**Một mã nguồn có phong cách lập trình tốt luôn được đánh giá cao hơn một mã nguồn không tuân theo phong cách lập trình.**

Mặc dù phong cách lập trình là không hoàn toàn bắt buộc tuy nhiên có thể coi nó có một sự liên hệ chặt chẽ và tất yếu đối với công việc lập trình. Vì lẽ đó mà các trường khi dạy lập trình cũng có nhắc đến điều này trong quá trình dạy, tuy nhiên thường rất hạn chế.

Có những tiêu chuẩn về phong cách lập trình riêng của từng ngôn ngữ, đôi khi sự tương đồng khá rõ nét vì không có sự khác biệt đáng kể. Phong cách lập trình ngoài mục đích giúp mã nguồn thêm trong sáng, rõ ràng còn để giúp phát huy tính chất làm việc tập thể dựa vào sự nhất quán theo một khuôn mẫu được quy ước. Vì thế không cần phải gò bó trong một lối viết nào đó, mỗi tập thể có thể linh hoạt tự quy ước cho mình một phong cách lập trình trên một mức độ nào đó.

Việc áp dụng phong cách lập trình còn thể hiện trình độ của lập trình viên, một mã nguồn có phong cách lập trình tốt luôn được đánh giá cao hơn một mã nguồn không tuân theo phong cách lập trình, cho dù trình độ giữa hai người viết là tương đương nhau. Vì thế các lập trình viên có ít nhiều kinh nghiệm đều có một phong cách lập trình khá hoàn thiện.

Phong cách lập trình còn phụ thuộc vào IDE (Integrated Development Environment) mà lập trình viên sử dụng. Các IDE cung cấp sẵn phong cách định dạng mã nguồn cho ngôn ngữ mà chúng hỗ trợ một cách tự động. Chẳng hạn như cách đóng mở ngoặc cặp ngoặc “{}”, cách thêm khoảng trắng giữa các biến, toán tử…

Mặc dù có thể linh hoạt tự đặt ra quy ước cho phong cách lập trình cho mỗi ngôn ngữ, trong mỗi tập thể, tuy nhiên xu hướng đó nên được hạn chế với mức tối thiểu có thể. Phần lớn các chuẩn mực về phong cách lập trình thường được nhà phát triển ngôn ngữ lập trình định rõ và bạn được khuyến cáo nên tuân theo đầy đủ, cơ bản như cách đặt tên lớp, đặt tên phương thức, các từ khóa. Bạn có thể thấy rằng các từ khóa trong VB.NET đa số là dạng viết hoa (capitalize) chữ cái đầu, trong khi C# là viết thường, cách đặt tên phương thức trong Java so với VB.NET và C# cũng khác nhau.

Tuy nhiên một số tiêu chuẩn này có thể bị hướng dẫn sai lạc bởi cách quy định đặt tên trong một số trường học. Ví dụ khi tạo một lớp Form mới, bạn được dạy là phải sử dụng tiền tố frm để xác định đó là một Form. Điều này có vẻ hợp lý nhưng nếu bạn phân biệt được giữa đối tượng và lớp thì bạn sẽ không đặt tên dạng này. Thứ nhất tên lớp thường không viết tắt, việc đặt tên lớp theo kiểu Hungary cũng có thể không cần thiết. Bạn hãy xem các lớp tạo sẵn như Button, Label, ngay cả các lớp có tên khá dài như ApplicationException, ConsoleCancelEventArgs… cũng không sử dụng viết tắt (mặc dù cũng có ngoại lệ như Char, AppDomain… nhưng là số ít). Thứ hai khá đơn giản là tên lớp phải capitalize chữ cái đầu tiên. Một điểm khác nữa là bạn có thể nhầm lẫn giữa Frame và Form (cũng có cách đặt tiền tố là frm), tuy nhiên Frame được dùng trong Java chứ không phải .Net.

