Лабораторна робота 2

Створено системою Doxygen 1.9.1

1 Звіт з лабораторної роботи №2

за дисципліною "Элементи хаотичної динаміки"
студента групи ПА-17-2
Панасенка Єгора Сергійовича
Кафедра комп'ютерних технологій
ФПМ, ДНУ, 2020-2021 навч.р.
Варіант 17

Звіт доступний за посиланням
https://gaurapanasenko.github.io/unilab_opt/EoCD_Lab2/html/index.html.
Вихідний код доступний за посиланням
https://github.com/gaurapanasenko/unilab/tree/master/08/EoCD_Lab2

1.1 Постановка задачі

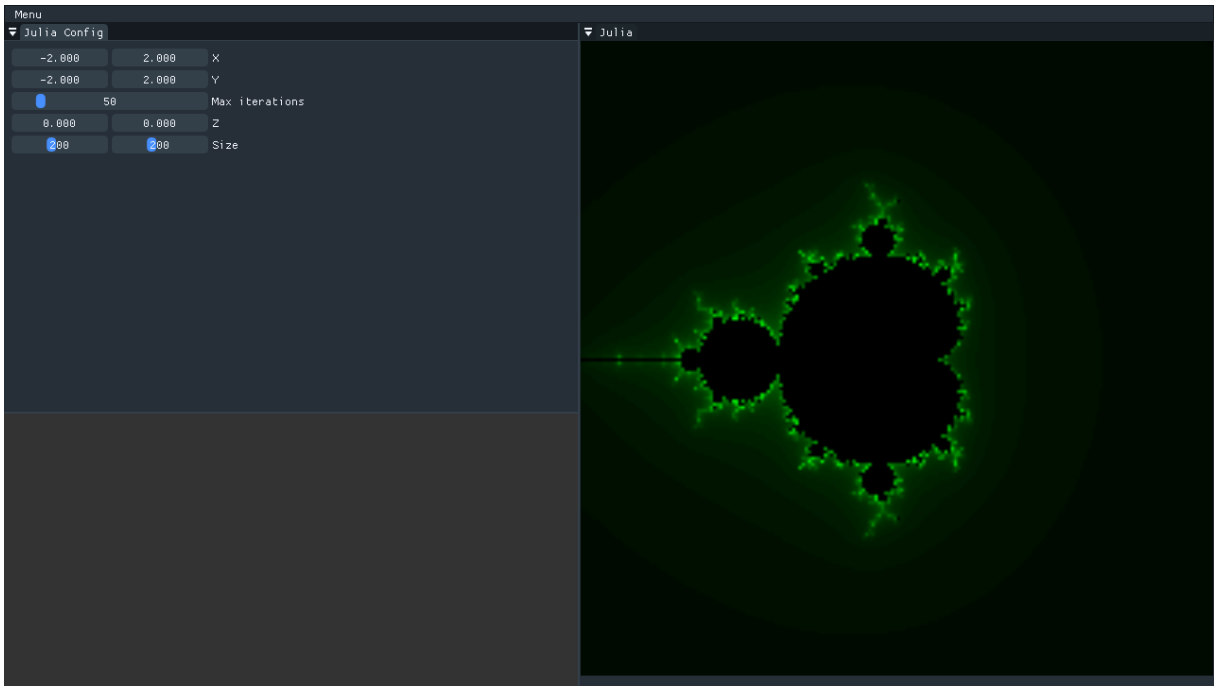
Написати алгоритм побудови заповнюючуої множини Жуліа для функцій $f(z) = z^2 + c_i$, де c_i будь-яке комплексне число, яке задовільняє умові $|c_i| < 2$.

