LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

1. Object oriented programming

* Phương pháp lập trình quan tâm tới **dữ liệu** và **đối tượng** hơn là **hàm** và **logic**.
* Một đối tượng được định nghĩa bởi trường dữ liệu đặc trưng bởi thuộc tính và phương thức.

Các thuật ngữ liên quan

* Class : Lớp
* Atribute : Thuộc tính
* Method, Behavior : Phương thức
* Object : Đối tượng

Tính chất của OOP:

* Encapsulation: Đóng gói
  + Chỉ được truy cập bên trong Class.
  + Ra ngoài thì ko thể thao tác với các thuộc tính được, giấu data của Class.
  + Phạm vi của thuộc tính – Access modifier
    - Private : Thuộc tính
    - Public : Behavior
    - Protected : Liên quan tới kế thừa
* Hàm tạo Contructor: khởi tạo khi khai báo hàm cùng tên Class, không có giá trị trả về.
  + Khi khai báo hay thực thi, đều không cần khai báo giá trị trả về
  + Khi ko khởi tạo thì mặc định cũng đã có.
  + Có thể vừa khai báo Constructor không có giá trị truyền vào và có cùng tồn tại.
    - Khi khai báo biến thì tùy vào có giá trị truyền vào không mà class thực hiện Hàm constructor tương ứng.
    - Khi chỉ khai báo Constructor có giá trị truyền vào, thì khi khai báo biến định dạng ko có giá trị truyền vào thì sẽ báo lỗi, do hiện tại chỉ có 1 Constructor có giá trị truyền vào.
* Hàm hủy Destructor: Không thể hủy, ko có giá trị trả về, có tên trùng với Class nhưng có thêm dấu ~.
  + Được gọi khi đối tượng kết thúc.
  + Khi được khởi tạo trong 1 End Closing Scoup hoặc hàm con(hàm if, …) thì khi hết hàm sẽ hủy đối tượng.
* Con trỏ This: Trong hàm Constructor, có thể khai báo các tham số có tên trùng với tên thuộc tính của lớp. this ->
* Mảng: Có thể khai báo cho cả mảng, mỗi phần tử đều có kiểu dữ liệu Class.
* Hàm Setter, Getter: Để thay đổi các thuộc tính đang để Private.
* Có thể gán 2 biến cùng Class
  + Gán từng thuộc tính cho nhau.

1. Biến Static

VD: Tự động gán giá trị cho 999 Sinh viên: SV01 -> SV02 -> SV03

* Biến static không thể khởi tạo, chỉ có thể khai báo.
* Biến dùng chung cho toàn bộ các biến khai báo Class, chia sẻ cho nhau.
* Nên dùng trong Hàm Constructor để có thể gán tự động ID cho các đối tượng.
* Thêm từ khóa static khi khai báo biến:

Static int dem; // chỉ khai báo chứ ko khởi tạo

1. Hàm bạn, lớp bạn

* Các hàm trong Class có thể truy xuất tới từng thuộc tính trong Class.
* Có các hàm khác Class nhưng cx có thể truy xuất tới thuộc tính private trong Class khác -> Friend Function.
* Lưu ý: khi muốn thay đổi giá trị một biến có sẵn, nhớ truyền vào địa chỉ

Update(SinhVien&)

* Khai báo hàm chính trước, sau đó mới có thể dùng được các hàm từ hàm bạn.

1. Nạp chồng toán tử - Operator Overloading

* Toán tử nhập xuất << >>
  + Cout << : ostream.
  + Cint >> : istream.
* Xây dựng bằng hàm friend:

friend istream& operator >> (istream &in, SinhVien& a);

friend ostream& operator << (ostream &out, SinhVien a);

* Nhớ khi nào truyền vào tham trị và tham chiếu.
* Kiểu trả về của toán tử.
* Sử dụng giá trị tham số truyền vào đầu tiên thay cho kí tự.

istream& operator >> (istream &in, SinhVien& a){

cout << "Nhap id : ";

in >> a.id;

cout << "Nhap ten :";

in.ignore();

getline(in, a.ten);

cout << "Nhap ngay sinh : ";

in >> a.ns;

cout << "Nhap diem so : ";

in>>a.gpa;

return in;

* In thay cho cint (Cũng ko cần thiết lắm).
* Có thể nạp chồng toán tử bằng cách là member của class:
  + bool operator < (SinhVien a); // So sánh class hiện tại với 1 class khác
* Hoặc dùng hàm friend
  + friend bool operator < (SinhVien a, SinhVien b); // Cần 2 giá trị truyền vào.
* Khi đó hàm sort(a.begin(), a.end()), sử dụng toán tử < để sắp xếp, sẽ sắp xếp giá trị thông qua toán tử < đã nạp chồng mà ta đã định nghĩa.