Phong cách lập trình thường không phân loại, tuy nhiên có thể được chia làm hai loại là về nội dung và hình thức. Về nội dung ví dụ như tên biến, hàm phải thể hiện đúng công dụng của nó; tên hàm, phương thức là động từ, tên lớp là danh từ,…Hai phần này thường được trình bày đan xen nhau và khó tách rời. Ở đây tôi sẽ trình bày thành nhiều phần với ngôn ngữ minh họa là C#, bạn cũng có thể tìm thấy sự tương đồng và áp dụng cho các loại ngôn ngữ lập trình khác, đặc biệt là các ngôn ngữ “họ hàng” của C# như Java. C++…

**1)  Tổ chức mã nguồn**

Một solution trong C# có thể bao gồm nhiều dự án (project), các project có cùng thư mục cha và mỗi project phải có một thư mục riêng. Trong cùng một dự án, bạn có thể tạo thêm một vài thư mục con tương ứng với mỗi namespace nếu cần thiết.

Một project gồm nhiều tập tin, thường mỗi tập tin là một lớp, tuy nhiên trong C# một lớp có thể bao gồm nhiều tập tin như lớp Form. Mỗi tập tin bạn chỉ nên chứa một lớp và giữ giữ cho số dòng không nên quá dài, trường hợp các lớp hoặc cấu trúc nhỏ bạn có thể ghi chung vào một tập tin, giữa những lớp này nên có một sự tương đồng nào đó. Ví dụ bạn tạo một namespace Shape bao gồm các lớp hình vuông, tròn, tam giác thì các lớp này có thể đặt trong cùng một tập tin.

Các lớp mà bạn ra phải có chức năng rõ ràng, không nên phân chia ra các phương thức tương tự ra nhiều lớp khác nhau. Phạm vi của các phương thức và thuộc tính cần xác định chính xác là public, internal hay private. Theo nguyên tắc OOP, lớp mã nguồn của bạn càng có tính độc lập càng tốt, hạn chế mọi sự truy xuất các biến toàn cục có thể bằng cách viết dưới dạng phương thức hoặc Properties, nếu không việc kiểm soát giá trị biến sẽ rất khó khăn, khó sửa lỗi và làm mất đi tính độc lập của chương trình.

**2) Chú thích (Comment)**

Khác với truyền thống, lập trình viên thường sử dụng cặp /\* \*/ để chú thích cho một đoạn code dài có nhiều dòng. Kiểu chú thích này được sử dụng trong Eclipse, nó có thể rất tiện lợi nhưng khi muốn bỏ comment một số dòng ở giữa bạn phải gõ lại cặp đóng mở tại vị trí đó. Hơn nữa chú thích dạng này không phân biệt rõ ràng được đoạn comment với những đoạn khác. Vì thể sau này, các lập trình viên sửa đổi lại ách chú thích này như sau:

*/\*\*  
\* Line 1  
\* Line 2  
\*/*

Cách chú thích này tương đối rõ ràng hơn cách cũ, việc uncomment cũng đỡ tốn thời gian hơn. Tuy nhiên với C#, trong hầu hết trường hợp, bạn nên sử dụng // để chú thích. IDE Visual C# cũng hỗ trợ chức năng Comment Selection (hoặc Comment Region trong SharpDevelop) để chú thích mỗi dòng bằng cách thêm // phía trước. Với cách này, bạn có thể tùy ý bỏ hoặc comment lại một số dòng nào đó**. Cũng cần chú ý, bạn không nên chú thích sát ngay sau các kí tự /, dành cho chúng một khoảng trắng trước khi bắt đầu dòng chú thích của bạn**. Ví dụ thay vì viết:

*//Đây là phương thức A*

Thì hãy sửa lại như sau:

*// Đây là phương thức A*

Ngoài ra bạn cũng nên tuân theo các quy tắc sau:

–          Việc thụt đầu dòng của đoạn chú thích phải tương ứng với phần code mà nó chú thích bên dưới.

–          Các dòng chú thích không nên quá dài, vượt quá phạm vi hiển thị của trình soạn thảo, hãy xuống dòng sau các dấu câu.