1.2 Теоретичні відомості

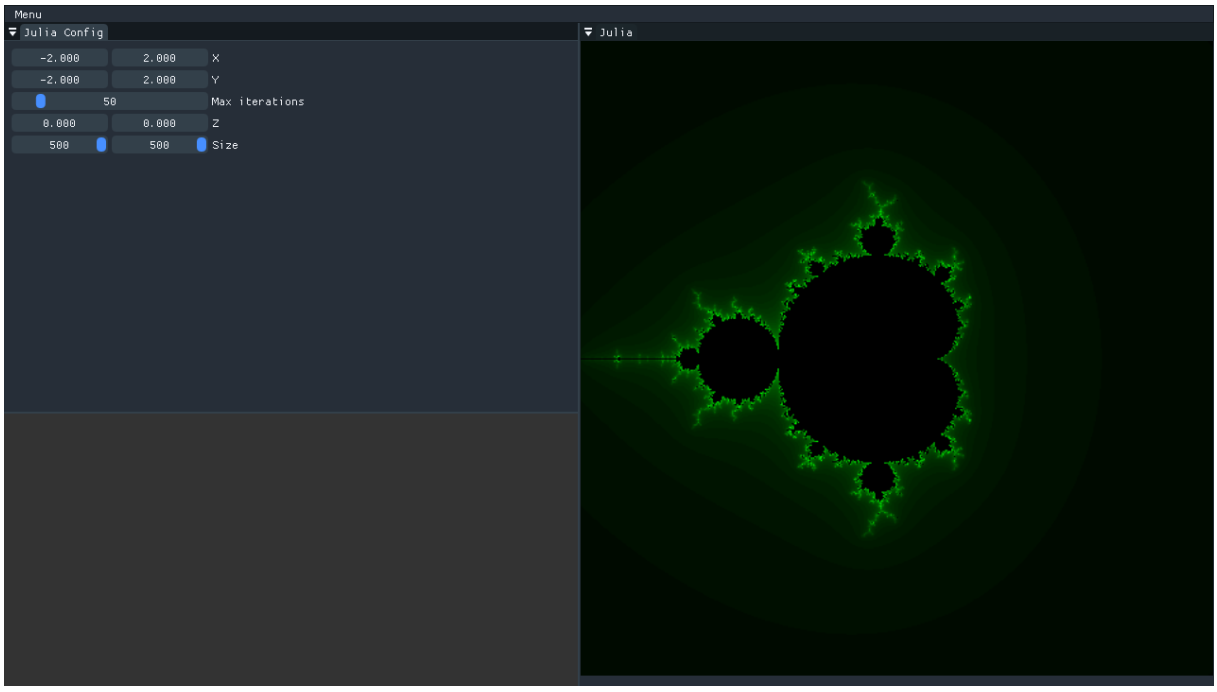
У голоморфній динаміці, множина Жуліа $J(f)$ раціонального відображення $f : CP^1 \to CP^1$ — множина точок, динаміка в околиці яких у певному сенсі нестійка відносно малих збурень початкового положення. У випадку, якщо f — поліном, розглядають також заповнену множину Жуліа — множину точок, що не прямують до нескінченності. Звичайна множина Жуліа при цьому є її межею.

1.3 Результат роботи програми

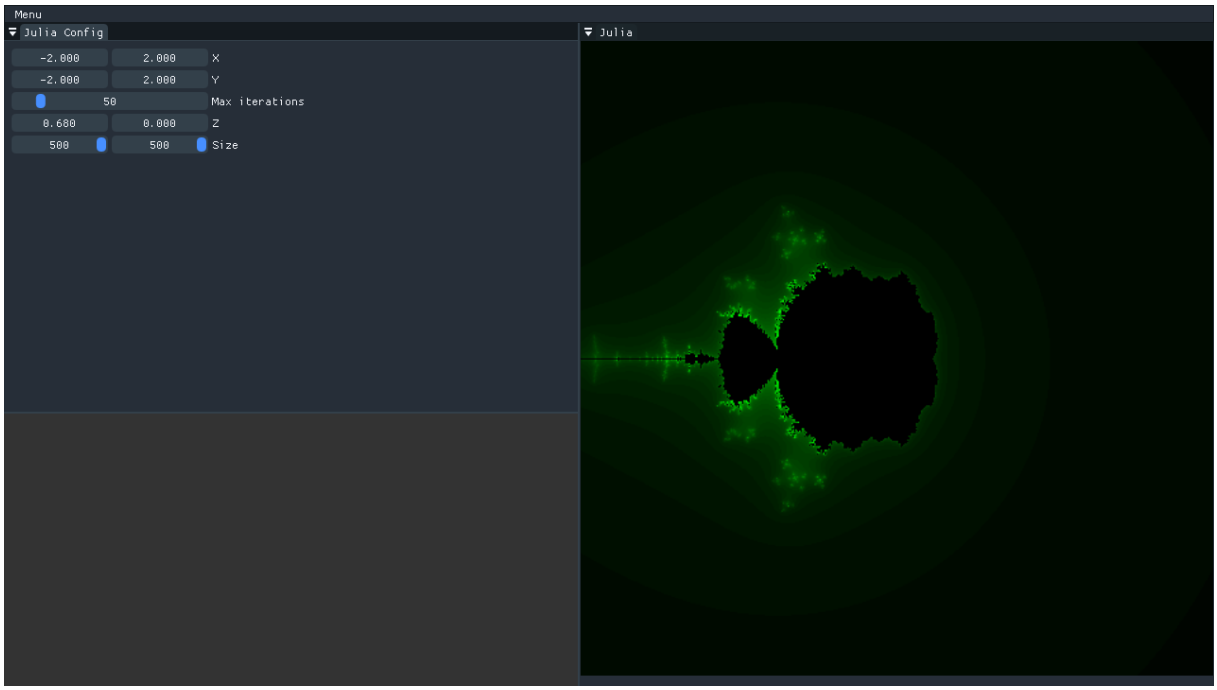
Програма дозволяє в інтерактивному режимі генерувати картинку на основі заповнюючуої множини жуліа. На при запуску програми отримаємо таке вікно.

