**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN ĐIỆN TỬ - VIỄN THÔNG  
\*\*\*\*\* 🙠🕮🙢 \*\*\*\*\***



**BÁO CÁO ĐỒ ÁN I** *ĐỀ TÀI:*  
**TÌM HIỂU VỀ FACTORY PATTERN VÀ FSM PATTERN**

***GVHD: Nguyễn Đức Minh***

***DANH SÁCH SINH VIÊN:***

***1. Đào Đức Dũng ĐTVT08-K62 20172492***

***2. Nguyễn Văn Đạt ĐTVT08-K62 20172458***

***3. Vũ Đức Thái ĐTVT08-K62 20172804* Hà Nội, 05/2020**

**Lời nói đầu**

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… Xuất phát từ thực tiễn này em sẽ khái quát lại một số những hiểu biết về các kiến thức liên quan đến lập trình hướng đối tượng. Em xin chân thành cảm ơn thầy đã tạo điều kiện và cung cấp những tài liệu cần thiết trong quá trình hoàn thành bài báo cáo này. Bài tiểu luận không tránh khỏi những thiếu sót. Em mong nhận được ý kiến đóng góp của thầy để bài báo cáo được hoàn thiện hơn nữa.

**MỤC LỤC**

[Phần 1: Tìm hiểu Factory pattern 4](#_Toc39312432)

[*1.1 Mục đích của Factory pattern* 4](#_Toc39312433)

[*1.2 Cách sử dụng Factory pattern* 4](#_Toc39312434)

[*1.3 Sơ đồ mô tả cấu trúc UML của Factory pattern* 4](#_Toc39312435)

[*1.4 Code chương trình và giải thích* 5](#_Toc39312436)

[*1.5 Sơ đồ UML của class Graph* 6](#_Toc39312437)

[Phần 2: Tìm hiểu về FSM pattern 7](#_Toc39312438)

[*2.1 Khái niệm về FSM* 7](#_Toc39312439)

[*2.2 Ứng dụng của FSM trong lập trình* 7](#_Toc39312440)

[*2.3 Code chương trình và giải thích* 7](#_Toc39312441)

[Kết luận 15](#_Toc39312443)

# 

# Phần 1: Tìm hiểu Factory pattern

## *1.1 Mục đích của Factory pattern*

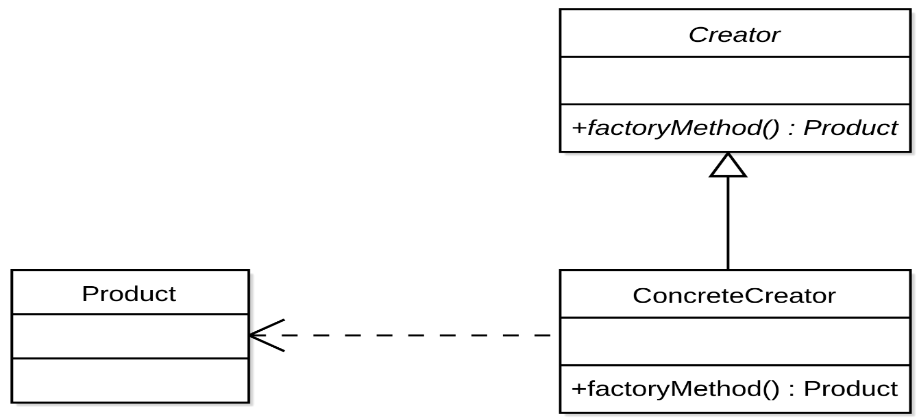
Factory Method pattern là thiết kế mẫu hướng đối tượng trong việc thiết kế phần mềm cho máy tính, nhằm giải quyết vấn đề tạo một đối tượng mà không cần chỉ ra một cách chính xác lớp nào sẽ được tạo. Factory method giải quyết vấn đề này bằng cách định nghĩa một phương thức cho việc tạo đối tượng, và các lớp con thừa kế có thể override để chỉ rõ đối tượng nào sẽ được tạo. Method factory thường được áp dụng cho những phương thức mà nhiệm vụ chính của nó là tạo ra đối tượng.

