TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

**VIỆN ĐIỆN TỬ - VIỄN THÔNG**


# ĐỒ ÁN I

**Đề tài:**

## Xây dựng chương trình quản lý hình học, quản lý đường nối tâm các hình, vẽ các hình và đường nối tâm

Giảng viên hướng dẫn: PGS.TS Nguyễn Đức Minh


| Sinh viên thực hiện | MSSV | Lớp |
|---|---|---|
| Vũ Đức Thái | 20172804 | ĐTVT.08-K62 |
| Đào Đức Dũng | 20172492 | ĐTVT.08-K62 |
| Nguyễn Văn Đạt | 20172458 | ĐTVT.08-K62 |


Hà Nội, 7/2020

# LỜI NÓI ĐẦU

Sự tiến bộ của khoa học kỹ thuật ngày nay, cùng với sự phát triển mạnh mẽ của nền kinh tế thế giới nói chung; công nghiệp hóa - hiện đại hóa ở Việt Nam cũng đang có sự tiến bộ từng ngày. Điều này thể hiện qua việc máy tính điện tử đang dần trở nên phổ biến và gần gũi với con người, giúp con người tăng hiệu suất làm việc. Mục tiêu hàng đầu của các doanh nghiệp hiện nay là phát triển các phần mềm hệ thống quản lý của mình giúp con người quản lý tài nguyên, tiết tiệm nhân lực và nâng cao hiệu suất làm việc. Để có thể hiểu hơn về kỹ thuật này chúng em lựa chọn đề tài "Xây dựng chương trình quản lý hình học, các đường nối tâm và vẽ các đối tượng" cho dự án của mình.

Ngôn ngữ lập trình C++ là một trong những ngôn ngữ lập trình hướng đối tượng mạnh và phổ biến hiện nay do tính mềm dẻo và đa năng của nó. Không chỉ các ứng dụng được viết trên C++ mà cả những chương trình hệ thống lớn đều được viết hầu hết trên C++. C++ là ngôn ngữ lập trình được viết trên nền tảng của C, không những khắc phục được một số đặc điểm của ngôn ngữ C mà quan trọng hơn, C++ cung cấp cho người sử dụng một phương tiện lập trình theo kỹ thuật mới: lập trình hướng đối tượng. Đây là kỹ thuật lập trình được sử dụng hầu hết trong các ngôn ngữ mạnh hiện nay, đặc biệt là các ngôn ngữ hoạt động trong môi trường Windows như Microsoft Access, Visual Basic, Visual Foxpro… . Bên cạnh đó chúng em thiết kế giao diện đồ họa trong C++, sử dụng Qt framework theo mô hình MVC (Model – View - Controller). Trong quá trình thực hiện cũng như nội dung bài báo cáo có không tránh khỏi những thiếu sót, chúng em rất mong được sự đóng góp của thầy và các để bài báo cáo được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn thầy Nguyễn Đức Minh đã tạo điều kiện hướng dẫn và cung cấp những tài liệu cần thiết trong quá trình hoàn thành bài báo cáo này.

***Chúng em xin chân thành cám ơn !***

# MỤC LỤC

# DANH MỤC HÌNH ẢNH



# DANH MỤC BẢNG BIỂU

# CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

***Mở đầu chương:*** Trong chương này, chúng ta sẽ giới thiệu khái quát nhất về những vấn đề được sử dụng trong dự án, các mục tiêu phạm vi sẽ đề cập, các hướng tiếp cận và các các thuật ngữ viết tắt trong báo cáo này.

## 1.1 Tổng quan

Chương trình được đặt tên là ShapeMana sẽ quản lý các đối tượng hình học gồm có: Circle (hình tròn), Square (hình vuông), Rectangle (hình chữ nhật), Oval (hình ellipse), Triangle (hình tam giác), Line (đoạn thẳng). Chương trình sẽ quản lý các cạnh nối tâm của các hình (đối tượng Edge) có nhiệm vụ nối tâm các hình theo yêu cầu của người dùng. Sau đó, Chương trình sẽ vẽ tất cả các đối tượng được người dùng nhập vào.

## 1.2 Mục tiêu và phạm vi

Mục tiêu của đề tài sẽ nhắm đến việc xây dựng một hệ thống quản lý tất cả các đối tượng hình học, các đường nối tâm. Nghiên cứu và tìm hiểu về Qt framework cụ thể như sau:

- Vận dụng nâng cao ngôn ngữ C++ (tất cả các khái niệm về OOP trong C++)
- Nghiên cứu cấu trúc cơ bản của một chương trình Qt cụ thể
- Nghiên cứu và vận dụng mô hình MVC trong Qt framework để xây dựng chương trình trên
- Nghiên cứu và sử dụng một số Design Patterns trong thư viện Shape – sẽ được trình bầy ở các nội dung sau
- Xây dựng chương trình ShapeMana có tất cả các chức năng cơ bản để quản lý các hình, các cạnh nối tâm (thêm, sửa, xóa, tìm kiếm,…) và sử dụng Qt để vẽ Các đối tượng lên màn hình.

## 1.3 Các phương pháp tiếp cận

### 1.3.1 Qt framework

Qt (được phát âm là “cute”, không phải “cu-te”) là một framework đa nền tảng, nó được sử dụng như một bộ cụ đồ họa, mặc dù nó cũng hữu ích để tạo ra những ứng dụng CLI (tương tác với người dùng đưa ra lệnh cho chương trình dưới dạng các dòng lệnh liên tiếp nhau). Nó chạy chủ yếu trên ba nền tảng chính đó là: các thiết bị máy tính để bàn (đa nền tảng), các nền tảng OS trên điện thoại (Symbian, Nokia Belle, Meego Harmattan, MeeGo or BB10) và các thiết bị nhúng. Các mảng khác về Android và IOS cũng đang được phát triển [1]

Qt được biết đến với một bô sưu tập ấn tượng các modules, gồm có:

- **QtCore** là một thư viện cơ bản nhất, cung cấp các bộ chứa (Containers), quản lý luồng (Thread Management), quản lý các sự kiện (Event Management), và nhiều thứ khác…
- **QtGui** và **QtWidgets** là một bộ công cụ GUI cho máy tính để bàn, nó cung cấp phần lớn các thành phần đồ họa để thiết kế các ứng dụng.
- **QtNetwork** là thư viện cung cấp một tập các lớp hữu dụng để giải quyết các vấn đề về kết nối mạng.
- **QtWebkit** là một bộ dụng cụ được sử dụng để cho phép sử dụng các trang web và các ứng dụng web trong ứng dụng Qt.
- **QtSQL** là một lớp trừu tượng có các đặc tính đầy đủ của SQL RDBM có khả năng mở rộng, hỗ trợ cho ODBC, SQLITE, MySQL và PostgreSQL có sẵn trong đầu ra của lớp.
- **QtXML** hỗ trợ việc phân tích XML đơn giản và DOM.
- **QtXmlPatterns** hỗ trợ cho XSLT, XPath, XQuery and các sơ đồ hợp được hợp thức hóa.

### 1.3.2 MVC Pattern (Software Design Pattern)

Model – View – Controller (MVC) là một mẫu thiết kế nhằm mục tiêu chia tách phần Giao diện và Code để dễ quản lý, phát triển và bảo trì. MVC chia ứng dụng phần mềm ra làm 3 phần có tương tác với nhau là Model (Dữ liệu), View (Giao diện), Controller (Điều khiển tương tác giữa Model và View) [2]

### 1.3.3 Design Patterns được sử dụng trong thư viện Shape

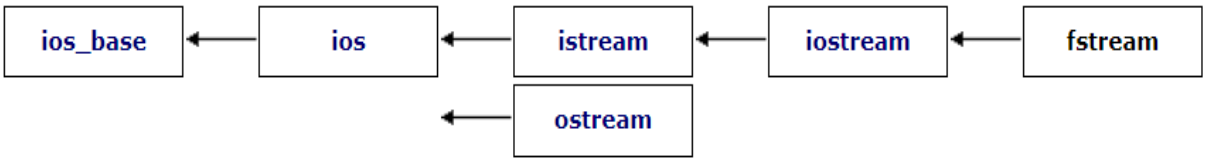
#### *1.3.3.1 Factory Method*

Factory Method là một mẫu thiết kế thuộc kiểu tạo đối tượng (Creational Design Pattern). Người dùng tạo đối tượng mà không cần mô tả rõ các đối tượng, người dùng sử dụng cùng một giao diện chung (Common Interface) để tạo ra một kiểu đối tượng mới.

