C programming for beginners

Lesson 3

December 15, 2008

Main (and final) task

- What are the values of c that hold

$$x_{n+1} = {x_n}^2 + c \quad (x, c\,\epsilon\,\mathbb{C})$$

bounded?

Complex numbers

```
typedef struct {
        double re, im;
} complex;

complex csum (complex a, complex b){
        complex c;
        c.re = a.re + b.re;
        c.im = a.im + b.im;
        return c;
}

complex cquad (complex a){
        complex c;
        c.re = a.re*a.re - a.im*a.im;
        c.im = 2 * a.re * a.im;
        return c;
}

double cmod (complex a){
        return sqrt(a.re*a.re + a.im*a.im);
}

complex sucession (complex x, complex c){
        return csum(cquad(x),c);

}
```

Complex numbers

```
typedef struct {
        double re, im;
} complex;

complex csum (complex a, complex b){
        complex c;
        c.re = a.re + b.re;
        c.im = a.im + b.im;
        return c;
}

complex cquad (complex a){
        complex c;
        c.re = a.re*a.re - a.im*a.im;
        c.im = 2 * a.re * a.im;
        return c;
}

double cmod (complex a){
        return sqrt(a.re*a.re + a.im*a.im);
}

complex sucession (complex x, complex c){
        return csum(cquad(x),c);

}
```

struct

```
struct complex {
        double re, im;
};
```

typedef

```
typedef struct {
        double re, im;
} complex;
```

Complex numbers

```
typedef struct {
        double re, im;
} complex;

complex csum (complex a, complex b){
        complex c;
        c.re = a.re + b.re;
        c.im = a.im + b.im;
        return c;
}

complex cquad (complex a){
        complex c;
        c.re = a.re*a.re - a.im*a.im;
        c.im = 2 * a.re * a.im;
        return c;
}

double cmod (complex a){
        return sqrt(a.re*a.re + a.im*a.im);
}

complex sucession (complex x, complex c){
        return csum(cquad(x),c);

}
```

struct

```
struct complex {
        double re, im;
};
```

typedef

```
typedef struct {
        double re, im;
} complex;
```

math.h

```
return sqrt(a.re*a.re + a.im*a.im);
```

Auxiliary functions

```
void rdata(char *myfile, double *x){
        FILE *f = fopen(myfile,"r");
        fscanf(f, "%lf %lf %lf\n", &x[0], &x[1], &x[2]);
        fscanf(f, "%lf %lf %lf", &x[3], &x[4], &x[5]);
        fclose(f);
        return;
}

int iterate(complex c, complex (*f)(complex,
complex)){
        complex x;
        int i = 0;
        x.re = 0;
        x.im = 0;
        while (i<ITERS && cmod(x)<BIG){
                x = f(x,c);
                i++;
        }
       return i;

}
```

Main program

```
int main(int argc, char *argv[]){
        int i, j, nf, nc;
        complex x,c;
        double cc[6];
        char *myfile;
        int *p;
        myfile = argv[1];
        rdata(myfile, cc);
        nf = ((cc[1]-cc[0])/cc[2])+1;
        nc = ((cc[4]-cc[3])/cc[5])+1;
        p = (int *) malloc(nf*nc*sizeof(int));
        for (i=0;i<nf;i++){
                c.im = cc[0]+i*cc[2];
                for (j=0;j<nc;j++){
                        c.re = cc[3]+j*cc[5];
                        *p = iterate(c,&sucession);
                        p++;
                }
        }
        p-=nf*nc;
        makeppm_ascii("mandelbrot.ppm", p, nf, nc);
        return 0;

}
```

Main program

```
int main(int argc, char *argv[]){
        int i, j, nf, nc;
        complex x,c;
        double cc[6];
        char *myfile;
        int *p;
        myfile = argv[1];
        rdata(myfile, cc);
        nf = ((cc[1]-cc[0])/cc[2])+1;
        nc = ((cc[4]-cc[3])/cc[5])+1;
        p = (int *) malloc(nf*nc*sizeof(int));
        for (i=0;i<nf;i++){
                c.im = cc[0]+i*cc[2];
                for (j=0;j<nc;j++){
                        c.re = cc[3]+j*cc[5];
                        *p = iterate(c,&sucession);
                        p++;
                }
        }
        p-=nf*nc;
        makeppm_ascii("mandelbrot.ppm", p, nf, nc);
        return 0;

}
```

