



## ALL Practice - Xử Lý Ảnh

tin học căn bản (Trường Đại học Cần Thơ)



Scan to open on Studocu

# Bài thực hành buổi 1

## I. Mục đích:

Cài đặt và Làm quen với QT Creator

Cài đặt tính năng load ảnh và chỉnh sửa độ sáng tối, ảnh âm bảng và phân ngưỡng ảnh.

## II. QT Creator

### 1. Download và cài đặt

Qt là ứng dụng cross-platform và là framework giao diện người dùng (UI) cho các nhà phát triển sử dụng ngôn ngữ C++ hoặc QML (ngôn ngữ kiểu CSS & JavaScript). Thế mạnh của Qt là đồ họa và xử lý ảnh. OpenGL cũng được tích hợp vào trong Qt tạo nên một thư viện hoàn hảo về đồ họa máy tính. Đi kèm với Qt là Qt Creator, một môi trường phát triển tích hợp hỗ trợ cho phát triển các ứng dụng Qt. Qt được sử dụng trong các phần mềm KDE trên các hệ thống GNU/Linux. Plasma

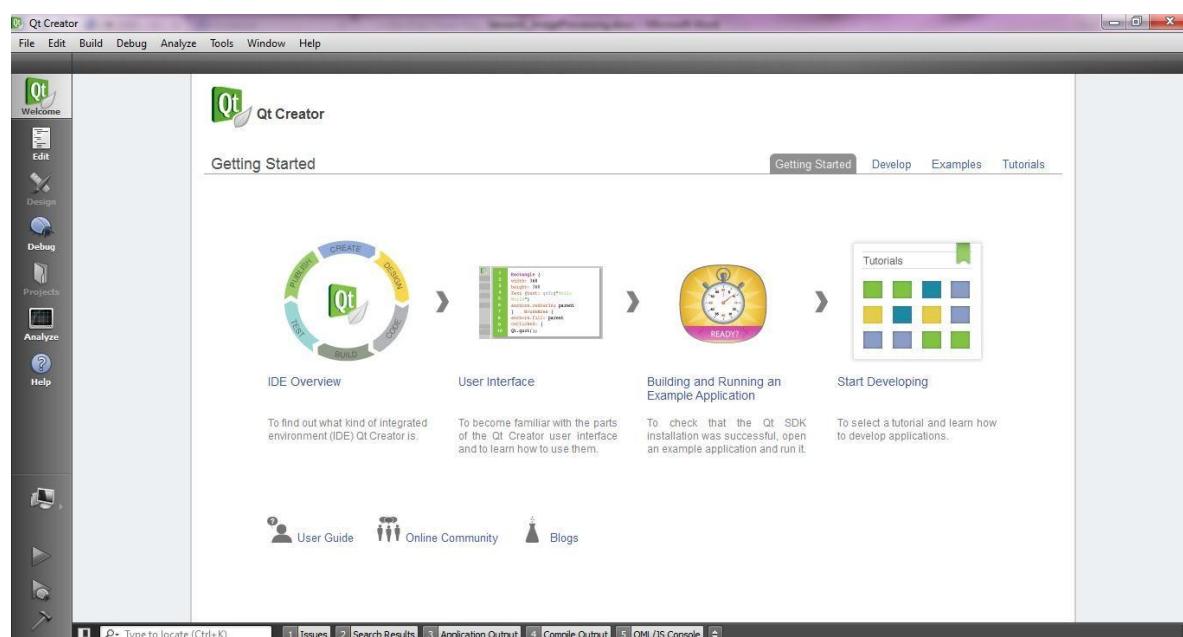


Workspaces là một ví dụ.

Download Qt từ trang <http://qt-project.org/downloads>. Phiên bản sử dụng trong môn học này của Qt là Qt 5.0. Trên Windows, nên sử dụng bản Qt 5.0.1 for Windows 32-bit (MinGW 4.7, 823 MB). Bản cài đặt này bao gồm thư viện Qt và môi trường phát triển tích hợp (IDE) Qt Creator.

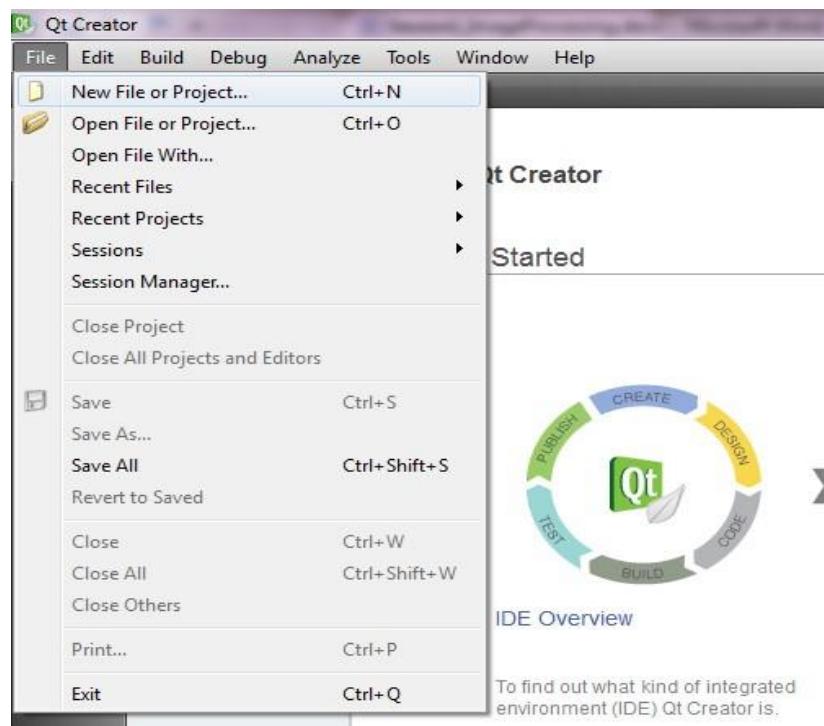
### 2. Sử dụng Qt Creator

Giao diện chính của Qt Creator như hình bên dưới.

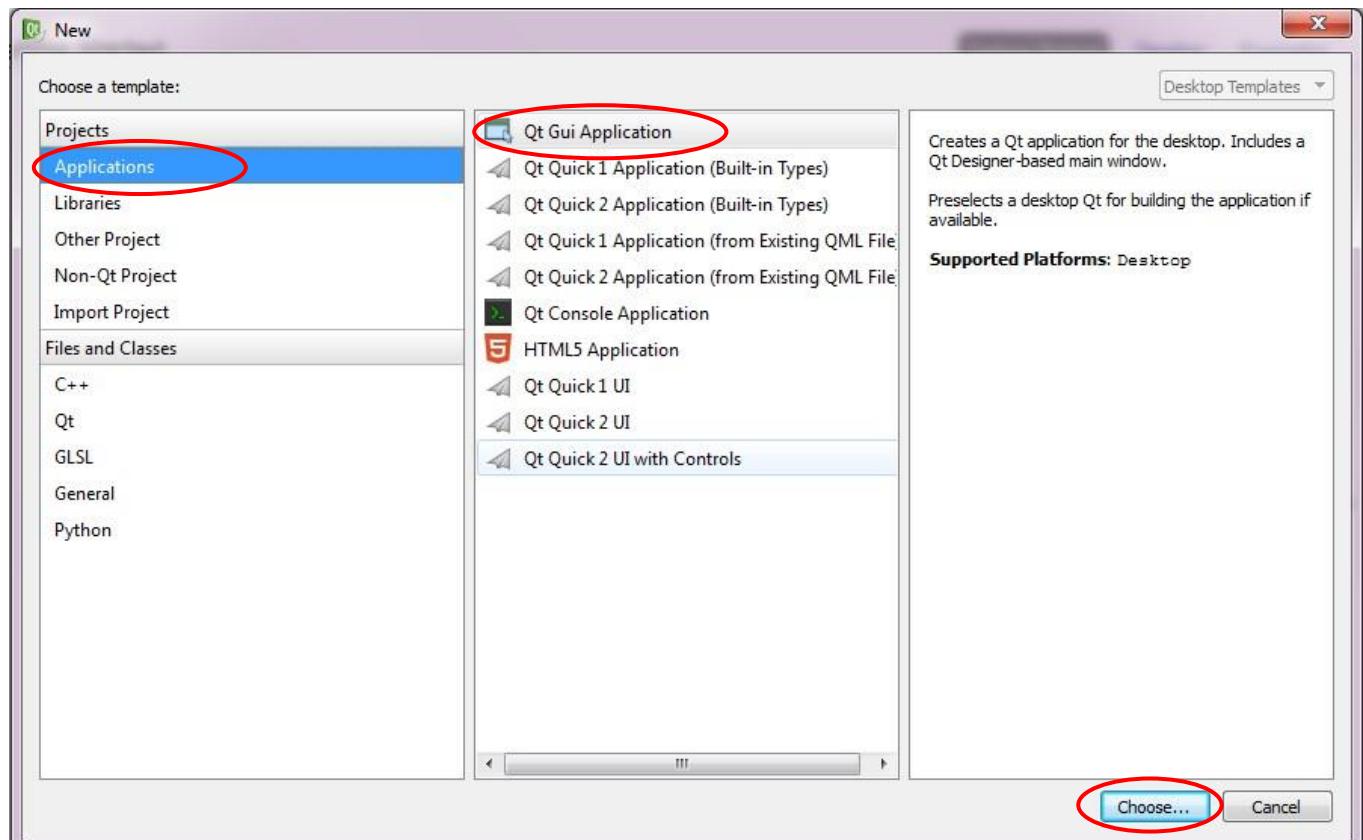


#### a. Tạo dự án mới (create Project)

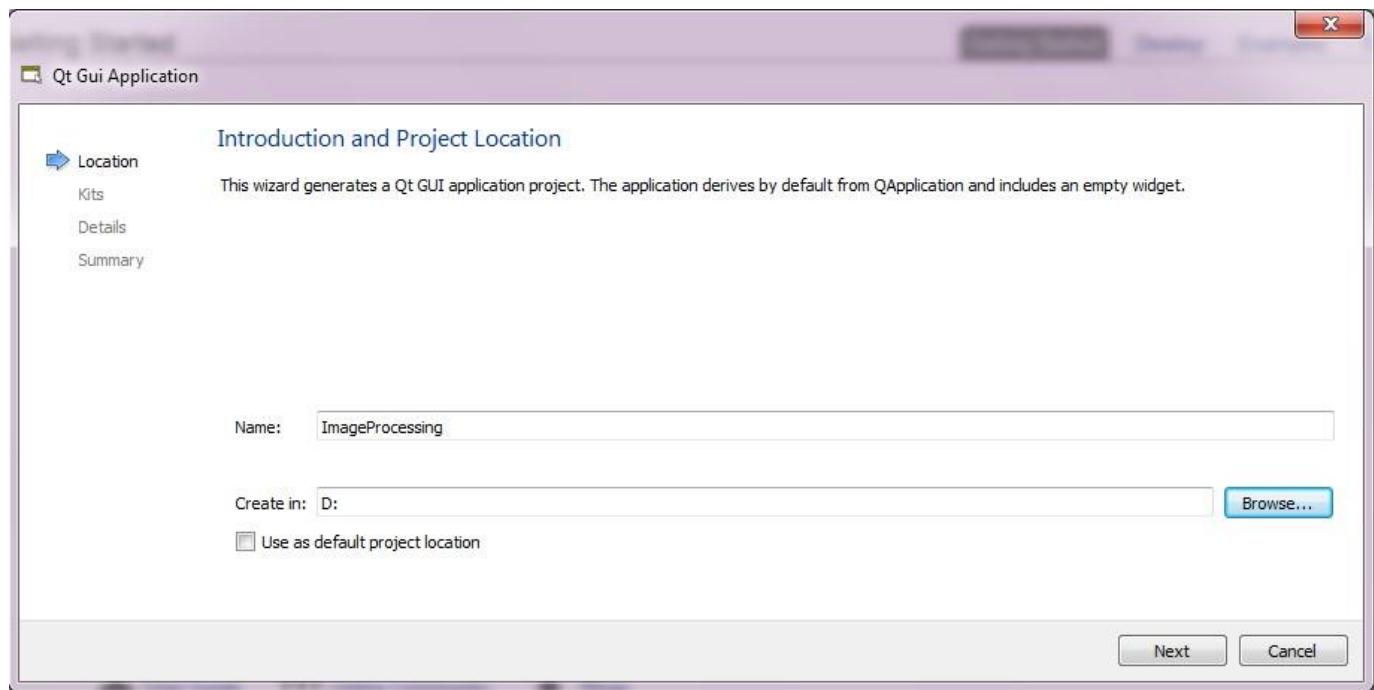
Click File => New File or Project (Ctrl+N)



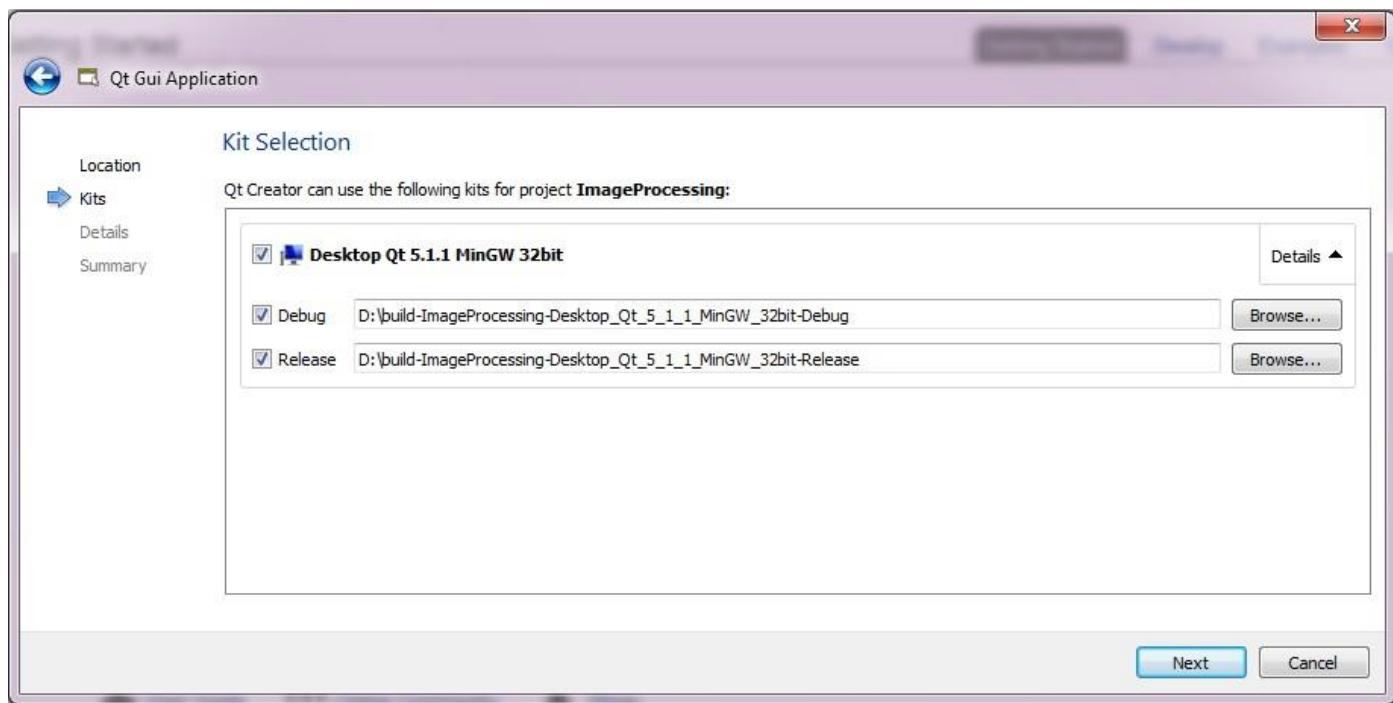
Chọn mẫu (template): **Qt C++ Project, Qt Gui Application**, kế đến click **Choose...**



Nhập tên dự án (Name) và thư mục chứa dự án (Create in), sau đó click **Next**.



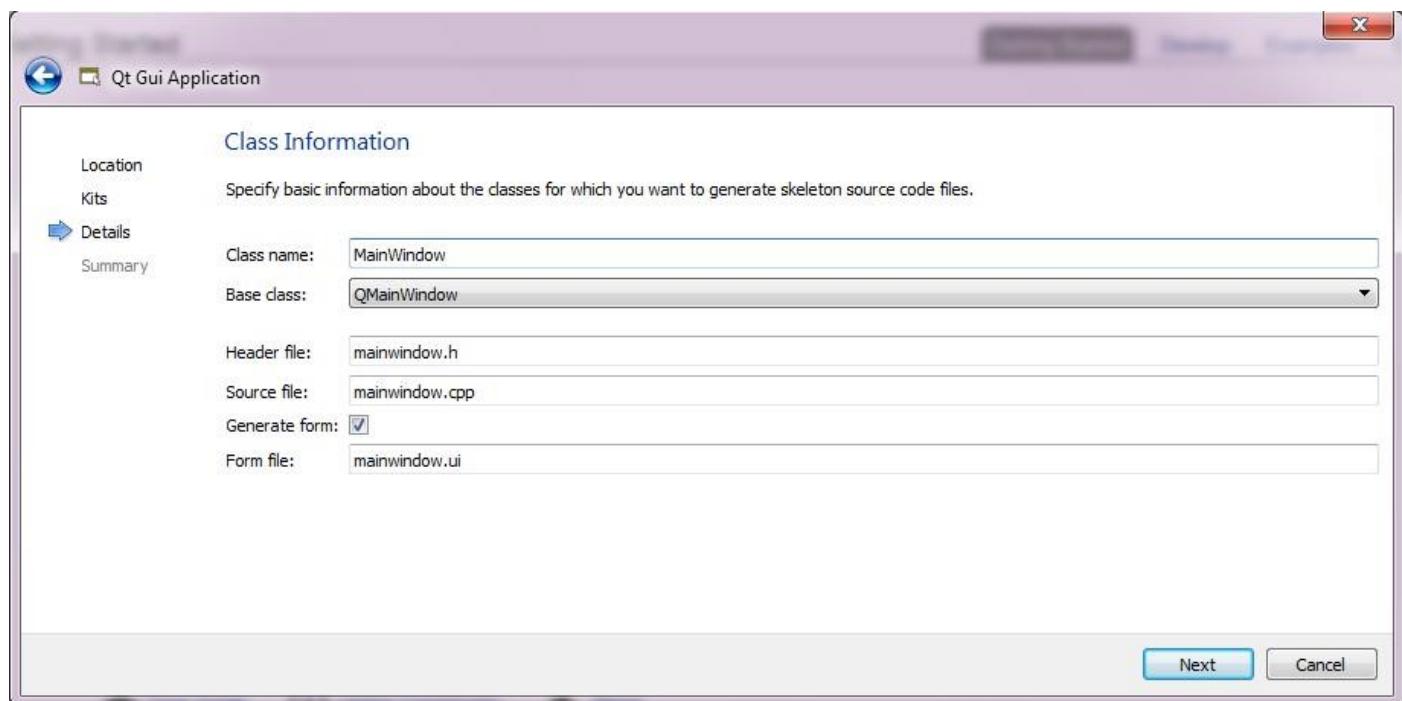
Chọn thư mục để Debug và Release.



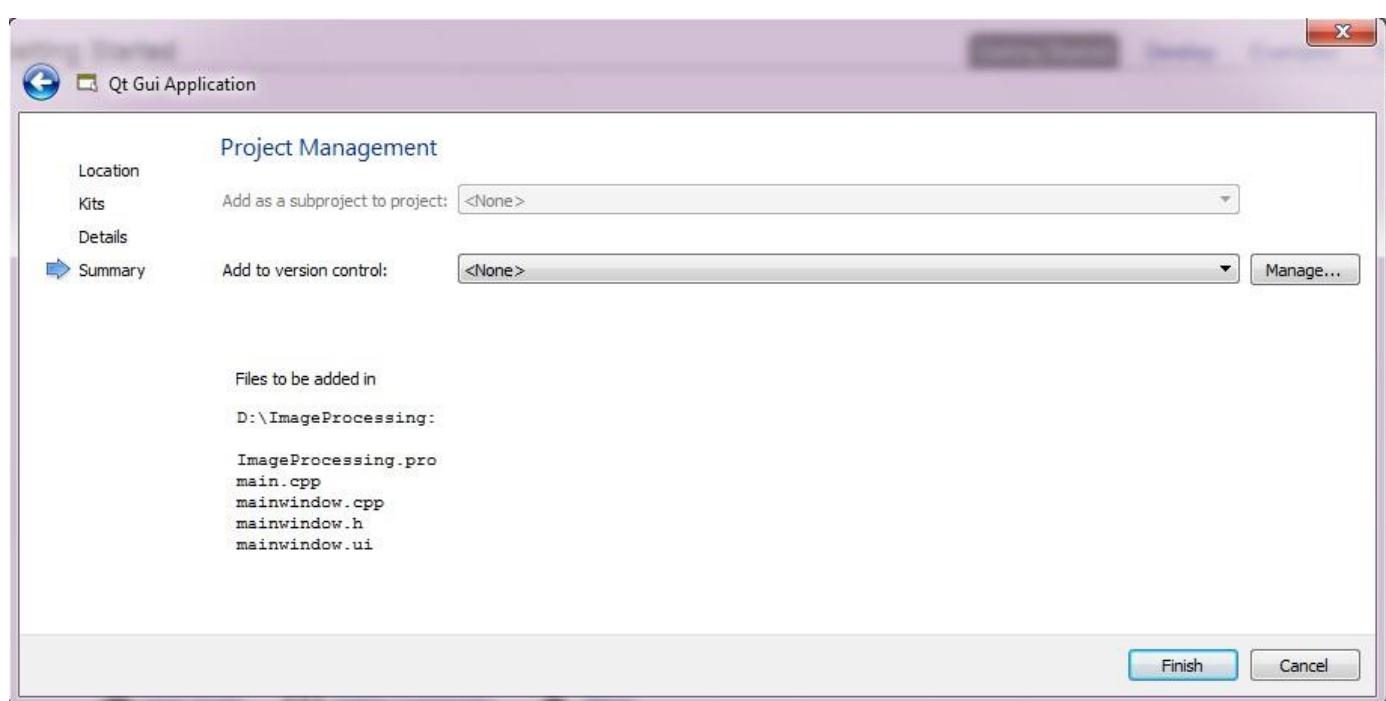
Chọn kiểu giao diện. Qt Creator cung cấp 3 kiểu ứng dụng giao diện đồ họa chính thừa kế từ các lớp cơ sở sau:

- **QMainWindow:** Kiểu giao diện này thường được sử dụng, giao diện gồm có Menu, Thanh công cụ (Toolbar), Cửa sổ trung tâm (Center Widget) và Thanh trạng thái (StatusBar)
- **QDialog:** Giao diện ứng dụng kiểu hộp thoại.
- **QWidget:** (Dành cho người dùng chuyên nghiệp), người dùng có thể tùy biến giao diện theo ý thích, nhưng phải lập trình khá nhiều.

