**LÝ THUYẾT – OOP**

1. **Lớp đối tượng có mấy tính chất ?**

* Có 4 tính chất:
  + Tính **trừu tượng** (Abstraction).
    - Khái quát hóa về 1 tập đối tượng có chung đặc điểm.
  + Tính **đóng gói** (Encapsulation).
    - Những đặc điểm liên quan sẽ được gọm lại một nhóm cùng tên
    - Che dấu thông tin
      * Nội bộ mới thấy và được sử dụng trực tiếp.
      * Nội bộ không nhìn thấy và không được sử dụng trực tiếp.
    - Chương trình không bị thay đổi bởi các đoạn chương trình khác.
  + Tính **kế thừa** (Inheritance)
    - Cơ chế cho phép lớp dẫn xuất:
      * có được thuộc tính và thao tác của lớp cơ sở.
    - Cài đặt cho phép giữa các quan hệ sau giữa các đối tượng:
      * Đặt biệt hóa – Tổng quát hoá.
    - Loại bỏ những đoạn chương trình lặp lại nhiều lần ở các lớp đối tượng.
  + Tính **đa hình** (Polymorphism).
    - Cơ chế cho phép một tên thao tác hoặc thuộc tính có thể định nghĩa lại và có nhiều cài đặt khác nhau ở các lớp đối tượng khác nhau.

1. **Lớp đối tượng (class) là gì ?**

* Lớp đối tượng hiểu đơn giản nhất là sự tích hợp của 2 thành phần:
  + Thành phần dữ liệu – các thông tin về chính đối tượng.
  + Thành phần hành động – các khả năng mà đối tượng có thể thực hiện.
* Một mô tả trừu tượng của các nhóm đối tượng cùng bản chất

1. **Đối tượng (object) là gì ?**

* Đối tượng thuộc về 1 lớp là sự thể hiện cụ thể những mô tả trừu tượng của lớp đó.
* Có thể có nhiều đối tượng cùng thuộc về 1 lớp.

1. **Phương thức (method) là gì ?**

* Khả năng mà đối tượng thuộc về lớp đó có thể thực hiện.
* Trong một lớp có thể có nhiều phương thức.
* Một lớp có tối thiếu 0 phương thức và không giới hạn số lượng phương thức.
* **Cùng 1 phương thức** có thể áp dụng cho **nhiều lớp đối tượng khác nhau**
  + - Tính **đa hình**

1. **Đối tượng hành động là gì ?**

* Đối tượng gọi thực hiện phương thức mà lớp đối nó thuộc về cung cấp.
* Để đối tượng hành động thực hiện phương thực ta dùng toán tử chấm (dot operator).

1. **Hãy kể các câu thành ngữ việc có nhiều lớp đối tượng ?**

* Cá không ăn muối cá ươn, con cãi cha mẹ trăm đường con hư.
* Râu tôm nấu với ruột bầu, chồng chang vợ húp gật đầu khen ngon.
* Cứu vật, vật trả ơn, cứu nhân, nhân trả oán.
* Vắng chủ nhà, gà vọc đuôi tôm.

1. **Hãy kể các câu thành ngữ thể hiện việc các đối tượng cùng thuộc 1 lớp ?**

* Hộ phụ sinh hổ tử.
* Cha nào con nấy.
* Nồi nào úp vung đó.
* Cá mè một lứa.
* Gió tầng nào gặp mây tầng đó.
* Ngưu tầm ngưu mã tầm mã.
* Trâu đồng nào ăn cỏ đồng nấy.

1. **Toán tử phạm vi dùng để làm gì ?**

* Dùng để truy cập đến **biến toàn cục** trong trường hợp có biến cục bộ cùng tên.
* Dùng để chỉ rõ phương thức thuộc lớp nào khi được **định nghĩa bên ngoài lớp** mà đối tượng thuộc về.

