Pointgivende aktivitet 3

Denne pointgivende aktivitet består af to opgaver, hvor der skal programmeres klasserne ComplexNumbers og Mandelbrot. Efter løsning af opgave 1 og 2 skal nedenstående main metode kunne køres (main.cpp er vedlagt opgaven). Som svar til opgaverne skal I **aflevere én zip-fil** indeholdende følgende filer:

output.pdf - Screenshot af jeres terminal (viser at ComplexNumber virker), sceenshot after imshow har vist første billede af Mandelbrot mængden, og screenshoot efter programmet har kørt lidt tid. Hvis ikke I har løst hele opgaven skal I udkommentere den del der ikke compiler og tage et screenshot af outputtet af det der compiler.
readme.txt – En txt fil der kort beskriver hvordan I har linket til OpenCV (med cmake, g++ terminal kald, pkg-config, etc.)
complexNumbers.h
Mandelbrot.h

Opgaven afleveres på itsLearning **senest 15. maj kl. 23.59.**

ComplexNumber

ComplexNumber klassen repræsenterer et komplekst tal og har følgende deklaration:

```
class ComplexNumber{
private:
    double _re;
    double _im;
public:
    ComplexNumber(double re, double im);
    void print();
    double magnitude();
    ComplexNumber operator+(ComplexNumber x);
    ComplexNumber operator*(ComplexNumber x);
};
```

Beskrivelse af metoder:

- re og im er henholdsvis den reelle og imaginære del af det komplekse tal.
- ComplexNumber(double re, double im) initialiserer et komplext tal
- print printer et reelt tal med cout. Hvis f.eks. re=1.5 og im=2.7 skal print udskrive 1.5+2.7i
- magnitude() returnere magnetuden af det komplekse tal
  - Husk: magnituden af et komplekst tal er sqrt(re^2 + im^2)
- + operatoren skal returnere summen af det komplekse tal selv og x
  - Husk summen af to komplekse tal er: (a+bi) + (c+di) = (a+c) + (b+d)i
- * operatoren skal returnere produktet af det komplekse tal og x
  - Husk produketet af to komplekse tal er: (a+bi) * (c+di) = (a*c -b*d) + (a*d + b*c)i

Mandelbrot

Baseret på https://da.wikipedia.org/wiki/Mandelbrotm%C3%A6ngden:

Mandelbrotmængden er en berømt fraktal opkaldt efter den franske matematiker Benoît B. Mandelbrot. Matematisk set er Mandelbrotmængden ikke vigtig, men den er vigtig for at forstå, hvordan simple formler kan frembringe komplekse strukturer.

Mandelbrotmængden er defineret som en mængde af komplekse punkter *{c}* for hvilke den rekursive talfølge $z_{n+1} = z_n^2 + c$ ikke går mod uendelig, når startværdien z_0 er 0.

For at beregne et billede af Mandelbrotmængden, antager man, at hvert punkt i billedet (row,col) modsvarer et specifikt komplekts tal c = row + col*i. Hver pixel svarer således til en specifik talfølge som kan undersøgt med hensyn til divergens.

Man kan vise, at talfølgen altid divergerer, hvis magnetuden af et z_n bliver større end 2. Eftersom det ikke er muligt at udføre et uendeligt antal iterationer på endelig tid vælges i praksis et forudbestemt, maksimalt antal iterationer, N, og hvis $|z_n|$ for en specifik talfølge ikke overstiger 2 inden for de N iterationer, så antager man at c tilhører mandelbrotmængden.

Når Mandelbrot mængden benyttes til at lave flotte videosekvenser gøre det ofte på følgende måde:

1. For hver pixel udregner man talfølgen og returnerer hvor mange iterationer, *n*,det tog før et tal i talfølgen blev over 2. I tilfælde af at et tal ikke når over 2 inden for de N iterationer returneres blot N.
2. En pixels intensitet udregnes på baggrund af antallet af iterationer til convergens ud fra følgende formel: intensitet = 255 * n/N.
    1. Hint: Benyt static_cast for at undgå integerdivision eller at resultatet der returneres er en float.