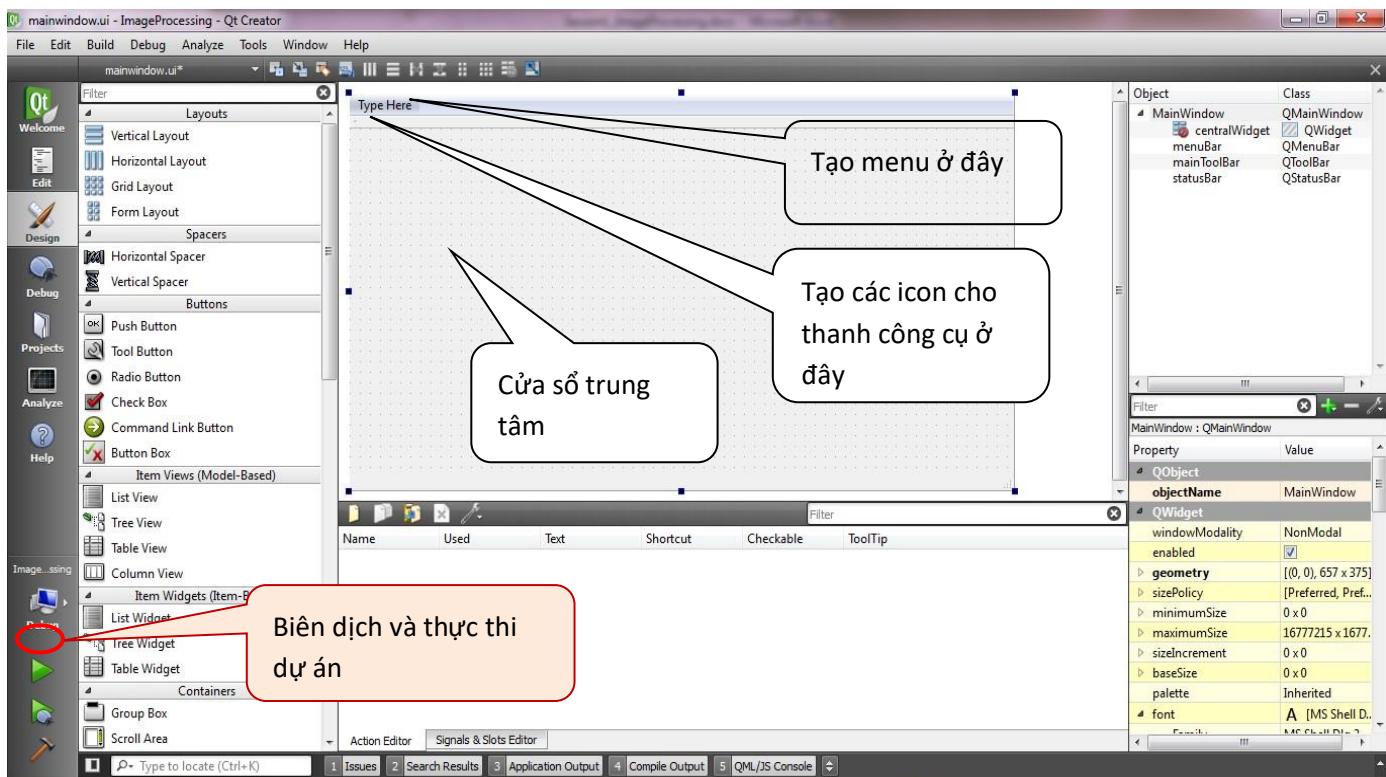
Nhập tên lớp cửa sổ chính (Class name): mặc định Qt Creator đặt tên lớp chính là MainWindow nếu lớp cơ sở là QMainWindow.



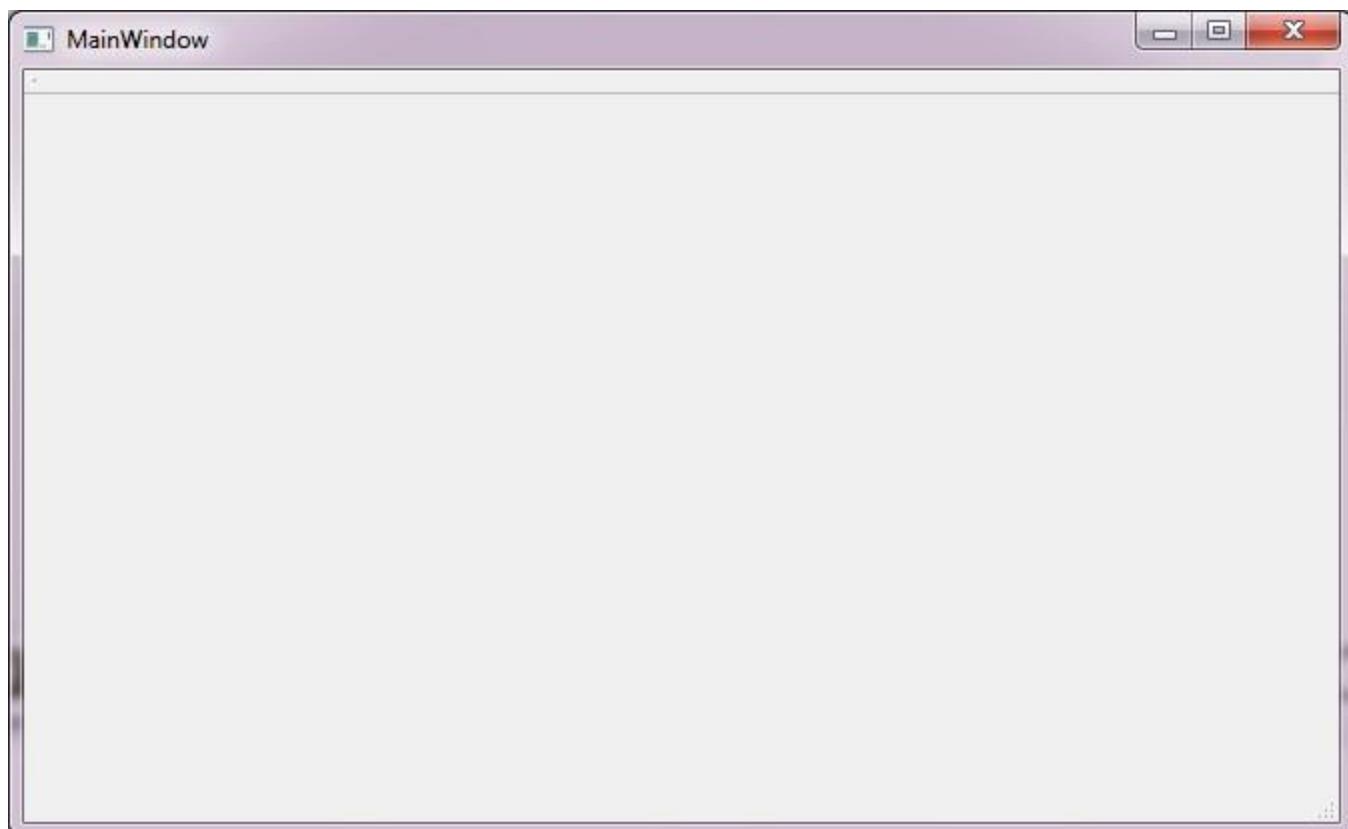
Click Next, chọn bộ quản lý phiên bản (version control). Có thể bỏ qua bước này.



Click **Finish** là hoàn tất.



Kết quả thực thi dự án: Ta được ứng dụng gồm khung Window, Thanh công cụ, cửa sổ trung tâm và thanh trạng thái.



## b. Tổ chức của dự án

Đóng giao diện, click icon **Edit** bên trái để xem tổ chức dự án.

```

1 #include "mainwindow.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7     MainWindow w;
8     w.show();
9
10    return a.exec();
11}
12

```

Dự án gồm các file sau:

- **ImageProcessing.pro**: file chính của dự án
- **mainwindow.ui**: định nghĩa giao diện giao diện đồ họa (theo cấu trúc XML)
- **mainwindow.h**: định nghĩa lớp MainWindow
- **main.cpp**: chứa hàm main()
- **mainwindow.cpp**: cài đặt các hàm có trong lớp MainWindow

## File ImageProcessing.pro

```

1 -----
2 #
3 # Project created by QtCreator 2015-09-17T18:46:17
4 #
5 -----
6
7 QT      += core gui
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = ImageProcessing
12 TEMPLATE = app
13
14
15 SOURCES += main.cpp \
16             mainwindow.cpp
17
18 HEADERS  += mainwindow.h
19
20 FORMS    += mainwindow.ui
21

```

Sử dụng 2 module:  
core và gui

Tên file thực thi

File cài đặt

File định nghĩa

File định nghĩa giao diện

## Filemainwindow.ui

The screenshot shows the Qt Creator interface with the mainwindow.ui file open in the central editor area. The code is XML-based UI definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>MainWindow</class>
<widget class="QMainWindow" name="MainWindow">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>657</width>
<height>375</height>
</rect>
</property>
<property name="windowTitle">
<string>MainWindow</string>
</property>
<widget class="QWidget" name="centralWidget">
<widget class="QMenuBar" name="menuBar">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>657</width>
<height>21</height>
</rect>
</property>
</widget>
</widget>
```

Khi biên dịch Qt sẽ sinh ra lớp Ui::MainWindow được dùng trong lớp MainWindow.

## Filemainwindow.h

The screenshot shows the Qt Creator interface with the mainwindow.h file open in the central editor area. The code defines a MainWindow class that inherits from QMainWindow. A callout box highlights the declaration of the ui::MainWindow pointer:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

**Định nghĩa lớp  
MainWindow kế thừa từ  
lớp QMainWindow**

**Biến ui dùng để truy xuất các  
thành phần giao diện (widget)  
như nút nhấn (QPushButton),  
nhãn (QLabel), ... có trong giao  
diện**

**This document is available free of charge on** **studocu**  
Downloaded by Ly Hoan (xqvbfxdlyvbgynr@hoavan.edu.vn)

## Filemainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

Tạo một đối tượng  
Ui::MainWindow gán nó cho  
biến ui. Biến này sẽ được sử  
dụng để truy xuất các thành  
phần có trong giao diện

Thu hồi vùng nhớ đã cấp  
phép cho ui.

## File main.cpp

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Tạo cửa sổ chính.

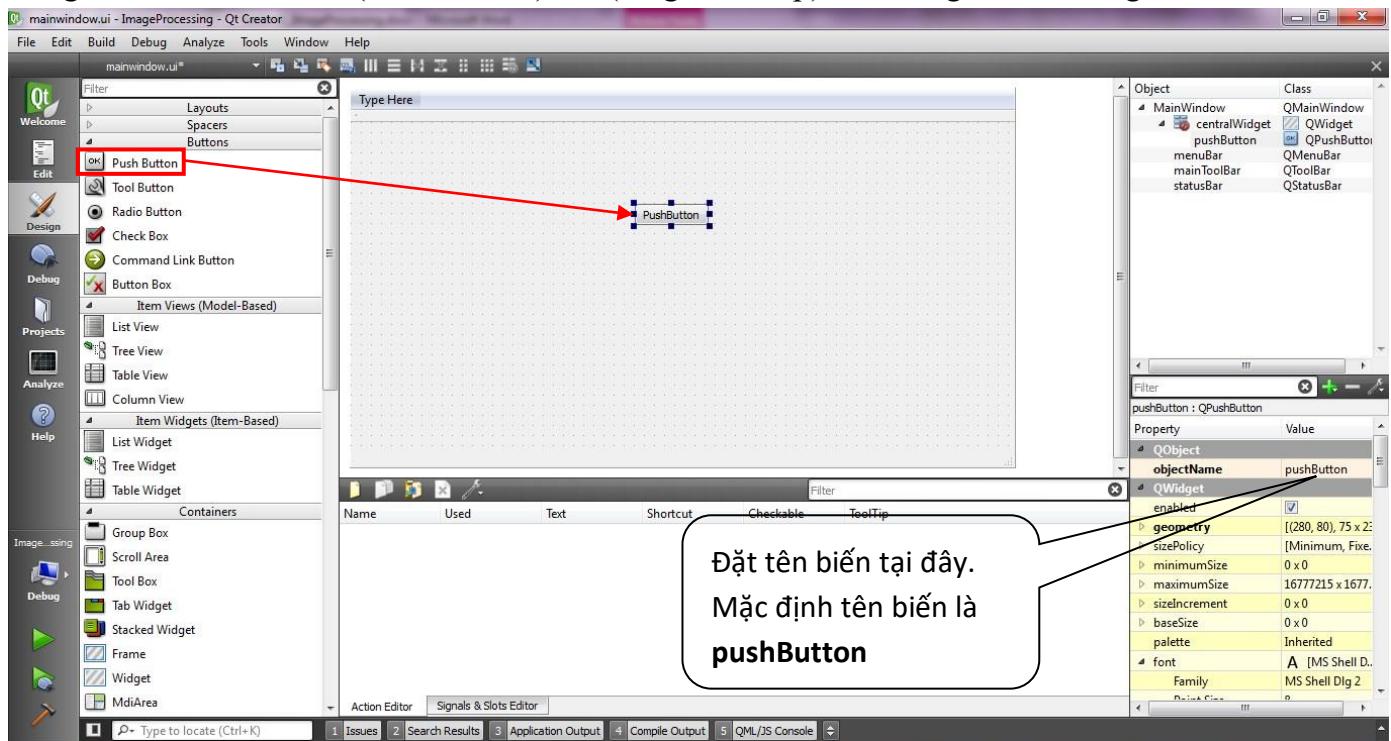
Hiển thị cửa sổ.

Vào vòng lặp sự kiện. Chờ cho đến khi  
cửa sổ cuối cùng được đóng thì thoát.

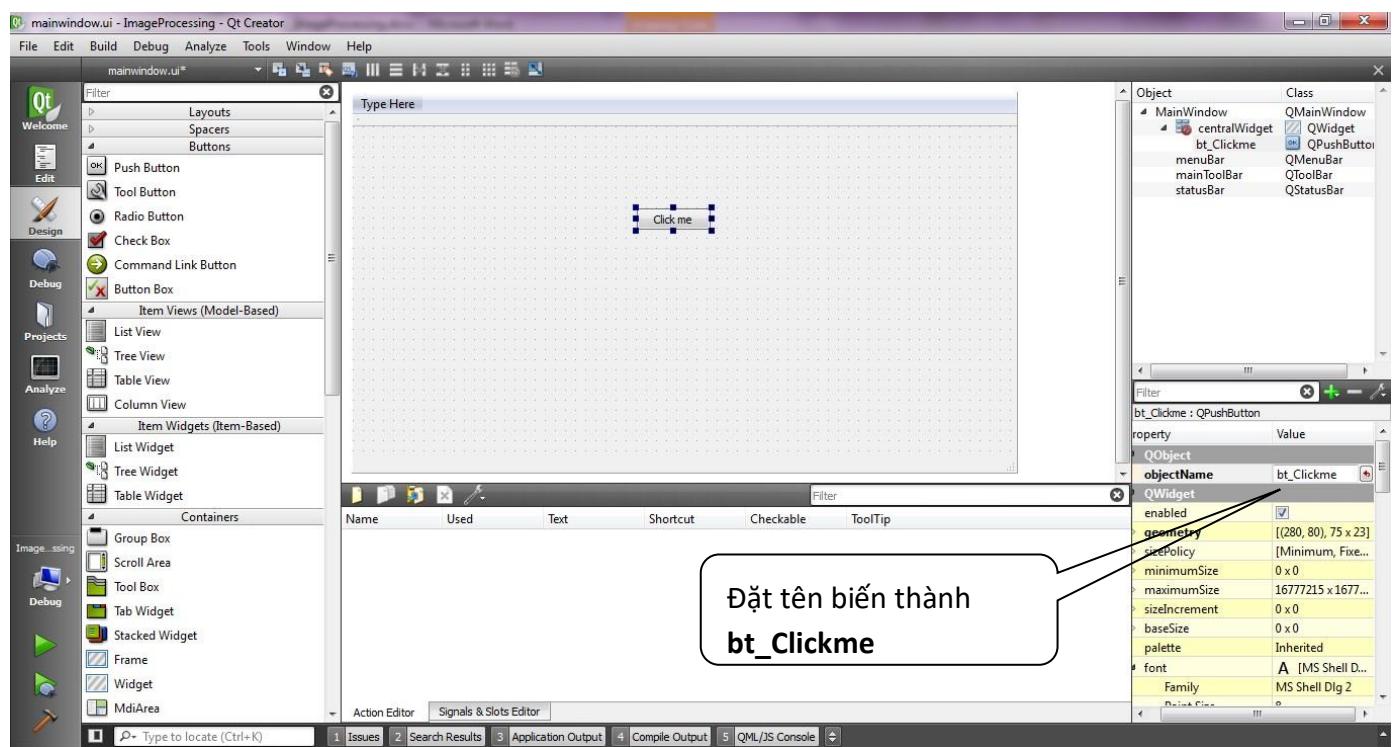
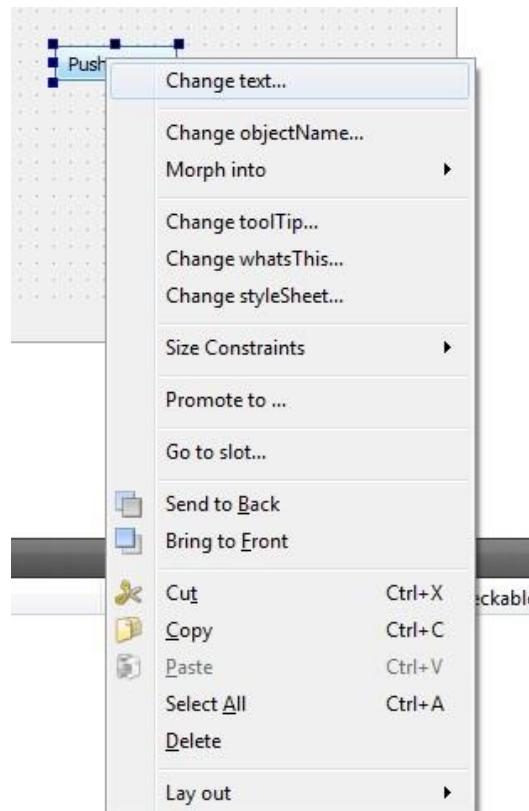
### c. Thiết kế giao diện và xử lý sự kiện

Trong phần này ta sẽ thiết kế một giao diện gồm một nút nhán (PushButton) có nhãn: “Click me!”, có tên (name): btnClick, khi click chuột vào sẽ hiện hộp thông báo “Hello world!”

**Thiết kế giao diện:** mở file **mainwindow.ui** ở chế độ Design, thêm một nút nhấn vào khung cửa sổ trung tâm. Kéo nút nhấn (Push Button) thả (drag and drop) vào trong cửa sổ trung tâm.



Click phải trên nút nhấn, chọn **Change text...** để thay đổi nhãn của nút nhấn.



Đổi nhãn thành Click me

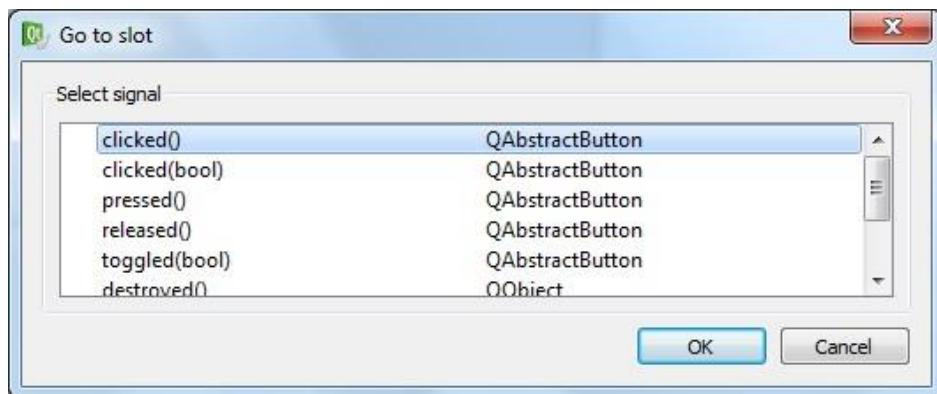
Xử lý sự kiện: Qt cung cấp cơ chế lập trình hướng sự kiện thông qua: **SIGNAL** và **SLOT**.

- **SIGNAL:** Sự kiện. Sự kiện có thể là sự kiện của hệ thống như click chuột, di chuyển chuột, bấm phím, ... Sự kiện cũng có thể do người lập trình tự định nghĩa và kích hoạt trong chương trình.

- SLOT: Hàm xử lý sự kiện. Về bản chất SLOT cũng là một phương thức (method). Để khai báo một phương thức là một slot ta thêm phát biểu **public slots:** hoặc **private slots:** trước phương thức tương ứng.

SIGNAL và SLOT được gắn kết với nhau qua hàm **connect**.

Để gắn kết sự kiện do một thành phần có trong giao diện phát ra, click phải trên thành phần đó, chọn **Go to slot...**, trong ví dụ này là click phải trên nút nhấn, chọn **Go to slot...**



Chọn tín hiệu/sự kiện **clicked()**. Sự kiện này xảy ra người dùng click chuột lên nút nhấn hoặc nhấn ENTER khi nút nhấn đang được kích hoạt.

