**CÂU HỎI LÝ THUYẾT**

**Câu 1:**

Trình bày khái niệm về đối tượng (object), lớp (class), thể hiện (instance), thành phần dữ liệu (data member), phương thức (method), thông điệp (message).

***Trả lời:***

- Class là một nhóm các đối tượng có cùng thuộc tính và các hành vi. Khi một phần tử dữ liệu được khai báo là phần tử của một lớp thì nó được gọi là một đối tượng (Object).  
- Đối tượng (object) thường là một thực thể trong thế giới thực, có các thuộc tính và đặc điểm riêng biệt. Khi ta khai báo một đối tượng thuộc một lớp thì đối tượng còn được gọi là thể hiện (instance) của lớp đó.

Đối tượng = Dữ liệu + phương thức

- Sự khác nhau giữa đối tượng và lớp

+Lớp như là một khuôn mẫu, đối tượng là một thực thể thể hiện dựa trên  
khuôn mẫu đó

+ Ví dụ: Lớp Car (các thuộc tính: màu, biển số, hãng,… Phương thức: chạy, dừng,…) thì các đối tượng của lớp Car này là xe Aufi, Volvo,…  
- Thành phần dữ liệu (data member), phương thức (method), thông điệp(message) :+ Thành phần dữ liệu hay còn gọi là dữ liệu thành viên của lớp là các dữ liệu thể hiện các thuộc tính đặc trưng của một lớp.

+ Phương thức: hay hàm là một đoạn chương trình thực thi một nhiệm vụ  
nào đó. Sau khi thực thi xong đoạn chương trình nó sẽ trả về một giá trị. Tùy  
vào lúc khai báo kiểu giá trị trả về mà nó sẽ trả về giá trị kiểu đó.  
+ Thông điệp: Các phương thức định nghĩa trong một lớp có thể được gọi  
bởi các đối tượng của lớp đó. Điều này được gọi là gửi một thông điệp  
(Message) cho đối tượng. Như vậy một thông điệp bao gồm 3 thành phần:  
đối tượng nhận thông điệp, phương thức được gọi từ đối tượng và các tham  
số truyền cho phương thức này.

**Câu 2:**

Trình bày các toán tử xác định phạm vi lớp và truy cập tới các thành viên. Ví dụ minh hoạ?

***Trả lời:***

- Khi khai báo các thành phần của lớp (các thuộc tính và các phương thức) có thể dùng các từ khoá private, public và protected để qui định phạm vi sử dụng của các thành phần .Nếu không qui định cụ thể (tức không dùng từ khóa) thì mặc định trình biên dịch C++ sẽ hiểu đó là private.

- Các thành phần private chỉ được sử dụng bên trong lớp (trong thân các phương thức của lớp) hoặc từ các hàm bạn. Các hàm không phải là phương thức của lớp hoặc từ các hàm bạn thì không được phép sử dụng các thành phần này. - Protected: trong phạm vi lớp đó và các lớp con kế thừa

- Public: tại mọi nơi nếu đối tượng tồn tại

VD: QuanLyThuVien

**Câu 3:**

Trình bày khái niệm về hàm khởi tạo, hàm huỷ, các tính chất. Ví dụ minh hoạ?

***Trả lời:***

**\* Hàm khởi tạo (constructor):** Khi một đối tượng được tạo, các dữ liệu thành viên của nó có thể được khởi tạo bởi một hàm khởi tạo, hàm sẽ được gọi tự động mỗi khi đối tượng của lớp đó được tạo. Một hàm khởi tạo là một hàm thành viên với tên giống như tên của lớp và không có kiểu trả về. Trong một lớp ta có thể xây dựng nhiều hàm khởi tạo, nhưng các hàm khởi tạo phải có số đối số truyền vào khác nhau  
***Các tính chất***:

- Được gọi một cách tự động khi đối tượng của lớp được tạo ra. Dùng để khởi tạo các giá trị ban đầu cho các thành phần dữ liệu của lớp.

