**Class là gì?**

Class hay lớp là một mô tả trừu tượng (abstract) của nhóm các đối tượng (object) có cùng bản chất, ngược lại mỗi một đối tượng là một thể hiện cụ thể (instance) cho những mô tả trừu tượng đó. Một class trong C++ sẽ có các đặc điểm sau:

* Một class bao gồm các thành phần dữ liệu (thuộc tính hay property) và các phương thức (hàm thành phần hay method).
* Class thực chất là một kiểu dữ liệu do người lập trình định nghĩa.
* Trong C++, từ khóa class sẽ chỉ điểm bắt đầu của một class sẽ được cài đặt.

Ví dụ về một class đơn giản, class Car. Một chiếc xe hơi vậy thì sẽ có chung những đặc điểm là đều có vô lăng, có bánh xe nhiều hơn 3, có động cơ… Đó là một class, một cái model hay mẫu mà người ta đã quy định là nếu đúng như vậy thì nó là xe hơi. Nhưng mà xe thì có thể có nhiều hãng khác nhau, BMW, Vinfast, Toyota… Thì mỗi hãng xe lại có những model xe khác nhau nhưng chúng đều là xe hơi. Vậy thì trong lập trình cũng vậy, class là quy định ra một mẫu, một cái model mà các thể hiện của nó (instance) hay đối tượng (object) phải tuân theo.

**Khai báo class và sử dụng class**

Ví dụ một class cơ bản:

class Person

{

public:

    string firstName; // property

    string lastName;  // property

    int age;          // property

    void fullname()

    { // method

        cout << firstName << ' ' << lastName;

    }

};

Cú pháp tạo object của một class và sử dụng các thuộc tính và phương thức:

Person person;

person.firstName = "Le";

person.lastName = "Duc Anh";

person.fullname(); // sẽ in ra màn hình là "Le Duc Anh"

**Access modifiers & properties declaration**

Access modifier là phạm vi truy cập của các thuộc tính và phương thức sẽ được khai báo bên dưới nó. Có 3 phạm vi truy cập trong C++ là public, private và protected.

• Các thuộc tính và phương thức khai báo public thì có thể được truy cập trực

tiếp thông qua instance của class đó. Các thuộc tính nên khai báo là public

nếu bạn không có ràng buộc điều kiện trước khi gán (người dùng có thể

thoải mái gán giá trị) hoặc bạn không cần xử lý trước khi trả về giá trị thuộc

tính;

• Các thuộc tính private thường được sử dụng khi bạn không mong muốn

người khác có thể tùy ý gán giá trị hoặc là bạn muốn xử lý trước khi trả về

giá trị.

• Đối với protected, các phương thức và thuộc tính chỉ có thể truy cập qua

các class kế thừa nó hoặc chính nó.

Ví dụ của access modifier:

class MyClass

{

public:

    int public\_property;

private:

    int \_private\_property;

};

**Method declaration**

Phương thức cũng giống như một hàm bình thường.

Đối với phương thức thì có hai cách định nghĩa thi hành: định nghĩa thi hành trong

lúc định nghĩa class và định nghĩa thi hành bên ngoài class.

Định nghĩa thi hành bên trong class:

class Animal

{

public:

    string sound;

    void makeNoise()

    {

        cout << sound;

    }

};

Định nghĩa thi hành bên ngoài class:

class Animal

{

public:

    string sound;

    void makeNoise();

};

void Animal::makeNoise()

{

    cout << sound;

}

**Constructor**

Constructor hay hàm dựng là một hàm đặc biệt, nó sẽ được gọi ngay khi chúng ta

khởi tạo một object.

class Person

{

public:

    string firstName;

    string lastName;

    int age;

    Person(string \_firstName, string \_lastName, int \_age)

    {

        firstName = \_firstName;

        lastName = \_lastName;

        age = \_age;

    }

    void fullname()

    {

        cout << firstName << ' ' << lastName;

    }

};

**Static member**

Static member hay thành viên tĩnh trong class C++ cũng tương tự như với static variable (biến tĩnh) trong function. Đối với function, sau khi thực hiện xong khối lệnh và thoát thì biến tĩnh vẫn sẽ không mất đi. Đối với class, thành viên tĩnh sẽ là thuộc tính dùng chung cho tất cả các đối tượng của class đó, cho dù là không có đối tượng nào tồn tại. Tức là bạn có thể khai báo nhiều object, mỗi object các thuộc tính của nó đều khác nhau nhưng riêng static thì chỉ có một và static member tồn tại trong suốt chương trình cho dù có hay không có object nào của nó hay nói ngắn gọn là dùng chung một biến static.