Ý tưởng ở đây sẽ sử dụng một phương thức tĩnh (Static Factory Function), nó sẽ tạo ra và trả về một những đối tượng cụ thể, ẩn đi những chi tiết từ phía người dùng [3].

### 1.3.4 Đọc ghi file text trong thư viện Shape

Phần đọc ghi file text sử dụng thư viện <fstream> của C++ để tạo ra các phương thức đọc ghi file text. [4]




**Hình 1.1: Input/Output file stream system**

## 1.4 Thuật ngữ viết tắt

Trong báo cáo này, chúng ta sẽ sử dụng một vài thuật ngữ viết tắt bằng tiếng anh, cụ thể được trình bầy trong *Bảng 1.1:*

| Từ viết tắt | Tiếng Anh | Tiếng Việt |
|---|---|---|
| MVC | Model-View-Controller | Mô hình - Giao diện – Điều khiển |
| OOP | Object-Oriented Programing | Lập trình hướng đối tượng |
| CLI | Command-Line Interface | Giao diện dòng lệnh |
| OS | Operating System | Hệ điều hành |
| GUI | Graphic User Interface | Giao diện đồ họa người dùng |
| RDBM | Relational Database Management System | Hệ thống quản lý cơ sở dữ liệu |
| DOM | Document Object Model | Mô hình đối tượng tài liệu |

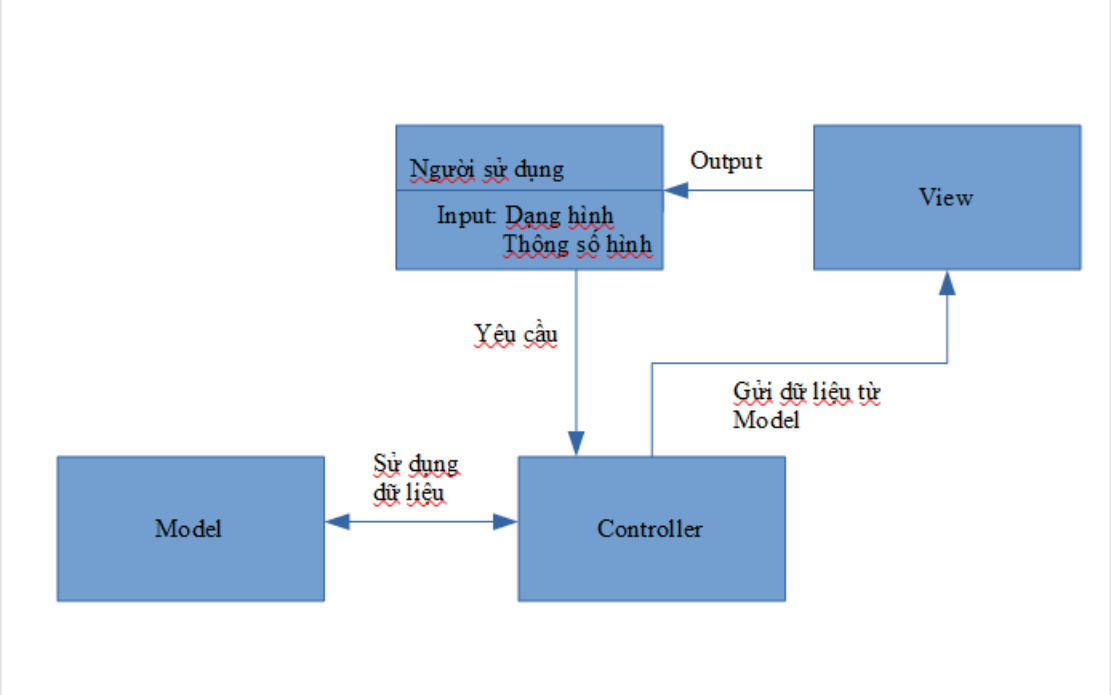
**Bảng 1.1: Bảng tra cứu thuật ngữ viết tắt**

***Kết luận chương:*** Vậy trong chương này, chúng ta đã có một cái nhìn tổng quan về chương trình, cũng như điểm qua một vài thuật ngữ sẽ được sử dụng trong dự án. Phần tiếp theo chúng ta sẽ nghiên cứu khái quát hơn về hệ thống, các khối chính được phân chia một cách rõ ràng hơn.

# CHƯƠNG 2: TỔNG QUAN VỀ HỆ THỐNG

***Mở đầu chương:*** Trong phần này, chúng ta sẽ phân tích tổng quan các khối chức năng chính của chương trình, được thiết kế theo mô hình MVC với đầu vào đầu ra của hệ thống tương tác trực tiếp với người sử dụng.

Chương trình sẽ sử dụng hai phương pháp chính là MVC Pattern và Factory Method. Với đầu vào của cả hai phương pháp là yêu cầu từ người sử dụng về loại và thông số hình, sau đó hệ thống sẽ thực thi các lệnh và đưa ra là các hình tương ứng với thông số và dạng mà người sử dụng yêu cầu.




**Hình 2.1: Sơ đồ khối về tổng quan của hệ thống**

Hình 2.1: Sơ đồ khối về tổng quan của hệ thốngHình 2.1 mô tả tổng quan về hệ thống, người sử dụng đưa ra yêu cầu cho khối Controller xử lý, khối Controller sử dụng dữ liệu từ Model và gửi dữ liệu đến cho người dùng thông qua View.

***Kết luận chương:*** Vậy trong chương 2 này, chúng ta đã có cái nhìn tổng quan về hệ thống được thiết kế theo mô hình MVC, các luồng dữ liệu vào ra của hệ thống. Trong chương tiếp theo sẽ trình bày về các tiêu chí thiết kế hệ thống.

# CHƯƠNG 3: TIÊU CHÍ THIẾT KẾ

***Mở đầu chương:*** Trong chương này, chúng ta sẽ trình bầy về các tiêu chí để thiết kế chương trình ShapeMana, giúp chúng ta định hình được mục tiêu lập trình, sử dụng các mẫu "Design Patterns" hợp lý để tránh hiểm họa lâu dài.

## 3.1 Chỉ tiêu kỹ thuật

Các phương pháp được sử dụng bao gồm MVC Pattern, Factory Method.

- MVC Pattern có chức năng chia tách phần giao diện và phần code để dễ quản lý, ngoài ra thành phần dữ liệu (Model) sẽ không ảnh hưởng nhiều đến giao diện của người dùng vì mô hình đưa ra Model để không cho người dùng thao tác trực tiếp vào dữ liệu mà phải thông qua Model, do vậy cho dù dữ liệu vật lý thay đổi cấu trúc nhưng cấu trúc của Model cho việc truy cập, xử lý, lưu trữ dữ liệu sẽ không bị ảnh hưởng. Đầu vào của thuật toán là các yêu cầu của người dùng thông qua giao diện, sau đó các dữ liệu sẽ được lưu trữ trong Database, cuối cùng yêu cầu được xử lý và dữ liệu được xuất ra cho người dùng thông qua giao diện.
- Factory Method thì có chức năng là quản lý và trả về các đối tượng theo yêu cầu, giúp cho việc khởi tạo đối tượng trở nên đơn giản hơn. Với đầu vào của thuật toán là các yêu cầu về việc tạo Shape mà người dùng muốn tạo, khi đó các subclass (class con) sẽ thực hiện các phương thức của supperclass (class cha) theo nghiệp vụ riêng của nó và trả về một trong những sub class đó. Factory class sử dụng if-else hoặc switch – case để xác định class con đầu ra.