- Tên hàm giống tên lớp, phạm vi truy cập luôn là public

- Không có giá trị trả về

- Trước khi thực hiện, đối tượng chưa thực sự tồn tại trong bộ nhớ, sau khi hoàn thành hàm khởi tạo, một thể hiện hợp lệ của lớp được lưu trong bộ nhớ.

Khi người dùng không khai báo và định nghĩa hàm khời tạo thì trình biên dịch sẽ gọi đến hàm khởi tạo mặc định

VD: Tamgiac…

**\* Hàm hủy (destructor)** của một lớp được gọi khi đối tượng được hủy bỏ nghĩa là khi sự thực hiện chương trình rời khỏi phạm vi mà trong đó đối tượng của lớp đó được khởi tạo. Tương tự hàm khởi tạo hàm hủy trùng tên lớp (có dấu ~ đằng trước) và không trả về giá trị. Hàm hủy không có các đối số truyền vào

VD: HamHuy…

**Khi nào các constructor và destructor được gọi?**

Các constructor và destructor được gọi một cách tự động. Thứ tự các hàm này được gọi phụ thuộc vào thứ tự trong đó sự thực hiện vào và rời khỏi phạm vi mà các đối tượng được khởi tạo. Một cách tổng quát, các destructor được gọi theo thứ tự ngược với thứ tự của các constructor được gọi. Các constructor được gọi của các đối tượng khai báo trong phạm vi toàn cục trước bất kỳ hàm nào (bao gồm hàm main()) trong file mà bắt đầu thực hiện. Các destructor tương ứng được gọi khi hàm main() kết thúc hoặc hàm exit() được gọi. Hoặc khi dùng con trỏ kiểu đối tượng, khi khai báo và xin cấp phát bộ nhớ cho con trỏ, hàm constructor sẽ được gọi, còn khi ta giải phóng vùng nhớ mà con trỏ đang chiếm giữ bằng lệnh delete , thì hàm destructor sẽ được gọi.

**Câu 4:**

Trình bày khái niệm về nạp chồng toán tử, cho ví dụ minh hoạ?.

***Trả lời:***

-Các lập trình viên có thể sử dụng các toán tử với các kiểu do người dùng định nghĩa. Mặc dù C++ không cho phép các toán tử mới được tạo, nó cho phép các toán tử đã tồn tại được đa năng hóa sao cho khi các toán tử này được sử dụng với các đối tượng của lớp, các toán tử có ý nghĩa thích hợp các kiểu mới. Hay nói cách khác, nạp chông toán tử là định nghĩ lại các toán tử cho các kiểu dữ liệu do người dùng tự định nghĩa.  
- Các toán tử được đa năng hóa bằng cách viết một định nghĩa hàm, khai báo của nó có dạng như sau: kieu operator ki\_hieu\_toan\_tu ( cac\_tham\_so );

- Để sử dụng một toán tử với các đối tượng của lớp, toán tử phải được đa năng hóa ngoại trừ hai điều. Điều thứ nhất toán tử gán có thể sử dụng với mọi lớp mà không cần đa năng hóa. Cách cư xử mặc định của toán tử gán là một phép gán thành viên của các thành viên dữ liệu của lớp. Điều thứ hai toán tử địa chỉ (&) cũng có thể được sử dụng với các đối tượng của bất kỳ lớp nào mà không cần đa năng hóa; Nó trả về địa chỉ của đối tượng trong bộ nhớ.

Các giới hạn của đa năng hóa toán tử

+ - \* / % ^ & |

~ ! = < > += -= \*=

/= %= ^= &= |= << >> >>=

<<= == != <= >= && || ++

-- ->\* , -> [] () new delete

- Các toán tử của không thể đa năng hóa:

. .\* :: ?: sizeof

- Thứ tự ưu tiên của một toán tử không thể được thay đổi bởi đa năng hóa.Ví  
dụ phép nhân phải được thực hiện trước phép cộng.

***Đa năng hóa toán tử hai ngôi***

Một toán tử hai ngôi có thể được đa năng hóa như là hàm thành viên không  
tĩnh với một tham số hoặc như một hàm không thành viên (hàm bạn) với hai  
tham số (một trong các tham số này phải là hoặc là một đối tượng lớp hoặc  
là một tham chiếu đến đối tượng lớp).