1. **Cấp phát bộ nhớ ?**

* Bằng toán tử **new.**
* Khai báo bằng biến con trỏ để giữ địa chỉ vùng nhớ sẻ được cấp phát.
* Miền giá trị của con trỏ là địa chỉ ô nhớ.
* Có 2 cách cấp phát bộ nhớ: <type\_data>\* p;
  + Cho 1 phần tử: p = new < type\_data >;
  + Cho n phần tử: p = new < type\_data> [n];
* Nếu cấp phát thành công:
  + Con trỏ P sẽ nắm giữ địa chỉ đầu tiên của vùng nhớ được cấp phát.
* Nếu cấp phát thất bại:
  + Con trỏ P sẽ năm giữ địa chỉ **NULL**.

1. **Giải phóng bộ nhớ ?**

* Bằng toán tử **delete**.
* Có 2 cách giải phóng bộ nhớ:
  + Giải phóng 1 phần tử: delete p;
  + Giải phóng n phần tử: delete[] p;

1. **Khai báo hàm có đối mặc định ?**

* **Mục đích**:
  + Gán giá trị mặc định cho các đối số của hàm.
  + Được sử dụng khi không truyền giá trị cho các đối số khi gọi hàm.
* Nếu **có khai báo nguyên mẫu hàm**:
  + Các đối mặc định phải được gán trong nguyên mẫu, trong định nghĩa không cần gán lại.
* Nếu **không khái báo nguyên mẫu hàm**:
  + Các đối tượng được gán trong vòng đầu của định nghĩa hàm.
* **Gọi hàm có đối mặc định**:
  + Các tham số thiếu trong lời gọi hàm phải tương ứng với các đối số mặc định.
    - (trái sang phải)
* Nếu trong lời gọi hàm cung cấp **đủ** tham số:
  + Dùng giá trị truyền vào.
* Nếu trong lời gọi hàm cùng cấp **không đủ** tham số:
  + Dùng giá trị mặc định.
* Có thể dùng hằng, biến toàn cục, hàm:
  + Gán giá trị cho đối số mặc định
* VD:
  + Void f(int d1, float d2, char\* d3 = “”, int d4 = 100, double d5 = 3.1)
    - f(3) => sai vì thiếu tham số cho đối số mặc định d2.
    - f(3, 3.2), f(3, 3,2, “C++”), f(3, 3,2, “C++”, 5), f(3, 3,2, “C++”, 5, 3.4)
      * => đúng

1. **Có mấy loại biến ?**

* Có 3 loại:
  + Giá trị: chứa dữ liệu.
  + Con trỏ: chứa địa chỉ.
  + Tham chiếu: dùng làm bí danh cho 1 biến giá trị.

1. **Nêu đặc điểm của biến tham chiếu ?**

* Không cấp phát vùng nhớ => không có địa chỉ riêng.
* Sử dụng chung vùng nhớ của biến mà nó tham chiếu.
* Khai báo rõ tham chiếu đến biến nào vì không có địa chỉ riêng.
* Không được khai báo mảng tham chiếu.

1. **Phân biệt truyền tham số đầu vào của hàm ?**

|  |  |
| --- | --- |
| **Tham trị** | **Tham chiếu** |
| + Gán giá trị sao chép từ các đối số trong lời gọi hàm cho các tham số đầu vào.  + Thao tác trên chính vùng nhớ của tham số đầu vào trong hàm.  + Không thay đổi giá trị của các đối số trong lời gọi hàm vì không thao tác trực tiếp trên các vùng nhớ của các đối số. | + Không tạo ra bản sao của các tham số đầu vào.  + Thao tác trực tiếp vùng nhớ của các tham số đầu vào – vùng nhớ của các đối số trong lời gọi hàm.  + Thay đổi giá trị các đối số trong lời gọi hàm.  + Cùng kiểu dữ liệu với đối số trong lời gọi hàm. |

1. **Cho biết ưu điểm của Inline Function ?**

* Tăng tốc độc chương trình do không thực hiệ các thao tác có tính thủ tục khi gọi hàm:
  + Cấp phát vùng nhớ cho các đối số, biến cục bộ.
  + Truyền dữ liệu cho các tham số của đối số.
  + Giải phóng vùng nhớ trước khi thoát khỏi hàm
  + ……….
* Chỉ dùng inline function cho hàm có kích thước nhỏ => tăng dung lượng bộ nhớ.
* Hàm có biến sau sẽ không xử lý theo inline:
  + Chứa biến static.
  + Đệ quy
  + Lệnh chu trình, goto, switch

1. **Thế nào là nạp chồng hàm (Overloading) ?**

* Dùng **cùng tên hàm** nhưng **khác các tham số đầu vào**:
  + Số lượng, thứ tự, Kiểu dữ liệu.
* Các hàm trùng tên chỉ khác giá trị trả về (type\_return) : KHÔNG CHẤP NHẬN.
* Sử dụng khi thục hiện các công viêc như nhau nhưng trên các đối tượng khác nhau.
* Dùng phép chuyển kiểu dữ liệu bộ tham số **trùng hoàn toàn** bộ đối số của hàm.