```

mainwindow.cpp - ImageProcessing - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects ImageProcessing Headers Sources Forms
mainwindow.h mainwindow.cpp mainwindowui.h
mainwindow.ui
mainwindow.cpp

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9 }
10
11 ~MainWindow()
12 {
13     delete ui;
14 }
15
16 void MainWindow::on_bt_Clickme_clicked()
17 {
18 }
19
20

```

Xem lại file **mainwindow.h** ta sẽ thấy Qt đã tự động thêm vào một slot tên **on\_bt\_Clickme\_clicked()**. Việc kết nối SIGNAL **clicked()** và SLOT **on\_bt\_Clickme\_clicked()** được thực hiện tự động khi ta gọi hàm **ui->setupUi(this)**.

mainwindow.h - ImageProcessing - Qt Creator

```

File Edit Build Debug Analyze Tools Window Help
Projects mainwindow.h ui:Ui::MainWindow *
Qt Welcome
Edit
Design
Debug
Projects
Analyze
Help
Image_using
Debug
Projects
Analyze
Help
Open Documents ImageProcessing.pro
main.cpp
mainwindow.cpp*
mainwindow.h*
mainwindow.ui*

```

```

1 ifndef MAINWINDOW_H
2 define MAINWINDOW_H
3
4 #include <QMainWindow>
5
6 namespace Ui {
7     class MainWindow;
8 }
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     explicit MainWindow(QWidget *parent = 0);
16     ~MainWindow();
17
18 private slots:
19     void on_bt_Clickme_clicked();
20
21 private:
22     Ui::MainWindow *ui;
23 }
24
25 #endif // MAINWINDOW_H
26

```

Issues Search Results Application Output Compile Output QML/JS Console

Giờ ta quay lại file **mainwindow.cpp** để định nghĩa code xử lý cho SLOT **on\_bt\_Clickme\_clicked()**

mainwindow.cpp - ImageProcessing - Qt Creator

```

File Edit Build Debug Analyze Tools Window Help
Projects mainwindow.cpp* MainWindow::on_bt_Clickme_clicked(): void
Qt Welcome
Edit
Design
Debug
Projects
Analyze
Help
Image_using
Debug
Projects
Analyze
Help
Open Documents ImageProcessing.pro
main.cpp
mainwindow.cpp*
mainwindow.h*
mainwindow.ui*

```

```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include " QMessageBox "
4
5 MainWindow::MainWindow(QWidget *parent) :
6     QMainWindow(parent),
7     ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10
11     MainWindow::~MainWindow()
12     {
13         delete ui;
14     }
15
16     void MainWindow::on_bt_Clickme_clicked()
17     {
18         QMessageBox::information(this, "Information", "Hello World");
19     }
20
21

```

include file QMessageBox để có thể sử dụng lớp QMessageBox

Gọi hàm information của QMessageBox để hiện câu thông báo.

Issues Search Results Application Output Compile Output QML/JS Console

Kết quả như hình bên dưới.

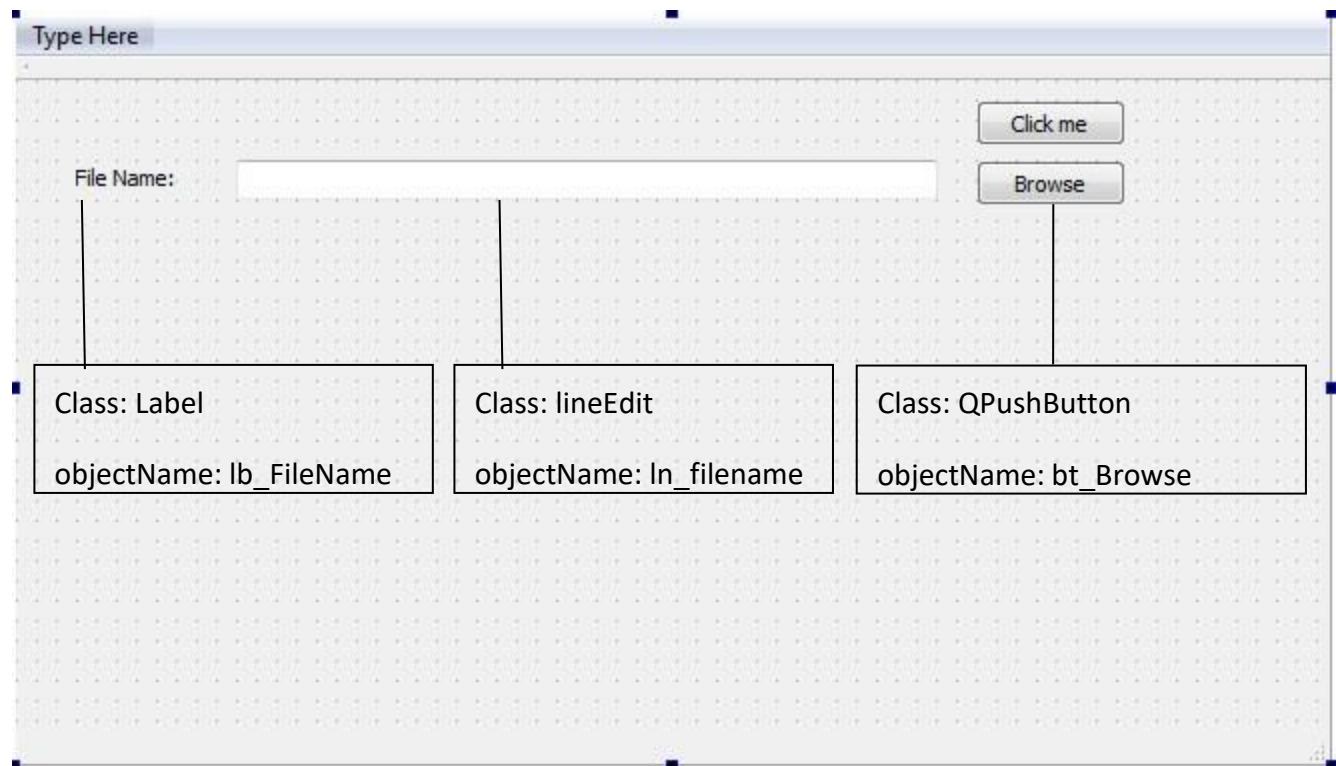


### 3. Đọc và hiển thị ảnh:

#### a. Đọc và hiển thị ảnh

Trong ví dụ này ta sẽ mở hộp thoại cho phép người dùng chọn một file ảnh, sau đó hiển thị ảnh lên một cửa sổ mới. Sử dụng lại dự án vừa rồi, ta thêm vào các đối tượng như hình bên dưới.

- Để mở hộp thoại cho phép chọn file ta sử dụng Hàm `getOpenFile()` của lớp `QFileDialog`.
- Để hiển thị ảnh, ta tạo một pixmap (kiểu `QPixmap`) từ file, tạo một label (kiểu `QLabel`), gán pixmap cho label và gọi hàm `show()` để hiển thị ảnh.



Code chương trình trong hình bên dưới. Mở file mainwindow.cpp và thêm vào các dòng lệnh sau:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include " QMessageBox"
#include "QFileDialog"
#include "QPixmap"
#include "QLabel"
```

Quay trở lại form giao diện đã thiết kế. Click chuột phải lên nút Browse => Go to slot => click() và thực hiện đoạn code sau cho nút Browse. Viết code cho phép mở hộp thoại chọn tên file và gán tên file vào ô ln\_filename. Để truy xuất một Widget có trong giao diện ta sử dụng biến ui. Hàm setText dùng để gán một chuỗi cho biến ln\_filename có kiểu QLineEdit.

```

void MainWindow::on_bt_Browse_clicked()
{
    QString filename = QFileDialog::getOpenFileName(this, "Open File",
                                                    "C:/CVIPtools/images",
                                                    "**.* All Files; ;*.bmp; ;*.jpeg; ;*.ppm; ;*.png; ;*.jpg");
    if (filename.isEmpty())
        return;
    ui->ln_filename->setText(filename);

    QPixmap pixmap(filename);
    QLabel *label = new QLabel();
    label->setPixmap(pixmap);
    label->setWindowTitle(QFileInfo(filename).fileName());
    label->show();
}

```

Viết hàm `DisplayImage(QImage &img, QString title)` để hiển thị ảnh khi cần thiết. Hàm `DisplayImage` gồm có 2 tham số: 1. `QImage &img` - ảnh cần hiển thị, `QString title` – tiêu đề file ảnh. Mở file mainwindow.h và thêm dòng lệnh như hình bên dưới:

```

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void DisplayImage(QImage &img, QString title);

```

Mở file mainwindow.cpp và thực hiện đoạn code sau cho hàm `DisplayImage(QImage &img, QString title)`.

```

void MainWindow::DisplayImage(QImage &img, QString title){
    QLabel *label = new QLabel();
    label->setPixmap(QPixmap::fromImage(img));
    label->setWindowTitle(title);
    label->show();
}

```

## b. Xử lý điểm ảnh

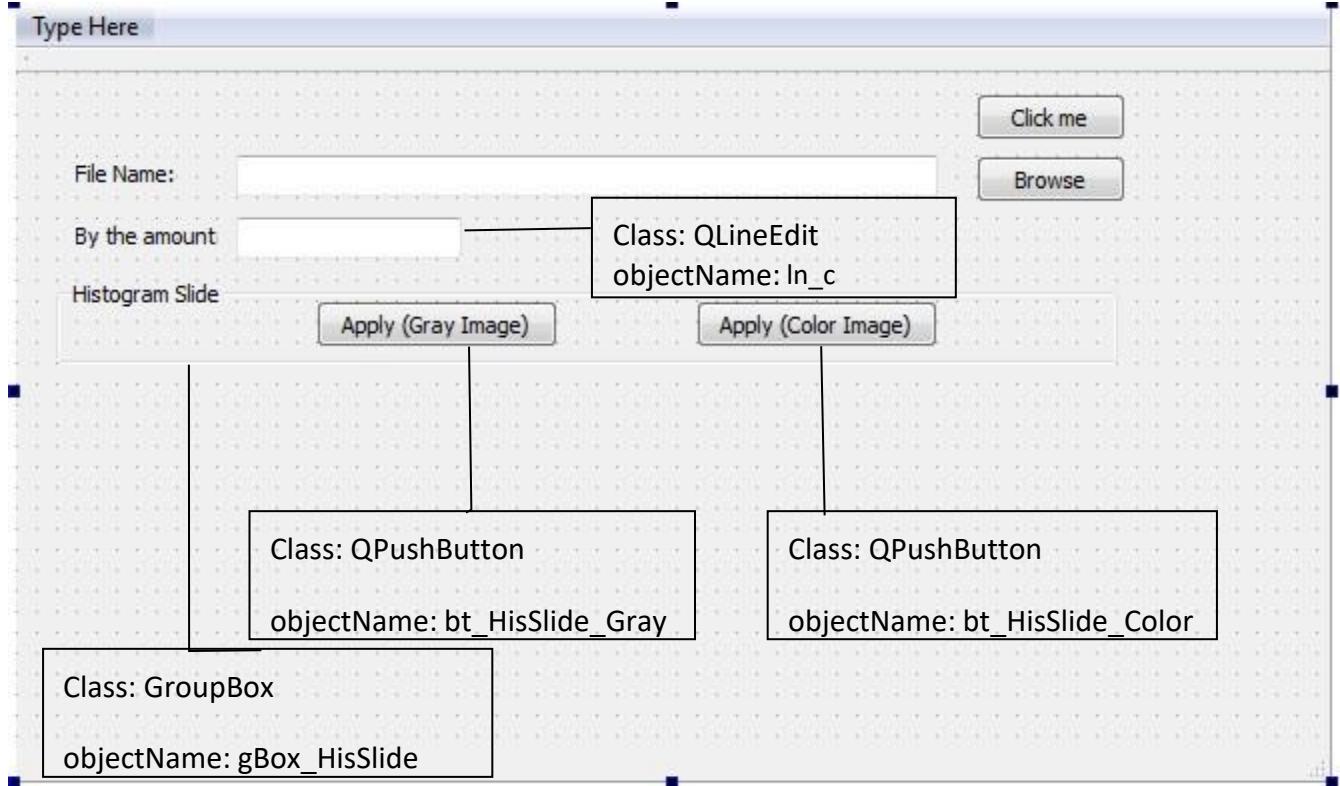
Lý thuyết: xem phần xử lý ảnh điểm.

Thực hành:

### - Thay đổi độ sáng tối của ảnh -

Tạo dự án mới tương tự như dự án trên -

Thiết kế giao diện như hình vẽ.



- Viết code xử lý làm thay đổi độ sáng tối của ảnh.

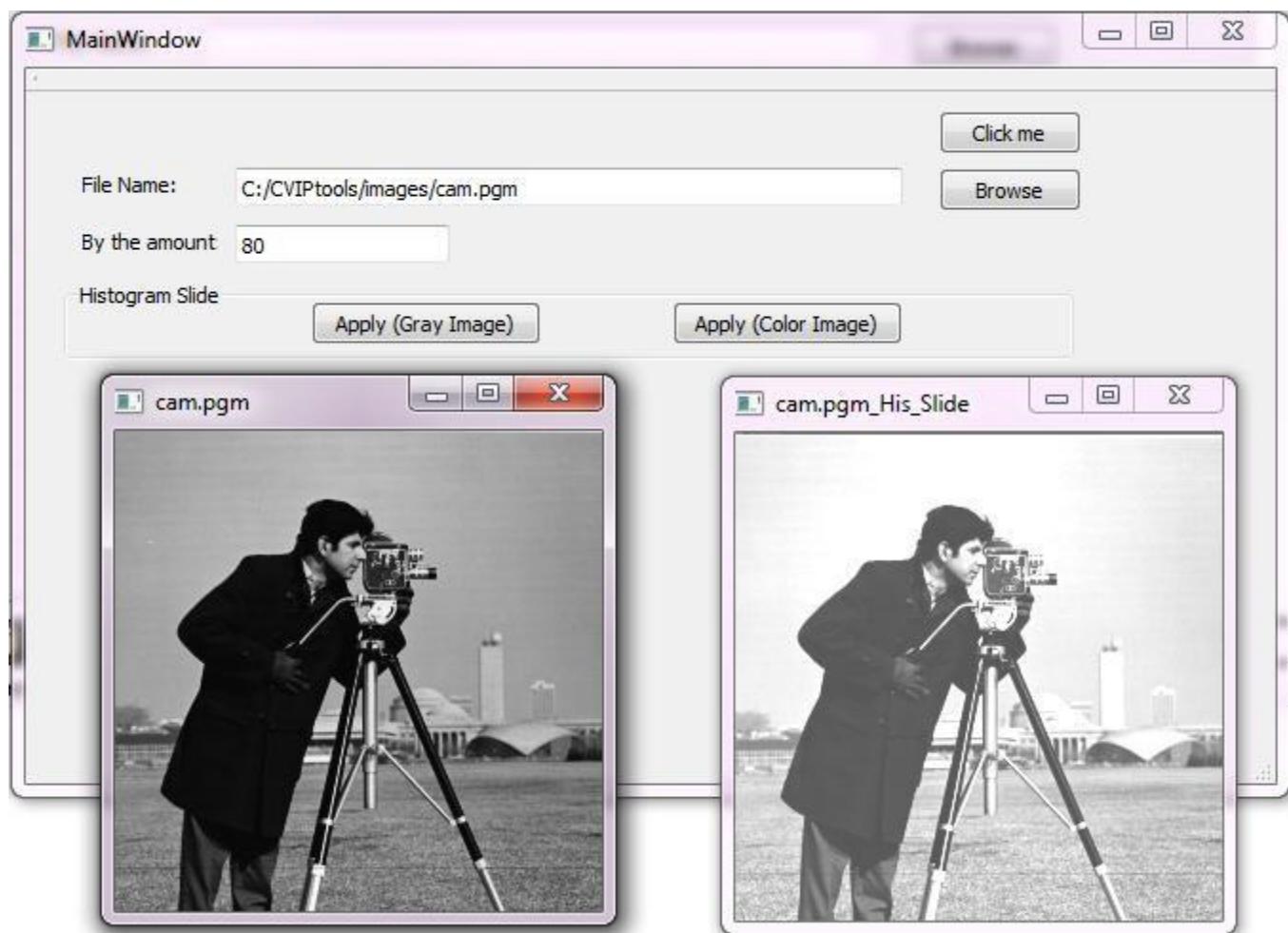
- Click phải trên nút Apply (Gray Image), Go to slot..., chọn sự kiện clicked(). Viết code xử lý và hiển thị ảnh xử lý. Duyệt từng điểm ảnh, lấy màu → chuyển thành mức xám, tăng giảm mức xám một lượng c (kiểm tra giới hạn 0..255), gán vào ảnh kết quả.

```

void MainWindow::on_bt_HisSlide_Gray_clicked()
{
    QString filename = ui->ln_filename->text();
    QImage image_in(filename); //Doc anh tu File
    QImage image_out(image_in.width(), image_in.height(), QImage::Format_ARGB32);
    int c = ui->ln_c->text().toInt(); //Doi chuoi thanh so
    for (int x=0; x<image_in.width(); x++)
        for (int y=0; y<image_in.height(); y++) {
            QRgb color = image_in.pixel(x, y); //Lay mau gia tri pixel(x,y)
            int gray_in = qGray(color); //Doi gia tri mau thanh muc xam
            int gray_out = gray_in + c; //Thay doi do sang toi
            if (gray_out>255) //Kiem tra gia muc xam 0...255
                gray_out=255;
            else if (gray_out<0)
                gray_out=0;
            QRgb color_out = qRgb(gray_out, gray_out, gray_out); //Anh xam (R=G=B)
            image_out.setPixel(x, y, color_out);
        }
    DisplayImage(image_in, QFileInfo(filename).fileName());
    DisplayImage(image_out, QFileInfo(filename).fileName()+"_His_Slide");
}

```

## Kết quả chạy thử chương trình



Ảnh gốc

Ảnh xử lý (trượt về phải 80)

Chương trình trên chỉ có thể xử lý và cho ra kết quả là ảnh xám. Ta sẽ cải tiến để có thể xử lý được ảnh màu.

**Ý tưởng:** Đổi không gian màu từ RGB  $\rightarrow$  HSV, xử lý thành phần độ sáng V (tăng/giảm V), kết hợp lại với H và S cũ  $\rightarrow$  RGB.

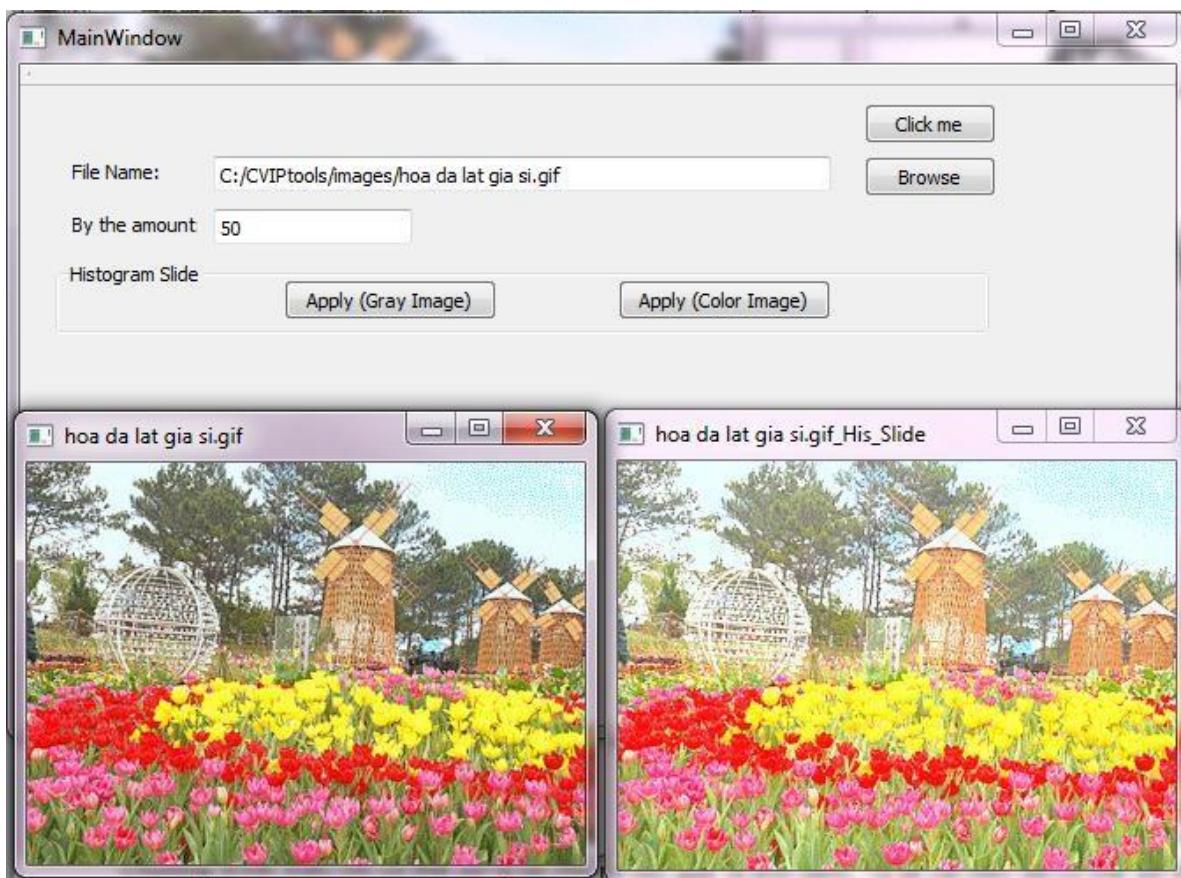
Code xử lý:

```

void MainWindow::on_bt_HisSlide_Clicked()
{
    QString filename = ui->ln_filename->text();
    QImage image_in(filename);
    QImage image_out(image_in.width(),image_in.height(),QImage::Format_ARGB32);
    int c = ui->ln_c->text().toInt();
    for (int x=0; x<image_in.width(); x++)
        for (int y=0; y<image_in.height(); y++) {
            QRgb rgb = image_in.pixel(x, y);
            QColor color_in(rgb);
            int h, s, v;
            color_in.getHsv(&h, &s, &v); //Lay 3 thanh phan h, s, v tu color
            int v_out = v + c;
            if (v_out>255)
                v_out=255;
            else if (v_out<0)
                v_out=0;
            QColor color_out = QColor::fromHsv(h, s, v_out);
            image_out.setPixel(x, y, color_out.rgb());
        }
    DisplayImage(image_in,QFileInfo(filename).fileName());
    DisplayImage(image_out,QFileInfo(filename).fileName()+"_His_Slide");
}

```

Kết quả chạy thử



#### 4. Bài tập :

Tạo một dự án mới. Thiết kế giao diện như hình bên dưới và thực hiện code cho chức năng:

1. Tạo ảnh âm bản cho ảnh xám và ảnh màu

Ví dụ : tạo ảnh âm bản cho ảnh xám :



Ví dụ : tạo ảnh âm bản cho ảnh màu :



2. Phân ngưỡng ảnh. Gợi ý: nhập vào giá trị ngưỡng, duyệt qua từng điểm ảnh, nếu giá trị điểm ảnh > giá trị ngưỡng => 255, và ngược lại đặt về 0.

Ví dụ về phân ngưỡng ảnh xám



Với những giá trị ngưỡng khác nhau sẽ cho ra các kết quả khác nhau.