Ví dụ: Với nạp chồng toán tử hai ngôi là hàm thành viên (nạp chồng các toán tử +, - trong bài Date, Time, Phanso)

sp3=sp1\*sp2.  
Chương trình hiểu là

sp3=sp1.operator\*(sp2).  
Chú ý: Thứ tự của đối số khi truyền.

***Đa năng hóa toán tử một ngôi***

Một toán tử một ngôi của lớp được đa năng hóa như một hàm thành viên  
không tĩnh với không có tham số hoặc như một hàm không thành viên với  
một tham số; Tham số đó phải hoặc là một đối tượng lớp hoặc là một tham  
chiếu đến đối tượng lớp.

Đa năng hóa một số toán tử đặc biệt

Các toán tử [ ], ( ), = là các toán tử đặc biệt và khi đa năng toán tử này thì  
hàm toán tử tương ứng phải là thành viên của một lớp.  
***Các trường hợp nạp chồng toán tử bắt buộc dùng hàm thành viên***Các toán tử gán (=), chỉ số con ([]), lệnh gọi hàm (“()”) và lựa chọn thành  
viên (->) phải được định nghĩa là các hàm thành viên  
***Các trường hợp nạp chồng toán tử bắt buộc phải dùng hàm bạn***: toán tử  
nhập, xuất; toán tử hai ngôi với hai toán hạng khác kiểu nhau (VD: cộng một  
số nguyên với một phân số)

*Giải thích: Vì sao toán tử nhập xuất phải nạp chồng bằng hàm bạn mà  
không được nạp chồng theo hàm thành viên :* vì để nạp chồng toán tử <<, >>  
cần có một toán hạng trái của kiểu ostream& (cout, cin), vì thế nó không thể  
là hàm thành viên

VD: PhanSo...

**Câu 5:**

Trình bày khái niệm về các thành viên tĩnh (static) của lớp, cho ví dụ minh hoạ?

***Trả lời:***

- Dữ liệu thành viên tĩnh của lớp rất hữu dụng trong trường hợp tất cả các đối tượng cùng một lớp muốn chia sẻ một thông tin chung. Dữ liệu thành viên tĩnh có phạm vi trong toàn lớp và tồn tại trong suốt chương trình. Dữ liệu thành viên tĩnh có thể được

dùng như là biến toàn cục của lớp. Dữ liệu static không được khởi tạo ngầm định, vì vậy phải khởi tạo nó một cách tường minh bên ngoài khai báo class. Được truy cập thông qua toán tử phạm vi.

Ví dụ 1: Dùng biến tĩnh count trong lớp hinhchunhat để biết có bao nhiêu biến đối tượng đã được khai báo có kiểu lớp hinhchunhat.

Class hinhchunhat {

int cd,cr;

public:  
static int count;

hinhchunhat(int cd1, int cr1);

int tinhdientich( );

};

hinhchunhat :: hinhchunhat(int cd1, int cr1)

{

cd = cd1;

cr = cr1;

count++;

};  
int hinhchunhat :: tinhdientich( )

{

return cd\*cr;

};

Ví dụ 2:

#include<iostream>

using namespace std;

class MyClass {

public:

static int x;

};

int MyClass::x = 10;

int main() {

MyClass m1;

MyClass m2;

m1.x = 20;

cout << m1.x << " " << m2.x;

return 0;

}

**-Static function:** có thể dùng mà không cần khởi tạo đối tượng; các hàm  
không phải static có thể truy cập được các biến static; còn các hàm static  
không thể truy cập được các biến không phải static

**Câu 6:**

Khái niệm về tính kế thừa, tính chất. Nêu ví dụ minh hoạ?

