**ÔN TẬP**

**1. Lập trình hướng đối tượng là gì?**

Lập trình hướng đối tượng (OOP - Object-Oriented Programming) là một phương pháp lập trình mà trong đó chương trình được tổ chức thành các đối tượng tương tác với nhau. OOP tập trung vào việc mô hình hóa các đối tượng trong thế giới thực và xử lý chúng dưới dạng các đối tượng trong chương trình.

Lợi ích:

- Khả năng tái sử dụng lại mã nguồn

- OOP mô hình hóa những thứ phức tạp dưới dạng cấu trúc đơn giản.

- Code OOP có thể sử dụng lại, giúp tiết kiệm tài nguyên.

- Giúp sửa lỗi dễ dàng hơn. So với việc tìm lỗi ở nhiều vị trí trong code thì tìm lỗi trong các lớp (được cấu trúc từ trước) đơn giản và ít mất thời gian hơn.

- Có tính bảo mật cao, bảo vệ thông tin thông qua đóng gói.

- Dễ mở rộng dự án.

**2. Các đặc điểm quan trọng của lập trình hướng đối tượng?**

Các đặc điểm quan trọng của lập trình hướng đối tượng bao gồm:

- **Tính trừu tượng (Abstraction):** Cho phép tạo ra các lớp trừu tượng mô tả tính chất chung của một nhóm đối tượng, và từ đó tạo ra các lớp cụ thể.

- **Tính đóng gói (Encapsulation):** Cho phép ẩn thông tin bên trong đối tượng và chỉ tiết lộ các phương thức công cộng để tương tác với đối tượng.

- **Tính kế thừa (Inheritance):** Cho phép tái sử dụng lại mã nguồn và xây dựng các lớp dẫn xuất từ các lớp hiện có.

- **Tính đa hình (Polymorphism):** Cho phép một phương thức có thể có nhiều hình thức thực hiện khác nhau tùy thuộc vào đối tượng gọi.

**3. Lớp đối tượng là gì? Đối tượng là gì?**

- Lớp đối tượng là một cấu trúc dữ liệu trong lập trình hướng đối tượng, mô tả các thuộc tính (biến thành viên) và hành vi (phương thức) của một đối tượng.

- Đối tượng là một thực thể cụ thể được tạo ra từ lớp đối tượng, có thể có trạng thái riêng và thực hiện các hành động.

**4. Phân biệt các phạm vi truy xuất private, protected, public?**

Phạm vi truy xuất trong lập trình hướng đối tượng:

- Private: Các thành viên private chỉ có thể được truy cập từ bên trong cùng một lớp.

- Protected: Các thành viên protected có thể được truy cập từ bên trong cùng một lớp và các lớp dẫn xuất từ lớp đó.

- Public: Các thành viên public có thể được truy cập từ bất kỳ nơi nào.

**5. Constructor là gì? Constructor mặc định?**

- Constructor là một phương thức đặc biệt trong một lớp được gọi để khởi tạo đối tượng. Nó có cùng tên với lớp và không có kiểu trả về. Constructor được gọi tự động khi một đối tượng mới được tạo.

- Constructor mặc định là constructor không có tham số.

**6. Destructor là gì?**

- Destructor là một phương thức đặc biệt trong một lớp được gọi khi đối tượng bị hủy. Nó có cùng tên với lớp, được đặt trước bởi ký tự `~` và không có tham số.

- Destructor được sử dụng để giải phóng tài nguyên mà đối tượng đã sử dụng.

**7. Kế thừa là gì?**

**-** Kế thừa có thể được định nghĩa là quá trình mà một lớp (class) có được các thuộc tính của một lớp khác

- Kế thừa trong lập trình hướng đối tượng là quá trình tạo ra các lớp dẫn xuất (lớp con) từ một lớp đã tồn tại (lớp cơ sở hoặc lớp cha).

- Lớp con kế thừa các thuộc tính và phương thức của lớp cha, cho phép tái sử dụng mã nguồn và mở rộng chức năng của lớp cha.

- Mối quan hệ 1-1 or nhiều-nhiều or đặc biệt hoá-đa dạng hoá

**8. Phân biệt các kiểu kế thừa private, protected, public?**

Các kiểu kế thừa trong lập trình hướng đối tượng:

**- Kế thừa private:** Các thành viên *private* của lớp cha *không được kế thừa cho lớp con*.

**- Kế thừa protected:** Các thành viên *protected* và *public* của lớp cha *được kế thừa cho lớp con với quyền truy cập là protected*.

**- Kế thừa public:** Các thành viên *protected* và *public* của lớp cha *được kế thừa cho lớp con với quyền truy cập là protected và public tương ứng.*

**9. Đa hình là gì?**

- Đa hình trong lập trình hướng đối tượng cho phép ***một phương thức có thể có nhiều hình thức thực hiện khác nhau tùy thuộc vào đối tượng gọi***.

- Điều này cho phép gọi phương thức dựa trên kiểu đối tượng thực tế thay vì kiểu đối tượng được khai báo.

**10. Lớp trừu tượng là gì? Phương thức thuần ảo là gì?**

- Lớp trừu tượng (abstract class) là một lớp ***mà không thể tạo ra các đối tượng trực tiếp từ nó, mà chỉ được sử dụng để làm lớp cơ sở cho các lớp dẫn xuất.*** Lớp trừu tượng chứa ***ít nhất một phương thức thuần ảo*** (pure virtual method). Nó có chung bản chất với nhiều lớp con khác.

- Hàm ảo của lớp trừu tượng -> các lớp con phải viết lại hàm đó -> ghi đè phương thức.