## Giao diện mẫu cho bài thực hành 1 – 2 – 3 – 4

MainWindow

Xử lý 1 ảnh

Tên file ảnh  Chọn

Ảnh âm bản

Mức sáng  > 0 tăng; < 0 giảm

Độ tương phản  > 1 tăng; 0 < c < 1 giảm

Chọn ngưỡng  > n trắng; < n đen

Xử lý 2 ảnh

Chọn ảnh 1  
 Chọn ảnh 2

Opacity ảnh 1  Cộng 2 ảnh  Trừ 2 ảnh

Opacity ảnh 2

# THỰC HÀNH BUỔI 2

## 1) Sửa bài tập thực hành buổi 1

### 1.1 Ảnh âm bản xám

Thêm đoạn chương trình sau vào nút xử lý của button Ảnh âm bản xám :

```
void MainWindow::on_btn_amban_xam_clicked()
{
    QImage image_in(ui->LineFileName->text());
    QImage image_out(image_in.width(),image_in.height(),QImage::Format_RGB32);

    for(int i = 0; i < image_in.width(); i++)
    {
        for(int j = 0; j < image_in.height(); j++)
        {
            QRgb rgb = image_in.pixel(i,j);
            int gray = qGray(rgb);
            QRgb color_in = qRgb(gray,gray,gray);
            image_in.setPixel(i,j,color_in);
            int out = 255 - gray;
            if(out > 255) out = 255;
            else if(out < 0) out = 0;

            QRgb color_out = qRgb(out,out,out);
            image_out.setPixel(i,j,color_out);
        }
    }

    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in));
    label_in->show();

    QLabel* label_out = new QLabel();
    label_out->setPixmap(QPixmap::fromImage(image_out));
    label_out->show();
}
```

### 1.2 Ảnh âm bản màu

Thêm đoạn chương trình sau vào nút xử lý của button Ảnh âm bản màu :

```
void MainWindow::on_btn_amban_mau_clicked()
{
    QImage image_in(ui->LineFileName->text());
    QImage image_out(image_in.width(),image_in.height(),QImage::Format_RGB32);

    for(int i = 0; i < image_in.width(); i++)
    {
        for(int j = 0; j < image_in.height(); j++)
        {
            QRgb rgb = image_in.pixel(i,j);
            QColor color(rgb);
            int red = color.red();
            int green = color.green();
            int blue = color.blue();
            int n_red = 255 - red;
            int n_green = 255 - green;
            int n_blue = 255 - blue;
            QRgb color_out = qRgb(n_red,n_green,n_blue);
            image_out.setPixel(i,j,color_out);
        }
    }

    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in));
    label_in->show();

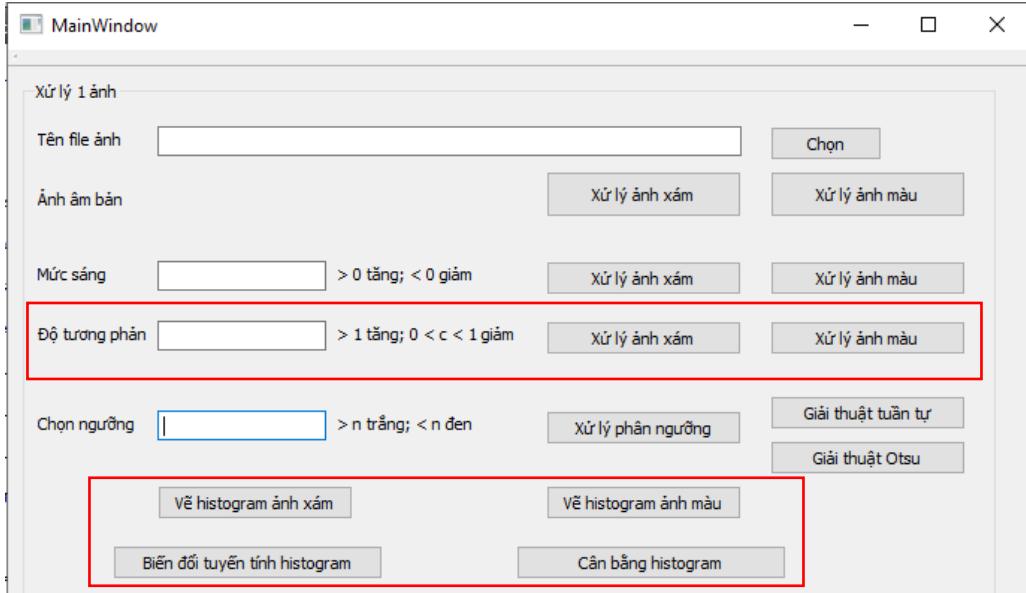
    QLabel* label_out = new QLabel();
    label_out->setPixmap(QPixmap::fromImage(image_out));
    label_out->show();
}
```

## 2.1 Phân ngưỡng ảnh xám :

Thêm đoạn chương trình sau vào nút xử lý của button phân ngưỡng :

```
QImage image_in(ui->LineFileName->text());  
  
QImage image_out(image_in.width(),image_in.height(),QImage::Format_RGB32);  
  
int c = ui->LinePhanNguong->text().toInt();  
  
for(int i = 0; i < image_in.width(); i++)  
{  
    for(int j = 0; j < image_in.height(); j++)  
    {  
        QRgb rgb = image_in.pixel(i,j);  
        int gray = qGray(rgb);  
        QRgb color_in = qRgb(gray,gray,gray);  
        image_in.setPixel(i,j,color_in);  
        int out = gray;  
        if(out > c) out = 255;  
        else if(out < c) out = 0;  
  
        QRgb color_out = qRgb(out,out,out);  
        image_out.setPixel(i,j,color_out);  
    }  
}  
  
QLabel* label_in = new QLabel();  
label_in->setPixmap(QPixmap::fromImage(image_in));  
label_in->show();  
  
QLabel* label_out = new QLabel();  
label_out->setPixmap(QPixmap::fromImage(image_out));  
label_out->show();
```

## 2) Nội dung thực hành buổi 2 :



Trong buổi thực hành này, chúng ta sẽ tiến hành thiết kế tính năng cho các Button như hình trên.

### 2.1 Histogram cho ảnh xám :

Chúng ta thêm vào chương trình một hàm Histogram\_GrayImage() để tính toán và vẽ tổ chức đồ ảnh ảnh xám như bên dưới.

```

void MainWindow::Histogram_GrayImage(const QImage &img, const int &his_height){

    int h[256];
    for (int i=0; i<256; i++)
        h[i]=0;

    QImage image_in(img);
    for (int x=0; x<img.width(); x++)
        for (int y=0; y<img.height(); y++){
            QRgb color = img.pixel(x,y);
            int gray = qGray(color);
            QRgb color_in = qRgb(gray,gray,gray);
            image_in.setPixel(x,y,color_in);
            h[gray]++;
        }
    int max=0;
    for (int i=0; i<256; i++)
        if (h[i]>max) max=h[i];
    QImage img_his = QImage(256,his_height,QImage::Format_RGB888);
    img_his.fill(Qt::white);

    int lineHeight;
    for (int i=0; i<256; i++) {
        lineHeight = his_height * h[i]/max; // tinh tien anh de histogram co do lon nhu mong muon
        for (int j=his_height-1; j>=his_height - 1 - lineHeight; j--) // toa do 0,0 nam phia tren ben trai
            img_his.setPixel(i,j,qRgb(0,0,0));
    }

    //QImage image_out(img.width(),img.height(),QImage::Format_RGB888);
    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in));
    label_in->show();

    // hien thi histogram
    QLabel* label_out = new QLabel();
    label_out->setPixmap(QPixmap::fromImage(img_his));
    label_out->show();
}

}

```

Sau đó tại sự kiện clicked của button vẽ tổ chức đồ cho ảnh xám, ta gọi hàm vừa cài đặt lên như sau :

```

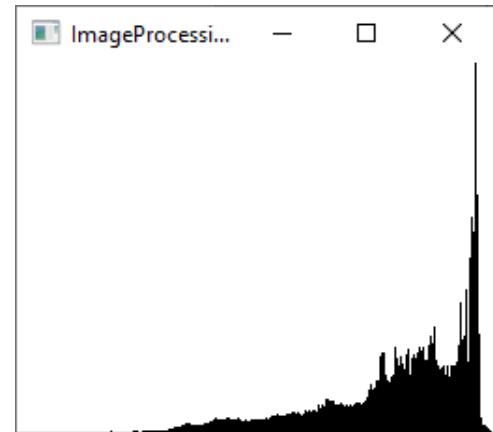
void MainWindow::on_btDrawl_clicked()
{
    QImage img(ui->LineFileName->text());
    Histogram_GrayImage(img, 200);
}

```

Kết quả :



Ảnh xám đầu vào



Tổ chức đồ của ảnh xám

## 2.2 Histogram cho ảnh màu :

Chúng ta thêm vào chương trình một hàm Histogram\_ColorImage() để tính toán và vẽ tổ chức đồ ảnh ảnh màu như bên dưới.

```

void MainWindow::Histogram_ColorImage(const QImage &img, const int &his_height){

    int hr[256];
    int hg[256];
    int hb[256];
    for (int i=0; i<256; i++)
    {
        hr[i]=0;
        hg[i]=0;
        hb[i]=0;
    }
    QImage image_in(img);
    for (int x=0; x<img.width(); x++)
        for (int y=0; y<img.height(); y++)
        {
            QRgb color = img.pixel(x,y);
            int green = qGreen(color);
            int red = qRed(color);
            int blue = qBlue(color);
            hr[red]++;
            hg[green]++;
            hb[blue]++;
        }
    int maxr = 0,maxg = 0,maxb = 0;
    for (int i=0; i<256; i++)
    {
        if (hr[i]>maxr) maxr=hr[i];
        if (hg[i]>maxg) maxg=hg[i];
        if (hb[i]>maxb) maxb=hb[i];
    }

    QImage img_hisr = QImage(256,his_height,QImage::Format_RGB888);
    img_hisr.fill(Qt::white);

    QImage img_histg = QImage(256,his_height,QImage::Format_RGB888);
    img_histg.fill(Qt::white);

    QImage img_hisb = QImage(256,his_height,QImage::Format_RGB888);
    img_hisb.fill(Qt::white);

    int liner,lineg,lineb;
    for (int i=0; i<256; i++) {
        liner = his_height * hr[i]/maxr;      // tinh tien anh de histogram co do lon nhu mong muon
        lineg = his_height * hg[i]/maxg;
        lineb = his_height * hb[i]/maxb;
        for (int j=his_height-1; j>=his_height - 1 - liner; j--) // toa do 0,0 nam phia tren ben trai
            img_hisr.setPixel(i,j,qRgb(255,0,0));

        for (int j=his_height-1; j>=his_height - 1 - lineg; j--) // toa do 0,0 nam phia tren ben trai
            img_histg.setPixel(i,j,qRgb(0,255,0));

        for (int j=his_height-1; j>=his_height - 1 - lineb; j--) // toa do 0,0 nam phia tren ben trai
            img_hisb.setPixel(i,j,qRgb(0,0,255));
    }

    //QImage image_out(img.width(),img.height(),QImage::Format_RGB888);
    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in));
    label_in->show();

    // hien thi histogram
    QLabel* label_out1 = new QLabel();
    label_out1->setPixmap(QPixmap::fromImage(img_hisr));
    label_out1->show();

    QLabel* label_out2 = new QLabel();
    label_out2->setPixmap(QPixmap::fromImage(img_histg));
    label_out2->show();

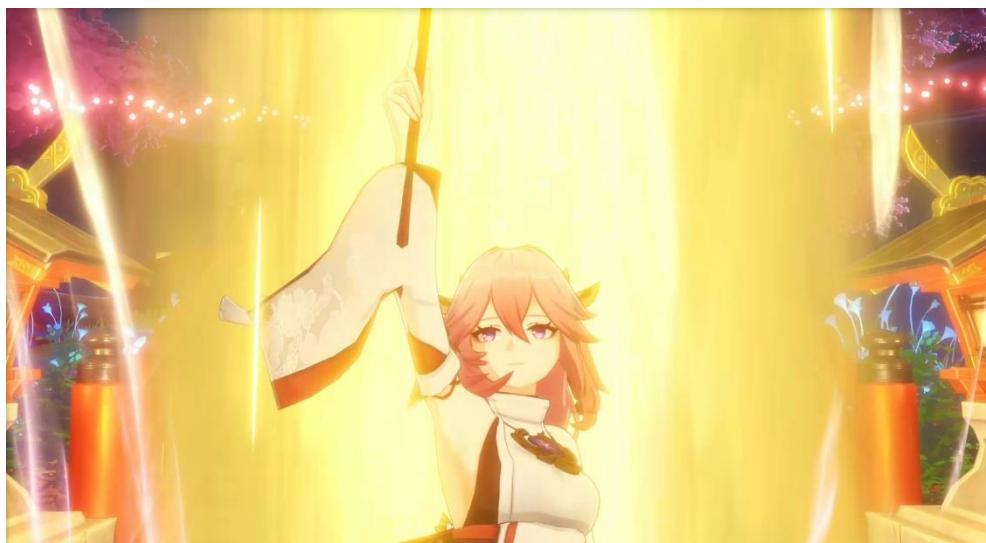
    QLabel* label_out3 = new QLabel();
    label_out3->setPixmap(QPixmap::fromImage(img_hisb));
    label_out3->show();
}

```

Sau đó tại sự kiện clicked của button vẽ tổ chức đồ cho ảnh màu, ta gọi hàm vừa cài đặt lên như sau :

```
void MainWindow::on_btDraw2_clicked()
{
    QImage img(ui->LineFileName->text());
    Histogram_ColorImage(img, 200);
}
```

Kết quả thu được :



Ảnh màu đầu vào



Tổ chức đồ ảnh màu

### 2.3 Căng tổ chức đồ ảnh xám ( Thay đổi độ tương phản )

Tại sự kiện clicked của button thay đổi độ tương phản của ảnh xám, ta thêm vào đoạn code sau :

```
void MainWindow::on_btnXuLy_2_clicked()
{
    QImage image_in(ui->LineFileName->text());

    QImage image_out(image_in.width(),image_in.height(),QImage::Format_RGB32);

    float c = ui->LineDoTuongPhan->text().toFloat();

    for(int i = 0; i < image_in.width(); i++)
    {
        for(int j = 0; j < image_in.height(); j++)
        {
            QRgb rgb = image_in.pixel(i,j);
            int gray = qGray(rgb);
            QRgb color_in = qRgb(gray,gray,gray);
            image_in.setPixel(i,j,color_in);
            int out = gray * c;
            if(out > 255) out = 255;
            else if(out < 0) out = 0;

            QRgb color_out = qRgb(out,out,out);
            image_out.setPixel(i,j,color_out);
        }
    }

    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in.scaled(640,480)));
    label_in->show();

    QLabel* label_out = new QLabel();
    label_out->setPixmap(QPixmap::fromImage(image_out.scaled(640,480)));
    label_out->show();
}
```

Kết quả thu được :



Ảnh xám gốc

Ảnh xám đã tăng độ tương phản với  $c = 1.5$

### 3) Bài tập thực hành buổi 2 :

3.1 ) Hãy thiết kế tính năng cảng tổ chức đồ (thay đổi độ tương phản) cho ảnh màu



Ảnh màu gốc

Ảnh màu đã tăng độ tương phản với  $c = 2$

3.2) Hãy thiết kế tính năng biến đổi tuyến tính tổ chức đồ cho ảnh xám (xem lại phần lý thuyết)

3.3) Hãy thiết kế tính năng cân bằng tự động tổ chức đồ cho ảnh xám (xem lại phần lý thuyết)

**TÀI LIỆU HƯỚNG DẪN THỰC HÀNH BUỔI 3****Học Phần : Xử Lý Ảnh****1) Sửa bài tập về nhà buổi 2****Câu hỏi 1 :** Xây dựng tính năng biến đổi tuyến tính tổ chức đồ

Sinh viên tham khảo đoạn code sau :

```

QImage MainWindow::SuaChuaHistogram(const QImage &imgin)
{
    int max=0;
    int min = 255;
    QImage image_in(imgin);
    QImage image_out(image_in.width(),image_in.height(),QImage::Format_Grayscale8);

    for (int x=0; x<imgin.width(); x++)
    {
        for (int y=0; y<imgin.height(); y++)
        {
            QRgb color = imgin.pixel(x,y);
            int gray = qGray(color);
            QRgb color_in = qRgb(gray,gray,gray);
            image_in.setPixel(x,y,color_in);
            if (gray > max) max = gray;
            if (gray < min) min = gray;
        }
    }

    for (int x=0; x<imgin.width(); x++)
    {
        for (int y=0; y<imgin.height(); y++)
        {
            int newgray = 0;
            QRgb color = imgin.pixel(x,y);
            int gray = qGray(color);
            newgray = (((gray - min)*255) / (max-min));
            QRgb color_out = qRgb(newgray,newgray,newgray);
            image_out.setPixel(x,y,color_out);
        }
    }

    return image_out;
}

void MainWindow::on_btSuaChua_clicked()
{
    QImage img(ui->LineFileName->text());
    Histogram_GrayImage(img, 200);

    QImage new_img = SuaChuaHistogram(img);
    Histogram_GrayImage(new_img, 200);

}

```

**Câu hỏi 2 :** Xây dựng tính năng cân bằng tự động tổ chức đồ

Sinh viên tham khảo đoạn code sau :

```

QImage MainWindow::CanBangHistogram(const QImage &imgin)
{
    double h[256];
    for (int i=0; i<256; i++)
        h[i]=0;

    QImage image_in(imgin);
    QImage image_out(image_in.width(),image_in.height(),QImage::Format_RGB32);
    for (int x=0; x<imgin.width(); x++)
    {
        for (int y=0; y<imgin.height(); y++)
        {
            QRgb color = imgin.pixel(x,y);
            int gray = qGray(color);
            h[gray]++;
        }
    }
    double hn[256];
    int numpix = imgin.width() * imgin.height();
    for (int i=0; i<256; i++)
        hn[i] = h[i]/(numpix);

    double c[256];
    for (int i=0; i<256; i++)
        c[i]=0;
    for (int i = 0; i < 256; i++)
    {
        for (int j = 0; j < i; j++)
        {
            c[i] = c[i] + hn[j];
        }
    }

    for (int x=0; x<imgin.width(); x++)
    {
        for (int y=0; y<imgin.height(); y++)
        {
            QRgb color = imgin.pixel(x,y);
            int gray = qGray(color);
            int newgray = (int)(c[gray] * 255);
            QRgb color_out = qRgb(newgray,newgray,newgray);
            image_out.setPixel(x,y,color_out);
        }
    }
}

return image_out;
}

void MainWindow::on_btCanBang_clicked()
{
    QImage img(ui->LineFileName->text());
    Histogram_GrayImage(img, 200);

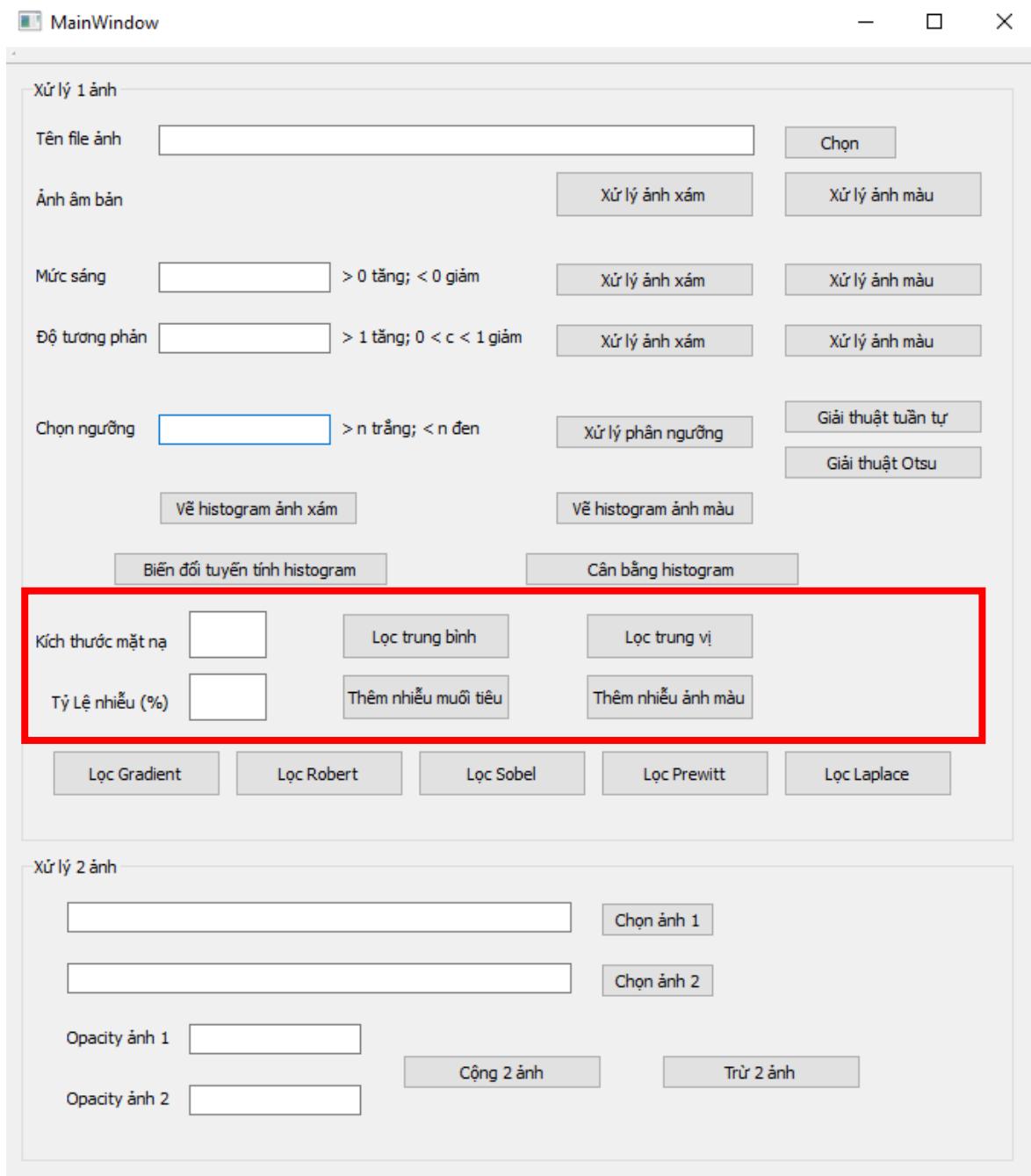
    QImage new_img = CanBangHistogram(img);
    Histogram_GrayImage(new_img, 200);
}

```

## 2) Nội dung thực hành buổi 3

Nội dung buổi thực hành thứ 3 sẽ xoay quanh việc tính tích chập của ảnh với các bộ lọc hạ thông như bộ lọc trung bình và bộ lọc trung vị. Sau đó xây dựng tính năng thêm nhiễu muối tiêu vào trong ảnh, và sử dụng bộ lọc Median để khử nhiễu muối tiêu cho ảnh.

Sinh viên tiến hành xây dựng giao diện chương trình như sau :



## 2.1) Bộ lọc trung bình :

Line Edit “Kích thước mặt nạ” sẽ do người dùng nhập vào. Nếu muốn kích thước mặt nạ là 3x3 thì người dùng nhập số 3 vào. Nếu là 5x5 thì nhập số 5 vào. Sau đó bấm vào nút lọc trung bình hoặc lọc trung vị thì chương trình sẽ đi tính tích chập của ảnh đầu vào với mặt nạ có kích thước đã chọn. Hàm để tính tích chập của ảnh với bộ lọc trung bình được viết như sau :

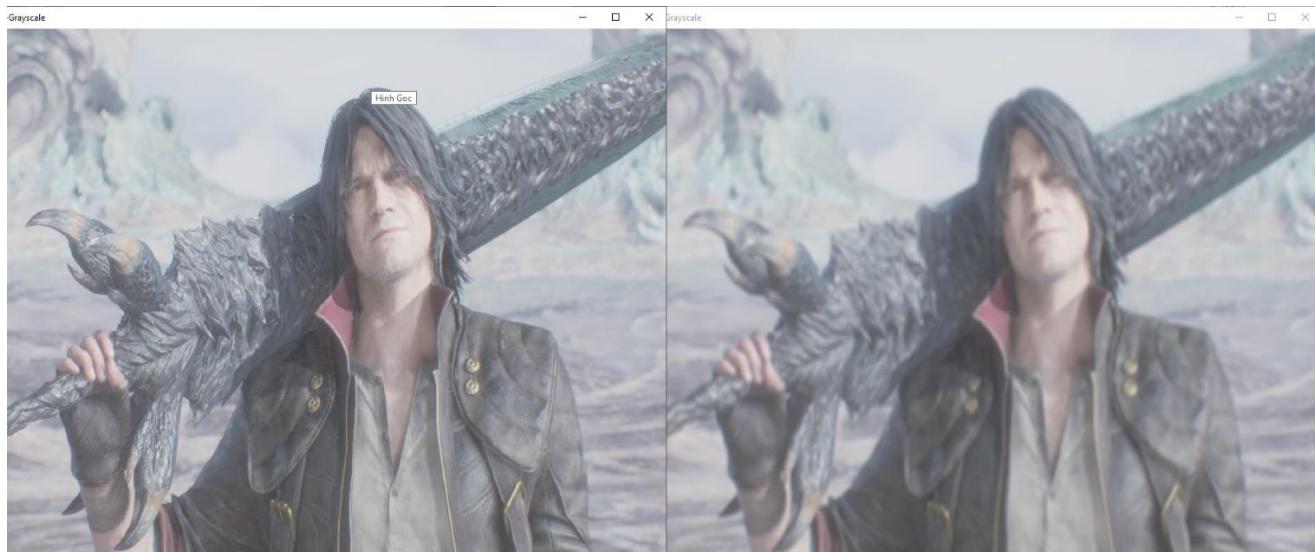
```
void MainWindow::on_btn_meanfilter_clicked()
{
    QImage image_in(ui->LineFileName->text());
    // khai báo cho mặt nạ chap 3x3
    int size = ui->ln_masksize->text().toInt();
    int margin = (size - 1)/2;
    int cells = size* size;

    QImage image_out(image_in.width()-margin,image_in.height()-margin,QImage::Format_RGB32);
    int sumR, sumG, sumB;
    QColor color;
    for (int x=margin; x<image_in.width() - margin; x++)
        for (int y=margin; y<image_in.height() - margin; y++){
            sumR = sumG = sumB = 0;
            for (int i=-margin; i<=margin; i++)
                for (int j=-margin; j<=margin; j++){
                    color = image_in.pixel(x+i, y+j);
                    sumR += color.red();
                    sumG += color.green();
                    sumB += color.blue();
                }
            image_out.setPixel(x,y,qRgb(sumR/cells, sumG/cells, sumB/cells));
        }

    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in.scaled(1024,768)));
    label_in->setToolTip("Hình Goc");
    label_in->show();

    QLabel *label = new QLabel();
    label->setPixmap(QPixmap::fromImage(image_out.scaled(1024,768)));
    label->setToolTip("Hình da xu ly");
    label->show();
    QString fname = QFileDialog::getSaveFileName();
    image_out.save(fname);
}
```

**Lưu ý :** Sinh viên cần nắm rõ ý nghĩa của các biến size, margin và cells. Chủ động thay đổi giá trị của kích thước mặt nạ để thấy rõ sự khác biệt.