***Trả lời:***

**- Kế thừa:** là khả năng cho phép xay dựng một lớp mới được thừa hưởng  
các thành phần từ một hay nhiều lớp đã có, trong lớp mới ta có thể bổ sung  
các thành phần hoặc định nghĩa lại các thành phần. Trong đó lớp được kế  
thừa gọi là lớp cơ sở (lớp cha), lớp kế thừa gọi là lớp dẫn xuất (lớp con).  
Lớp dẫn xuất có thể định nghĩa lại hàm thành viên ủa lớp cơ sở (ghi đè),  
muốn gọi đến phương thức của lớp cơ sở, dùng toán tử ::

Void SinhVien::Nhap() {

Nguoi::Nhap();  
cin>>nganh;  
}  
**-Ví dụ: QuanLyThuVien**

**-Cú pháp:**

class <tên lớp con>:[kiểu dẫn xuất]<tên lớp cha1>

,[kiểu dẫn xuất]<tên lớp cha 2>

…..  
{  
// Các thành phần của lớp con

};  
**Kế thừa đơn**: Là tiến trình tạo một lớp tù một lớp đã có sẵn.

VD:

**#include <iostream>**

**using namespace std;**

**class A**

**{**

**public:**

**void display()**

**{**

**cout << "Base class content.";**

**}**

**};**

**class B : public A**

**{**

**};**

**class C : public B**

**{**

**};**

**int main()**

**{**

**C obj;**

**obj.display();**

**return 0;**

**}**

**Đa kế thừa:** Một lớp có thể được dẫn xuất từ nhiều lớp cơ sở, sự dẫn xuất  
như vậy được gọi là đa kế thừa. Đa kế thừa có nghĩa là một lớp dẫn xuất kế  
thừa các thành viên của các lớp cơ sở khác nhau.

VD: DaKeThua1

**Các phương thức không được kế thừa:** Hàm khởi tạo, hàm hủy, toán tử  
gán, hàm bạn, nạp chồng toán tử new

**Hàm khởi tạo:** hàm khởi tạo của lớp cha không được kế thừa. Mỗi đối  
tượng của lớp con có thể là một đối tượng của lớp cha, do vậy, khi gọi hàm  
khởi tạo của lớp con sẽ kéo theo gọi hàm khởi tạo của lớp cha (thứ tự gọi:  
hàm khởi tạo lớp cha->hàm khởi tao lớp con). Nếu xây dựng hàm khởi tạo  
của lớp con, phải gọi hàm khởi tạo của lớp cha một cách tường minh  
**Hàm hủy:** hàm hủy của lớp cha không được kế thừa, các hàm huy được  
thực hiện ngược lại so với hàm khởi tạo, hàm huy của lớp dẫn xuất được thi  
hành trước hàm hủy của lớp cơ sở

**Câu 7:**

Tính đa hình, các đặc trưng của phương thức ảo, ví dụ minh hoạ?.

***Trả lời:***

Lớp trừu tượng là lớp có một hoặc nhiều phương thức thuần ảo không làm gì  
cả và cũng không thể tạo ra đối tượng thuộc lớp đó. Nó chỉ được dùng làm lớp cơ bản cho các lớp khác.

- Hàm thuần ảo: là hàm chỉ khai báo chứ không có các định nghĩa, và được  
gán giá trị 0.

Kiểu khai báo:

virtual kieu\_tra\_ve tenham( ) = 0;

- Toán tử ảo: Toán tử thực chất cũng là một hàm nên có thể tạo ra các toán  
tử ảo trong một lớp.

Hàm toán tử ảo khai báo như sau:

virtual kieu\_tra\_ve operator ki\_hieu\_toan\_tu (cac\_tham\_so);

Các đặc trưng của phương thức ảo:

- Phương thức ảo không thể là các hàm thành viên tĩnh

- Khóa virtual phải viết trước kiểu trả về của phương thức.

- Phải định nghĩa hàm ảo trong lớp nó được khai báo đầu tiên

- Không cần thiết phải ghi rõ từ khóa virtual khi định nghĩa một phương thức  
ảo trong lớp dẫn xuất

- Hàm định nghĩa lại trong lớp dẫn xuất phải có cùng số đối số và kiểu các  
đối số như đã được khai báo trong lớp cơ bản

VD: DaHinh!...

**Câu 8:**

Trình bày vấn đề hình thoi trong đa kế thừa? Ví dụ minh họa?