## 3.2 Môi trường hoạt động (Environments)

Môi trường được sử dụng để chạy phân mềm là Qt, đây là một Application Framework. Mục tiêu của Qt là tạo ra một framework có khả năng thiết kế những phần mềm có thể chạy trên nhiều nền tảng phần mềm lẫn phần cứng khác nhau mà không phải thay đổi nhiều về code.

***Kết luận chương:*** Sau khi phân tích xong, chúng ta đã có cái nhìn tổng thể cho chương trình và các cách thiết kế sẽ được sử dụng. Trong chương tiếp theo chúng ta sẽ phân tích hệ thống kỹ hơn.

# CHƯƠNG 4: THIẾT KẾ HỆ THỐNG

***Mở đầu chương:*** Trong chương này, chúng ta sẽ trình bầy chi tiết về cấu trúc dữ liệu, thuật toán sẽ sử dụng trong chương trình. Định hướng những tài nguyên sẽ sử dụng và sơ đồ chi tiết các kiến trúc.

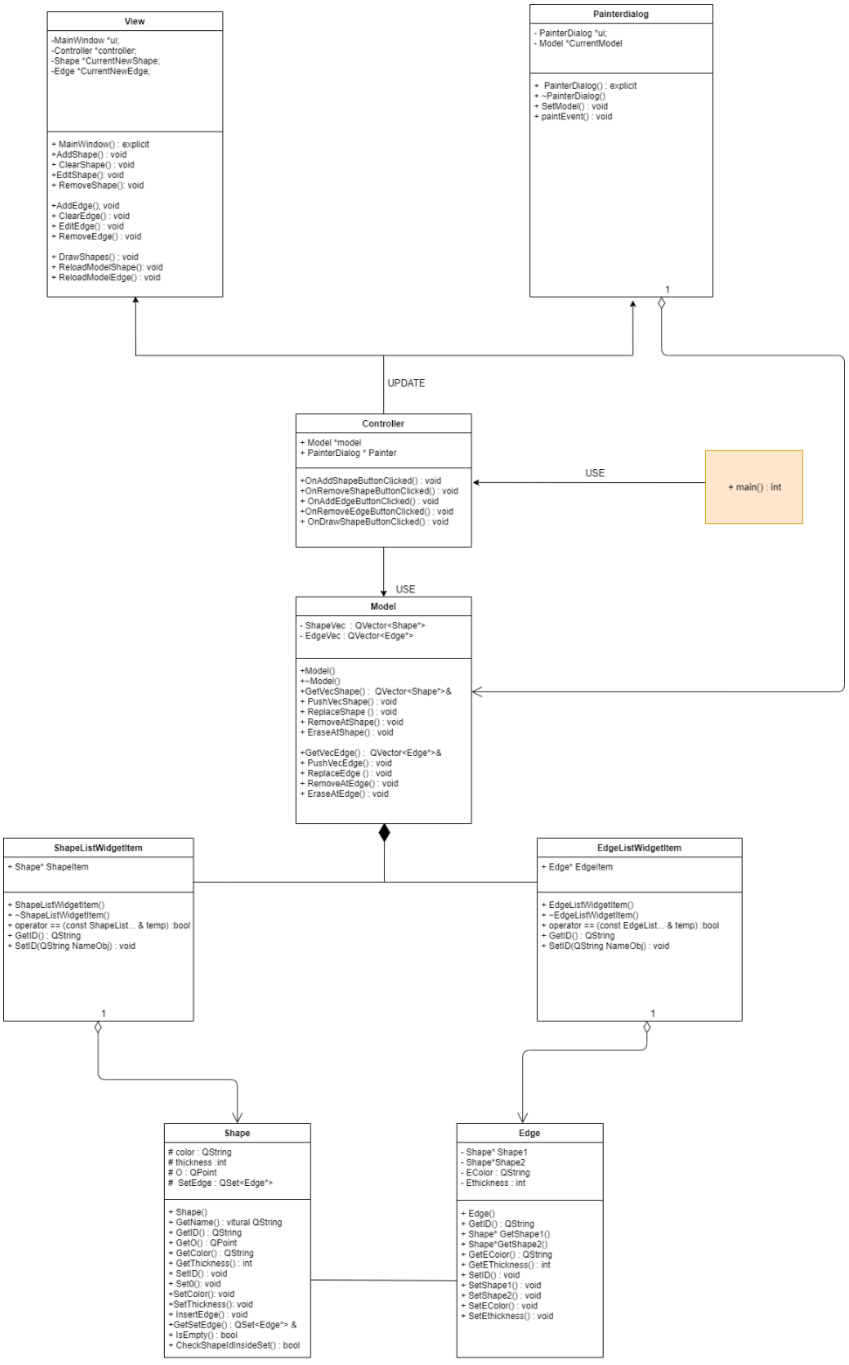
## 4.1 Cấu trúc dữ liệu

Chi tiết về cấu trúc dữ liệu của hai phương pháp MVC Pattern và Factory Method:

- Phương pháp MVC Pattern với đầu vào là các thông số về hình dạng (Shape) và cạnh nối giữa các hình (Edge) do người dùng yêu cầu thông qua View, sau đó các dữ liệu về Shape và Edge sẽ được lưu trữ trong Database, cuối cùng yêu cầu được xử lý và đưa ra màn hình các hình vẽ và các cạnh nối tâm tương ứng với yêu cầu của người dùng.
- Với phương pháp Factory Method thì đầu vào sẽ là các yêu cầu của người dùng về loại hình mà người dùng muốn tạo (Circle, Rectangle, Square, Oval, Straight Line, Triangle), và đầu ra của thuật toán là các class con, chính là các hình và thông tin của hình mà người dùng yêu cầu.

## 4.2 Thuật toán

Sơ đồ UML của phương pháp MVC Pattern:

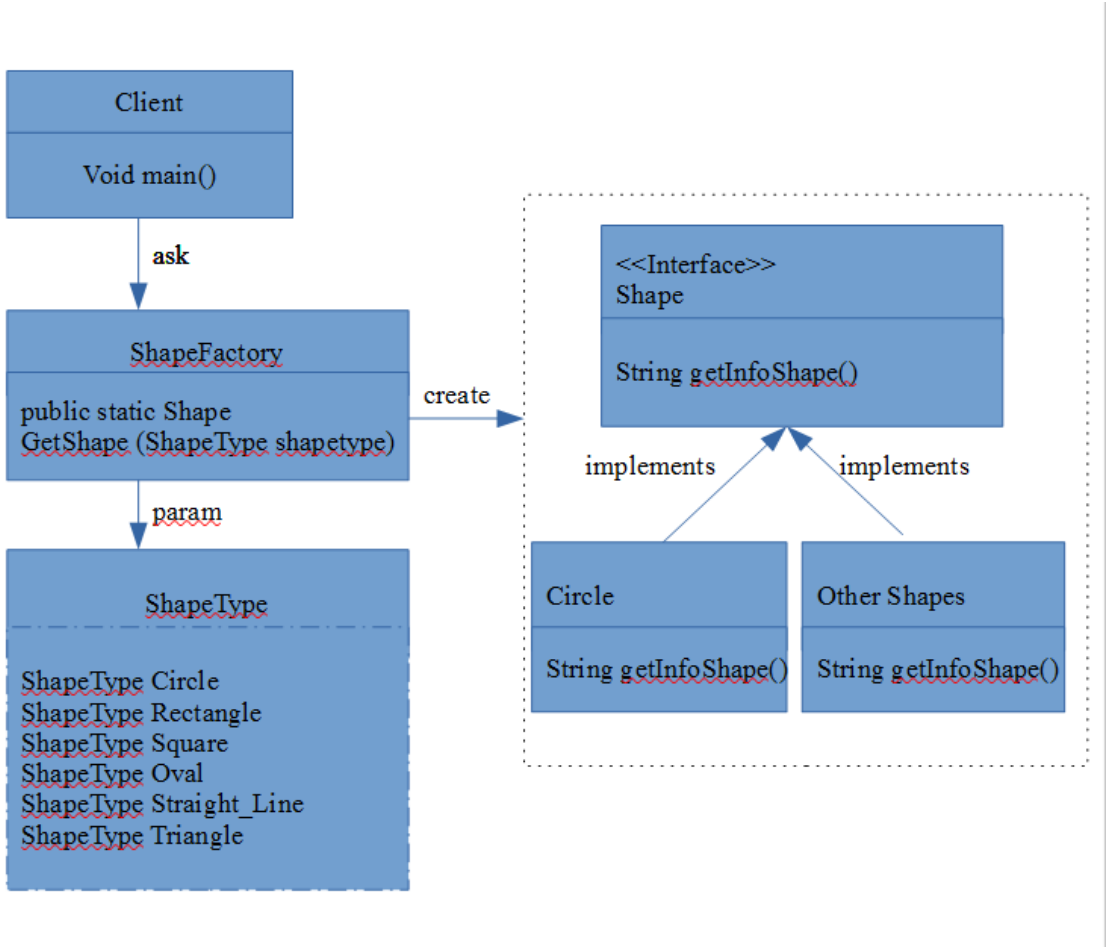



**Hình 4.1: Sơ đồ UML của phương pháp MVC Pattern**

Do sơ đồ UML có kích thước lớn dẫn đến hình ảnh chưa được rõ nên nhóm sẽ đặt đường link đến sơ đồ UML ở dưới. (Trong phần phụ lục 1)

***Lỗi! Không tìm thấy nguồn tham chiếu.*** mô tả mô hình MVC của chương trình ShapeMana bao gồm 3 khối Model chứa các dữ liệu của Shape và Edge dưới dạng vector, khối Controller chứa các hàm thao tác với dữ liệu và khối View chứa giao diện với người dùng.

Sơ dồ UML của phương pháp Factory Pattern:




**Hình 4.2: Sơ đồ UML của phương pháp Factory Method**

*Hình 4.2* mô tả hoạt động của phương pháp Factory Method với chương trình ShapeMana cụ thể. Với đầu vào là yêu cầu của người sử dụng và đầu ra là các hình và các thông tin của hình mà người sử dụng yêu cầu.

***Kết luận chương:*** Vậy trong chương này, chúng ta đã trình bầy về cấu trúc dữ liệu, cũng như thuật toán, cũng như việc khai thác sử dụng tài nguyên trong chương trình. Phần tiếp theo sẽ là bước triển khai dự án.

# CHƯƠNG 5: TRIỂN KHAI THỰC HIỆN

***Mở đầu chương:*** Trong chương này sẽ trình bầy về cách thức triển khai chương trình từ những cơ sở phân tích, định hướng ở các chương trên.

## 5.1 Cấu trúc chung

Phần này trình bày cấu trúc chung của các tệp chương trình dùng để triển khai thuật toán. Sử dụng các cấu trúc dữ liệu, thuật toán ở trong chương 4.

## 5.2 Một số đoạn mã quan trọng

### 5.2.1 Tạo Model cho Shape và Edge

Model là nơi chứa các dữ liệu của lớp Shape và lớp Edge. Các thuộc tính của Shape và Edge được lưu trữ dưới dạng Container : QVector.

```
#ifndef MODEL_H
#define MODEL_H

#include<QVector>
#include "controller.h"
#include "shape.h"

class Model
{
//    friend class Controller;
private:
    QVector<Shape*> ShapeList;
//    QVector<Edge*>  EdgeList;
public:
    Model();
    ~Model();
    // Declare method to react the database ...
public:
    QVector<Shape*> GetList();
    void PushShapeList(Shape* temp);
};

#endif // MODEL_H
```

**Hình 5.1: Model chứa dữ liệu của Shape và Edge**

*Hình 5.1* mô tả khối Model cho chương trình Shape, tại đây Shape và Edge được lưu dưới dạng vector là ShapeList và EdgeList.

Các thuộc tính chung của Shape : Point O, Qstring color, Qstring thickness. Mỗi subclass ( Circle, Square, Rectangle, Over,…. ) có những thuộc tính riêng biệt của chúng được khai báo ở từng subclass và kế thừa những thuộc tính của lớp Shape. Các object này được lưu trữ ở QVector <Shape*> Shapelist.

Các thuộc tính của Edge: Qstring EColor, Qstring EThickness, Shape* shape1, Shape* shape2. Các object của lớp Edge được lưu trữ ở Qvector <Edge*> Edgelist.

### 5.2.2 Tạo Controller cho chương trình

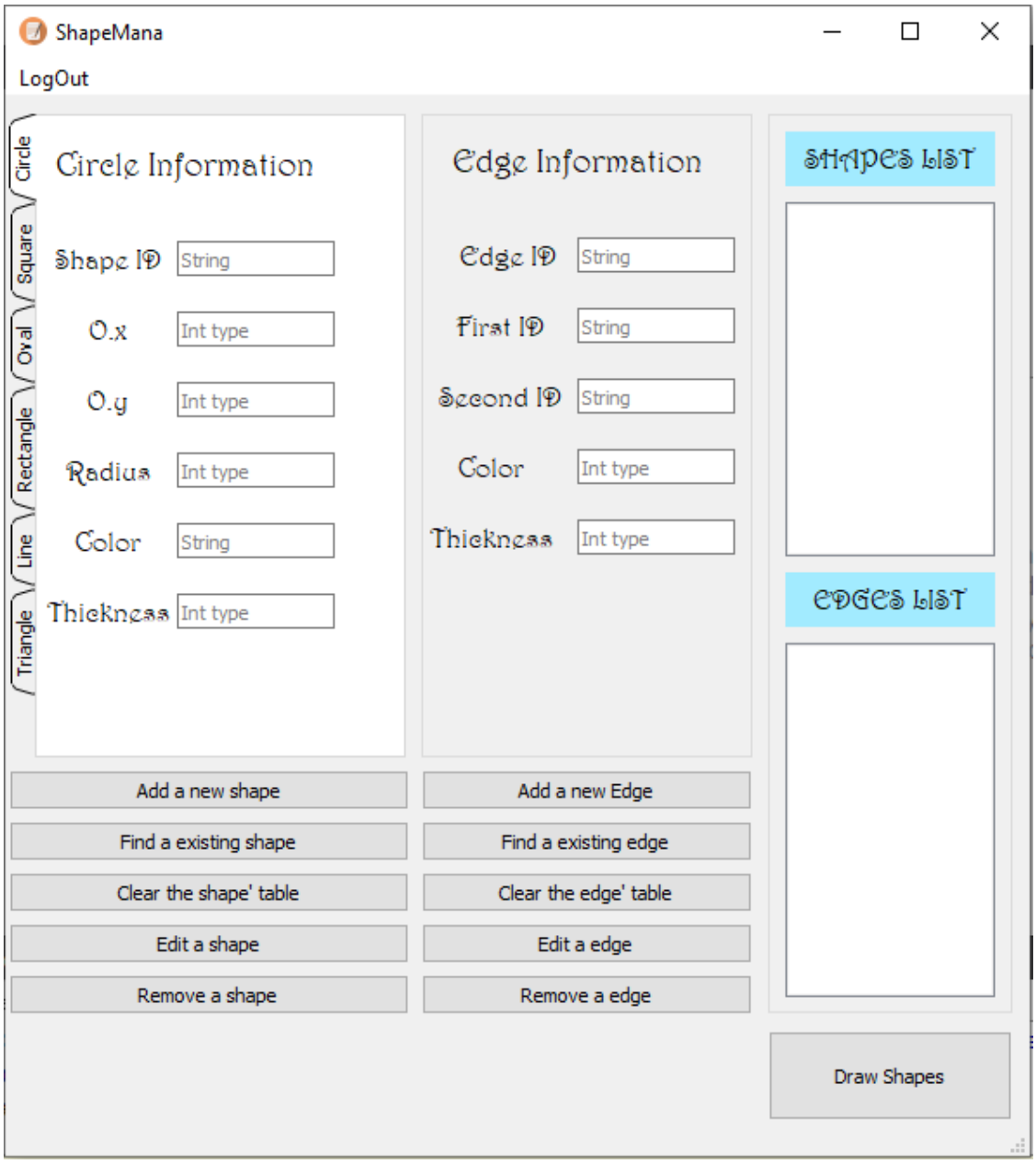
```
#include "controller.h"
#include "mainwindow.h"
#include "model.h"
#include <QMessageBox>

//using namespace Ui;
Controller::Controller(Model* m)
    :   model(m)
{};
Controller::~Controller()
{
    delete model;
};
void Controller::OnAddShapeButtonClicked(MainWindow *view)
{
    this->model->PushShapeList(view->CurrentNewShape);
};
//void Controller::OnReloadModel()
//{
//void (*ptr)() = model->GetList();
//};
```