Ảnh gốc

This document is available free of charge on

Ảnh đã chap với bộ lọc trung bình 11x11

## 2.2) Bộ lọc trung vị :

Sinh viên cài đặt tính năng này như sau :

```
void MainWindow::on_btn_medianfilter_clicked()
{
    QImage image_in(ui->LineFileName->text());
    // khai báo cho mat na chap
    int size = ui->ln_masksizes->text().toInt();
    int margin = (size - 1)/2;
    int cells = size* size;

    QImage image_out(image_in.width()-margin,image_in.height()-margin,QImage::Format_RGB32);

    for (int x=margin; x<image_in.width()-margin; x++)
        for (int y=margin; y<image_in.height()-margin; y++) {
            int h[cells];
            int k=0;
            for (int i=-margin; i<=margin; i++)
                for (int j=-margin; j<=margin; j++){
                    QRgb color = image_in.pixel(x+i,y+j);
                    int gray = qGray(color);
                    h[k] = gray;
                    k++;
                }

            qSort(h, h + cells);
            int meanV = h[(cells - 1)/2];
            image_out.setPixel(x,y,qRgb(meanV,meanV,meanV));
        }

    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in.scaled(1024,768)));
    label_in->setToolTip("Hình Gốc");
    label_in->show();

    QLabel *label = new QLabel();
    label->setPixmap(QPixmap::fromImage(image_out.scaled(1024,768)));
    label->setToolTip("Hình đã xử lý");
    label->show();
    QString fname = QFileDialog::getSaveFileName();
    image_out.save(fname);
}
```

Lưu ý : Cần xem lại phần lý thuyết về bộ lọc Median.

Kết quả thu được



Ảnh bị nhiễu muối tiêu



Ảnh kết quả đã được khử nhiễu

### 2.3) Thêm nhiễu muối tiêu vào ảnh

Sinh viên cần khai báo thêm 2 hàm sau vào file mainwindow.h và mainwindow.cpp

```
int MainWindow::Random(int n)
{
    return rand()%n;
}

QImage MainWindow::imNoise_Gray(QImage &image_in, float level)
{
    QImage image_out = image_in;

    int noisePoint = image_in.width() * image_in.height() * level;

    for (int i = 0; i < noisePoint; i++)
    {
        int x = Random(image_in.width());
        int y = Random(image_in.height());
        int a = Random(2);
        if(a == 0)
            image_out.setPixel(x,y,qRgb(0,0,0));
        else
            image_out.setPixel(x,y,qRgb(255,255,255));
    }

    return image_out;
}
```

Sau đó, tại sự kiện click của nút thêm nhiễu, ta thêm đoạn code như sau :

```
void MainWindow::on_btn_addnoise_clicked()
{
    QImage image_in(ui->LineFileName->text());
    QImage image_inGray(image_in.width(),image_in.height(),QImage::Format_RGB32);

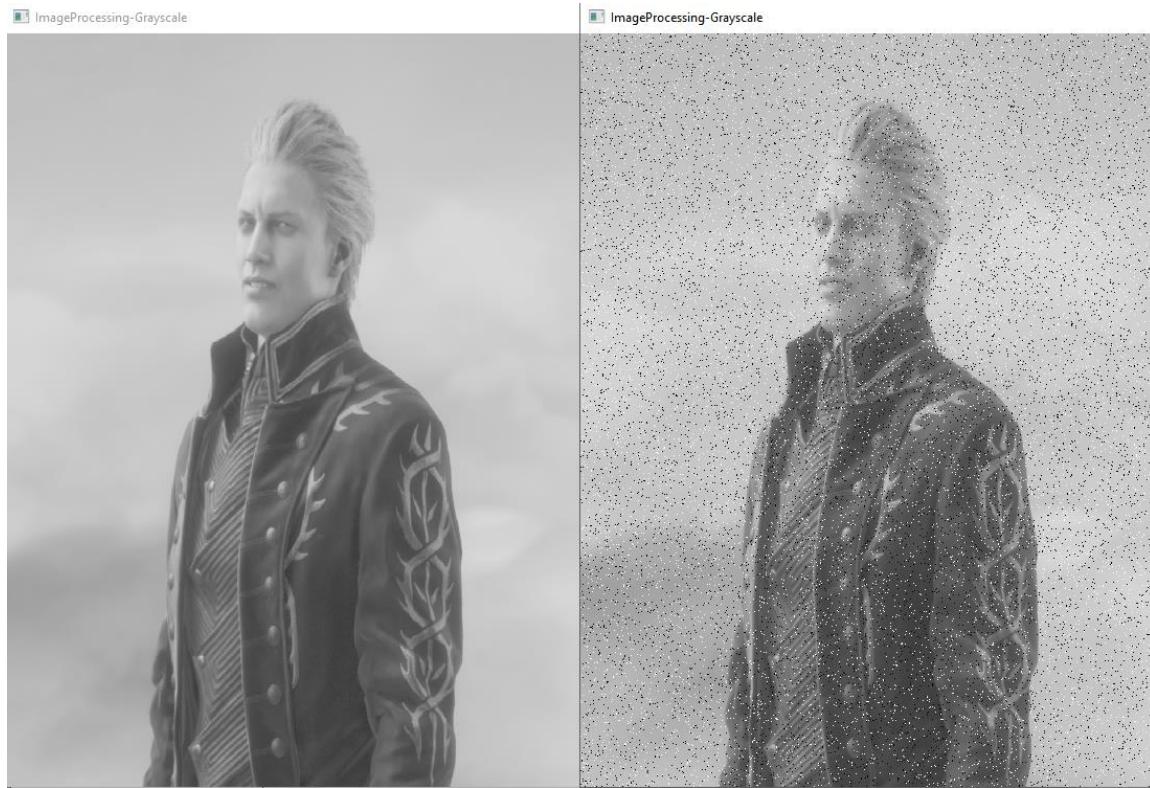
    for(int i = 0; i < image_in.width(); i++)
    {
        for(int j = 0; j < image_in.height(); j++)
        {
            QRgb rgb = image_in.pixel(i,j);
            int gray = qGray(rgb);
            QRgb color_in = qRgb(gray,gray,gray);
            image_inGray.setPixel(i,j,color_in);
        }
    }
    float percentage = (ui->ln_percentage->text().toInt())/100;

    QImage image_out = imNoise_Gray(image_inGray, percentage);

    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_inGray.scaled(1024,768)));
    label_in->setToolTip("Hình Góc");
    label_in->show();

    QLabel *label = new QLabel();
    label->setPixmap(QPixmap::fromImage(image_out.scaled(1024,768)));
    label->setToolTip("Hình da xu ly");
    label->show();
    QString fname = QFileDialog::getSaveFileName();
    image_out.save(fname);
}
```

### Kết quả thu được



Ảnh trước khi thêm nhiễu

Ảnh sau khi thêm nhiễu

### 3) Bài tập về nhà

Bài 1 : Xây dựng tính năng lọc trung bình đối với ảnh xám

Bài 2 : Xây dựng tính năng thêm nhiễu trên ảnh màu.

# Bài thực hành buổi 4

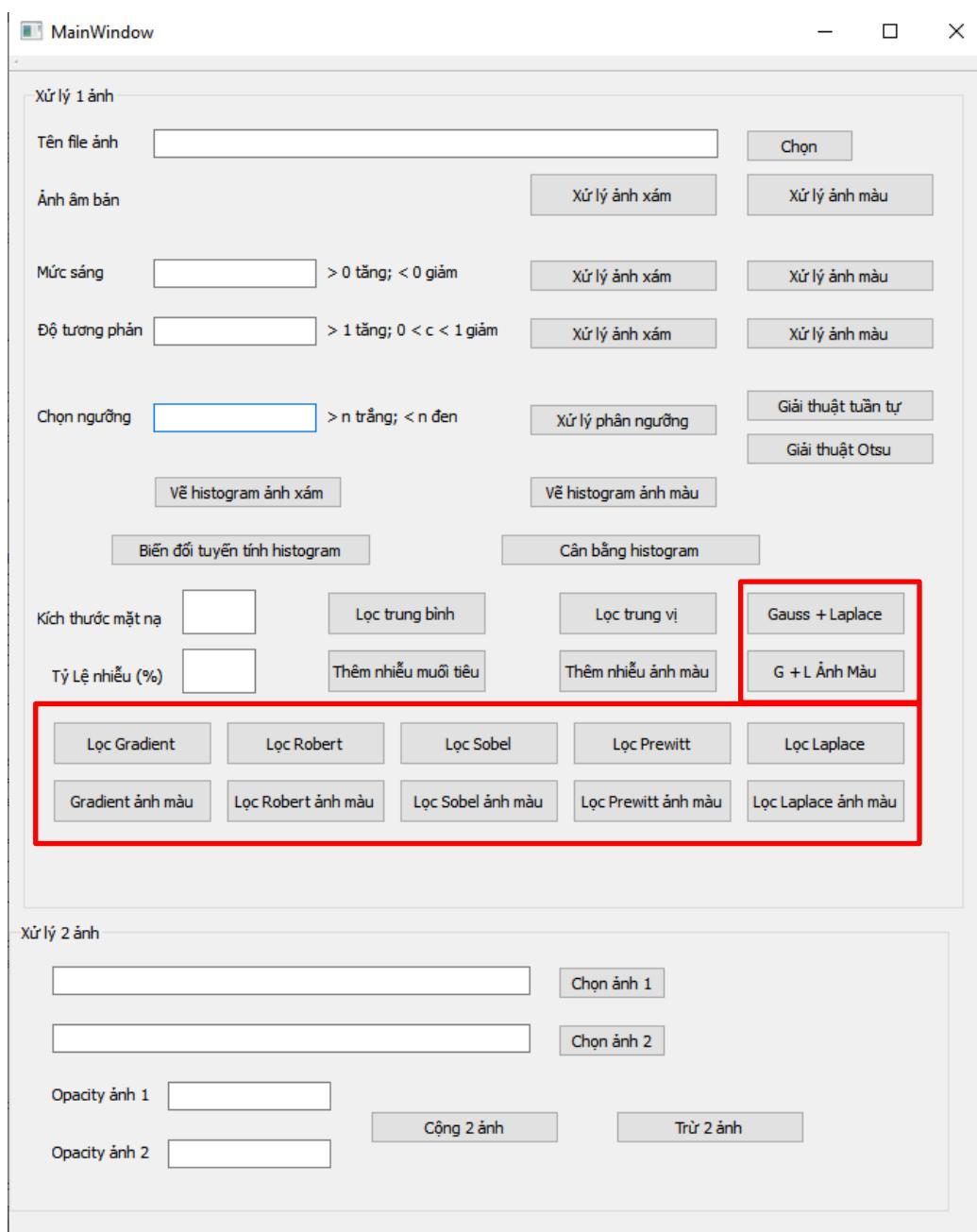
## Học Phân : Xử Lý Ảnh

### 1. Mục tiêu:

- Thực hành các phương pháp lọc đường biên: Bộ lọc Gradient, Bộ lọc Robert, Bộ lọc Sobel, lọc Sobel và lọc Laplace
- Thực hành phương pháp phân ngưỡng tự động bằng giải thuật tuần tự và giải thuật Otsu

### 2. Thực hành:

Sinh viên tiến hành thiết kế giao diện như hình



## 2.1) Lọc đường biên bằng phương pháp Gradient

Tại sự kiện clicked của nút “ Lọc Gradient ”, sinh viên tiến hành nhập vào đoạn code sau:

```
void MainWindow::on_btn_gradient_clicked()
{
    QImage image_in(ui->LineFileName->text());
    QImage image_out_Gx(image_in.width()-1,image_in.height(),QImage::Format_RGB32);
    QImage image_out_Gy(image_in.width(),image_in.height()-1,QImage::Format_RGB32);

    for (int i = 0; i < image_in.width() - 1; i++)
    {
        for(int j = 0; j < image_in.height(); j++)
        {
            QRgb color1 = image_in.pixel(i,j);
            int gray1 = qGray(color1);

            QRgb color_in = qRgb(gray1,gray1,gray1);
            image_in.setPixel(i,j,color_in);

            QRgb color2 = image_in.pixel(i+1,j);
            int gray2 = qGray(color2);
            int newgray = abs (gray2 - gray1);
            image_out_Gx.setPixel(i,j,qRgb(newgray,newgray,newgray));
        }
    }

    for (int i = 0; i < image_in.width(); i++)
    {
        for(int j = 0; j < image_in.height()-1; j++)
        {
            QRgb color1 = image_in.pixel(i,j);
            int gray1 = qGray(color1);

            QRgb color_in = qRgb(gray1,gray1,gray1);
            image_in.setPixel(i,j,color_in);

            QRgb color2 = image_in.pixel(i,j+1);
            int gray2 = qGray(color2);
            int newgray = abs (gray2 - gray1);
            image_out_Gy.setPixel(i,j,qRgb(newgray,newgray,newgray));
        }
    }

    QLabel* label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in.scaled(1024,768)));
    label_in->show();

    QLabel *label_Gx = new QLabel();
    label_Gx->setPixmap(QPixmap::fromImage(image_out_Gx.scaled(1024,768)));
    label_Gx->show();

    QLabel *label_Gy = new QLabel();
    label_Gy->setPixmap(QPixmap::fromImage(image_out_Gy.scaled(1024,768)));
    label_Gy->show();
}
```

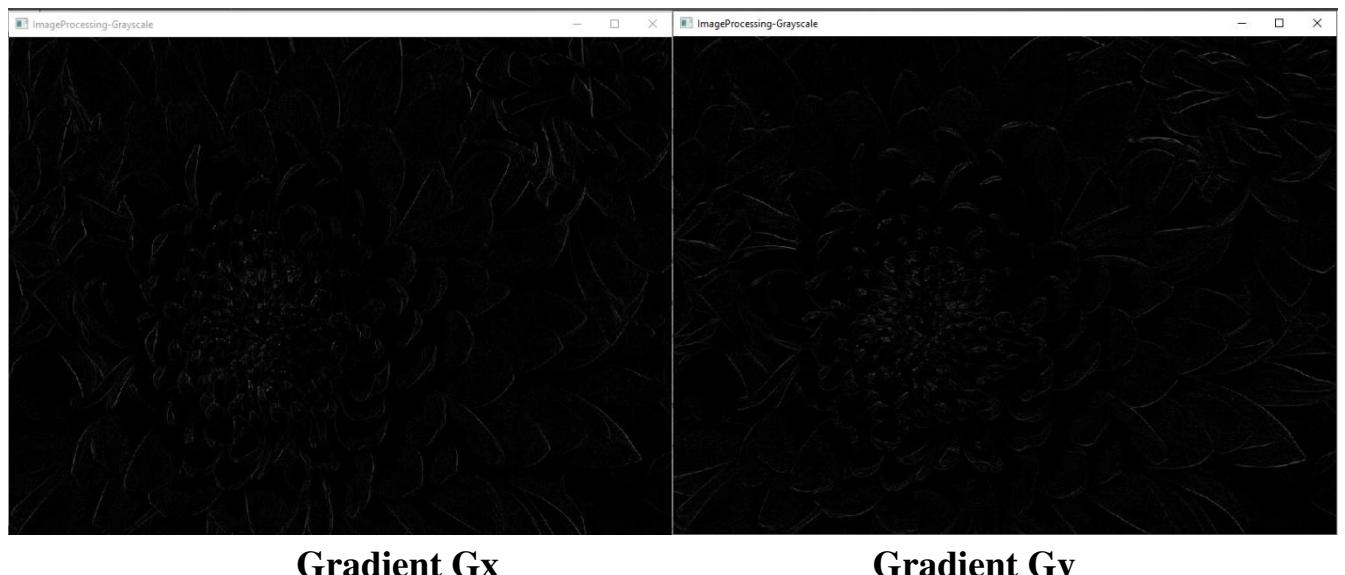
Đoạn code trên sẽ tiến hành tính đạo hàm Gx và Gy của ảnh đầu vào và xuất ra màn hình thành 2 ảnh phân biệt cho người dùng.

Để hiểu rõ nguyên lý sinh viên cần xem lại phần lý thuyết về nguyên lý lọc Gradient của chương 5 – Phát hiện đường biên.

Kết quả thu được :



Ảnh gốc



Gradient Gx

Gradient Gy

## 2.2) Lọc đường biên bằng bộ lọc Robert

Nhắc lại kiến thức : Bộ lọc Robert là một bộ lọc đặc biệt có kích thước  $2 \times 2$  với tác dụng làm nổi bật những đường biên theo hướng chéo

$$\begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix} \quad \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

Gx                    Gy

Tại sự kiện clicked của nút “Lọc Robert”, ta thêm vào đoạn code sau :

```

void MainWindow::on_btn_robert_clicked()
{
    QImage image_in(ui->LineFileName->text());
    QImage image_out_Gx(image_in.width()-1,image_in.height()-1,QImage::Format_RGB32);
    QImage image_out_Gy(image_in.width()-1,image_in.height()-1,QImage::Format_RGB32);

    for (int i = 0; i <image_in.width()-1 ; i++)
    {
        for (int j = 0; j<image_in.height()-1 ; j++)
        {
            QRgb color1 = image_in.pixel(i,j);
            int gray1 = qGray(color1);
            QRgb color2 = image_in.pixel(i+1,j);
            int gray2 = qGray(color2);
            QRgb color3 = image_in.pixel(i,j+1);
            int gray3 = qGray(color3);
            QRgb color4 = image_in.pixel(i+1,j+1);
            int gray4 = qGray(color4);

            int newgray_gx = abs(gray1 - gray4);
            int newgray_gy = abs(gray2 - gray3);

            image_out_Gx.setPixel(i,j,qRgb(newgray_gx,newgray_gx,newgray_gx));
            image_out_Gy.setPixel(i,j,qRgb(newgray_gy,newgray_gy,newgray_gy));
        }
    }

    QLabel *label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in.scaled(1024,768)));
    label_in->show();

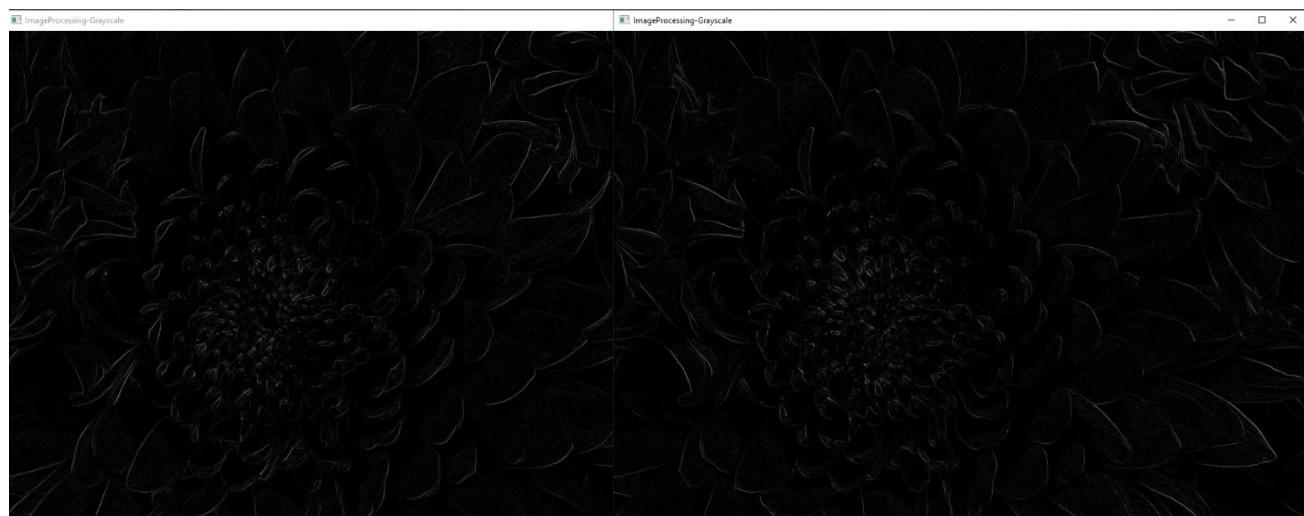
    QLabel *label_Gx = new QLabel();
    label_Gx->setPixmap(QPixmap::fromImage(image_out_Gx.scaled(1024,768)));
    label_Gx->show();

    QLabel *label_Gy = new QLabel();
    label_Gy->setPixmap(QPixmap::fromImage(image_out_Gy.scaled(1024,768)));
    label_Gy->show();
}

```

Đoạn chương trình trên sẽ tìm ra các đường biên theo 2 hướng xéo và lưu vào các ảnh Gx và Gy. Sinh viên hãy liên hệ với bộ lọc Robert để hiểu hơn đoạn chương trình trên.