–          Không nên chú thích quá dài dòng hoặc những đoạn không cần thiết, bản thân mã nguồn đã tự nói về công dụng của nó.

–          Không nên thêm dòng trống khi không cần thiết, chỉ nên có quá một dòng trống phân cách giữa hai phần.

Chúng ta sẽ đi vào chi tiết trong từng phần sau.

**a) Chú thích đầu tài liệu:**

Các tập tin mã nguồn thường được chú thích phía trên để giới thiệu thông tin cơ bản của chương trình. Bởi vì mỗi lớp có thể là một module riêng biệt nên việc giới thiệu thông tin chương trình trong mỗi lớp là cần thiết. Điều này không bắt buộc nhưng bạn nên bổ sung chúng khi dự án của mình tương đối phức tạp. Có một vài dạng được sử dụng, tuy nhiên thông dụng nhất là cách viết sau:

Mỗi phần thông tin và mỗi dòng được đánh thứ tự nên được cách nhau bởi một dòng trắng. Một số lập trình viên còn chú thích phiên bản, IDE sử dụng, tên lập trình viên, ngày tháng sửa đổi…

**b) Chú thích đầu lớp:**

Việc chú thích đầu mỗi lớp rất cần thiết để mô tả về công dụng của lớp. Về điểm này, các IDE hiện nay đều hỗ trợ khá tốt, IDE sẽ tự động tạo ra đoạn ghi chú thích cho bạn bằng cách gõ một vài kí tự quy ước trước tên lớp, tên phương thức, tên thuộc tính,… Trong Visual C# hoặc Sharp Develop, bạn sử dụng /// để IDE tạo ra một đoạn chú thích tối thiểu dạng XML như sau:

Với lớp, bạn chỉ cần chú thích công dụng của lớp, hoặc chi tiết hơn, việc đánh số để liệt kê các phương thức và thuộc tính của lớp đôi khi cũng không quá dư thừa. Ví dụ:

**c) Chú thích đầu phương thức, thuộc tính:**

Cũng với cách chú thích cho lớp, IDE tự động sinh ra cho bạn đoạn ghi chú thích tùy thuộc vào phương thức của bạn có tham số hay không, kiểu trả về là gì. Giả sử bạn có một phương thức:

Dựa vào tên phương thức và tham số bạn cũng có thể đoán được chức năng của nó là gì. Tuy nhiên khi bắt đầu chú thích, IDE sẽ tạo ra cho bạn một đoạn chú thích như sau

Bạn có thể thấy đoạn chú thích khá dài này cung cấp đầy đủ phần chú thích cho phương thức lẫn tham số và kiểu trả về. Công việc của bạn là bổ sung vào những chỗ khuyết nếu cần thiết:

Với các Properties và biến toàn cục lớp, bạn cũng có thể dùng cách chú thích tương tự. Bạn cũng có thể tự thêm một số thông tin chú thích khác phía sau như exception, value…Ví dụ để chỉ ra phương thức của mình ném ra ngoại lệ FileNotFound, bạn có thể dòng sau:

**d) Chú thích dòng:**

Như đã nói trước, bạn sử dụng // để chú thích cho một hoặc nhiều dòng. Chỉ cần chọn các dòng cần chú thích và nhấn nút Comment Selection trên thanh công cụ hoặc dùng phím tắt Ctrl + E, C (nhấn C sau E). Bạn có thể thắc mắc vậy việc hỗ trợ cách chú thích /\* \*/ là không cần thiết nữa? Thực ra là nó vẫn có công dụng mà kiểu chú thích // không thể thay thế được. Đó là chú thích tạm thời một phần dòng lệnh, thường dùng khi đang debug chương trình. Ví dụ như bạn có một dòng lệnh với nhiều phép tính:

*int z = a + b – c \* d;*

Bạn muốn loại b ra khỏi câu lệnh thì có thể chú thích như sau:

*int z = a /\* + b \*/ – c \* d;*

Chú thích cho biến:

*int* value; *// Giá trị mục chọn*