**Hình 5.2: Class Controller**

*Hình 5.2* mô tả khối Controller của chương trình Shape, Controller chứa Model và View. Hàm khởi tạo Controller (Model *m), void OnaddShapeButtonClicked (MainWindow * view).

Khi nhận được một yêu cầu từ user lên View. Từ View sẽ gửi lại yêu cầu cho Controller để nó xử lý nếu cần truy xuất đến dữ liệu thì Controller sẽ gọi đến Model.

### 5.2.3 Tạo View cho chương trình




**Hình 5.3: Giao diện chính cho Users**

*Hình 5.3* mô tả khối View trong mô hình MVC, và để hiển thị dữ liệu trong mô hình chúng em sử dụng QlistWidgetItem và sử dụng hộp thoại thông báo QmessageBox để phản hồi lại với user.

- Line Edit để nhập các thuộc tính của Shape và Edge đồng thời cũng sử dụng Line Edit để hiển thị dữ liệu.

- Textlabel.

- TabWidget để cho người dùng chọn các Shape cần thao tác.

- Các slots:

+ AddShape, FindShape, ClearShape, EditShape, RemoveShape.

+ AddEdge, FindEdge, ClearEdge, EditEdge.

-QListWidgetItem để hiển thị danh sách các Shape và Edge.

=> Đây là phần người dùng sẽ thao tác và nhận dữ liệu.

***Kết luận chương:*** Vậy trong chương triển khai này, chúng ta đã đề cập đến những phần quan trọng của MVC. Tuy nhiên vẫn chưa đầy đủ, chỉ mang tính chất thể hiện một phần nhỏ, chi tiết sẽ được thể hiện trong mã nguồn của chương trình kèm theo báo cáo.

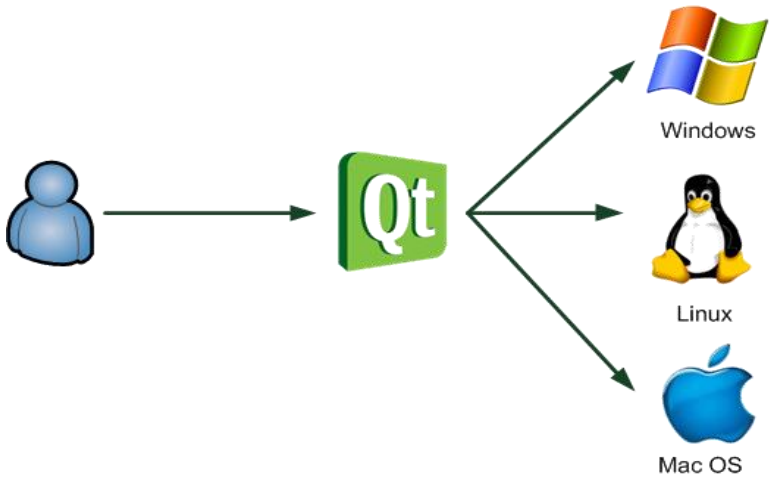
# CHƯƠNG 6: KẾT QUẢ THỰC NGHIỆM

***Mở đầu chương:*** Trong phần này chúng ta sẽ trình bầy về kết quả thực nghiệm sau khi hoàn thiện chương trình, với các bộ dữ liệu đầu vào sẽ sử dụng để kiểm thử và đầu ra của chương trình sẽ được kiểm chứng.

## 6.1 Môi trường kiểm tra thuật toán

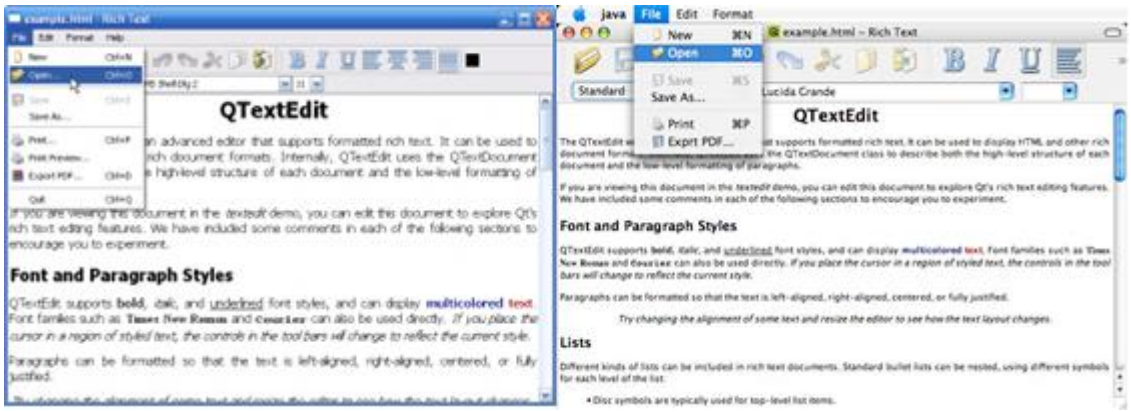
**Qt là một framework** đa nền tảng, được tích hợp trên **nhiều hệ điều** hành như:

- Windows
- Linux
- Mac Os




**Hình 6.1: Qt được tích hợp trên nhiều hệ điều hành**

Khi người lập trình viên viết bằng Qt và Qt được dịch các câu lệnh ra tùy theo hệ điều hành. Dựa trên cơ chế này, cửa sổ mà người lập trình viên tạo ra sẽ thay đổi phù hợp với từng hệ điều hành như trong dưới đây:

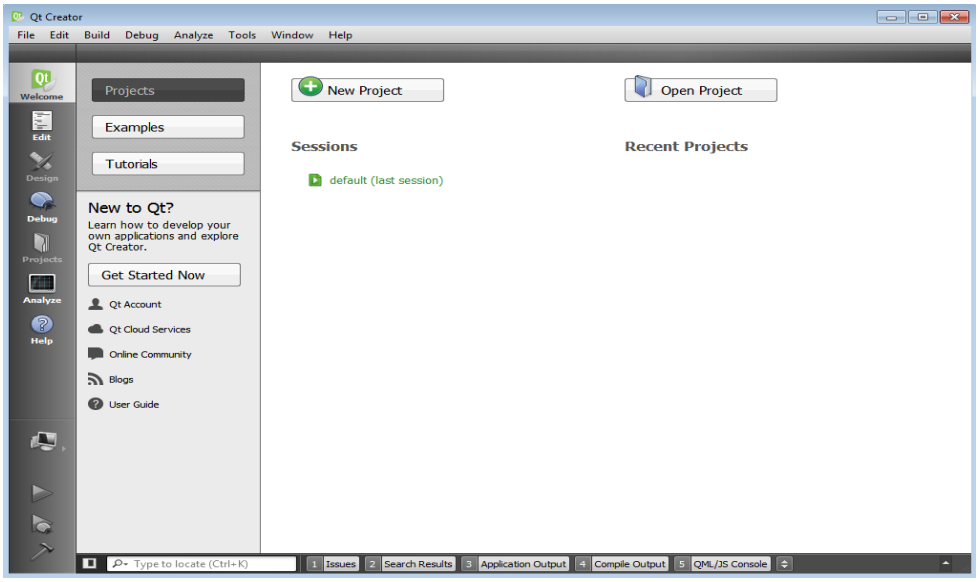



**Hình 6.2: Cửa sổ thay đổi tùy theo hệ điều hành**

Trong các dự án Qt người phát triển ứng dụng có thể sử dụng nhiều môi trường (IDE) để biên dịch và chạy thuật toán như: Qt Creator, Visual Studio for Qt5,…. Qua quá trình tìm hiểu thì chúng em nhận thấy **Qt Creator** tích hợp nhiều công cụ như:

