Mandelbrot set

A little bit of theory

The equation for Mandelbrot set is: $f_c(z) = z^2 + c$

All we have to know (and realise) is in this table:

$z_0 = 0$	$z_{n+1} = z_n^2 + c$	$c = a + bi$	$i^2 = -1$

When plugged into the main equation, we get:
$z_1 = 0^2 + c \iff z_1 = a + bi$

So all we have now is the complex number $c; c \in \mathbb{C}$.

But when we continue:
$z_2 = z_1^2 + c \iff z_2 = c^2 + c \iff z_1 = (a + bi)^2 + c$

Then $(a + bi)^2$ *using* $A^2 + 2AB + B^2$

$(a + bi)^2 = a^2 + 2abi + b^2 \cdot i^2 = a^2 - b^2 + 2abi$

$z_2 = a^2 - b^2 + 2abi + a + bi$
Which is, as we can see, another complex number with real part $a^2 - b^2$ and imaginary one $2ab$.

This process'll continue to the endless, so we have to set limitation → you can set whatever you want, but on PC with resolution 1920x1080p you dont see more iterations than **30**. So it's an ideal limit. `#define LIMIT 30`

Programming integration

Now we have to integrate all that simple math into code.

First, it's fine to know the size of terminal window.

```
#include <sys/ioctl.h>
#include <complex.h>

int width, height;

void getResolution() {
    struct winsize wnsz;
    ioctl(0, TIOCGWINSZ, &wnsz);

    width = (wnsz.ws_row);
    height = (wnsz.ws_col);
}
```

Now we have to represent "pixels". It's gonna be `x` and `y` loop in `callMandelbrot()` function.

```
[...]

void callMandelbrot() {
    getResolution();

    for (int y = 1; y <= width; y++) {
        for (int x = 1; x <= height; x++) {
            // rest
        } printf("\r\n");
    }
}

[...]
```

Lets integrate the math...
We can call the parameters same as in the theory section.

[...]