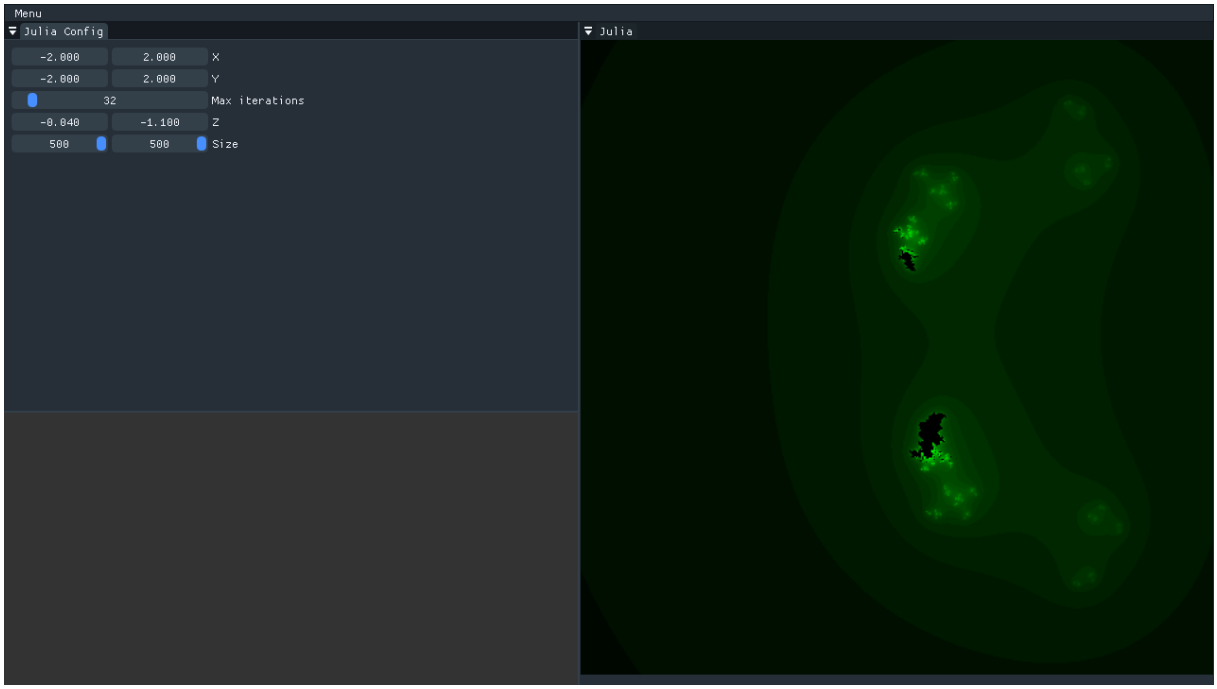


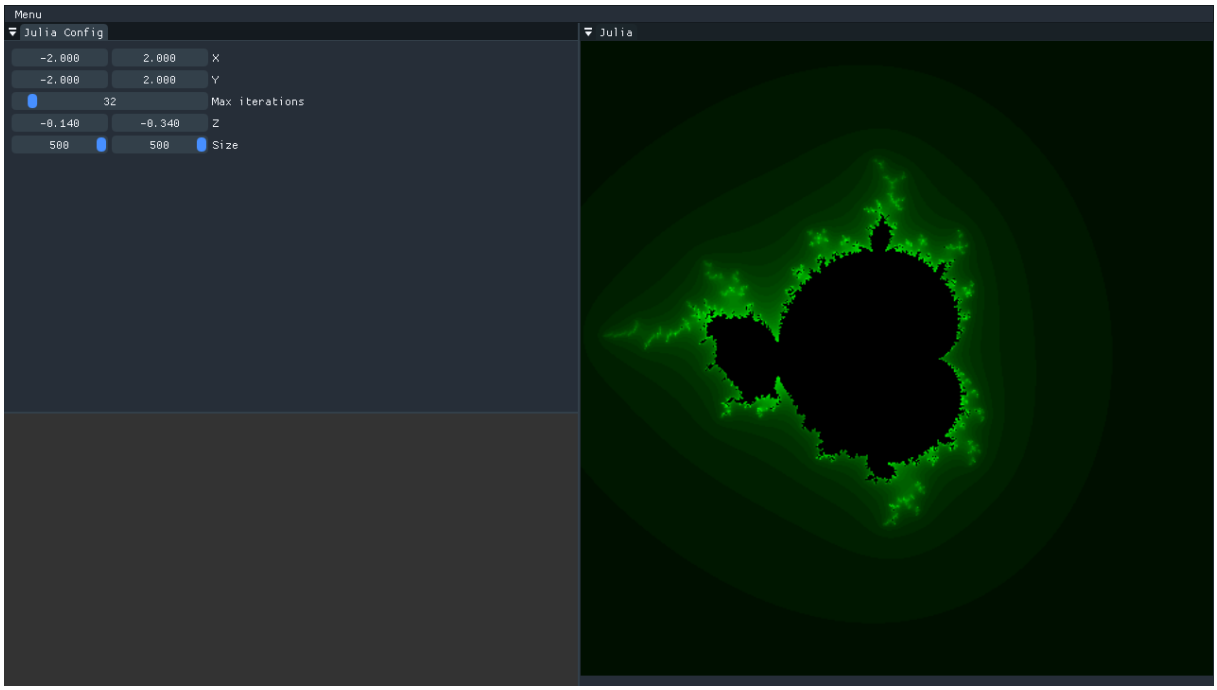
Тут генерується зображення 200x200 пікселів. Тепер збільшимо якість зображення до 500x500 пікселів.