*- Mục đích của Factory method:* Định nghĩa interface để sinh ra đối tượng nhưng cho lớp con quyết định lớp nào được sinh ra đối tượng

## *1.2 Cách sử dụng Factory pattern*

Factory pattern này giúp thay thế việc khởi tạo đối tượng một cách trực tiếp (sử dụng toán tử new) bằng việc goi đến một factory method. Về thực chất, toán tử new vẫn được sử dụng để tạo ra đối tượng mới, nhưng việc thực hiện lời gọi này là do phương thức kia đảm nhận. Các đối tượng mới được tạo ra nhờ factory method còn được gọi là **“**product**”**. Từ đó chúng ta có thể override factory method này để chỉ rõ đối tượng nào sẽ được khởi tạo.

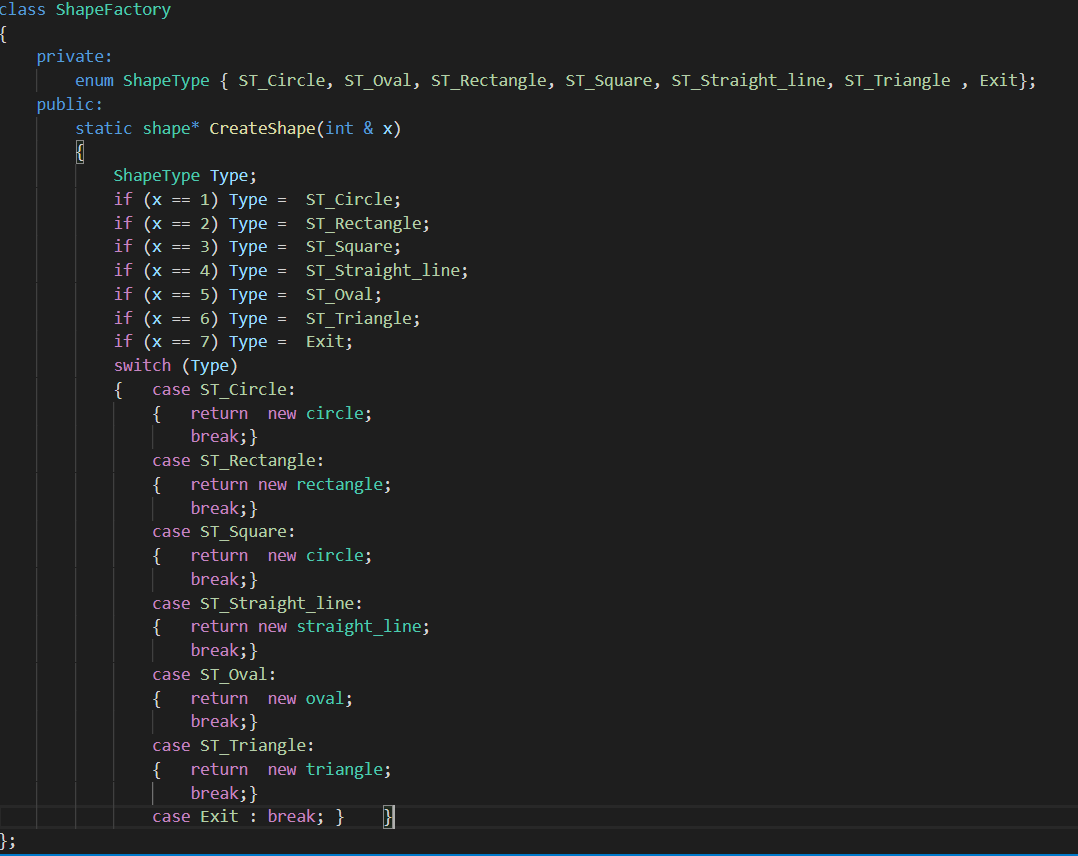
## *1.3 Sơ đồ mô tả cấu trúc UML của Factory pattern*



*Mô tả Factory pattern bằng UML*

## *1.4 Code chương trình và giải thích*

*\* Code chương trình:*



*\* Giải thích chương trình:*

* Trong private sẽ liệt kê các kiểu của Shape do người lập trình tự định nghĩa, bao gồm ST\_Circle (hình tròn), ST\_Oval (hình bầu dục), ST\_Rectangle (hình chữ nhật), ST\_ Square (hình vuông), ST\_Straight\_line (đường thẳng), và ST\_Triangle (hình tam giác).
* Trong public sẽ khai báo một hàm static CreateShape với đầu vào là một con trỏ kiểu nguyên x, x nằm trong khoảng từ 1 đến 6 tương ứng với các hình mà người sử dụng muốn tạo. Khi x từ 1 đến 6 thì chương trình sẽ trả về một đối tượng hình mới tương ứng, ví dụ như khi x bằng 1 thì chương trình sẽ trả về một đối tượng circle mới.