- ❖ Chương trình viết mã lệnh (Code editor)
- ❖ Chương trình sửa lỗi (Debugger)
- ❖ Chương trình thiết kế giao diện bằng thao tác kéo thả

Qt Creator mang đến một giao diện rõ ràng, đơn giản, cho phép người dùng thiếp lập một số dự án mới bằng cách sử dụng phương pháp từng bước tiện dụng, đặc biệt đối với người chưa có kinh nghiệm vì họ sẽ được trợ giúp trong toàn bộ quá trình.



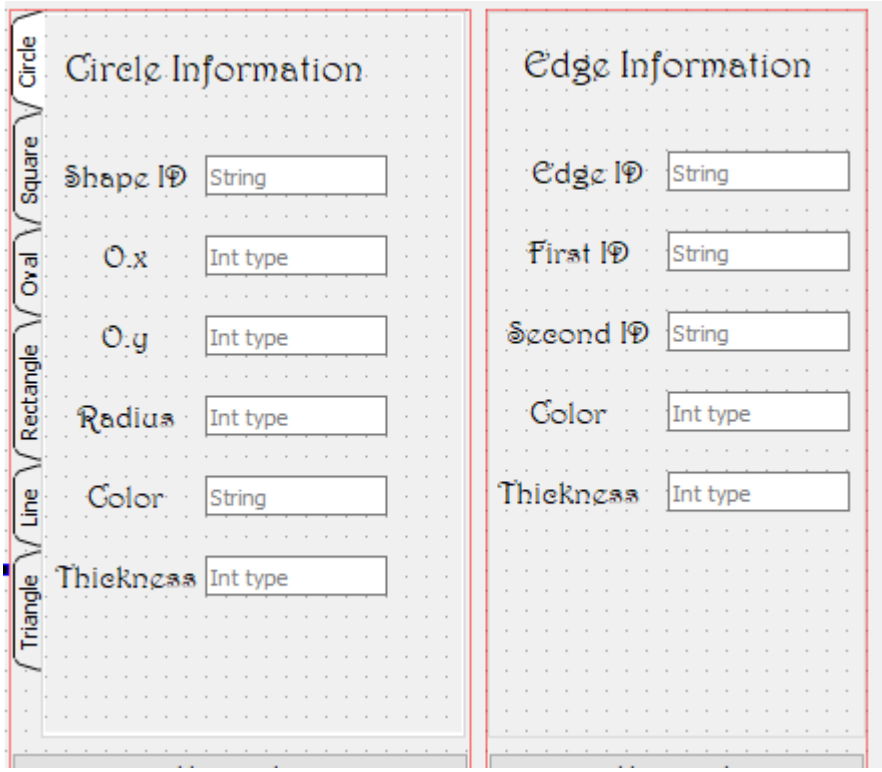

**Hình 6.3: Giao diện của Qt Creator**

Ngôn ngữ lập trình với Qt Creator: **C++.**

## 6.2 Bộ dữ liệu đầu vào

### 6.2.1 Dữ liệu được nhập từ bàn phím

Trên giao diện của mình chúng em có tạo giao diện để người dùng có thể nhập trực tiếp dữ liệu, thông số của một hình bất kỳ từ bàn phím mà họ muốn sử dụng, thực hiện các chức năng của chương trình.




**Hình 6.4: Giao diện nhập dữ liệu từ bàn phím**

Giao diện nhập dữ liệu từ bàn phím gồm ba phần chính:

- ❖ Chọn hình để nhập các thông số dữ liệu của hình
- ❖ Nhập thông số dữ liệu cho các hình
- ❖ Nhập thông số dữ liệu cho các cạnh nối tâm giữa hai hình

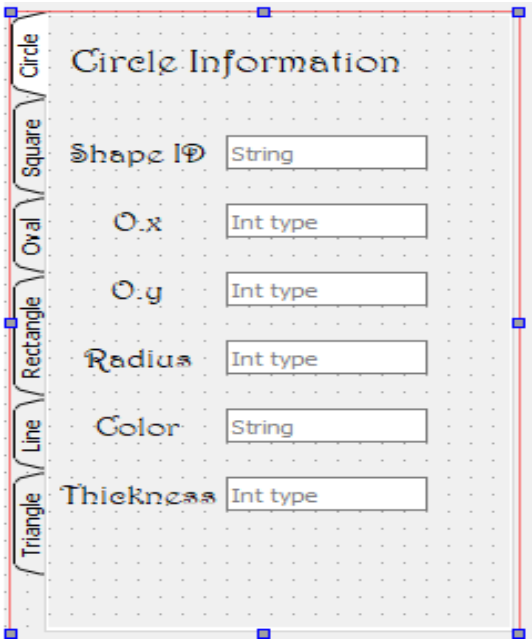
***Chọn hình để nhập dữ liệu***

Cho phép người dùng có thể click vào các tab: Circle, Square, Oval, Rectangle, Line, Triangle để có thể nhập dữ liệu hình tương thích.

***Thông số dữ liệu đầu vào cho shape***

Trong giao diện này các hình: Circle, Square, Oval, Rectangle, Line, Triangle chúng có các thuộc tính chung đó là:

- Tên hình: Shape ID
- Tọa độ tâm: O.x, O.y
- Màu sắc: Color
- Độ dày: Thickness




**Hình 6.5: Nhập dữ liệu cho Shape**

Những thuộc tính chung của các hình được thể hiện trong bảng dưới đây:

| Đối tượng | Thuộc tính chung | Kiểu dữ liệu |
|---|---|---|
| Shape | Shape ID | String |
| | O.x | Int |
| | O.y | Int |
| | Color | String |
| | Thickness | Int |

**Bảng 6.1: Thuộc tính chung của Shape**

Những thuộc tính cụ thể của từng hình: Circle, Square, Rectangle, Oval, Line, Triangle được thể hiện trong bảng dưới đây:

| Đối tượng | Thuộc tính riêng | Kiểu dữ liệu |
|---|---|---|
| Cirle | Radius | Int |
| Square | Edge | Int |
| Oval | Width | Int |
| | Heigth | Int |
| Rectangle | Width | Int |
| | Heigth | Int |
| Line | A.x, A.y | Int |
| | B.x, B.y | Int |
| Triangle | A.x, A.y | Int |
| | B.x, B.y | Int |
| | C.x, C.y | Int |

**Bảng 6.2: Thuộc tính riêng của từng Shape**

Khi người dùng nhập dữ liệu, mỗi hình sẽ có từng thuộc tính riêng biệt của nó nhưng vẫn có các thuộc tính chung của hình. Vd: Hình tròn (Circle) sẽ có thuộc tính riêng là Radius nhưng Hình vuông (Square) sẽ không có thuộc tính đó mà nó có thuộc

tính Edge. Thuộc tính riêng của từng hình sẽ quyết định hình đó sẽ là hình gì, như thế nào.