Тепер спробуємо задати деякі початкові значення z, наприклад z=0.68, z=-0.04-1.1, z=-0.14-0.34.


Menu
Julia Config
-2.000
2.000
X
-2.000
2.000
Y
50
Max iterations
0.680
0.000
Z
500
500
Size
Julia


Menu
Julia Config
-2.000
2.000
X
-2.000
2.000
Y
32
Max iterations
-0.040
-1.100
Z
500
500
Size
Julia



Також можна побачити відео демонстрацію програми перейшовши за посиланням:
https://youtu.be/LXtTFXJbMHg
Або можна побачити відео перейшовши на HTML версію звіту.

2 Алфавітний покажчик простору імен

2.1 Простір імен

Повний список просторів імен.

3 Алфавітний покажчик класів

3.1 Класи

Класи, структури, об'єднання та інтерфейси з коротким описом.

4 Покажчик файлв

4.1 Файли

Повний список файлів.

5 Опис простору імен

5.1 Простір імен main

Змінні

- pmin
- pmax
- qmin
- qmax
- ppoints
- qpoints
- int max_iterations = 10
- int infinity_border = 10
- image = np.zeros((ppoints, qpoints), dtype="uint8")
- pp = np.linspace(pmin, pmax, ppoints)
- qq = np.linspace(qmin, qmax, qpoints)
- int z = 0
- int c = p + 1j * q
- int k = cv2.waitKey(0) & 0xFF

5.1.1 Опис змінних

5.1.1.1 c int main.c = p + 1j * q

Див. визначення в файлі main.py, рядок 20

5.1.1.2 image main.image = np.zeros((ppoints, qpoints), dtype="uint8")

Див. визначення в файлі main.py, рядок 11

5.1.1.3 infinity_border int main.infinity_border = 10

Див. визначення в файлі main.py, рядок 9

5.1.1.4 k int main.k = cv2.waitKey(0) & 0xFF

Див. визначення в файлі main.py, рядок 30

5.1.1.5 max_iterations int main.max_iterations = 10

Див. визначення в файлі main.py, рядок 8

5.1.1.6 pmax main.pmax

Див. визначення в файлі main.py, рядок 6

5.1.1.7 pmin main.pmin

Див. визначення в файлі main.py, рядок 6

5.1.1.8 pp main.pp = np.linspace(pmin, pmax, ppoints)

Див. визначення в файлі main.py, рядок 13

5.1.1.9 ppoints main.ppoints

Див. визначення в файлі main.py, рядок 7

5.1.1.10 qmax main.qmax

Див. визначення в файлі main.py, рядок 6

5.1.1.11 qmin main.qmin

Див. визначення в файлі main.py, рядок 6

5.1.1.12 qpoints main.qpoints

Див. визначення в файлі main.py, рядок 7

5.1.1.13 qq main.qq = np.linspace(qmin, qmax, qpoints)

Див. визначення в файлі main.py, рядок 14

5.1.1.14 z int main.z = 0

Див. визначення в файлі main.py, рядок 19

6 Класи

6.1 Клас Image

Зберігає самі піскелі, та розмір картинки.

#include <image.h>

Загальнодоступні елементи

- Image (shared_ptr< const pixel_t[]> data, const int size[2])
  Конструктор, який поєднує пікселі картинки, та її розмір у цей клас.

Загальнодоступні статичні елементи

- static Image julia (const int size[2], const Julia &jul)
  Метод який генерує нову картинку множини Жуліа.

Загальнодоступні атрибути

- shared_ptr< const pixel_t[]> data
  Пікселі картинки.
- const int size [2]
  Розмір картинки

6.1.1 Детальний опис

Зберігає самі піскелі, та розмір картинки.

Див. визначення в файлі image.h, рядок 24

6.1.2 Конструктор(и)

6.1.2.1 Image() Image::Image (
shared_ptr< const pixel_t[]> data,
const int size[2])

Конструктор, який поєднує пікселі картинки, та її розмір у цей клас.

Аргументи

data	пікселі картинки
size	розмір картинки

Див. визначення в файлі image.cpp, рядок 5
```
00007     : data(data), size{size[0], size[1]}
00008 {
00009 }
```

6.1.3 Опис методів компонент

6.1.3.1 julia() Image Image::julia (
const int size[2],
const Julia & jul) [static]

Метод який генерує нову картинку множини Жуліа.

