m, i, j, k, layers;

double w, h, xdist, ydist, curve; // layer height(w), height of the base (h)

// xdist n ydist are for the horizontal and vertical distanc ein the array

// curve tells us how much the peg bulges out

string name1 = "pegs\_info.txt";

cout<<"\nTaking information from text file "<<name1<<endl;

fileRead.open(name1, ios::in);

getline(fileRead,line); // first line removed

fileRead>>n>>m;

cout<<"\nThe number of Rows is: "<<n<<"\nThe number of Columns is: "<<m;

rect\*\* id = new rect\*[n];

rect\*\* od = new rect\*[n];

rect vari, varo;

fileRead>>vari.l>>vari.b>>varo.l>>varo.c[0]>>varo.c[1]>>xdist>>ydist>>w>>h>>curve;

double dif;

dif = varo.l-vari.l;

varo.b = vari.b + dif;

vari.c[0] = varo.c[0] + dif/2;

vari.c[1] = varo.c[1] + dif/2; // making vari and varo complete

layers = floor(h/w);

cout<<"\nNumber of layers is: "<<layers;

fileRead.close();

for(i=0;i<n;i++)

{

id[i] = new rect[m];

od[i] = new rect[m];

} // completing the 2D array dynamic declaration

for(i=0;i<n;i++)

{

for(j=0;j<m;j++)

{

id[i][j] = vari;

od[i][j] = varo; // initialize all values

od[i][j].c[0] = varo.c[0] + (j\*xdist); // vertical columns marked by m x-spacings

od[i][j].c[1] = varo.c[1] + (i\*ydist); // horizontal rows marked by n y-spacings

id[i][j].c[0] = od[i][j].c[0] + dif/2;

id[i][j].c[1] = od[i][j].c[1] + dif/2; // id is just added dif/2 to all od

}

}

cout<<"\nInitialization done. Starting print.\n";

double length[2], breadth[2], temp;

// length out and in and breadth out and in 2 centers for the in and out

int lb2 = layers/2;

cout<<"\nlb2 = "<<lb2;

for(k=0;k<layers;k++)

{

z = (k+1)\*w;

cout<<"\n\nLayer "<<k<<endl;

temp = ((lb2-k)\*(lb2-k)\*curve)/(lb2\*lb2);

length[0] = varo.l - temp;

breadth[0] = varo.b - temp; // subtract to make outer walls curve in

length[1] = vari.l + temp;

breadth[1] = vari.b + temp; // add to make inner walls curve out

for(i=0;i<n;i++)

{

for(j=0;j<m;j++)

{

od[i][j].l = length[0];

od[i][j].b = breadth[0];

id[i][j].l = length[1];

id[i][j].b = breadth[1]; // adjusting the dims for all

od[i][j].c[0] += temp/2;

od[i][j].c[1] += temp/2; // add temp to move start point inward

// inner dimensions self adjust due to the hole function

e = mk\_hole(id[i][j],od[i][j],w,e);

od[i][j].c[0] -= temp/2;

od[i][j].c[1] -= temp/2; // revert back after print

cout<<"\nHole ( "<<i<<", "<<j<<" )";

}

}

}

return e;

}

## Main Function

All these functions are called one at a time to make various items. Depending on the costing and time frame of the prints, you can modify the codes for printing the whole assembly at once if needed.

#include "ses5headers.h"

int main ()

{

double x0,y0,e0;

cout<<"Enter the name of the file: ";

cin>>line;

gcodeSes4 = line + ".gcode";

// Reading Top File

read\_top\_file();

// write middle section of code

e0 = 0;

x0 = 10;

y0 = 10;

z = 0.2; // current z value

fileMain.open(gcodeSes4, ios::app);

fileMain<<"\nG1 F1200 X"<<x0<<" Y"<<y0<<" E"<<e0<<" \n"; // setting original position and speed

fileMain.close();

// e0 = make\_stick\_row(e0);

// e0 = make\_base(e0);

e0 = make\_pegs(e0);

cout<<"\n\nThe final e0 value is: "<<e0<<endl;

//Reading Bottom File

read\_bottom\_file();

}