Kết quả thu được :



Robert Gx

Robert Gy

### 2.3) Lọc đường biên bằng bộ lọc Sobel

Nhắc lại kiến thức : bộ lọc Sobel là sự kết hợp của việc làm trơn ảnh và lấy đạo hàm của ảnh

## Sobel:

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Ta thêm đoạn code sau vào sự kiện clicked của nút “ Lọc Sobel ” :

```
void MainWindow::on_btn_sobel_clicked()
{
    QImage image_in(ui->LineFileName->text());
    QImage image_out(image_in.width(),image_in.height(),QImage::Format_RGB32);
    QImage image_out2(image_in.width(),image_in.height(),QImage::Format_RGB32);

    int maskSize = 3;
    int margin = maskSize / 2;
    //int cells = maskSize * maskSize;

    int mask[maskSize][maskSize];
    mask[0][0] = -1; mask[0][1] = 0; mask[0][2] = 1;
    mask[1][0] = -2; mask[1][1] = 0; mask[1][2] = 2;
    mask[2][0] = -1; mask[2][1] = 0; mask[2][2] = 1;
    int mask2[maskSize][maskSize];
    mask2[0][0] = -1; mask2[0][1] = -2; mask2[0][2] = -1;
    mask2[1][0] = 0; mask2[1][1] = 0; mask2[1][2] = 0;
    mask2[2][0] = 1; mask2[2][1] = 2; mask2[2][2] = 1;
    QRgb color;
    int sum1, sum2;
    for (int y = margin; y < image_in.height() - margin; y++)
        for (int x = margin; x < image_in.width() - margin; x++)
    {
        sum1 = sum2 = 0;
        for (int j = -margin; j <= margin; j++)
            for (int i = -margin; i <= margin; i++)
            {
                color = image_in.pixel(x + i, y + j);
                int gray = qGray(color);
                sum1 += gray * mask[j+margin][i+margin];
                sum2 += gray * mask2[j+margin][i+margin];
            }
        image_out.setPixel(x, y, qRgb(abs(sum1), abs(sum1), abs(sum1)));
        image_out2.setPixel(x, y, qRgb(abs(sum2), abs(sum2), abs(sum2)));
    }

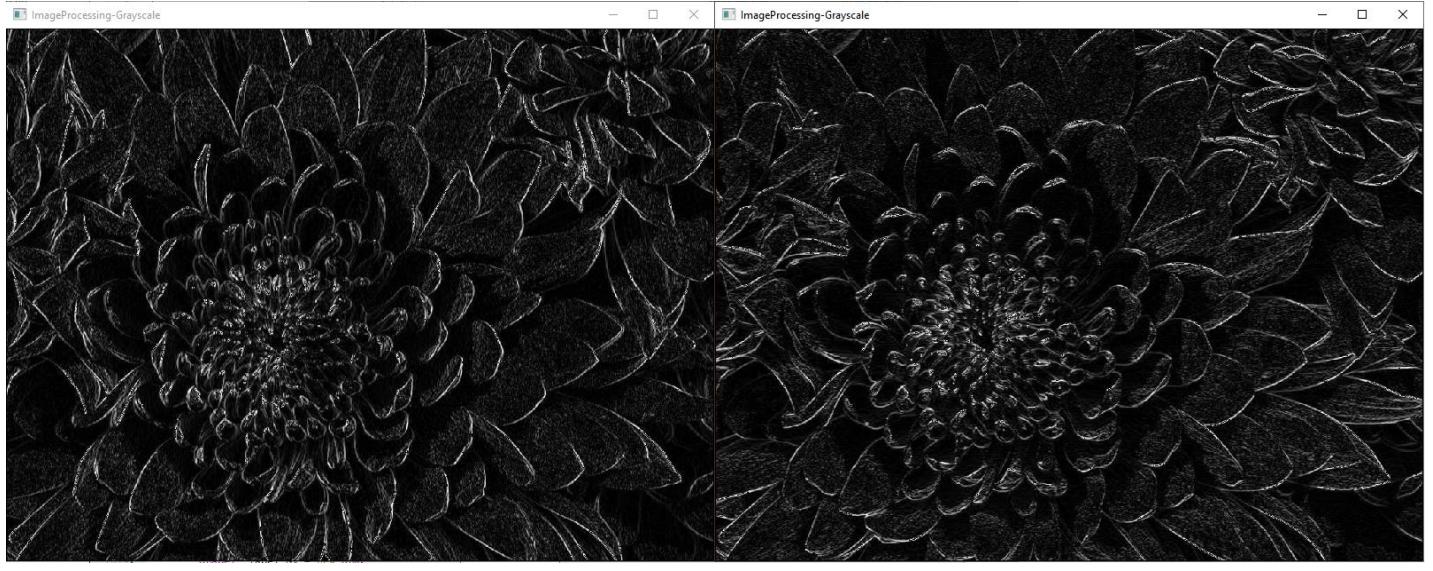
    QLabel *label_in = new QLabel();
    label_in->setPixmap(QPixmap::fromImage(image_in.scaled(1024,768)));
    label_in->show();

    QLabel *label_Gx = new QLabel();
    label_Gx->setPixmap(QPixmap::fromImage(image_out.scaled(1024,768)));
    label_Gx->show();

    QLabel *label_Gy = new QLabel();
    label_Gy->setPixmap(QPixmap::fromImage(image_out2.scaled(1024,768) ));
    label_Gy->show();
}
```

Đoạn chương trình trên sẽ tiến hành tính tích chập của ảnh với 2 bộ lọc Sobel để thu được kết quả Gx và Gy. Sinh viên so sánh kết quả với các bộ lọc trước đó và đưa ra kết luận.

Kết quả thu được :



Sobel Gx

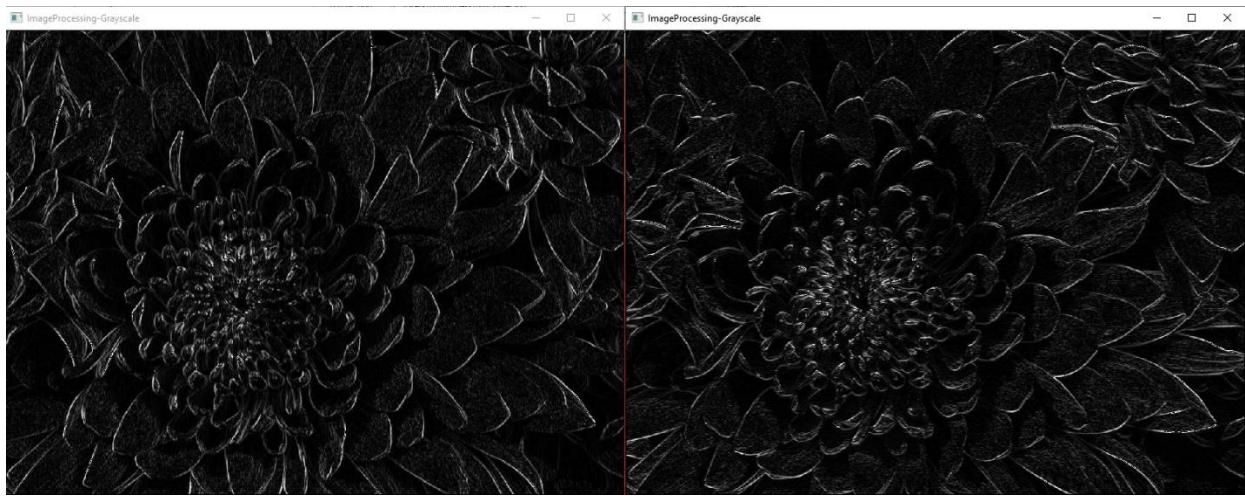
Sobel Gy

### 3) Bài tập

**Bài 1 :** Thiết kế chức năng cho bộ lọc Prewitt cho ảnh xám

-1	-1	-1
0	0	0
1	1	1

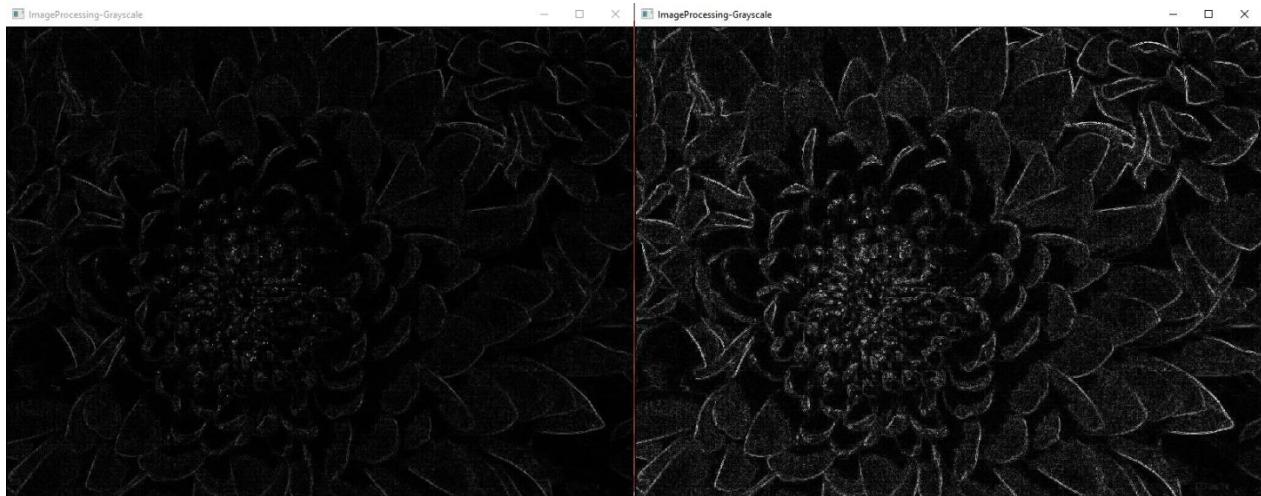
-1	0	1
-1	0	1
-1	0	1



**Bài 2 :** Thiết kế chức năng cho bộ lọc Laplace cho ảnh xám

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

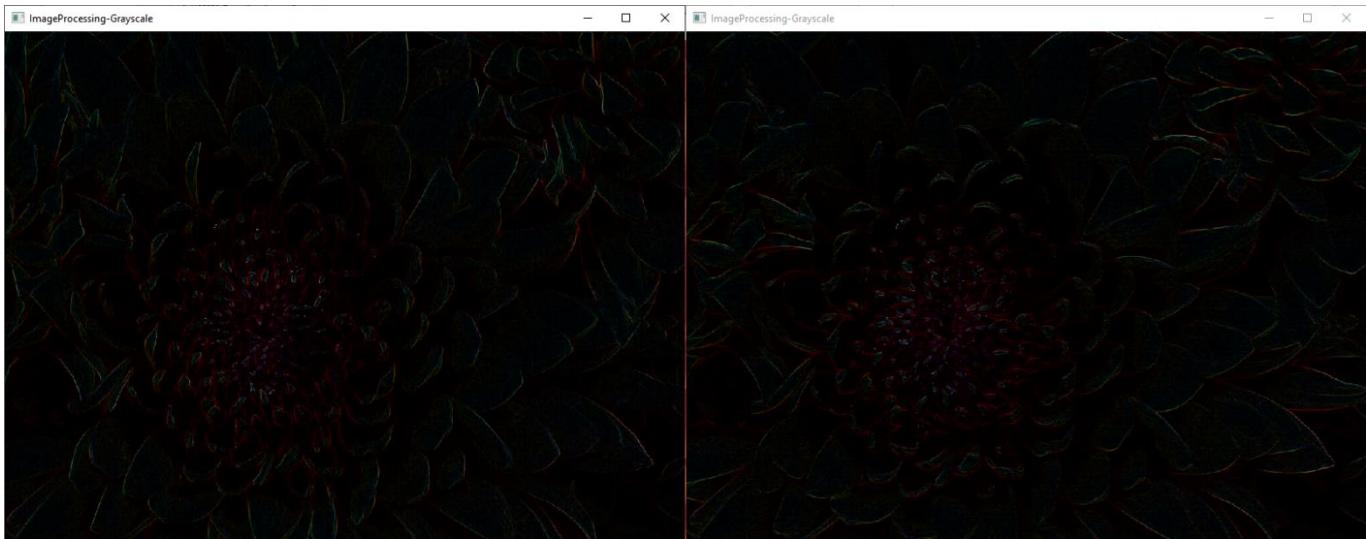


**Bài 3 :** Thiết kế tính năng cho Bộ lọc Gauss kết hợp Laplace cho ảnh xám

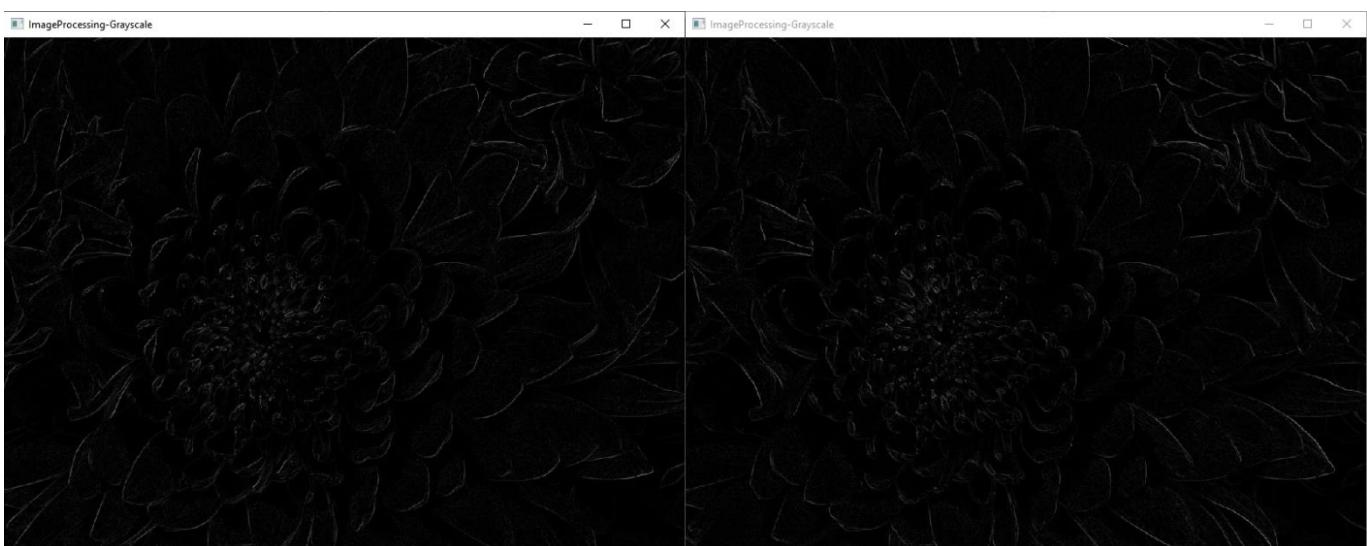
0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0



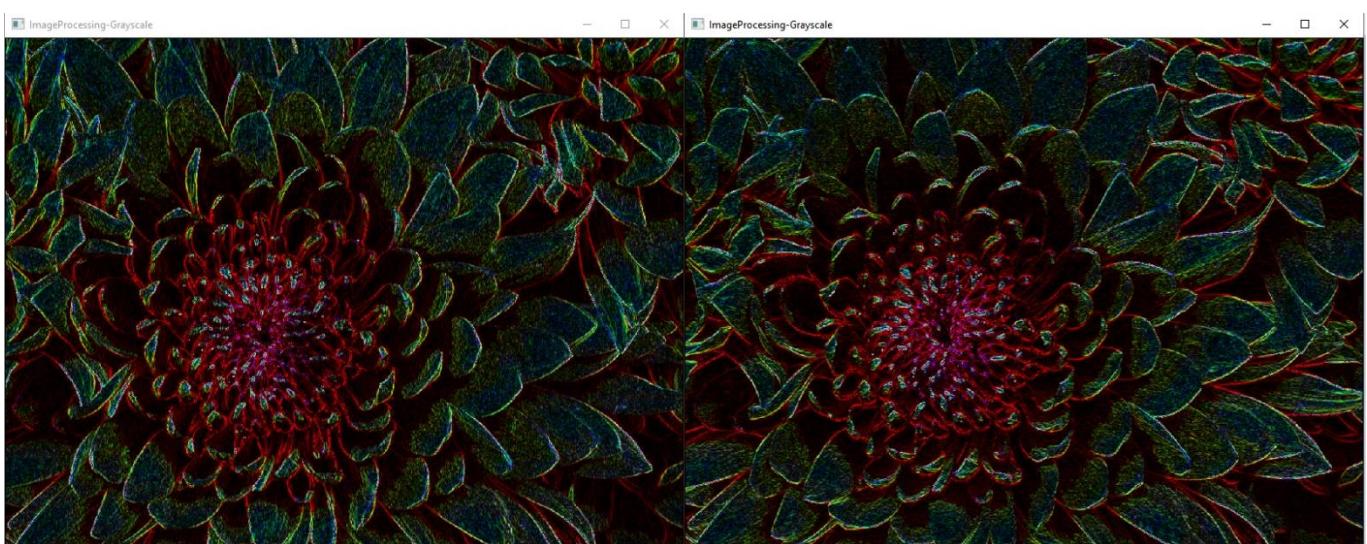
**Bài 4 :** Thiết kế tính năng Lọc Gradient cho ảnh màu



**Bài 5 :** Thiết kế tính năng Lọc Robert cho ảnh màu



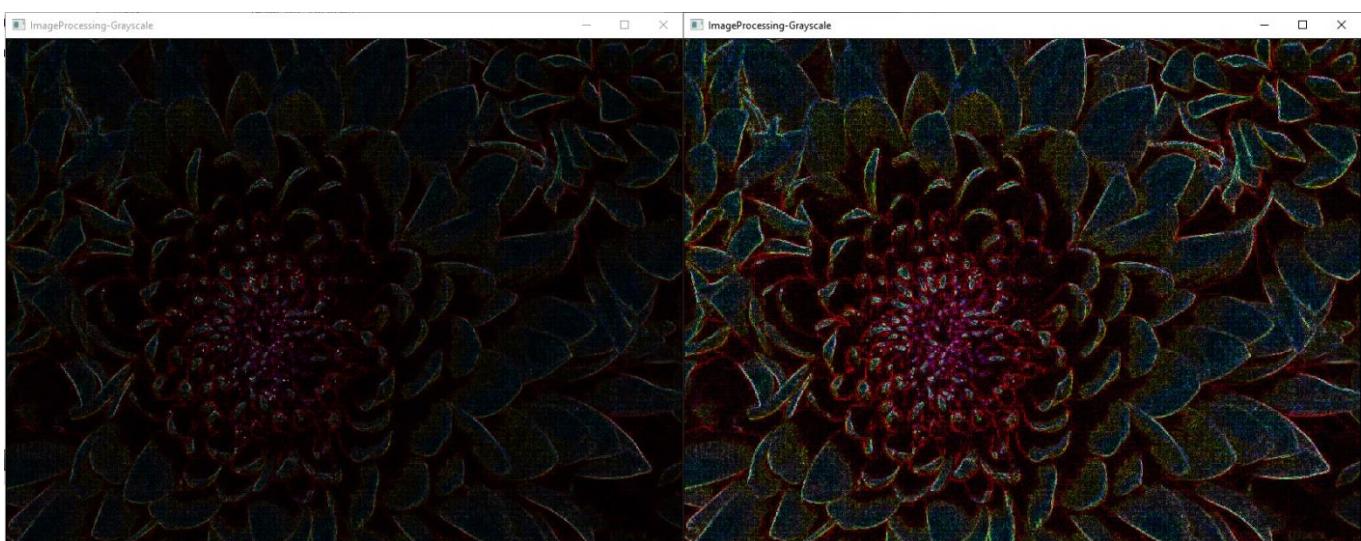
**Bài 6 :** Thiết kế tính năng Lọc Sobel cho ảnh màu



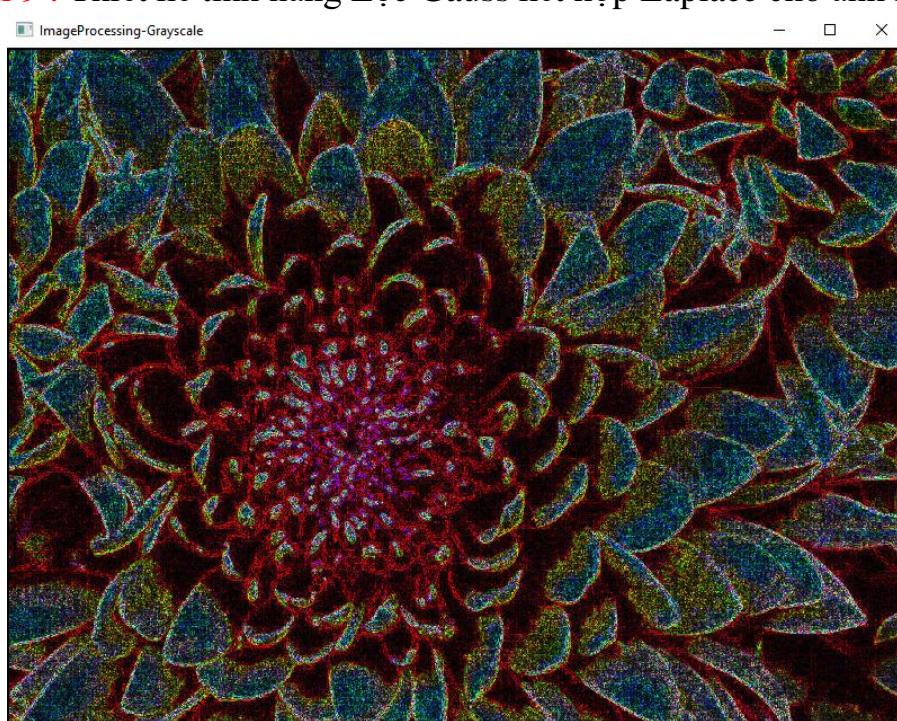
**Bài 7 :** Thiết kế tính năng Lọc Prewitt cho ảnh màu



Bài 8 : Thiết kế tính năng Lọc Laplace cho ảnh màu



Bài 9 : Thiết kế tính năng Lọc Gauss kết hợp Laplace cho ảnh màu



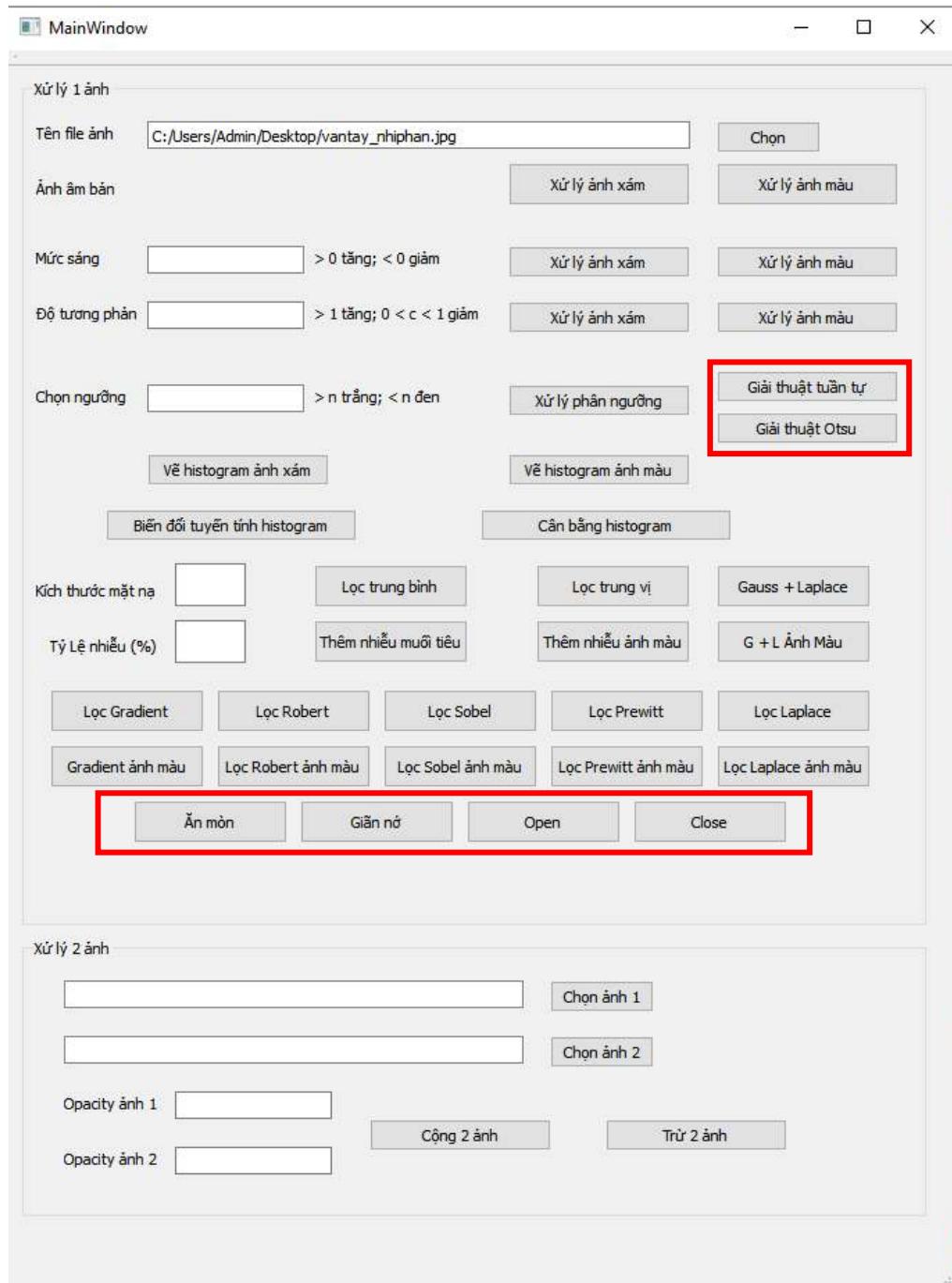
# Bài thực hành buổi 5

## 1. Mục tiêu:

- Thực hành phương pháp phân ngưỡng tự động bằng giải thuật tuần tự và giải thuật Otsu
- Thực hành các phép xử lý hình thái như phép ăn mòn, giãn nở, Open, Close
- Xây dựng một chương trình Xử lý ảnh và tích hợp các tính năng đã có lên chương trình mới.