Аргументи

size	розмір картинки
jul	вхідні параметри, які будуть використовуватися для генерації

Повертає

нова картинка множини Жуліа

Див. визначення в файлі image.cpp, рядок 11
```
00012 {
00013     const int width = size[0], height = size[1];
00014     std::shared_ptr<pixel_t[]> out_data(new pixel_t[size[0] * size[1]]);
```

```
00015     std::pair<float, float> coord;
00016     std::complex<float> z(jul.getZ()[0], jul.getZ()[1]), c;
00017     int maxIters = jul.getMaxIterations();
00018     float inf = jul.getInfinityBorder();
00019     int i, j, k, cur = 0;
00020     for (i = 0; i < width; i++) {
00021         for (j = 0; j < height; j++) {
00022             coord = jul.getCoords(i, j, size);
00023             cur = 0;
00024             z = std::complex<float>(jul.getZ()[0], jul.getZ()[1]);
00025             c = std::complex<float>(coord.first, coord.second);
00026             for (k = 0; k < maxIters; k++) {
00027                 z = pow(z, 2) + c;
00028                 if (abs(z) > inf) {
00029                     cur = k * 255 / maxIters;
00030                     break;
00031                 }
00032             }
00033             out_data[j * width + i][0] = 0 * i / width;
00034             out_data[j * width + i][1] = cur;
00035             out_data[j * width + i][2] = 0 * j / height;
00036             /*for (k = 0; k < 3; k++)
00037                 out_data[j * width + i][k] = cur;*/
00038             out_data[j * width + i][3] = 255;
00039         }
00040     }
00041     return {out_data, size};
00042 }
```

6.1.4 Компонентні дані

6.1.4.1 data shared_ptr<const pixel_t[]> Image::data

Пікселі картинки.

Див. визначення в файлі image.h, рядок 30

6.1.4.2 size const int Image::size[2]

Розмір картинки

Див. визначення в файлі image.h, рядок 34

Документація цих класів була створена з файлів:

- image.h
- image.cpp

6.2 Клас Julia

Зберігає параметри для генерації множини Жуліа та створює вікно для їх налаштування.

#include <julia.h>

Загальнодоступні елементи

- Julia ()

  Сторює клас із начальними параметрами. За замовчанням виставлено:
- const float * getMin () const

  Забираємо мінімальні значення по обом осям.
- const float * getMax () const

  Забираємо максимальними значення по обом осям.
- int getMaxIterations () const

  Забираємо максимальну кількість ітерацій.
- float getInfinityBorder () const

  забираємо границю переходу у безкінечність.
- const float * getZ () const

  забираємо початкове значення комплексного числа z
- std::pair< float, float > getCoords (int x, int y, const int size[2]) const

  Перетворює координати пікселів у координати множини Жуліа.
- bool imConfig ()

  Створює вікно для редагування параметрів.

Приватні дані

- float minValue [2]

  Мінімальне значення по осі X та по осі Y.
- float maxValue [2]

  Максимальне значення по осі X та по осі Y.
- int maxIterations

  Максимальна кількість ітерацій яку потрібно виконати, перед тим як вважати що ми знаходимося за множиною Жуліа.
- float infinityBorder

  Вважаємо що, якщо норма комплексного числа вишла за це число, то ми знаходимося за множиною Жуліа.
- float z [2]

  Початкове значення комплексного числа z.

6.2.1 Детальний опис

Зберігає параметри для генерації множини Жуліа та створює вікно для їх налаштування.

Див. визначення в файлі julia.h, рядок 9

6.2.2 Конструктор(и)

6.2.2.1 Julia() Julia::Julia ( )

Сторює клас із начальними параметрами. За замовчанням виставлено:

- координатна сітка у діапазонах [-2, 2] по обом осям,
- максимальна кількіст ітерацій 50,
- границя переходу у безкінечність 10.

Див. визначення в файлі julia.cpp, рядок 5

```
00006     : minValue{-2, -2}, maxValue{2, 2},
00007       maxIterations(50), infinityBorder(10), z{0, 0}
00008 {
00009
00010 }
```

6.2.3 Опис методів компонент

6.2.3.1 getCoords() std::pair< float, float > Julia::getCoords (int x, int y, const int size[2]) const

Перетворює координати пікселів у координати множини Жуліа.

Аргументи

x	номер пікселя по осі X
y	номер пікселя по осі Y
size	розмір картинки

Повертає

координати множини Жуліа

Див. визначення в файлі julia.cpp, рядок 36

```
00037 {
00038     std::pair<float, float> out(0, 0);
00039     out.first = (maxValue[0] - minValue[0]) * x / size[0] + minValue[0];
00040     out.second = (maxValue[1] - minValue[1]) * y / size[1] + minValue[1];
00041     return out;
00042 }
```

6.2.3.2 getInfinityBorder() float Julia::getInfinityBorder ( ) const

забираємо границю переходу у безкінечність.

Повертає

границя переходу у безкінечність

Див. визначення в файлі julia.cpp, рядок 27
```
00028 {
00029     return infinityBorder;
00030 }
```

6.2.3.3 getMax() const float * Julia::getMax ( ) const

Забираємо максимальними значення по обом осям.

Повертає

массив із мксимальними значеннями

Див. визначення в файлі julia.cpp, рядок 17
```
00018 {
00019     return maxValue;
00020 }
```

6.2.3.4 getMaxIterations() int Julia::getMaxIterations ( ) const

Забираємо максимальну кількість ітерацій.