***Thông số dữ liệu đầu vào cho edge***

Người dùng còn có thể nhập dữ liệu đầu vào để có thể tạo cạnh nối tâm giữa hai hình


**Hình 6.6: Nhập dữ liệu đầu vào cửa Edge**

Các thuộc tính của cạnh nối tâm được thể hiện trong bảng dưới đây:

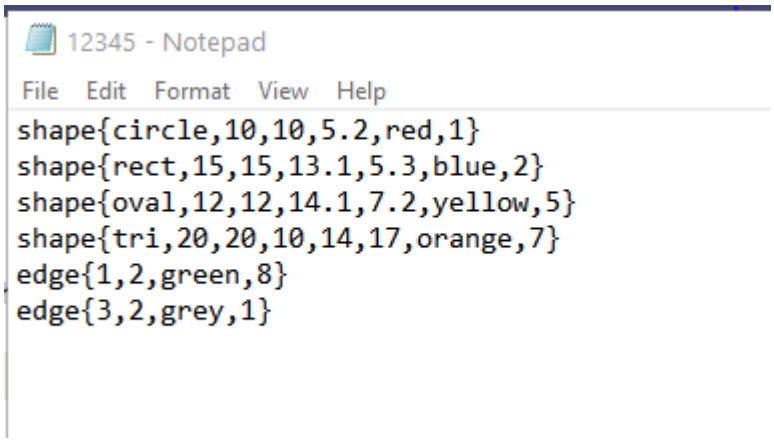
| Đối tượng | Thuộc tính | Kiểu dữ liệu |
|---|---|---|
| Edge | Edge ID | String |
| | First ID | Sting |
| | Second ID | String |
| | Color | Int |
| | Thickness | Int |

**Bảng 6.3: Thuộc tính của Edge**

Cạnh nối tâm chỉ tạo khi người dùng đã tạo hai hình trước đó.

### 6.2.2 Dữ liệu được nhập từ file text

Bộ dữ liệu đầu vào có thể được truyền vào từ file text dạng .txt




**Hình 6.7: Bộ dữ liệu từ file text**

Một file text có định dạng như sau:

- Mỗi dòng chứa các thông số của Shape hoặc Edge.
- Các thông số được đặt trong dấu ngoặc nhọn {}.
- Các thông số cách nhau bởi dấu phẩy (,).
- Thông số đầu tiên là loại shape n.
- Tùy thuộc vào thông số đầu tiên về loại shape các thông số sau tương ứng với vị trí, kích thước, mầu và độ dầy cạnh.
- Dòng chứa cạnh nối shape có 2 thông số đầu tiên là index của shape (bắt đầu từ 1).

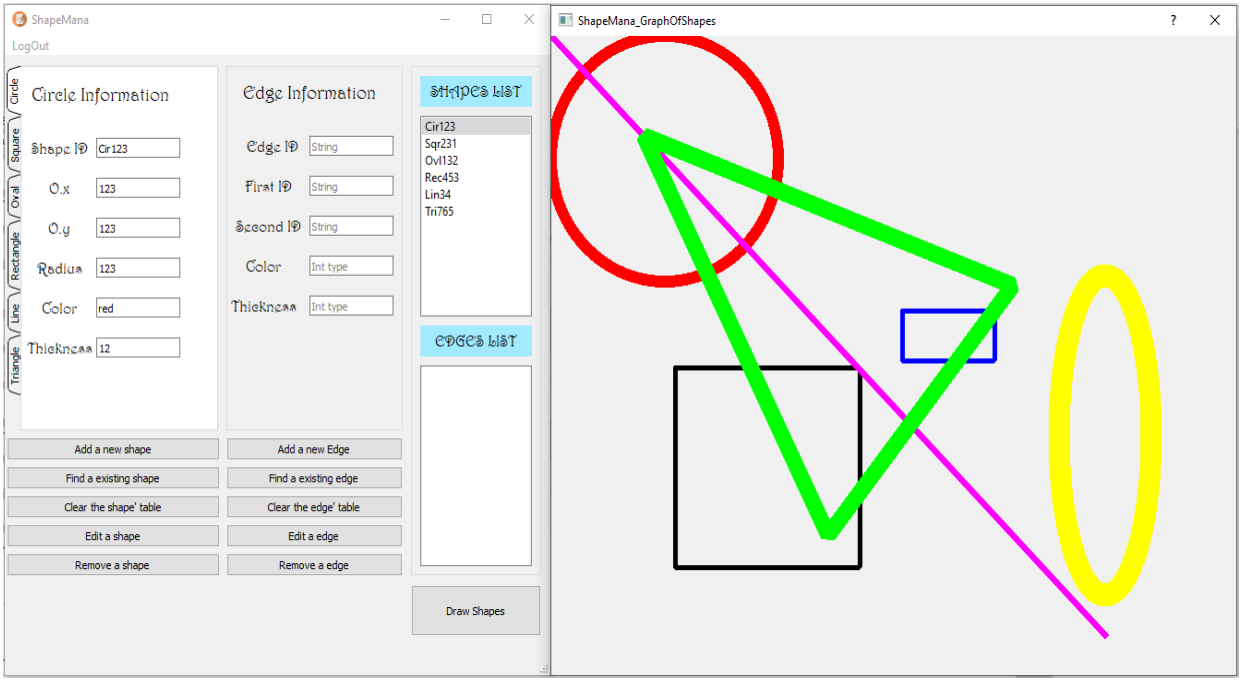
## 6.3 Kết quả

Ở phần này chúng ta sẽ sử dụng dữ liệu đầu vào là nhập từ bàn phím (giống như đã đề cập ở tiểu mục 6.2.1 ) với các dữ liệu và kết quả được trình bầy ở dưới đây

### 6.3.3 Bộ dữ liệu của Shape từ bàn phím

| Circle | | | | | | | |
|---|---|---|---|---|---|---|---|
| Shape ID | O.x | O.y | Radius | | Color | | Thickness |
| Cir123 | 123 | 123 | 123 | | red | | 12 |
| Square | | | | | | | |
| Shape ID | O.x | O.y | Edge | | Color | | Thickness |
| Sqr231 | 234 | 432 | 200 | | black | | 5 |
| Oval | | | | | | | |
| Shape ID | O.x | O.y | Width | Height | Color | | Thickness |
| Ovl132 | 600 | 400 | 100 | 320 | yellow | | 23 |
| Rectangle | | | | | | | |
| Shape ID | O.x | O.y | Width | Height | Color | | Thickness |
| Rec453 | 430 | 300 | 100 | 50 | blue | | 5 |
| Line | | | | | | | |
| Shape ID | A.x | A.y | B.x | B.y | Color | | Thickness |
| Lin34 | 0 | 0 | 600 | 600 | magenta | | 7 |
| Triangle | | | | | | | |
| Shape ID | A.x | A.y | B.x | B.y | C.x | C.y | Color | Thickness |
| Tri765 | 100 | 100 | 300 | 500 | 500 | 250 | green | 17 |