Pointers II

```
int *p, *q;
...
p = (int *) malloc(nf*nc*sizeof(int));
...
q = p;
...
p++; ... p--; ...
...
p = q;
```

Main program

```
int main(int argc, char *argv[]){
        int i, j, nf, nc;
        complex x,c;
        double cc[6];
        char *myfile;
        int *p;
        myfile = argv[1];
        rdata(myfile, cc);
        nf = ((cc[1]-cc[0])/cc[2])+1;
        nc = ((cc[4]-cc[3])/cc[5])+1;
        p = (int *) malloc(nf*nc*sizeof(int));
        for (i=0;i<nf;i++){
                c.im = cc[0]+i*cc[2];
                for (j=0;j<nc;j++){
                        c.re = cc[3]+j*cc[5];
                        *p = iterate(c,&sucession);
                        p++;
                }
        }
        p-=nf*nc;
        makeppm_ascii("mandelbrot.ppm", p, nf, nc);
        return 0;

}
```

Pointers II

```
int *p, *q;
...
p = (int *) malloc(nf*nc*sizeof(int));
...
q = p;
...
p++; ... p–; ...
...
p = q;
```

Reseting the pointer

```
p-=nf*nc;
```

makeppm ?

File Edit View Terminal Tabs Help

PPM Format Specification(5) PPM Format Specification(5)

NAME

PPM - Netpbm color image format

DESCRIPTION

This program is part of **Netpbm**(1).

The PPM format is a lowest common denominator color image file format.

It should be noted that this format is egregiously inefficient. It is highly redundant, while containing a lot of information that the human eye can't even discern. Furthermore, the format allows very little information about the image besides basic color, which means you may have to couple a file in this format with other independent information to get any decent use out of it. However, it is very easy to write and analyze programs to process this format, and that is the point.

It should also be noted that files often conform to this format in every respect except the precise semantics of the sample values. These files are useful because of the way PPM is used as an intermediary format. They are informally called PPM files, but to be absolutely precise, you should indicate the variation from true PPM. For example,

:

makeppm ?

File Edit View Terminal Tabs Help

```
       Here is an example of a small image in this format.
       P3
       # feep.ppm
       4 4
       15
        0  0  0    0  0  0    0  0  0   15  0 15
        0  0  0    0 15  7    0  0  0    0  0  0
        0  0  0    0  0  0    0 15  7    0  0  0
       15  0 15    0  0  0    0  0  0    0  0  0

       There is a newline character at the end of each of these lines.

       Programs that read this  format  should  be  as  lenient  as  possible,
       accepting anything that looks remotely like a PPM image.

       All  characters  referred  to  herein  are encoded in ASCII.  'newline'
       refers the the character known in ASCII as Line Feed or LF.   A  'white
       space'  character  is space, CR, LF, TAB, VT, or FF (I.e. what the ANSI
       standard C isspace() function calls white space).

COMPATIBILITY
       Before April 2000, a raw format  PPM  file  could  not  have  a  maxval
:
```