Повертає

максимальна кількість ітерацій

Див. визначення в файлі julia.cpp, рядок 22
```
00023 {
00024     return maxIterations;
00025 }
```

6.2.3.5 getMin() const float * Julia::getMin ( ) const

Забираємо мінімальні значення по обом осям.

Повертає

массив із мінімальни значеннями

Див. визначення в файлі julia.cpp, рядок 12
```
00013 {
00014     return minValue;
00015 }
```

6.2.3.6 getZ() const float * Julia::getZ ( ) const

забираємо початкове значення комплексного числа z

Повертає

значення комплексного числа z

Див. визначення в файлі julia.cpp, рядок 32

```
00032                           {
00033     return z;
00034 }
```

6.2.3.7 imConfig() bool Julia::imConfig ( )

Створює вікно для редагування параметрів.

Повертає

перевірка, чи були змінені параметри у класі.

Див. визначення в файлі julia.cpp, рядок 44

```
00045 {
00046     bool changed = false;
00047     changed |= ImGui::DragFloatRange2("X", minValue, maxValue+1, 0.01, -3, 3);
00048     changed |= ImGui::DragFloatRange2("Y", minValue+1, maxValue, 0.01, -3, 3);
00049     changed |= ImGui::SliderInt("Max iterations", &maxIterations, 1, 400);
00050     changed |= ImGui::DragFloat2("Z", z, 0.01, -3, 3);
00051     return changed;
00052 }
```

6.2.4 Компонентні дані

6.2.4.1 infinityBorder float Julia::infinityBorder [private]

Вважаємо що, якщо норма комплексного числа вишла за це число, то ми знаходимося за множиною Жуліа.

Це число означає що всі значення норми вище за це ми будемо вважати безкінечністю.

Див. визначення в файлі julia.h, рядок 31

6.2.4.2 maxIterations int Julia::maxIterations [private]

Максимальна кількість ітерацій яку потрібно виконати, перед тим як вважати що ми знаходимося за множиною Жуліа.

Див. визначення в файлі julia.h, рядок 23

6.2.4.3 maxValue float Julia::maxValue[2] [private]

Максимальне значення по осі X та по осі Y.

Див. визначення в файлі julia.h, рядок 18

6.2.4.4 minValue float Julia::minValue[2] [private]

Мінімальне значення по осі X та по осі Y.

Див. визначення в файлі julia.h, рядок 14

6.2.4.5 z float Julia::z[2] [private]

Початкове значення комплексного числа z.

Див. визначення в файлі julia.h, рядок 35

Документація цих класів була створена з файлів:

- julia.h
- julia.cpp

7 Файли

7.1 Файл image.cpp

```
#include <algorithm>
#include <complex>
#include "image.h"
```

7.2 image.cpp

```
00001 #include <algorithm>
00002 #include <complex>
00003 #include "image.h"
00004
00005 Image::Image(std::shared_ptr<const pixel_t[]> data,
00006              const int size[])
00007     : data(data), size{size[0], size[1]}
00008 {
00009 }
00010
00011 Image Image::julia(const int size[], const Julia &jul)
00012 {
00013     const int width = size[0], height = size[1];
00014     std::shared_ptr<pixel_t[]> out_data(new pixel_t[size[0] * size[1]]);
00015     std::pair<float, float> coord;
00016     std::complex<float> z(jul.getZ()[0], jul.getZ()[1]), c;
00017     int maxIters = jul.getMaxIterations();
00018     float inf = jul.getInfinityBorder();
00019     int i, j, k, cur = 0;
00020     for (i = 0; i < width; i++) {
00021         for (j = 0; j < height; j++) {
00022             coord = jul.getCoords(i, j, size);
00023             cur = 0;
00024             z = std::complex<float>(jul.getZ()[0], jul.getZ()[1]);
00025             c = std::complex<float>(coord.first, coord.second);
00026             for (k = 0; k < maxIters; k++) {
00027                 z = pow(z, 2) + c;
00028                 if (abs(z) > inf) {
00029                     cur = k * 255 / maxIters;
00030                     break;
00031                 }
00032             }
00033             out_data[j * width + i][0] = 0 * i / width;
00034             out_data[j * width + i][1] = cur;
00035             out_data[j * width + i][2] = 0 * j / height;
00036             /*for (k = 0; k < 3; k++)
00037                 out_data[j * width + i][k] = cur;*/
00038             out_data[j * width + i][3] = 255;
00039         }
00040     }
00041     return {out_data, size};
00042 }
```

7.3 Файл image.h

#include <memory>
#import "julia.h"

Класи

- class Image

  Зберігає самі піскелі, та розмір картинки.

Макровизначення

- #define COMP 4

Визначення типів

- typedef unsigned char color_t

  Тип кольору, задає інтенсивність кольору у вигляді цілого числа у діапазоні значень [0, 255].
- typedef color_t pixel_t[COMP]

  Тип пікселя, задає що у пікселя буде 4 канали інтенсивності: красний, зелений, синій та прозорість.