Klassens deklaration er som følger:

```
class Mandelbrot{
private:
  int canvas_size;
  cv::Mat canvas;
  cv::Point2d center;
  double _limit;
  int iterationsToDiverge(ComplexNumber x, int N);
public:
  Mandelbrot(int canvas_size, double initial_limit, cv::Point2d center);
  cv::Mat getCanvas();
  void drawImageToCanvas(int N);
  void animate(double rate, int N, int steps);
};
```

Opgaven er som følger:

1. Opret klassen Mandelbrot med ovenstående constructor. Constructoren skal initialisere de fire private variable på følgende måde:
   - _canvas_size sættes til canvas_size
   - _limit sættes til initial_limit
   - _center sættes til center
   - _canvas initialiseres til cv::Mat(_canvas_size, _canvas_size, CV_8UC1)
2. Lav en getter metoden getCanvas til _canvas
3. Lav metoden iterationsToDivergence. Metoden skal indeholde et loop der udregner talrækken $z_{n+1} = z_n^2 + c$. I hver iteration udregnes magnetuden af z_n. Metoden skal returnere hvor mange iterationer det tog før magnetuden af z_n blev større end 2 eller N hvis det maksimale antal iterationer blev nået.
4. Lav metoden drawImageToCanvas. Metoden skal iterere igennem hver pixel i _canvas og sætte pixel værdien til 255*n/N, hvor n er antal iterationer udregnet i iterationsToDivergence.
   1. Hint: Billedet er _canvas_size x _canvas_size pixels. Dette billede dækker et _limit x _limit område af det komplekse plan med _center i midten. Dvs. det komplekse tal der svarer til pixel (r,c) kan udregnes ved (eksempel givet for reel del af det komplekse tal, men det er identisk for den imaginere del, blot med c erstattet med r, og x erstattet med y):

      step = _limit / _canvas_size

      re = _center.x - _limit/2 * c*step
   2. Hint: du kan iterere gennem et billede og sætte pixelværdierne vha. to forloop:

```
for(int c = 0; c < _canvas.cols; c++){
  for(int r = 0; r < _canvas.rows; r++){
    _canvas.at<uchar>(cv::Point(c, r)) = yourIntegerValueHere;

  }
}
```

   3. Obs. Metoden opdaterer kun pixelværdierne i _canvas. Den viser dem ikke på skærmen.
5. Lav metoden animate. Metoden skal køre et loop med steps iterationer. I hver iteration skal der ske følgende:
   1. kald drawImageToCanvas(N) for at updatere mandelbrot canvaset

2. kald cv::imshow("Mandelbrot", _canvas) for at vise billedet på skærmen
3. kald char c = cv::waitKey(1) for at vente i 1ms og registrere et tastatur input
4. updatér _limit så den får værdien _limit*zoom_rate
5. Test om brugerens har trykket på ’q’ tasten og afslut loopet hvis det er tilfældet
    1. Hint:

```
if(c == 'q'){
    break;
 }
```

Kør programmet med følgende main metode:

```
#include <iostream>
#include <vector>

// Opencv
#include <opencv2/core/core.hpp>
#include <opencv2/core/mat.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/imgcodecs.hpp>

// User headers
#include "complex_number.h"
#include "mandelbrot.h"

int main(){

    ComplexNumber a(5, 2);
    a.print();
    ComplexNumber b(3,1);
    (a+b).print(); //8+3i
    (a*b).print(); //13+11i
    std::cout << a.magnitude() << std::endl; //5.38516

    //This center point provides a nice visualization
    int N = 100;
    cv::Point2f center = cv::Point2f(-0.160701788300366, 1.037566551323223);
    Mandelbrot mandelbrot(500, 2, center);
    mandelbrot.drawImageToCanvas(N);
    cv::imshow("Mandelbrot",mandelbrot.getCanvas());
    cv::waitKey(-1);//press any key to start animation
    mandelbrot.animate(0.98, N, 400);

  return 0;
}
```