- Phương thức thuần ảo là một phương thức không có định nghĩa trong lớp cơ sở, mà chỉ có định nghĩa trong các lớp dẫn xuất.

**11. Đa năng hóa toán tử là gì?**

Đa năng hóa toán tử (operator overloading) là ***quá trình định nghĩa lại các toán tử*** để hoạt động với các kiểu dữ liệu tùy chỉnh. Khi ta đa năng hóa toán tử, ta có thể định nghĩa cách toán tử hoạt động với các đối tượng của lớp mà ta định nghĩa. Điều này giúp tăng tính linh hoạt và dễ đọc của mã nguồn.

**12. Quá tải hàm?**

- Nhiều hàm có cùng tên nhưng có nhiều tham số.

- Là một khái niệm cho phép bạn định nghĩa nhiều hàm có cùng tên nhưng có các danh sách tham số khác nhau.

- Khi gọi một hàm với các đối số tương ứng, trình biên dịch sẽ xác định và chọn hàm phù hợp dựa trên các đối số được truyền vào.

**13. Điều kiện** **nào khởi tạo đối tượng tự động?**

**- Lớp có hàm khởi tạo (constructor) mặc định (không có tham số):** Khi khai báo một biến thuộc lớp đó trong một hàm hoặc khối lệnh, đối tượng tự động sẽ được khởi tạo tự động bằng hàm khởi tạo mặc định.

**- Lớp có hàm khởi tạo có tham số:** Khi khai báo một biến thuộc lớp đó trong một hàm hoặc khối lệnh và cung cấp các đối số cho hàm khởi tạo, đối tượng tự động sẽ được khởi tạo bằng hàm khởi tạo tương ứng với các tham số truyền vào.

**- Lớp có hàm khởi tạo có tham số mặc nhiên (default arguments**): Khi khai báo một biến thuộc lớp đó trong một hàm hoặc khối lệnh mà không cung cấp các đối số cho hàm khởi tạo, đối tượng tự động sẽ được khởi tạo bằng hàm khởi tạo có tham số mặc nhiên.

class MyClass

{

public:

MyClass() {

// Hàm khởi tạo mặc định

}

MyClass(int x) {

// Hàm khởi tạo có tham số

}

MyClass(int y, int z = 0) {

// Hàm khởi tạo có tham số mặc nhiên

}

};

}

};

**14. Hàm bạn:**

- Hàm bạn là loại hàm *không phải là hàm thành viên của một lớp* nhưng có thể truy cập vào các thành phần, kể cả private của lớp đó.

- Một hàm có thể là bạn của nhiều lớp. Lúc đó, nó có quyền truy cập tới tất cả các thuộc tính của các đối tượng trong các lớp này.

- Lớp bạn (friend class): Khi một lớp có lớp bạn thì tất cả hàm thành viên của lớp bạn sẽ trở thành hàm bạn của lớp đó.

**15. Đa hình thời gian chạy/biên dịch**

**- Đa hình thời gian chạy (Runtime Polymorphism):**

+ Đa hình thời gian chạy là khả năng của một đối tượng để xử lý các phương thức theo hình thức của đối tượng thực tế được liên kết tại thời điểm chạy.

+ Đa hình thời gian chạy thường được đạt được thông qua việc sử dụng kỹ thuật ghi đè phương thức (method overriding) và con trỏ hướng đối tượng (polymorphic pointers) hoặc tham chiếu (polymorphic references).

+ Đa hình thời gian chạy cho phép chúng ta thực hiện các phương thức cùng tên trong các lớp khác nhau, nhưng hành vi của phương thức sẽ được xác định tại thời điểm chạy dựa trên đối tượng đang tham chiếu đến.

+ Đa hình thời gian chạy thường được sử dụng trong kịch bản đa hình động, khi ta muốn thay đổi hành vi của đối tượng trong quá trình chạy chương trình.

**- Đa hình thời gian biên dịch (Compile-time Polymorphism):**

+ Đa hình thời gian biên dịch là khả năng của một phương thức để có thể xử lý các đối số khác nhau dựa trên kiểu dữ liệu của chúng tại thời điểm biên dịch.

+ Đa hình thời gian biên dịch thường được đạt được thông qua việc sử dụng các kỹ thuật như quy tắc quy định (function overloading) và các mẫu hàm (function templates).

+ Đa hình thời gian biên dịch cho phép chúng ta sử dụng các phương thức cùng tên nhưng có cách hoạt động khác nhau dựa trên loại và số lượng các đối số được truyền vào.

+ Đa hình thời gian biên dịch thường được sử dụng trong kịch bản đa hình tĩnh, khi ta đã biết trước các kiểu dữ liệu và tham số của phương thức tại thời điểm biên dịch.

**16. Giao diện:**

- Tập hợp các phương thức của 1 đối tượng.

- Là một chức năng mà người lập trình có thể thêm vào bất kỳ lớp nào => cung cấp phương tiện cho người lập trình tương tác với đối tượng đó.

- Một chức năng bao gồm 01 hoặc nhiều phương thức

- Quan hệ với lớp: “A can do b”.

+ A là lớp hiện thực (implement) b.

+ b là một chức năng của A.

- Ví dụ: Xe tải có thể chở hàng, Xe Buýt có thể chở khách...

- Trong C++, việc khai báo một interface có nghĩa là chúng ta *khai báo một lớp với hàm thuần ảo và một phương thức hủy ảo (virtual destructor).*

- Có thể hiểu Interface là *một bản mô tả hành vi hoặc khả năng của một class* mà không đưa ra cách thực hiện cụ thể của class đó như thế nào. => Bắt buộc các lớp khác phải cài đặc lại phương thức này để hoàn thiện lớp.