makepgm

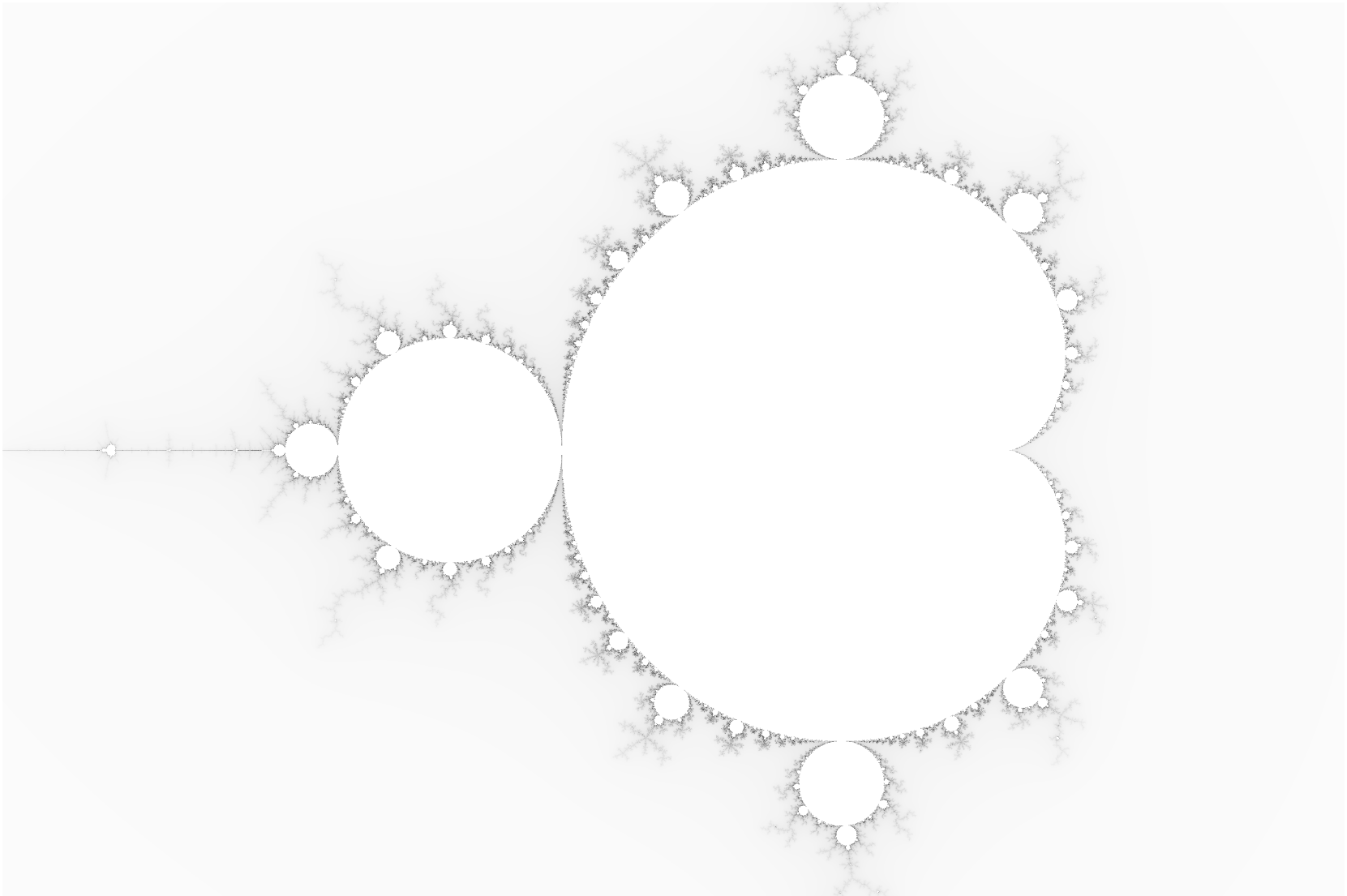
```
int makepgm(char *myfile, int *p, int nf, int nc){
        int i, j;
        FILE *of;
        of = fopen(myfile,"w");
        fprintf(of, "P2\n# Mandelbrot Set\n %d %d\n %d\n", nc, nf, MAXACOLOR);
        for (i=0;i<nf;i++){
                for (j=0;j<nc;j++){
                        fprintf(of," %d ", ((*p)==ITERS)?MAXACOLOR:(int) (MAXACOLOR - (*p)*MAXACOLOR/ITERS));
                        p++;
                }
                fprintf(of, "\n");
        }
        fclose(of);
        return 0;

}
```

if .. else

(*condition*?*expression*:*expression*)