7.3.1 Опис макровизначень

7.3.1.1 COMP #define COMP 4

Див. визначення в файлі image.h, рядок 6

7.3.2 Опис визначень типів

7.3.2.1 color_t typedef unsigned char color_t

Тип кольору, задає інтенсивність кольору у вигляді цілого числа у діапазоні значень [0, 255].

Див. визначення в файлі image.h, рядок 14

7.3.2.2 pixel_t typedef color_t pixel_t[COMP]

Тип пікселя, задає що у пікселя буде 4 канали інтенсивності: красний, зелений, синій та прозорість.

Див. визначення в файлі image.h, рядок 19

7.4 image.h

```
00001 #ifndef IMAGE_H
00002 #define IMAGE_H
00003 #include <memory>
00004 #import "julia.h"
00005
00006 #define COMP 4
00007
00008 using namespace std;
00009
00014 typedef unsigned char color_t;
00019 typedef color_t pixel_t[COMP];
00020
00024 class Image
00025 {
00026 public:
00030     shared_ptr<const pixel_t[]> data;
00034     const int size[2];
00035
00042     Image(shared_ptr<const pixel_t[]> data, const int size[2]);
00049     static Image julia(const int size[2], const Julia& jul);
00050 };
00051
00052
00053 #endif // IMAGE_H
```

7.5 Файл julia.cpp

```
#include <cstring>
#include <imgui.h>
#include "julia.h"
```

7.6 julia.cpp

```
00001 #include <cstring>
00002 #include <imgui.h>
00003 #include "julia.h"
00004
00005 Julia::Julia()
00006     : minValue{-2, -2}, maxValue{2, 2},
00007       maxIterations(50), infinityBorder(10), z{0, 0}
00008 {
00009
00010 }
00011
00012 const float *Julia::getMin() const
00013 {
00014     return minValue;
00015 }
00016
00017 const float *Julia::getMax() const
00018 {
00019     return maxValue;
00020 }
00021
00022 int Julia::getMaxIterations() const
00023 {
00024     return maxIterations;
00025 }
00026
00027 float Julia::getInfinityBorder() const
00028 {
00029     return infinityBorder;
00030 }
00031
00032 const float *Julia::getZ() const {
00033     return z;
00034 }
00035
00036 std::pair<float, float> Julia::getCoords(int x, int y, const int size[]) const
00037 {
00038     std::pair<float, float> out(0, 0);
00039     out.first = (maxValue[0] - minValue[0]) * x / size[0] + minValue[0];
00040     out.second = (maxValue[1] - minValue[1]) * y / size[1] + minValue[1];
00041     return out;
00042 }
00043
00044 bool Julia::imConfig()
00045 {
00046     bool changed = false;
00047     changed |= ImGui::DragFloatRange2("X", minValue, maxValue+1, 0.01, -3, 3);
00048     changed |= ImGui::DragFloatRange2("Y", minValue+1, maxValue, 0.01, -3, 3);
00049     changed |= ImGui::SliderInt("Max iterations", &maxIterations, 1, 400);
00050     changed |= ImGui::DragFloat2("Z", z, 0.01, -3, 3);
00051     return changed;
00052 }
```

7.7 Файл julia.h

#include "map"

Класи

- class Julia

  Зберігає параметри для генерації множини Жуліа та створює вікно для їх налаштування.

7.8 julia.h

```
00001 #ifndef JULIA_H
00002 #define JULIA_H
00003 #include "map"
00004
00009 class Julia
00010 {
00014     float minValue[2];
00018     float maxValue[2];
```

```
00023     int maxIterations;
00031     float infinityBorder;
00035     float z[2];
00036 public:
00045     Julia();
00050     const float *getMin() const;
00055     const float *getMax() const;
00060     int getMaxIterations() const;
00065     float getInfinityBorder() const;
00070     const float *getZ() const;
00078     std::pair<float, float> getCoords(int x, int y, const int size[2]) const;
00083     bool imConfig();
00084 };
00085
00086 #endif // JULIA_H
```

7.9 Файл main.cpp

```
#include <string>
#include <memory>
#include <sstream>
#include "gui/app.h"
#include "gui/texture.h"
#include "julia.h"
#include "image.h"
```

Макровизначення

- #define PROJECT_NAME "IaMP_Lab2"

Функції

- int main (int, char **)

7.9.1 Опис макровизначень

7.9.1.1 PROJECT_NAME #define PROJECT_NAME "IaMP_Lab2"

Див. визначення в файлі main.cpp, рядок 10

7.9.2 Опис функцій

7.9.2.1 main() int main (int , char **)