## *1.5 Sơ đồ UML của class Graph*

Do sơ đồ UML của class Graph có kích thước lớn nên nhóm sẽ để đường link dẫn đến file driver:

<https://drive.google.com/file/d/1-CTt4IXH0_hsI9wBn1iIZuw5Fqo_M40F/view?fbclid=IwAR19cpIzOzom8E9RK9SKl_-XHKyJnW3wtPmj01CyLgXoIEmNHrcYLE_qMNc>

# Phần 2: Tìm hiểu về FSM pattern

## *2.1 Khái niệm về FSM*

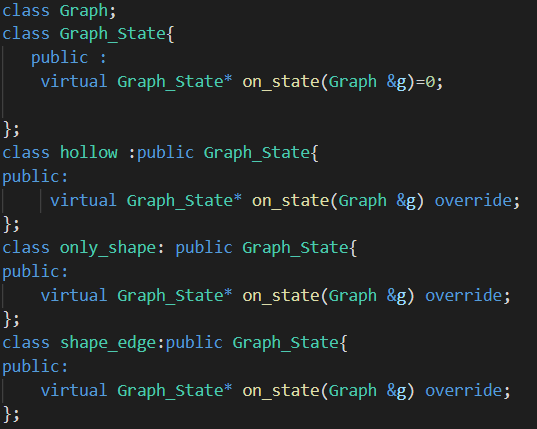
FSM (Finite State Machine) – máy trạng thái hữu hạn là một mô hình toán học biểu diễn trạng thái của hệ, trong đó số trạng thái là hữu hạn. Từ mỗi trạng thái, máy có thể chuyển đổi qua 1 số trạng thái cố định khác, dựa trên sự kiện input.

## *2.2 Ứng dụng của FSM trong lập trình*

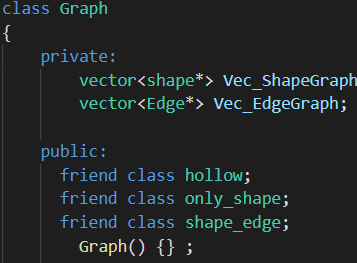
* FSM mô tả các trạng thái, sự kiện và quá trình chuyển đổi giữa các trạng thái, nên FSM có thể sử dụng để quản lý trạng thái của object, hoặc workflow.
* Chỉ có thể có một trạng thái tại một thời điểm, và từ 1 trạng thái chỉ có thể chuyển đổi qua một số trạng thái được quy định trước.

## *2.3 Code chương trình và giải thích*

Đầu tiên là khởi tạo các class bao gồm class Graph, Graph\_State, hollow, only\_shape, shape\_edge:



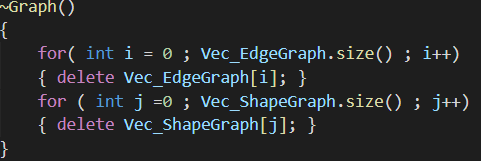
# Class Graph\_State để chỉ trạng thái của Graph với 1 hàm ảo on\_state. Class hollow được kế thừa từ class Graph\_State và để chỉ trạng thái rỗng của Graph. Sau đó là class only\_shape được kế thừa từ class Graph\_State và dùng để chỉ trạng thái có duy nhất 1 shape tron Graph. Và trạng thái cuối cùng là shape\_edge được kế thừa từ class Graph\_State và để chỉ trạng thái có cả shape và edge trong Graph.

 Sau đó là định nghĩa class Graph:

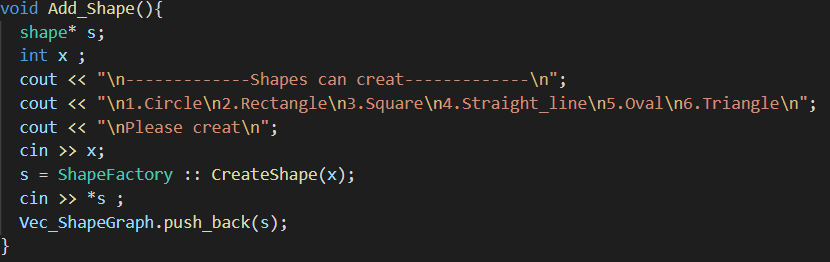
Class Graph sẽ bao gồm 2 vector Vec\_ShapeGraph và Vec\_EdgeGraph, đây đồng thời cũng là hai biến trạng thái của Graph. Ngoài ra class Graph còn là bạn của class hollow, only\_shape và shape\_edge.