1. **Thế nào là nạp chồng toán tử ?**

* Dùng khi thục hiện trên các dữ liệu không chuẩn mặc định của chương trình như (stuct, mảng,..)
* Số đối của hàm nạp chồng toán tử phụ thuộc số toán hạng tham gia vào phép toán được định nghĩa nạp chồng.

1. **Tính chất hàm thành phần ?**

* Có quyền truy cập đến tất cả các thành phần private của các đối tượng, tham chiếu đối tượng hay con trỏ đối tượng có cùng kiểu lớp đối tượng
  + - * + Gọi là **tham số hình thức**.

1. **Con trỏ this ?**

* Định nghĩa trong các **hàm thành phần** dùng để **xác định địa chỉ của đối tượng dùng làm tham số ngầm định** cho hàm thành phần

=> Con trỏ this **tham chiếu** đến đối tượng **đang gọi hàm thành phần**.

* Bên trong thân của một phương thức, this là con
* trỏ đối tượng thuộc về lớp mà phương thức đó thuộc về.
* Bên trong thân của một phương thức, this là con trỏ đối tượng giữ địa chỉ của đối tượng đang gọi thực hiện phương thức.
* Hơn nữa, **\*this chính là đối tượng đang gọi thực hiện phương thức.**
* **Ghi nhớ:**
  + **Miền giá trị** của một biến con trỏ là **địa chỉ ô nhớ**.
  + **Miền giá trị** của một biến con trỏ đối tượng là **địa chỉ ô nhớ**.

1. Sơ lược về thư viện iostream ?
2. **Thế nào là phương thức thiết lập (constructors) và trình bày các đặc điểm của phương thức và phân loại ?**

* Là một **phương thức đặc biệt** dùng **thiết lập thông tin ban đầu** cho các đối tượng thuộc về lớp **ngay khi đối tượng được khai báo**
* **Constructor** thường có **thuộc tính public**.
* **Các đặc điểm**: **6 tính chất**
  + **Tên** phương thức **trùng** với **tên lớp**.
  + Được **tự động gọi thực hiện** ngay **khi** đối tượng **được khai báo**.
  + **Không** có **giá trị trả về** (no return type).
  + Trong **quá trình sống** của đối tượng thì chỉ có **1 lần duy nhất** một phương thức **thiết lập được gọi thực hiện** => khi **đối tượng ra đời**.
  + Có thể có **nhiều phương thức thiết lập** 1 lớp.
  + Các phương thức thiết lập thuộc nhóm phương thức khởi tạo.
* **Phân loại**: **3 nhóm**
  + Phương thức thiết lập **mặc định** **(defualt constructor)**:
    - **Thiết lập thông tin ban đầu** cho đối tượng thuộc về lớp **bằng những giá trị mặc định** (do người lập trình quyết định).
    - **Không có đối số** nào **được cung cấp**.
    - **Mặc định** sẽ là **phương thức rỗng** khi **chương trình** được **gọi thục hiện**.
    - **Luu ý:**
      * ***Không có constructor mặc định*** nhưng có constructor khác
        + **Biên dịch báo lỗi** nếu không cung cấp tham số.
  + Phương thức thiết lập **sao chép** **(copy constructor)**:
    - **Nhận tham số đầu vào** là một **đối tượng cùng thuộc về lớp**. Tạo ra đối tượng mới **hoàn toàn giống** với **đối tượng đối số đầu vào.**
    - Có **tham số** là **hằng tham chiếu** đến đối tượng thuộc chính lớp này.
  + Phương thức thiết lập **nhận tham số đầu vào** **(parameterized constructor)**:
    - **Không phải** phương thức thiết lập mặc định và sao chép.
* **Ý Nghĩa**:
  + **Khởi tạo giá trị ban đầu** cho các **đối tượng thuộc về lớp** **ngay khi** đối tượng được **khai báo.**
  + **Ép kiểu** đối tượng từ lớp đối tượng này sang đối tượng thuộc lớp đối tượng khác.

