# **Arduino IDE: ComplexNumbers.ino**

**Reference**: **Complex Numbers**, [Studies.pdf](http://darcy.rsgc.on.ca/ACES/ICS3U/STUDIES.pdf), pp.4-5

Here’s a sample sketch I’ve prepared to lay the foundation for introduction to the Mandebrot Set.

// PROJECT  :ComplexNumbers

// PURPOSE  :Arithmetics of Complex Numbers for ICS3U

// COURSE   :ICS3U

// AUTHOR   :C. D'Arcy

// DATE     :2020 05 11

// MCU      :328P

// STATUS   :Working

// REFERENCE:pp. 4-5, <http://darcy.rsgc.on.ca/ACES/ICS3U/STUDIES.pdf>

//---------------------------------------------------------------------

**// NOTES    :There are many design options for modeling a comprehensive**

**//          :Complex number class. What appears below was designed to be**

**//          :understandable by the average ACES hardware student.**

**//          :Software-inclined ACES, designing for maximum efficiency**

**//          :would/will create something different.**

//---------------------------------------------------------------------

#define PRECISION 6   //number of decimal places in the display

#define DURATION 500  //timeout interval between display of outputs

//---------------------------------------------------------------------

// a struct provide a simple binding of the real and imaginary coefficients

struct Complex{

  float  re;

  float  im;

};

//---------------------------------------------------------------------

**Complex z0 = {0, 0};**        //the origin of the Complex Plane

**Complex z1 = {0.2,0.2};**     //a sample in the 1st quadrant

**Complex z2 = {0.5, 0.5};**    //another sample in the 1st quadrant

**Complex z3 = {-0.5,0.5};**    //a sample in the 2nd quadrant

**Complex z4 = {2,-1};**        //a sample in the 4th quadrant

//---------------------------------------------------------------------

void setup() {

  **Serial**.begin(9600);

  displayComplex(z0);       //present the complex numbers we've defined

  displayComplex(z1);       // "

  displayComplex(z2);       // "

  displayComplex(z3);       // "

  displayComplex(z4);       // "

  **Serial**.println(z0.re,PRECISION);

  displayComplex(add(z0,z1));             //sum

  displayComplex(subtract(z0,z1));        //difference

  displayComplex(multiply(z1,z2));        //product

//  displayComplex(divide(z1,z2));        //see if you can develop this function

  displayComplex(square(z4));             //multiply a complex number by itself

  displayComplex(squareAdd(z2,z3));       //sample compound operation

  **Serial**.println(magnitude(z4),PRECISION);//the distance from the origin

//---------------------------------------------------------------------

//let's begin to examine iterative feedback with complex numbers

  Complex z = {0.0,0.0};                  //initialize z at the origin

  Complex c = z1;                         //initialize c to a fixed complex number

  while (true){                           //repeat, forever

    displayComplex(z);                    //display the progress

    delay(DURATION);                      //admire...

    z = squareAdd(z,c);                  //z <- z^2 + c

  }

}

void loop() {}

**//----Arithmetic functions that operate on complex numbers------------**

Complex add(Complex za, Complex zb){

  Complex zc = {za.re+zb.re,za.im+zb.im};

  return zc;

}

Complex subtract(Complex za, Complex zb){

  Complex zc = {za.re-zb.re,za.im-zb.im};

  return zc;

}

Complex multiply(Complex za, Complex zb){

  Complex zc = {za.re\*zb.re-za.im\*zb.im,za.re\*zb.im+za.im\*zb.re};

  return zc;

}

Complex square (Complex za){

  Complex zc = multiply(za,za);

  return zc;

}

Complex squareAdd(Complex za, Complex zb){

  Complex zc = add (square(za),zb);

  return zc;

}

float magnitude(Complex za){

  return sqrt(za.re\*za.re+za.im\*za.im);

}

void displayComplex(Complex z){

  **Serial**.print("z=");

  **Serial**.print(z.re,PRECISION);

  **Serial**.print(z.im<0.0?'-':'+');

  **Serial**.print(abs(z.im),PRECISION);

  **Serial**.println("i");

}