## 2. Thực hành :

Sinh viên bổ sung các nút sau vào giao diện của chương trình :



## a. Phân ngưỡng tự động bằng giải thuật tuần tự

Viết chương trình cho phép tự động tìm ngưỡng bằng giải thuật tuần tự, sau đó tiến hành phân ngưỡng cho ảnh dựa vào kết quả ngưỡng tìm được.

Tại sự kiện clicked của button “Giải thuật tuần tự”, sinh viên hãy nhập vào đoạn code sau :

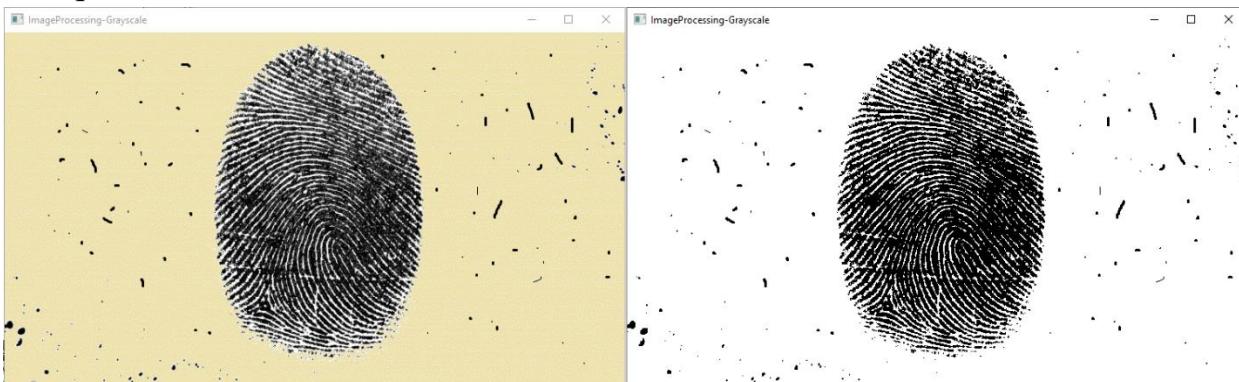
```
QImage image_in(ui->LineFileName->text());
QImage image_out(image_in.width(),image_in.height(),QImage::Format_RGB32);
int c = 127;
int newc = c;
do{
    int G1 = 0;
    int count1 = 0;
    int G2 = 0;
    int count2 = 0;
    c = newc;
    for(int i = 0; i < image_in.width(); i++)
    {
        for(int j = 0; j < image_in.height(); j++)
        {

            QRgb rgb = image_in.pixel(i,j);
            int gray = qGray(rgb);
            QRgb color_in = qRgb(gray,gray,gray);
            image_in.setPixel(i,j,color_in);
            if(gray > c)
            {
                G1 = G1 + gray;
                count1++;
            }
            else if(gray <= c)
            {
                G2 = G2 + gray;
                count2++;
            }
        }
    }
    int M1 = G1/count1;
    int M2 = G2/count2;
    newc = (M1 + M2)/2;

} while (abs(newc - c) !=0);|
```

**Lưu ý :** Đoạn code trên chỉ tiến hành áp dụng giải thuật tuần tự để tìm ra ngưỡng tự động. Sinh viên hãy kết hợp với tính năng phân ngưỡng đã làm ở các bài trước để tiến hành phân ngưỡng với giá trị ngưỡng tự động vừa tìm ra được

Kết quả thu được :



Anh gốc

Anh được phân ngưỡng tự động

### b. Xử lý hình thái nhị phân bằng phép toán Ăn mòn – Erosion

Cho đoạn code sau :

```
int kernel[3][3] = {{0,1,0},{1,1,1},{0,1,0}};
int margin = 1;
int c_gray;
QImage image_out2(image_in.width(),image_in.height(),QImage::Format_RGB32);
image_out2.fill(Qt::white);

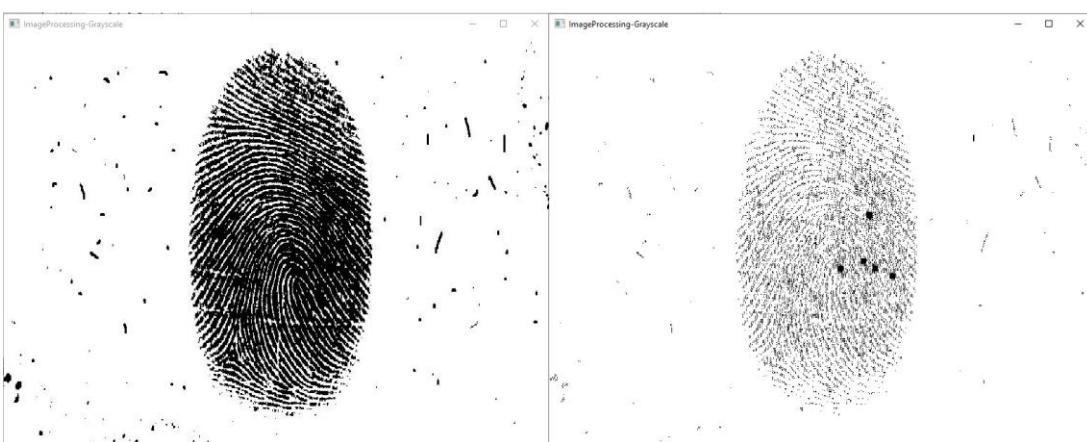
for(int x = margin;x <=image_in.width() - margin; x++)
    for(int y = margin;y <= image_in.height() - margin; y++)
    {
        int OK = 1;

        for (int i = -margin; i <= margin; i++)
            for (int j = -margin; j <=margin; j++)
            {
                c_gray = qGray(image_out.pixel(x+i,y+j));
                OK = OK && (kernel[i][j] == 0 || c_gray ==0);

            }
        if(OK) image_out2.setPixel(x,y,qRgb(0,0,0));
    }
```

Hãy tích hợp đoạn code này vào sự kiện clicked của tính năng “Ăn mòn”

Kết quả thu được:



Anh nhị phân đầu vào

Anh kết quả của phép ăn mòn

### c. Xử lý hình thái nhị phân bằng phép toán Giãn nở – Dilation

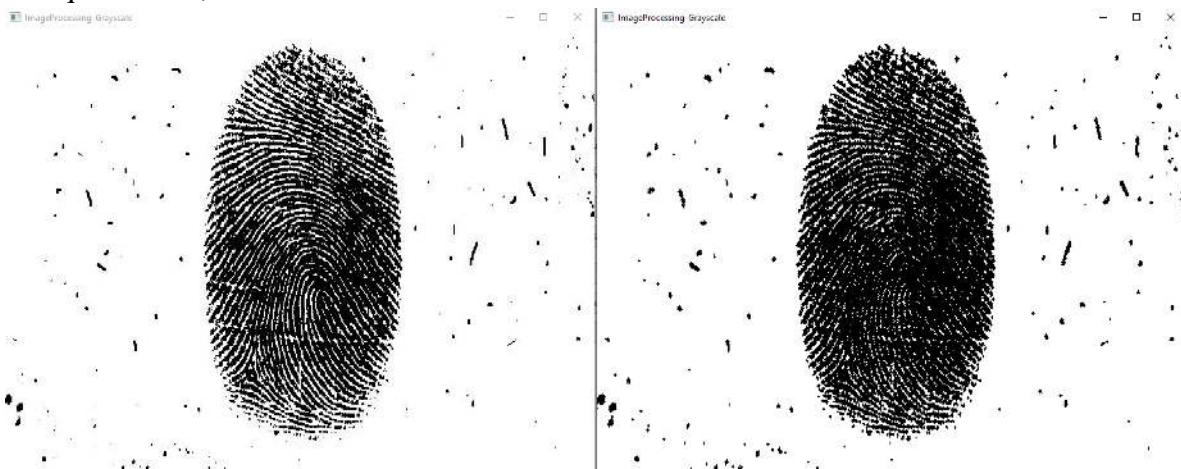
Cho đoạn code sau :

```
int kernel[3][3] = {{0,1,0},{1,1,1},{0,1,0}};
int margin = 1;
int c_gray;
QImage image_out2(image_in.width(),image_in.height(),QImage::Format_RGB32);
image_out2.fill(Qt::white);

for(int x = margin;x <=image_in.width() - margin; x++)
    for(int y = margin;y <= image_in.height() - margin; y++)
    {
        c_gray = qGray(image_out.pixel(x,y));
        if(c_gray ==0)
        {
            for (int i = -margin; i <= margin; i++)
                for (int j = -margin; j <=margin; j++)
                {
                    if(kernel[i + margin][j + margin])
                        image_out2.setPixel(x + i,y + j,qRgb(0,0,0));
                }
        }
    }
```

Hãy tích hợp đoạn code này vào sự kiện clicked của tính năng “ Giãn nở”.

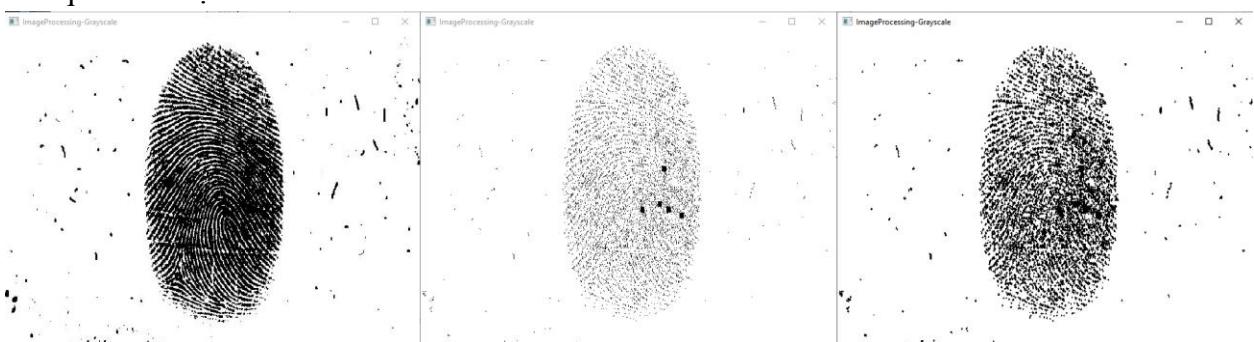
Kết quả thu được :



### 3. Bài tập :

**Bài 1 :** Hãy kết hợp phép ăn mòn + giãn nở để thu được phép toán Open

Kết quả thu được như sau :



**Bài 2 :** Hãy kết hợp phép giãn nở + ăn mòn để thu được phép toán Close  
Kết quả thu được như sau



Ảnh nhị phân đầu vào

Ảnh kết quả bước giãn nở

Ảnh kết quả phép Close

**Bài 3 :** Cho lưu đồ giải thuật của thuật toán OTSU như sau :

---

**Algorithm 1:** Thresholding segmentation using Otsu's method or manual input

---

```

input : input_image (grayscale), overridden_threshold
output: output_image
1 read(input_image)
2 N = input_image.width × input_image.height
      initialize variables
3 threshold, var_max, sum, sumB, q1, q2, μ1, μ2 = 0
4 max_intensity = 255
5 for i=0; i <= max_intensity; i++ do
6   histogram[value] = 0
      accept only grayscale images
7 if num_channels(input_image) > 1 then
8   return error
      compute the image histogram
9 for i=0; i<N; i++ do
10  value = input.image[i]
11  histogram[value] += 1
12 if manual threshold was entered then
13  threshold = overridden_threshold
14 else
      auxiliary value for computing μ2
15  for i=0; i<=max_intensity; i++ do
16    sum += i × histogram[i]
      update q_i(t)
17  for t=0; t<=max_intensity; t++ do
18    q1 += histogram[t]
19    if q1 == 0 then
20      continue
21    q2 = N - q1
      update μ_i(t)
22    sumB += t × histogram[t]
23    μ1 = sumB/q1
24    μ2 = (sum - sumB)/q2
      update the between-class variance
25    σ_b^2(t) = q1(t)q2(t)[μ1(t) - μ2(t)]^2
      update the threshold
26    if σ_b^2(t) > var_max then
27      threshold = t
28      var_max = σ_b^2(t)
      build the segmented image
29 for i=0; i < N; i++ do
30  if input.image[i] > threshold then
31    output.image[i] = 1
32  else
33    output.image[i] = 0
34 return output.image
  
```

---

Hãy viết chương trình tìm ngưỡng tự động bằng giải thuật OTSU dựa trên lưu đồ này.

#### 4. BÀI TẬP TỔNG HỢP :

Sinh viên tạo một dự án mới và tiến hành tích hợp các tính năng đã làm ở các buổi 1,2,3,4,5 vào chung một chương trình. Tạo Project mới với tên là “ImageProcessingTools”. Mở file main.cpp và chỉnh lại code như hình bên dưới để hiển thị form maximize.

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.showMaximized();

    return a.exec();
}
```

Khai báo mdiArea và khai báo hàm void DisplayImage(QImage &image, QString title); bằng cách mở file mainwindow.h lên và thực hiện đoạn code như hình bên dưới:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QMdiArea>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

    QMdiArea *mdiArea;

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private:
    Ui::MainWindow *ui;

};
```

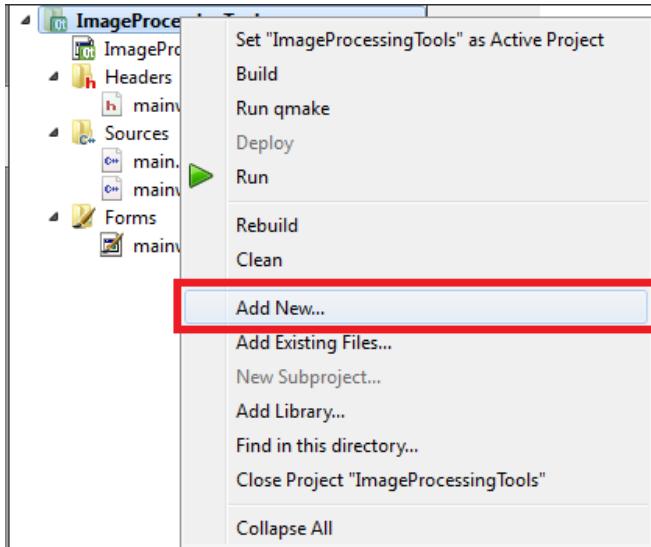
Mở file mainwindow.cpp và thực hiện đoạn code sau:

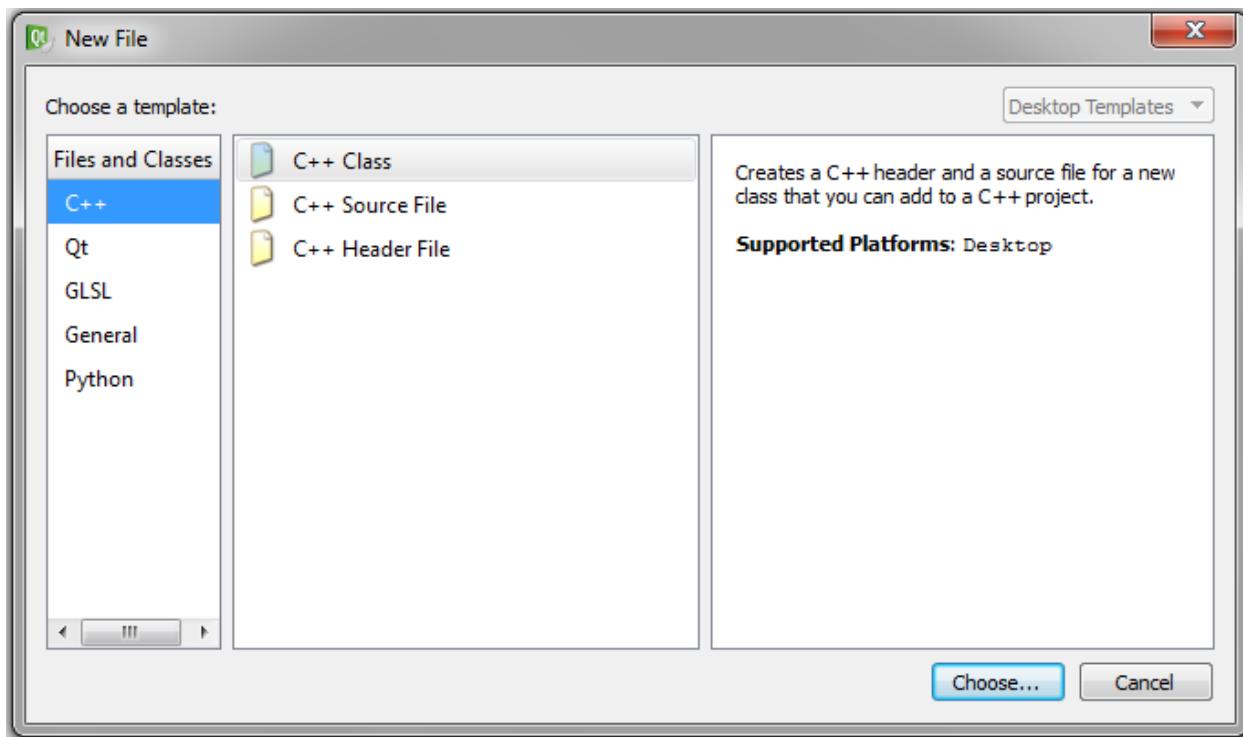
```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    mdiArea = new QMdiArea(this);
    setCentralWidget(mdiArea);
    mdiArea->setVerticalScrollBarPolicy(Qt::ScrollBarAsNeeded);
    mdiArea->setHorizontalScrollBarPolicy(Qt::ScrollBarAsNeeded);
}

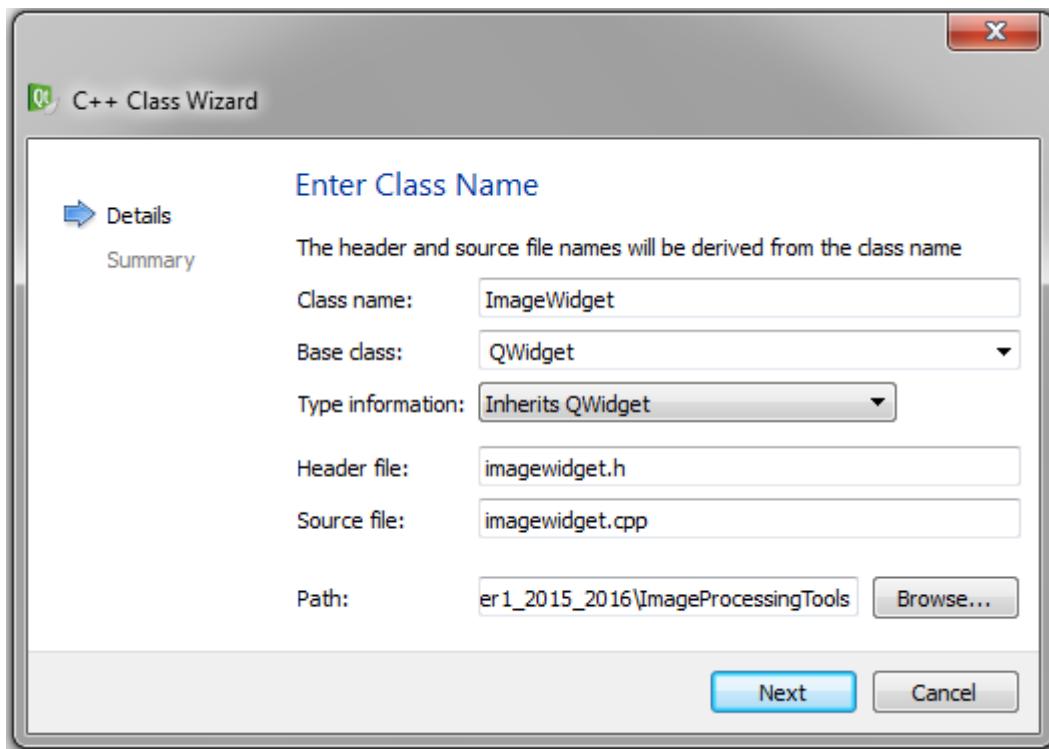
MainWindow::~MainWindow()
{
    delete ui;
}
```

Tạo mới 1 control ImageWidget bằng cách nhấp chuột phải vào tên projet và chọn “Add new” như trong hình.