Див. визначення в файлі main.cpp, рядок 14

```
00015 {
00016     App app(PROJECT_NAME);
00017     Julia jul;
00018     shared_ptr<const Image> image;
00019     const Texture texture;
00020     int size[2] = {200, 200};
00021     auto tex_id = (void*)(intptr_t)(texture.id);
00022     bool changed = true;
00023     image = make_shared<Image>(Image::julia(size, jul));
00024     texture.update(*image);
00025 
00026     // Main loop
00027     while (!app.should_closed())
00028     {
00029         app.begin_loop();
00030 
00031         ImVec2 sz(image->size[0], image->size[1]);
00032         ImGui::SetNextWindowSize(sz, ImGuiCond_FirstUseEver);
00033         ImGui::PushStyleVar(ImGuiStyleVar_WindowPadding, ImVec2(0, 0));
00034         ImGui::Begin("Julia");
00035         ImVec2 cont_sz = ImGui::GetContentRegionAvail();
00036         ImVec2 img_size(cont_sz.x, image->size[1] * cont_sz.x / image->size[0]);
00037         ImGui::Image(tex_id, img_size);
00038         ImGui::End();
00039         ImGui::PopStyleVar();
00040 
00041         ImGui::Begin("Julia Config", NULL, ImGuiWindowFlags_AlwaysAutoResize);
00042         changed = jul.imConfig();
00043         changed |= ImGui::SliderInt2("Size", size, 0, 500);
00044         if (changed) {
00045             image = make_shared<Image>(Image::julia(size, jul));
00046             texture.update(*image);
00047         }
00048         ImGui::End();
00049 
00050         app.end_loop();
00051     }
00052 
00053     return 0;
00054 }
```

7.10 main.cpp

```
00001 #include <string>
00002 #include <memory>
00003 #include <sstream>
00004 
00005 #include "gui/app.h"
00006 #include "gui/texture.h"
00007 #include "julia.h"
00008 #include "image.h"
00009 
00010 #define PROJECT_NAME "IaMP_Lab2"
00011 
00012 using namespace std;
00013 
00014 int main(int, char**)
00015 {
00016     App app(PROJECT_NAME);
00017     Julia jul;
00018     shared_ptr<const Image> image;
00019     const Texture texture;
00020     int size[2] = {200, 200};
00021     auto tex_id = (void*)(intptr_t)(texture.id);
00022     bool changed = true;
00023     image = make_shared<Image>(Image::julia(size, jul));
00024     texture.update(*image);
00025 
00026     // Main loop
00027     while (!app.should_closed())
00028     {
00029         app.begin_loop();
00030 
00031         ImVec2 sz(image->size[0], image->size[1]);
00032         ImGui::SetNextWindowSize(sz, ImGuiCond_FirstUseEver);
00033         ImGui::PushStyleVar(ImGuiStyleVar_WindowPadding, ImVec2(0, 0));
00034         ImGui::Begin("Julia");
00035         ImVec2 cont_sz = ImGui::GetContentRegionAvail();
```

```
00036         ImVec2 img_size(cont_sz.x, image->size[1] * cont_sz.x / image->size[0]);
00037         ImGui::Image(tex_id, img_size);
00038         ImGui::End();
00039         ImGui::PopStyleVar();
00040 
00041         ImGui::Begin("Julia Config", NULL, ImGuiWindowFlags_AlwaysAutoResize);
00042         changed = jul.imConfig();
00043         changed |= ImGui::SliderInt2("Size", size, 0, 500);
00044         if (changed) {
00045             image = make_shared<Image>(Image::julia(size, jul));
00046             texture.update(*image);
00047         }
00048         ImGui::End();
00049 
00050         app.end_loop();
00051     }
00052 
00053     return 0;
00054 }
```

7.11 Файл main.py

Простори імен

- main

Змінні

- main.pmin
- main.pmax
- main.qmin
- main.qmax
- main.ppoints
- main.qpoints
- int main.max_iterations = 10
- int main.infinity_border = 10
- main.image = np.zeros((ppoints, qpoints), dtype="uint8")
- main.pp = np.linspace(pmin, pmax, ppoints)
- main.qq = np.linspace(qmin, qmax, qpoints)
- int main.z = 0
- int main.c = p + 1j * q
- int main.k = cv2.waitKey(0) & 0xFF

7.12 main.py

```
00001 #!/usr/bin/env python3
00002 
00003 import cv2
00004 import numpy as np
00005 
00006 pmin, pmax, qmin, qmax = -2, 2, -2, 2
00007 ppoints, qpoints = 500, 500
00008 max_iterations = 10
00009 infinity_border = 10
00010 
00011 image = np.zeros((ppoints, qpoints), dtype="uint8")
00012 
00013 pp = np.linspace(pmin, pmax, ppoints)
00014 qq = np.linspace(qmin, qmax, qpoints)
00015 print("hi")
00016 
00017 for ip, p in enumerate(pp):
00018     for iq, q in enumerate(qq):
00019         z = 0
00020         c = p + 1j * q
00021         for k in range(max_iterations):
00022             z = z**2 + c
```

```
00023             if abs(z) > infinity_border:
00024                 image[iq,ip] = k * 20
00025                 break
00026
00027 cv2.imshow("img", image)
00028
00029 while True:
00030     k = cv2.waitKey(0) & 0xFF
00031     if k == ord("q"):
00032         break
```

7.13 Файл mainpage.dox