**Bảng 6.4: Bộ dữ liệu Shape nhập từ bàn phím**



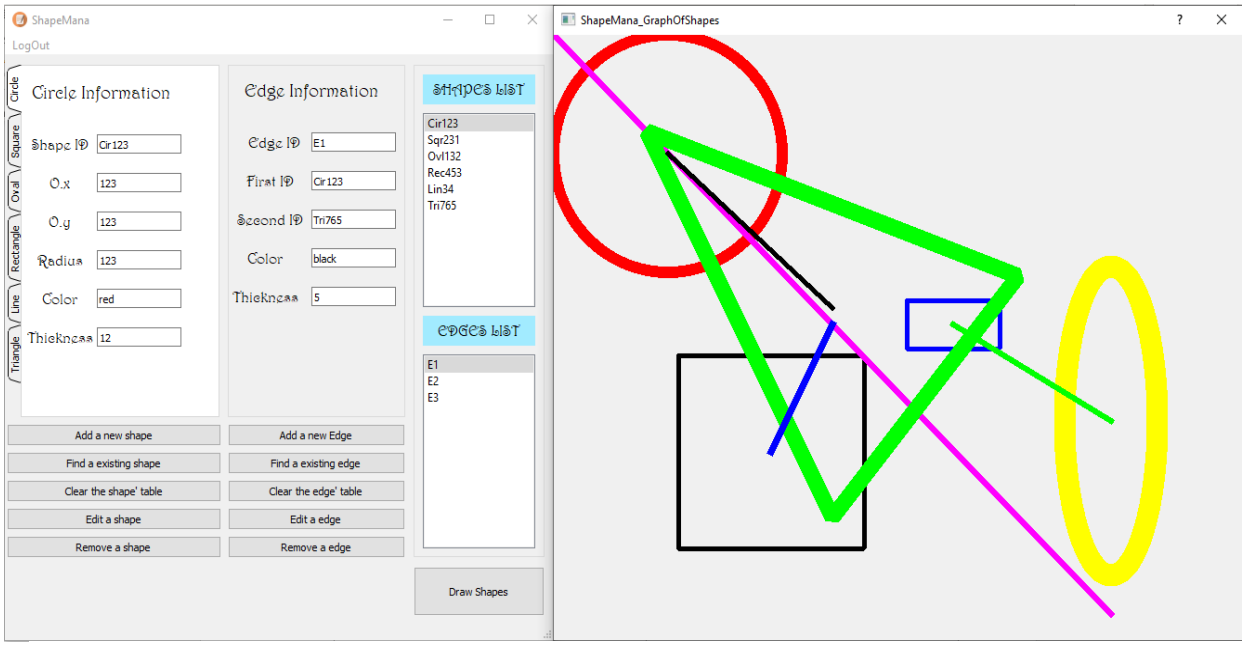

**Hình 6.8: Hình ảnh sau khi đã nhập dữ liệu các Shape và vẽ lên Dialog**

Hình 6.8 mô tả việc vẽ các đối tượng hình học lên màn hình (với bộ dữ liệu đầu vào như ở trên) với các thông số nhập vào từ bàn phím. Các đối tượng hình học hiện tại sẽ được hiện ID phía bên phải (trong phần SHAPES LIST) để người dùng tiện kiểm soát và quản lý (có thể thêm mới, sửa thông số, xóa hình,…).

### 6.3.4 Bộ dữ liệu của Edge từ bàn phím

| Edge | | | | |
|---|---|---|---|---|
| Edge ID | First ID | Second ID | Color | Thickness |
| E1 | Cir123 | Tri765 | black | 5 |
| E2 | Ovl132 | Rec453 | green | 6 |
| E3 | Lin34 | Sqr231 | blue | 7 |

**Bảng 6.5: Bộ dữ liệu Edge nhập từ bàn phím**




**Hình 6.9: Hình ảnh sau khi nhập thêm dữ liệu Edge và vẽ lên Dialog**

Hình 6.9 mô tả việc sau khi nhập thêm thông tin các Edge (đường nối tâm của hai hình với nhau) và vẽ chúng lên Dialog. Các đối tượng Edge hiện tại sẽ được hiện ID phía bên phải (trong phần EDGES LIST) để người dùng tiện kiểm soát và quản lý (có thể thêm mới, sửa thông số, xóa hình,…).

***Kết luận chương:*** Như vậy chúng ta đã trình bầy các cách kiểm thử chương trình dựa trên môi trường chạy, bộ dữ liệu đầu vào, đầu ra của chương trình,… Qua đó chúng ta khẳng định chương trình đã hoạt động tốt các chức năng cơ bản dựa trên mục tiêu ban đầu của dự án. Tuy nhiên một vài chức năng vẫn cần hoàn thiện thêm như: xóa, tìm kiếm.

# CHƯƠNG 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

***Mở đầu chương:*** Trong chương cuối này, chúng ta sẽ đưa ra kết luận cuối cùng cho dự án, những ưu điểm, nhược điểm của dự án, các vấn đề còn tồn đọng, bài học kinh nghiệm và hướng phát triển trong tương lai.

## 7.1 Tóm tắt kết quả đạt được và các vấn đề tồn tại

- ❖ Kết quả đạt được:
  - Đã thiết kế được giao diện thực hiện đúng và đầy đủ các tính năng mà thầy yêu cầu.
  - Các thành viên trong nhóm đã giúp đỡ nhau, đóng góp ý kiến để hoàn thành nội dung đề tài.
  - Qua môn đồ án I được sự dẫn dắt của thầy Nguyễn Đức Minh, mọi người đã hiểu sâu hơn, đầy đủ hơn về ngôn ngữ lập trình C++ và OOP, biết sử dụng các thuật toán trong C++. Từ đó vận dụng vào thiết kế giao diện bằng framwork Qt và đặc biệt sử dụng mô hình M-V-C trong quá trình thiết kế giao diện.
- ❖ Các vấn đề tồn tại:
  - Sản phẩm cần được thiết kế đẹp hơn, ngọn ngàng hơn, để người dùng có thể dễ dàng sử dụng các chức năng.
  - Chưa tối ưu hóa code.
  - Kỹ năng phân tích, giải quyết vấn đề còn thiếu.
  - Kỹ năng làm việc nhóm cần được cải thiện.

## 7.2 Bài học kinh nghiệm

- ❖ Qua đây chúng em rút ra được một số kinh nghiệm:
  - Viết code chỉ cần sai lỗi nhỏ, thiếu một dấu phẩy thì nó cũng không chạy.
  - Cần comment code để mọi người trong team cũng như người xem có thể đọc hiểu được.
  - Cần cải thiện một số kỹ năng như: làm việc nhóm hiệu quả, kỹ năng tìm kiếm tài liệu, cần học tiếng anh thật tốt.

## 7.3 Hướng phát triển

- Mở rộng thêm các chức năng của giao diện để người dùng có thêm sự lựa chọn.
- Thiết kế giao diện đăng nhập để tăng tính bảo mật cho ứng dụng.

***Kết luận chương:*** Vậy chúng ta đã đưa ra được các nhận xét cho chương trình, bài học kinh nghiệm và hướng phát triển trong tương lai. Qua dự án này chúng em rất cám ơn Thầy Nguyễn Đức Minh đã tận tình chỉ dạy những kiến thức quý báu để hoàn thành dự án này, các thành viên trong nhóm đều nỗ lực, học hỏi được những kỹ năng cần thiết để thiết kế một chương trình phần mềm hoàn chỉnh. Dù đã cố gắng hoàn thiện đề tài tốt nhất, nhưng do thời gian eo hẹp và kiến thức còn hạn chế nên sản phẩm sẽ không tránh khỏi những thiếu sót, chúng em rất mong nhận được sự đóng góp ý kiến tận tình của thầy và quý bạn!

Chúng em xin chân thành cám ơn!

# TÀI LIỆN THAM KHẢO

[1] Qt Company, "Qt_for_Beginners," Qt Company, 12 4 2020. [Online]. Available: https://wiki.qt.io/Qt_for_Beginners.

[2] Wikipedia, "MVC," Wikipedia, 16 6 2020. [Online]. Available: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller.

[3] Geeksforgeeks, "Geeksforgeeks," 12 6 2020. [Online]. Available: https://www.geeksforgeeks.org/design-patterns-set-2-factory-method/?ref=lbp.

[4] cplusplus, "cplusplus," cplusplus, 18 06 2020. [Online]. Available: https://www.cplusplus.com/reference/fstream/fstream/?kw=fstream.

# PHỤ LỤC

**Phụ Lục 1**: Đường dẫn UML của hệ thống

https://drive.google.com/file/d/1lroABb79g604S7jzkUDns9SUbA_Q60IB/view?fbclid=IwAR3G25QRHumLGczlySaZBNPiIgCa0y2CLXDZkhR4EBxn2xE9cC2u1mMIoXI