Tiếp theo là định nghĩa các hàm trong class Graph:

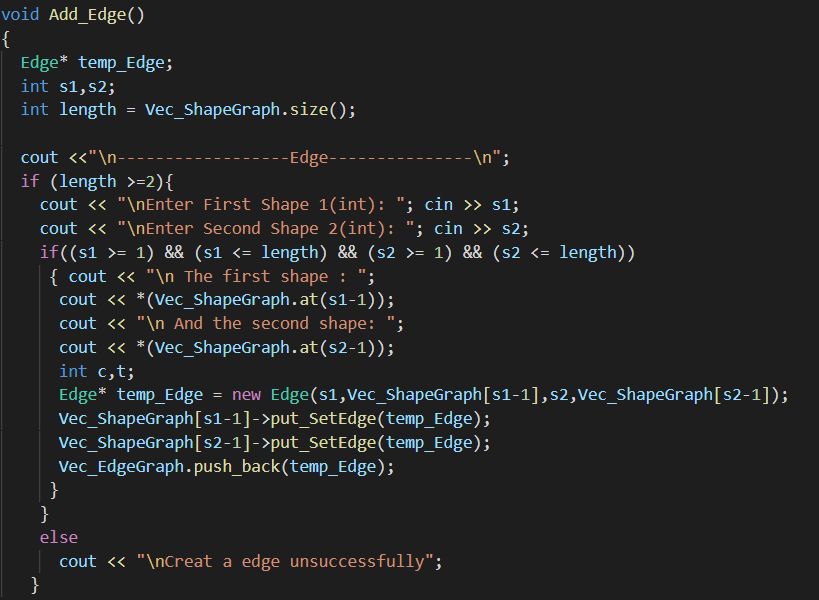
* Hàm hủy sẽ sử dụng vòng lặp đến size của 2 vector để xóa các phần tử



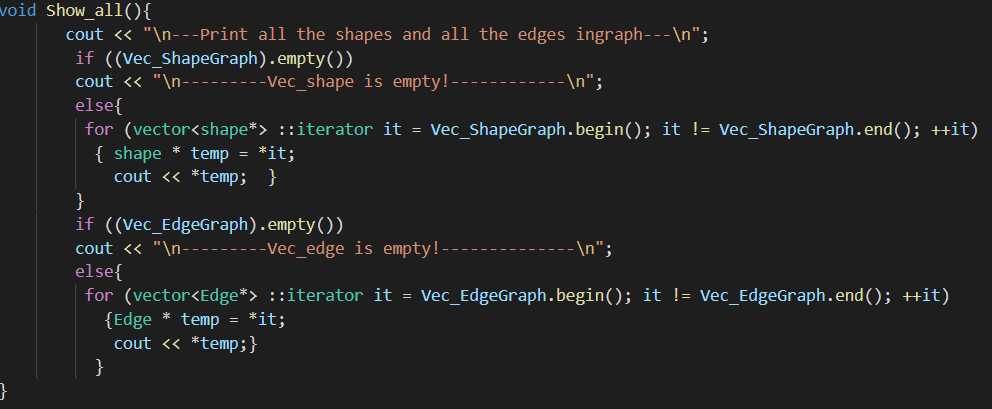
* Hàm thêm một shape vào trong Graph, hàm này sử dụng hàm khởi tạo của ShapeFactory, và người dùng sẽ có 6 lựa chọn tương ứng với 6 hình khác nhau:



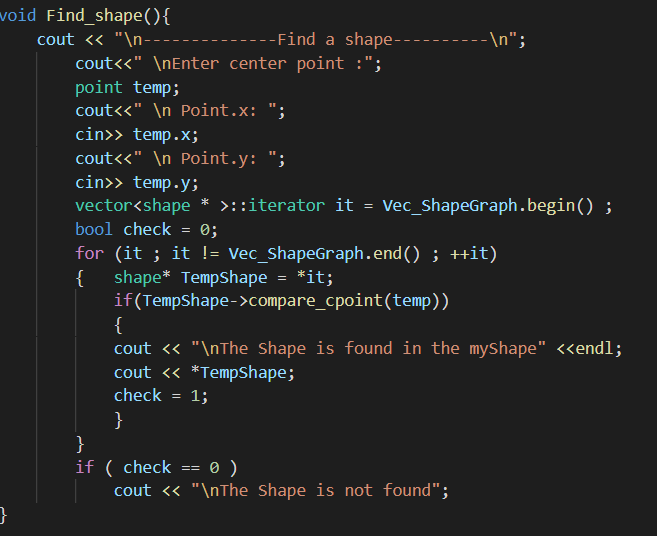
* Hàm thêm một edge vào trong Graph, hàm này yêu cầu số lượng shape trong Graph phải lớn hơn 2 nếu không sẽ báo lỗi, nếu số lượng shape lớn hơn 2 thì sẽ yêu cầu người dùng nhập chỉ số của 2 shape muốn tính:



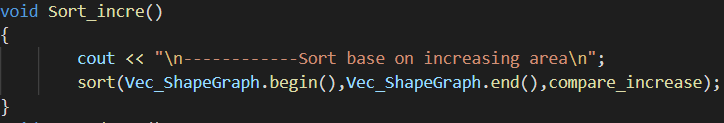
* Hàm hiển thị Graph, hàm này sẽ sử dụng vòng lặp chạy từ phần tử đầu đến phần tử cuối của 2 vector biến trạng thái và in ra màn hình dữ liệu của các shape và edge có trong Graph:



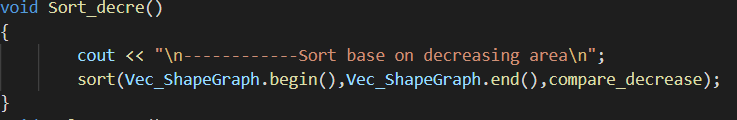
* Hàm tìm kiếm, yêu cầu người dùng nhập vào tọa độ hình muốn tìm kiếm, nếu tìm được sẽ trả về dữ liệu của hình đó còn không thì báo lỗi:



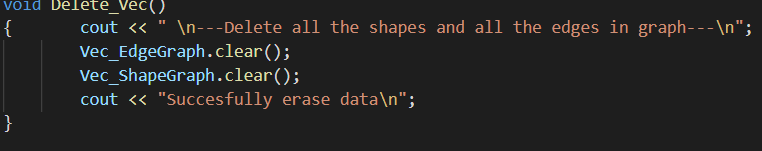
* Hàm sắp xếp theo chiều tăng dần của diện tích shape:



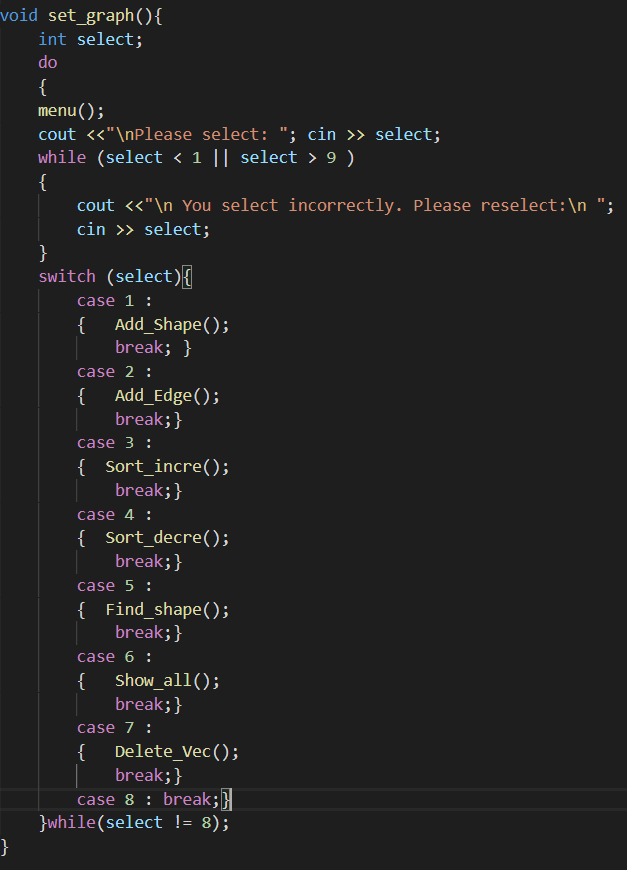
* Hàm sắp xếp theo chiều giảm dần của diện tích shape:



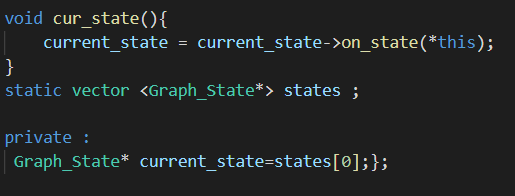
* Hàm xóa dữ liệu có trong Graph, toàn bộ shape và edge có trong Graph sẽ bị khóa khi người dùng sử dụng hàm này:



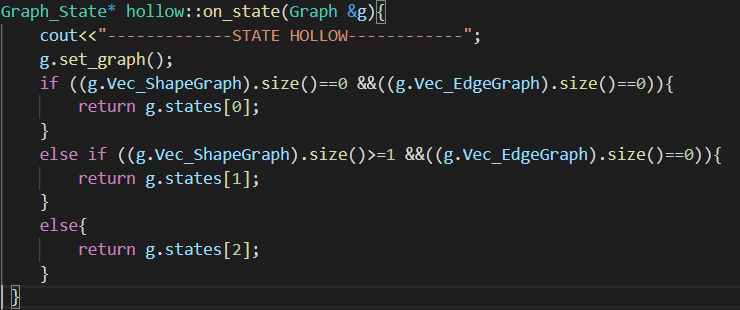
* Hàm thiết lập cho Graph



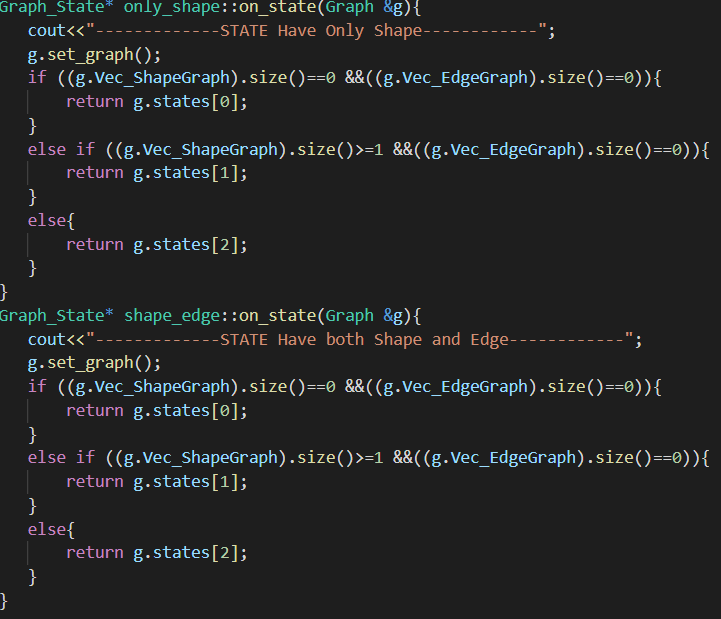
Sau đó sẽ thiết lập cho trạng thái ban đầu của Graph, sử dụng một vector có kiểu là con trỏ Graph\_State, và trạng thái hiện tại ban đầu được đặt là trạng thái state[0], tương ứng với trạng thái hollow (rỗng) của Graph:



Tiếp theo các trạng thái của Graph sẽ được định nghĩa, đầu tiên là trạng thái rỗng (hollow) của Graph, sau khi thiết lập cho Graph thì chương trình sẽ kiểm tra size của 2 vector biến trạng thái, nếu cả 2 vector đều bằng không thì sẽ trả về trạng thái rỗng (hollow), nếu vector Vec\_ShapeGraph có size lớn hơn 1 và vector Vec\_EdgeGraph có size bằng 0 thì trả về trạng thái only\_shape, và các trường hợp còn lại trả về trạng thái edge\_shape:



Tương tự như trạng thái rỗng (hollow), các trạng thái còn lại cũng có cấu trúc giống như trên:



# Kết luận

Lập trình hướng đối tượng được xây dựng dựa trên nền tảng của khái niệm lập trình có cấu trúc và sự trừu tượng hóa dữ liệu. Do đó, hai kĩ thuật lập trình Factory pattern và Finite State Machine pattern là hai kĩ thuật rất quan trọng trong lập trình hướng đối tượng. Đồng thời các kỹ thuật này sẽ làm nền tảng cho lập trình các hệ thống nhúng (embedded system), một kiến thức rất quan trọng với sinh viên ngành điện tử - viễn thông. Qua đó, nhóm em xin gửi lời cảm ơn chân thành đến thầy Nguyễn Đức Minh đã tạo điều kiện và cung cấp các tài liệu cần thiết cho nhóm trong quá trình học tập và nghiên cứu.