Gray scale

range.dat
-1 1 0.001
-2 1 0.001



makeppm

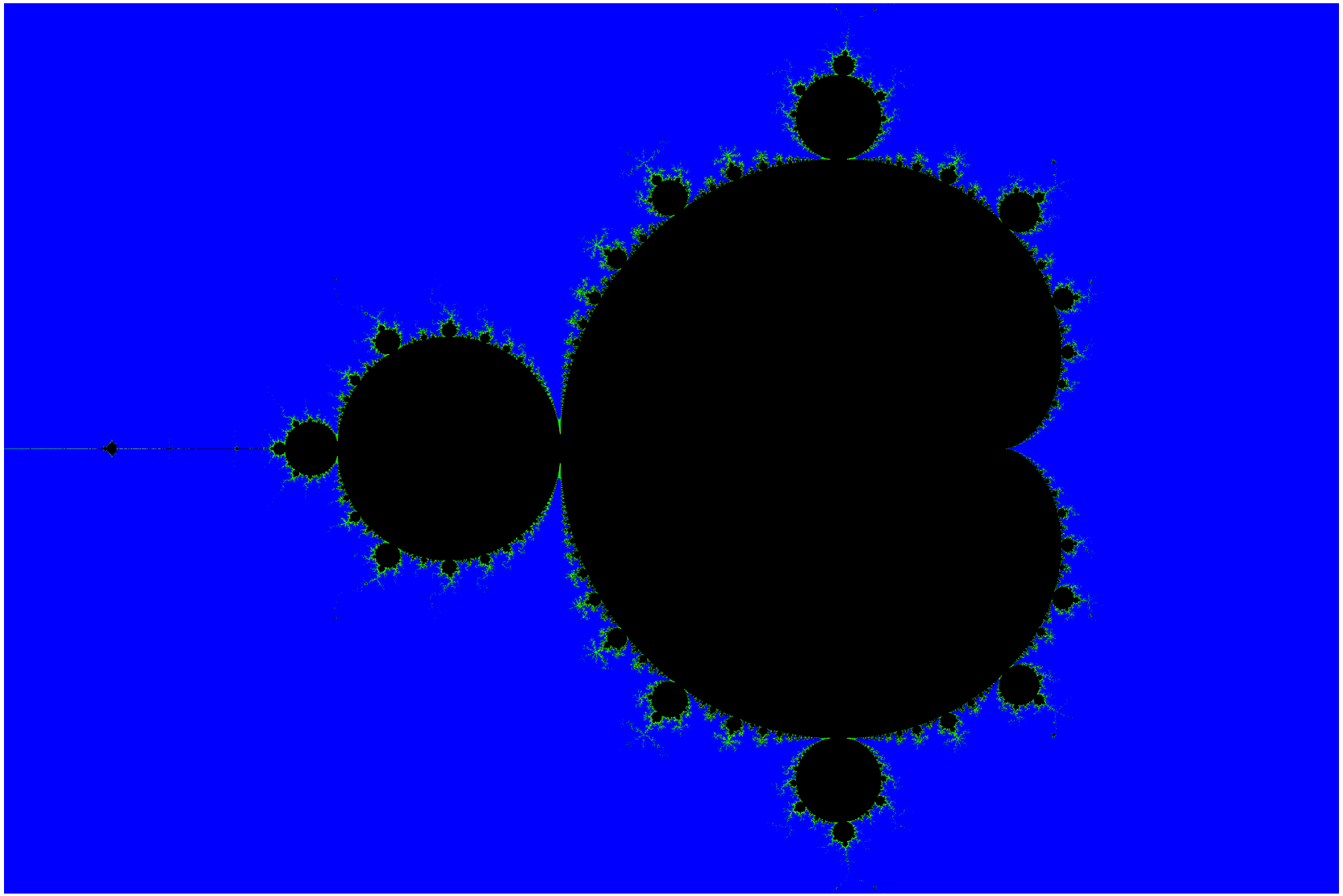
```
int makeppm_ascii(char *myfile, int *p, int nf, int nc){
        int i, j;
        irgbcolor mycolor;
        FILE *of;
        of = fopen(myfile,"w");
        fprintf(of, "P3\n# Mandelbrot Set\n %d %d\n %d\n", nc, nf, MAXACOLOR);
        for (i=0;i<nf;i++){
                for (j=0;j<nc;j++){
                        mycolor = geticolor(*p);
                        fprintf(of," %d %d %d ", mycolor.r, mycolor.g, mycolor.b);
                        p++;
                }
                fprintf(of, "\n");
        }
        fclose(of);
        return 0;

}
```

makeppm

```
typedef struct{ int r,g,b;} irgbcolor;

irgbcolor geticolor (int c){
        irgbcolor thiscolor;
        if (c==ITERS){
                thiscolor.r = 0;
                thiscolor.g = 0;
                thiscolor.b = 0;
        }else{
                thiscolor.r = (int) ((c<(ITERS/2))?0:((MAXACOLOR/3)*((4*c*c)/(ITERS*ITERS) - 1)));
                thiscolor.g = (int)
((c<(ITERS/2))?((4*MAXACOLOR/(ITERS*ITERS))*c*c):((4*MAXACOLOR/3)*(1-(c*c)/(ITERS*ITERS))));
                thiscolor.b = (int) ((c<(ITERS/2))?(MAXACOLOR*(1 - (4*c*c)/(ITERS*ITERS))):0);
        }
        return thiscolor;
}
```