1. **Thế nào là phương thức phá hủy (destructors) và trình bày các đặc điểm của phương thức và phân loại ?**

* Là một **phương thức đặc biệt** dùng để **thu hồi lại tất cả tài nguyên đã cấp phát** cho đối tượng **khi đối tượng hết phạm vị hoạt động** (scope)
* **Destructor** có **thuộc tính public**.
* **Câu thần chú**: **“Dọn dẹp ‘xác chết’ của đối tượng khi đối tượng ‘đi bán muối”** .
* **Các đặc điểm**:  **7 tính chất**
  + Tên **trùng** với **tên lớp** và có **dấu ‘~’ ngay đầu** phương thức.
  + **Không** có **giá trị trả về** (no return type).
  + **Không** có **tham số đầu vào**.
  + Được **tự động** **gọi thực hiện** khi đối tượng **hết phạm vi sử dụng**.
  + Có **DUY NHẤT 1** phương thức phá hủy trong 1 lớp.
  + Trong **quá trình sống** của đối tượng thì chỉ có **1 lần duy nhất** một phương thức **phá hủy được gọi thực hiện.**
  + **Thuộc** nhóm **phương thức xử lý**.

1. **Muốn thao tác với dữ liệu private thì có những phương thức nào?**

* Phương thức **TRUY VẤN**:
  + Không thay đổi trạng thái hiện tại của đối tượng.
  + **Quy ước đặt tên**:
    - **Tiền tố “get”** rồi đến **tên thành viên** cần **truy vấn**.
    - **Tiền tố “is”** cho **truy vấn có điều kiện**
    - **Loại truy vấn khác** nên **có mô tả riêng.**
* Phương thức **CẬP NHẬT**:
  + **Thay đổi trạng thái** đối tượng bằng cách sửa đổi 1 hoặc nhiều dữ liệu thành viên.
  + Dạng đơn giản nhất là gán 1 giá trị cho dữ liệu thành viên.
  + **Quy ước đặt tên**:
    - Tiền tố **“set”** + **tên thành viên** cần **sửa.**

1. **Trình bày về thành viên tĩnh – static member ?**

* Đứng trước dữ liệu được khai báo => được xem như biến toàn cục. (trong C).
* Đứng trước dữ liệu hoặc một phương thức của một lớp => giá trị được lưu lại và có ý nghỉa cho đối tượng khác cùng lớp này. (trong C++)
* Có thể là public, private, protected.
* **Đối với class**:
  + – Dùng để **khai báo thành viên dữ liệu dùng chung** cho mọi sự thể hiện của lớp.
    - Một bản duy nhất tồn tại trong suốt chương trình.
    - Dùng chung cho tất cả thể hiện của lớp.
    - Bất kể lớp đó có bao nhiêu thể hiện.
* **Phương thức static**:
  + Ngoài ý nghĩa tương tự với dữ liệu – Sự khác biệt cơ bản:
    - **Cho phép truy cập** đến **các phương thức static** khi **chưa khai báo đối tượng (thông qua tên lớp)**.
  + **Phạm vị hoạt động**:
    - **Public**:
      * **Truy cập thông qua bất kì đối tượng** nào của lớp đó, hoặc truy cập **thông qua tên lớp sử dụng toán tử phạm vị “::”**
      * **Thêm** vào **đầu tên lớp** và **toán tử phạm vi** cho **thành viên dữ liệu**.
    - **Private và Protected:**
      * **Truy cập thông qua hàm thành viên public** của lớp hoặc là **hàm bạn (friend)** của lớp
      * Khi **đối tượng không tồn tại** => **hàm thành viên public** phải **được cung cấp** và **được gọi** bởi thêm vào đầu tiên với **tên lớp** và **toán tử định vị phạm vị**.
        + **Các thành viên lớp tĩnh TỒN TẠI ngay cả khi lớp đối tượng không tồn tại.**