Đặt tên Class là “ImageWidget” và chọn Base là QWidget



Mở file imagewidget.h và khai báo thêm các dòng lệnh như hình bên dưới:

```
#ifndef IMAGEWIDGET_H
#define IMAGEWIDGET_H

#include <QWidget>

class ImageWidget : public QWidget
{
    Q_OBJECT

    QImage _image;
    void virtual paintEvent(QPaintEvent *);
```

```
public:
    explicit ImageWidget(QWidget *parent = 0);

    QImage getImage();
    void setImage(const QImage &img);
```

```
signals:

public slots:
};

#endif // IMAGEWIDGET_H
```

Mở file imagewidget.cpp và viết code cho các hàm mới vừa tạo.

```
#include "imagewidget.h"
#include "QWidget"
#include "QPainter"

ImageWidget::ImageWidget(QWidget *parent) :
    QWidget(parent)
{
```

```
    QImage ImageWidget::getImage()
    {
        return _image;
    }

    void ImageWidget::setImage(const QImage &img)
    {
        _image = img;
        repaint();
    }

    void ImageWidget::paintEvent(QPaintEvent *pe)
    {
        QWidget::paintEvent(pe);
        QPainter painter(this);
        painter.drawImage(this->rect(), _image);
    }
}
```

Mở file mainwindow.h và khai báo hàm `void DisplayImage(QImage &image, QString title);`

```
#define MAINWINDOW_H

#include <QMainWindow>
#include <QMdiArea>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

    QMdiArea *mdiArea;

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

    void DisplayImage(QImage &image, QString title);
}

private:
    Ui::MainWindow *ui;
};

};
```

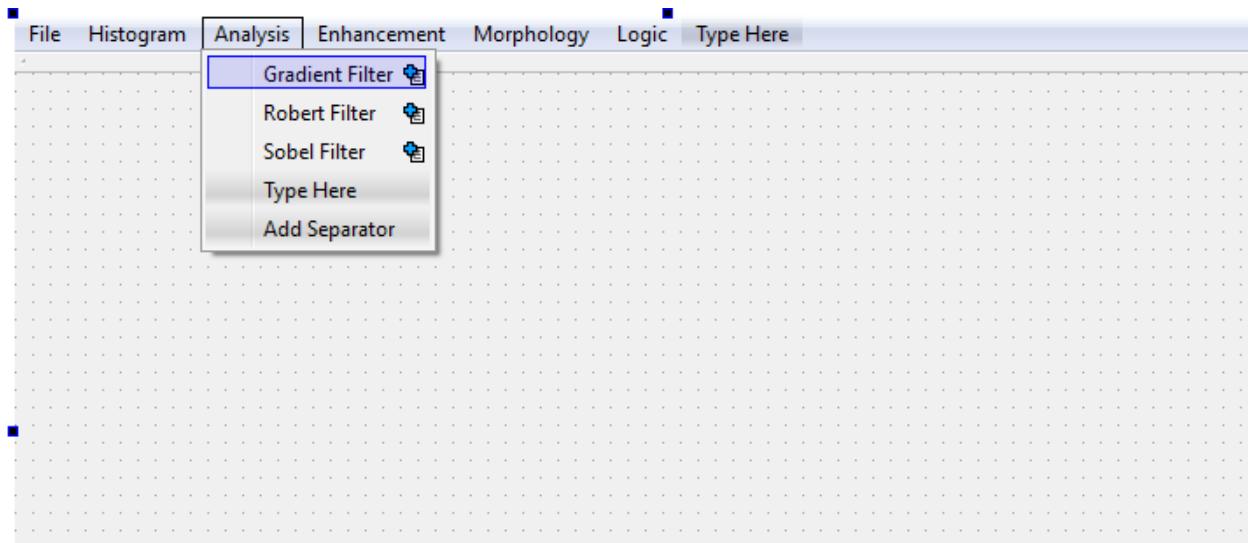
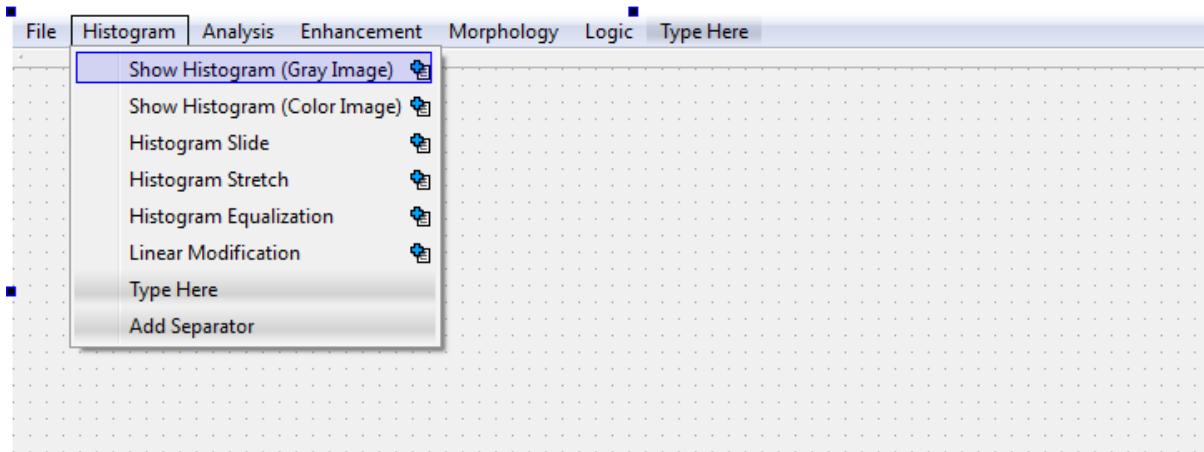
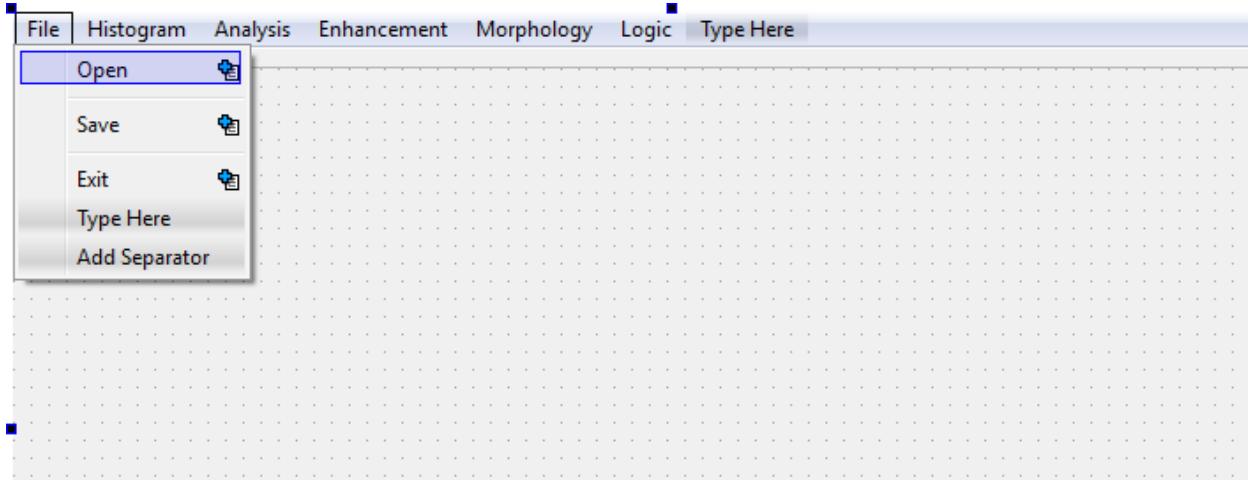
Mở file mainwindow.cpp và thêm vào các thư viện sau đây:

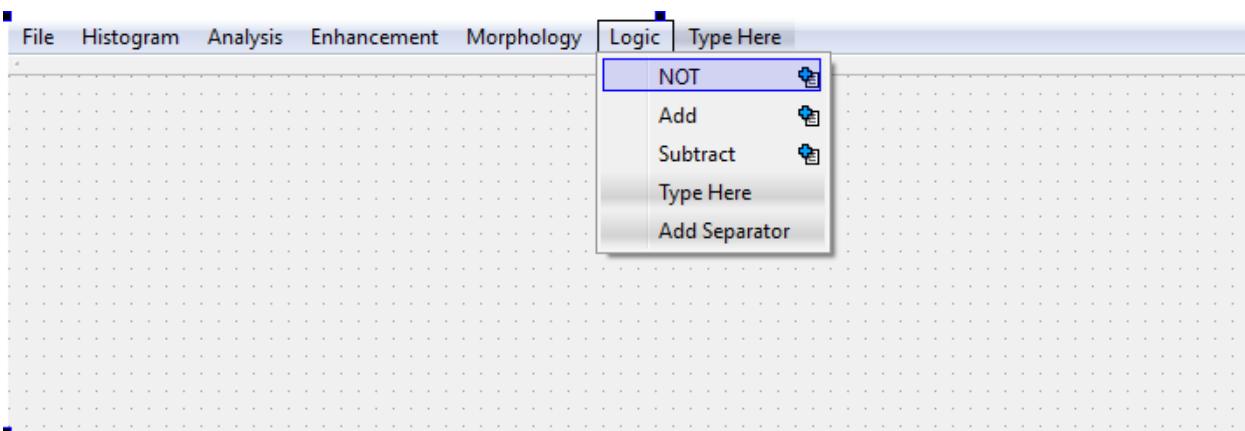
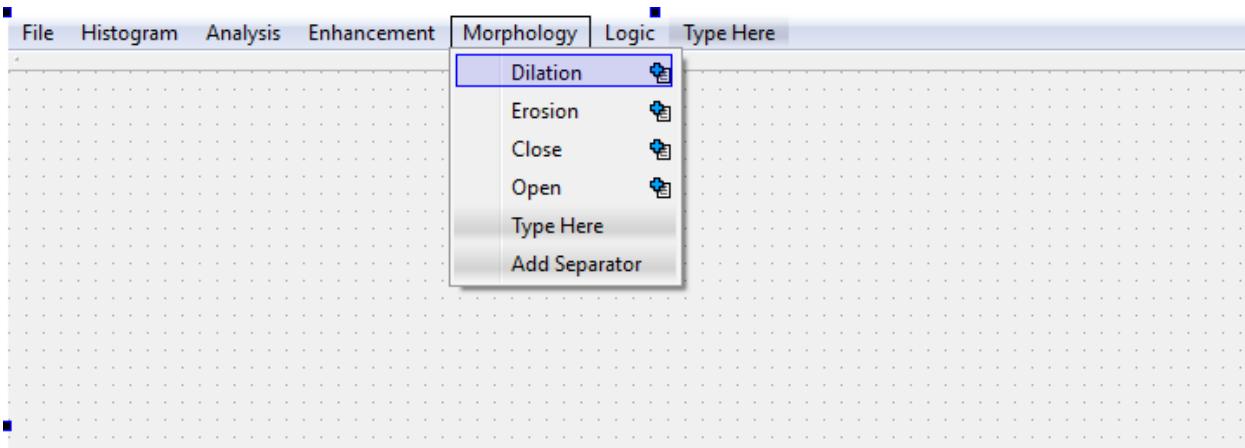
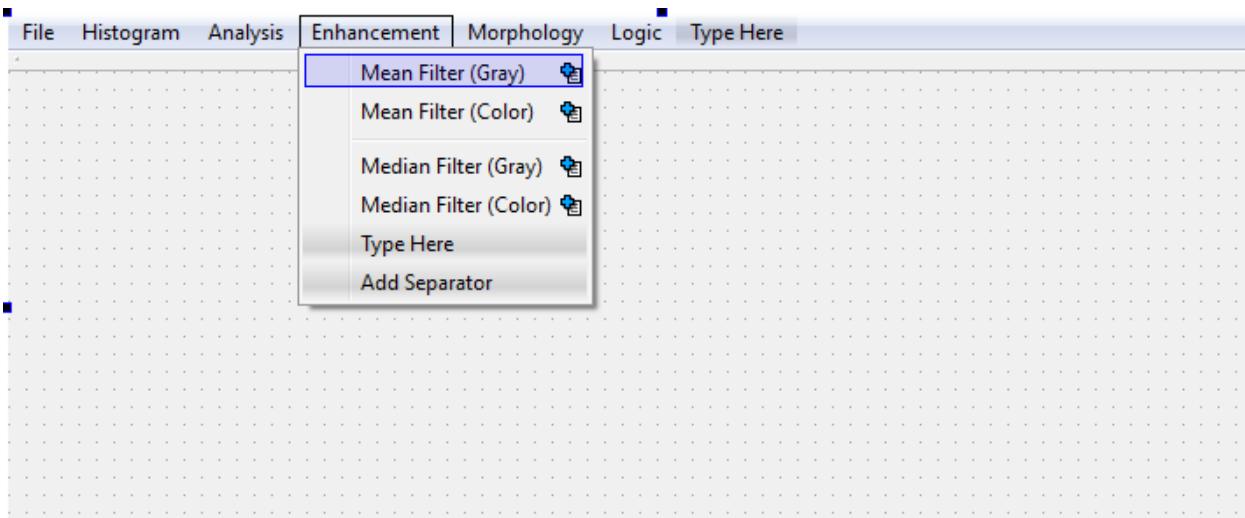
```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "QMdiSubWindow"
#include "imagewidget.h"
#include "QFileDialog"
```

Và viết code cho hàm `void DisplayImage(QImage &image, QString title);`

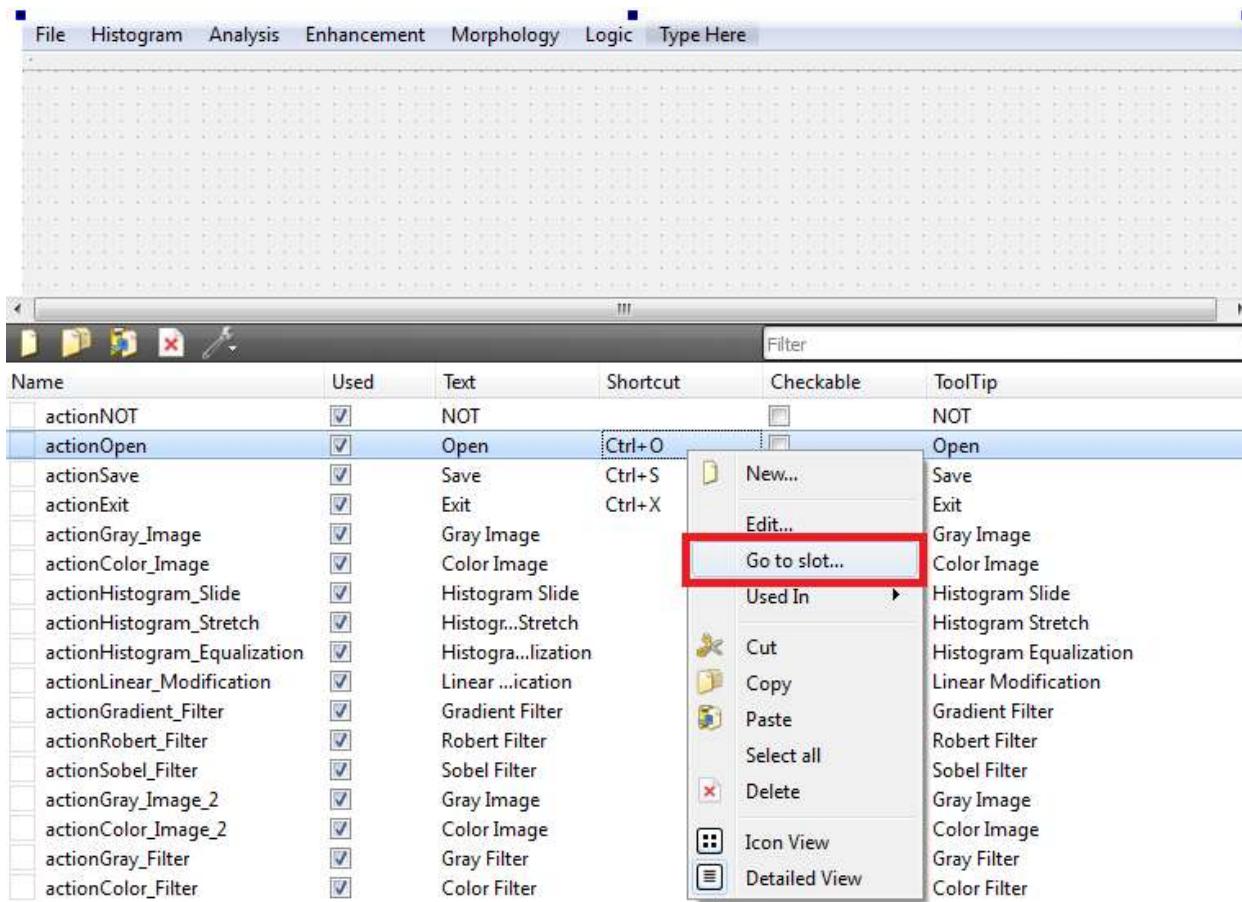
```
void MainWindow::DisplayImage(QImage &image, QString title)
{
    if (image.isNull())
        return;
    QMdiSubWindow *subWindow = new QMdiSubWindow();
    subWindow->setAttribute(Qt::WA_DeleteOnClose);
    mdiArea->addSubWindow(subWindow);
    subWindow->setWindowTitle(title);
    ImageWidget *imagewidget = new ImageWidget();
    imagewidget->setImage(image);
    subWindow->setWidget(imagewidget);
    subWindow->resize(image.size());
    subWindow->show();
}
```

Mở filemainwindow.ui và thiết kế menu như các hình bên dưới:





Viết code cho chức năng “open” trên thanh menu (viết tại file mainwindow.cpp). Để chọn xử lý sự kiện khi click chuột vào menu “File->open”, trong danh sách các đối tượng của menu phía bên dưới màn hình, nhấp phải vào đối tượng “Open”, chọn “Go to slot...” như hình



Chọn signal “triggered()” và viết hàm xử lý như bên dưới:

```
void MainWindow::on_actionOpen_triggered()
{
    QString fileName = QFileDialog::getOpenFileName(this, "Open File",
                                                    "C:/CVIPtools/images",
                                                    "*.* All Files; *.bmp; *.jpeg; *.ppm; *.png; *.jpg");
    QImage image(fileName);
    DisplayImage(image, QFileInfo(fileName).fileName());
}
```

Thực hiện tương tự và code cho chức năng Save.

```
void MainWindow::on_actionSave_triggered()
{
    QMdiSubWindow *activeWindow = mdiArea->currentSubWindow();
    if (activeWindow == 0)
        return;
    ImageWidget *activeWidget = (ImageWidget*)activeWindow->widget();
    QImage image = activeWidget->getImage();
    QString selectedFilter;
    QString fileName = QFileDialog::getSaveFileName(this, "Save image",
                                                    "C:/CVIPtools/images",
                                                    "BMP (*.bmp);;PNG (*.png);;JPEG (*.jpg);;All files (*.*)",
                                                    &selectedFilter);
    if (selectedFilter == "BMP (*.bmp)")
        image.save(fileName, "BMP");
    else if (selectedFilter == "PNG (*.png)")
        image.save(fileName, "PNG");
    else if (selectedFilter == "JPEG (*.jpg)")
        image.save(fileName, "JPEG");
    else
        image.save(fileName);
}
```

Code cho chức năng Exit

```
void MainWindow::on_actionExit_triggered()
{
    QApplication::exit();
}
```

Hướng dẫn thực hiện tổng hợp code các buổi thực hành 1, 2, 3 và 4. Sinh viên mở file mainwindow.h và khai báo hàm draw\_His\_Gray(QImage &image\_in)

```
private:  
    Ui::MainWindow *ui;  
  
    QImage draw_His_Gray(QImage &image_in);
```

Mở file mainwindow.cpp và thực hiện code cho chức draw\_His\_Gray(QImage &image\_in)

```
QImage MainWindow::draw_His_Gray(QImage &image_in)  
{  
    const int HEIGHT = 128;//Chieu cao to chuc do  
    QImage histogram(256, HEIGHT, QImage::Format_ARGB32);  
    histogram.fill(Qt::white);//Khoi tao to chuc do mau trang  
    int h[256];  
    for (int i=0; i<256; i++)//Khoi tao 256 mang luu tru so luong pixel  
        h[i]=0; //tuong ung voi muc xam 0...255  
    for (int x=0; x<image_in.width(); x++)  
        for (int y=0; y<image_in.height(); y++) {  
            QRgb rgb = image_in.pixel(x, y);  
            int gray = qGray(rgb);  
            h[gray]++; //Dem so luong pixel tuong ung voi muc xam 0...255  
        }  
    int max=0;  
    for (int i=0; i<256; i++)  
        if (h[i]>max) max=h[i];  
    int lineheight = 0;  
    for (int x=0; x<256; x++) {  
        lineheight = HEIGHT*h[x]/max; //Tinh ty le  
        for (int y=HEIGHT-1; y>HEIGHT-1-lineheight; y--)  
            histogram.setPixel(x, y, qRgb(0, 0, 255)); //Ve to chuc do  
    }  
    return histogram;  
}
```

Để chọn xử lý sự kiện khi click chuột vào menu “Histogram->Show Histogram (Gray Image)”. Trong danh sách các đối tượng của menu phía bên dưới màn hình, chọn action tương ứng chức năng Show\_Histogram\_Gray\_Image và thực hiện đoạn code sau:

```
void MainWindow::on_actionShow_Histogram_Gray_Image_triggered()  
{  
    QMdiSubWindow *activeWindow = mdiArea->currentSubWindow();  
    if (activeWindow == 0)  
        return;  
    ImageWidget *activeWidget = (ImageWidget*) activeWindow->widget();  
    QImage image_in = activeWidget->getImage();  
    QImage image_out = draw_His_Gray(image_in);  
    DisplayImage(image_out, activeWindow->windowTitle()+"_Histogram");  
}
```

Sinh viên mở file mainwindow.h và khai báo hàm draw\_His\_Color(QImage &image\_in)

```
private:  
    Ui::MainWindow *ui;  
  
    QImage draw_His_Gray(QImage &image_in);  
    QImage draw_His_Color(QImage &image_in);
```

Mở file mainwindow.cpp và thực hiện code cho chức draw\_His\_Color(QImage &image\_in)

```
QImage MainWindow::draw_His_Color(QImage &image_in)  
{  
    const int HEIGHT = 128;  
    int h_red[256], h_blue[256], h_green[256];  
    for (int i=0; i<256; i++){  
        h_red[i] = 0;  
        h_green[i] = 0;  
        h_blue[i] = 0;  
    }  
    for (int x=0; x<image_in.width(); x++)  
        for (int y=0; y<image_in.height(); y++) {  
            QColor color = image_in.pixel(x,y);  
            int red = color.red();  
            int green = color.green();  
            int blue = color.blue();  
            h_red[red]++;  
            h_green[green]++;  
            h_blue[blue]++;  
        }  
    int max = 0;  
    for (int i=0; i<256; i++){  
        if (max < h_red[i]) max = h_red[i];  
        if (max < h_green[i]) max = h_green[i];  
        if (max < h_blue[i]) max = h_blue[i];  
    }  
    QImage histogram = QImage(256, HEIGHT*3,QImage::Format_RGB32);  
    histogram.fill(Qt::white);  
    int lineHeight;  
    for (int i=0; i<256; i++){  
        lineHeight = HEIGHT*h_red[i]/max;  
        for (int j=HEIGHT-1; j>HEIGHT-lineHeight; j--)  
            histogram.setPixel(i,j,qRgb(255,0,0));  
        lineHeight = HEIGHT * h_green[i]/max;  
        for (int j=2*HEIGHT-1; j>2*HEIGHT-lineHeight; j--)  
            histogram.setPixel(i,j,qRgb(0,255,0));  
        lineHeight = HEIGHT * h_blue[i]/max;  
        for (int j=3*HEIGHT-1; j>3*HEIGHT-lineHeight; j--)  
            histogram.setPixel(i,j,qRgb(0,0,255));  
    }  
    return histogram;  
}
```

Để chọn xử lý sự kiện khi click chuột vào menu “Histogram->Show Histogram (Color Image)”. Trong danh sách các đối tượng của menu phía bên dưới màn hình, chọn action tương ứng chức năng Show\_Histogram\_Color\_Image và thực hiện đoạn code sau:

```
void MainWindow::on_actionShow_Histogram_Color_Image_triggered()
{
    QMdiSubWindow *activeWindow = mdiArea->currentSubWindow();
    if (activeWindow == 0)
        return;
    ImageWidget *activeWidget = (ImageWidget*) activeWindow->widget();
    QImage image_in = activeWidget->getImage();
    QImage image_out = draw_His_Color(image_in);
    DisplayImage(image_out, activeWindow->windowTitle()+"_Histogram");
}
```

### Bài tập:

Hãy tích hợp tất cả các tính năng đã học vào công cụ ImageProcessingTools.