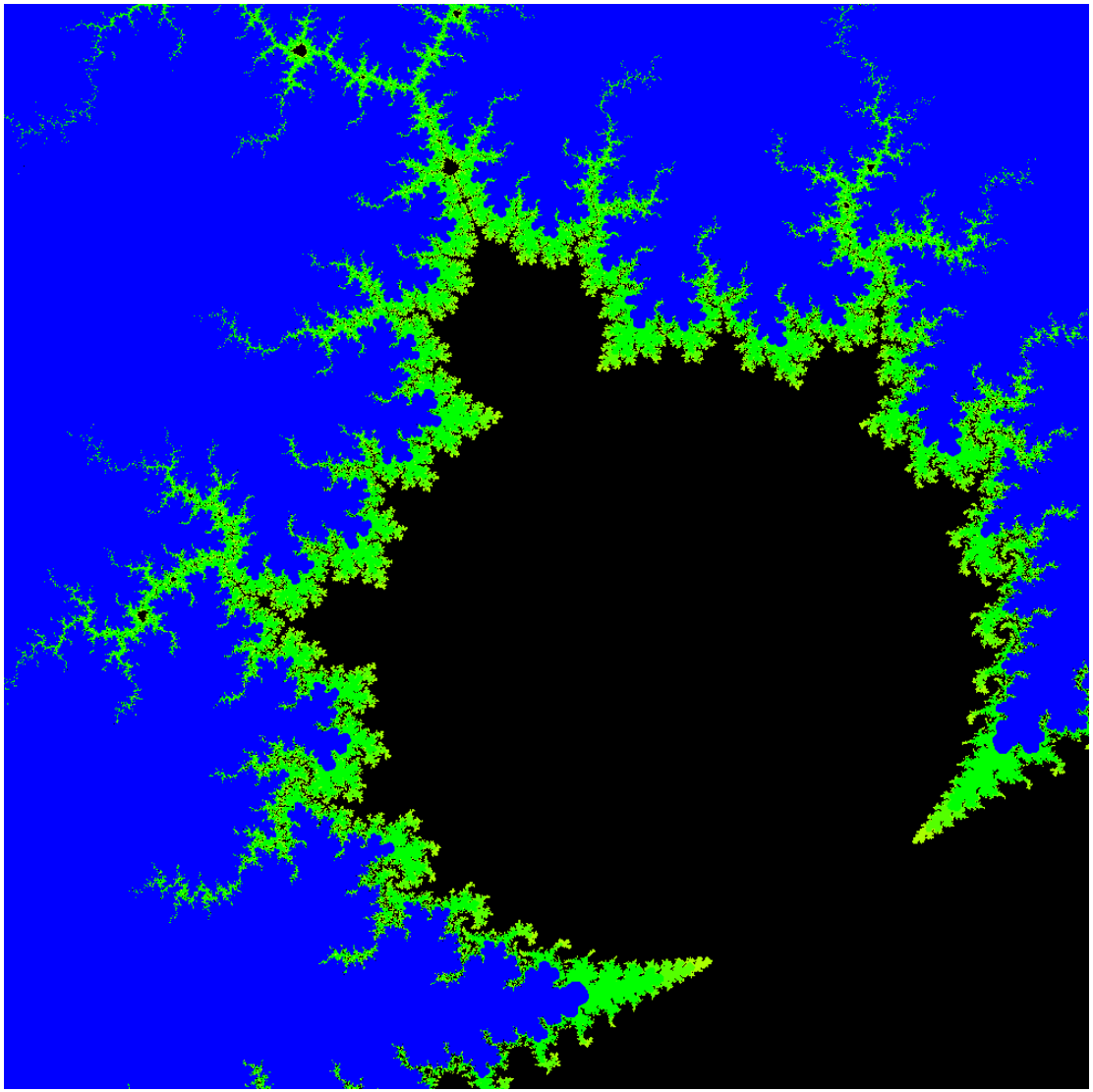
Final Result

range.dat
-1 1 0.001
-2 1 0.001



Zoom

range1.dat
-0.3 0.2 0.0001
-1.2 1.1 0.0001



Summary

What you don't know won't hurt you

Now you have the basic tools.
Go and learn everything else by yourself.