***Trả lời:***

-Mô tả: Lớp B, C đều kế thừa từ lớp A; lớp D kết thừa lớp B và lớp C. Lớp B và C đều là bản sao của lớp A. Một đối tượng của D nếu cố gắng truy cập vào một phương thức của lớp A, thì trình biên dịch sẽ không biết quyết định sử dụng bản sao nào và sẽ báo lỗi

-dùng virtual để khắc phục

VD: HinhThoiKeThua1 & HinhThoiKeThua2…

**Câu 9:**

Trình bày các tính chất của OOP? Ví dụ minh họa?

***Trả lời:***

- **Đóng gói (Encapsulation)**

Gói thông tin và dữ liệu vào trong 1 lớp (class). Nó được coi như 1 cơ chế che giấu dữ liệu (data hiding) bởi chúng được che giấu đi ở dạng private để đảm bảo các dữ liệu được sử dụng đúng mục đích, đúng cách thông qua các hàm và phương thức dạng public do class cung cấp

- **Kế thừa (Inheritance)**

Nó cho phép xây dựng 1 lớp mới dựa trên các định nghĩa của lớp đã  
có. Lớp cha có thể chia sẻ dữ liệu cho các phương thức lớp con. Các lớp con k cần định nghĩa lại và có thể mở rộng các thành phần kế thừa cũng như thêm các thành phần mới. Tái sử dụng mã nguồn 1 cách tối ưu, tận dụng được mã nguồn

- **Đa hình(Polymorphism)**

Hiện tượng các đối tượng thuộc lớp khác nhau có thể hiểu cùng 1  
thông điệp theo các cách khác nhau

- **Trừu tượng(Abstraction)**

Chọn ra các thuộc tính, phương thức của đối tượng cần cho việc giải quyết bài toán đang lập trình mà k cần sd hết các thuộc tính

**Câu 10:**

Trình bày các thành phần của một lớp? Ví dụ minh họa?

Một class sẽ gồm những thành phần dữ liệu (**thuộc tính hoặc property**) và phương thức (**hàm thành phần hoặc method**)

VD: AnPham

**Câu 11:**

Phân biệt tham trị (Value Type) và tham biến (Reference Type)? Minh họa cách sử dụng các kiểu tham số này?

***Trả lời:***

- **Value:**

**+** các tham số cục bộ là bản sao của các đối số được truyền vào

+ các thay đổi được thực hiện trong hàm đối với các biến này không ảnh hưởng đến giá trị gốc

VD:

void doubleTheValue(int a)

{  
a=a\*2;  
}  
int main()

{

int a = 5;

cout<<”Value of a before calling double function “<<a<<”\n”;

doubleTheValue( a);

cout<<”Value of a after calling double function “<<a<<”\n”;

}

 Kết quả của hai lần in ra giá trị a đều là 5. Vì: giá trị của a đượctruyền trong hàm và giá trị của a sau khi gán cho a\*2 là hai giá trị trỏ đến các địa chỉ khác nhau

- **Reference:** cả biến được truyền vào hàm và biến được sử dụng trong hàm đều trỏ đến một ô nhớ

**+** tham số cục bộ là tham chiếu đến ô nhớ lưu trữ các đối số truyền vào ban đầu

+ các thay đổi đối với các biến này trong hàm sẽ ảnh hưởng đến bản gốc

VD:  
void doubleTheValue(int &a)

{  
a=a\*2;  
}  
int main()

{

int a = 5;

cout<<”Value of a before calling double function “<<a<<”\n”;

doubleTheValue( a);

cout<<”Value of a after calling double function “<<a<<”\n”;

}

**Câu 12:**

Giải thích ý nghĩa từ khóa this? Ví dụ minh họa?

***Trả lời:***

**-**Mọi đối tượng có thể truy cập vào địa chỉ của nó thông qua con this

**-**Bên trong hàm thành viên, con trỏ this được sử dụng để tham chiếu đến đối tượng đang gọi

**-**Sử dụng con trỏ this để phân biệt giữa biến của lớp với biến cục bộ của phương thức có cùng tên. VD: hàm khởi tạo có tham số có tên các tham số trùng tên với các thuộc tính của class

VD: TamGiac…

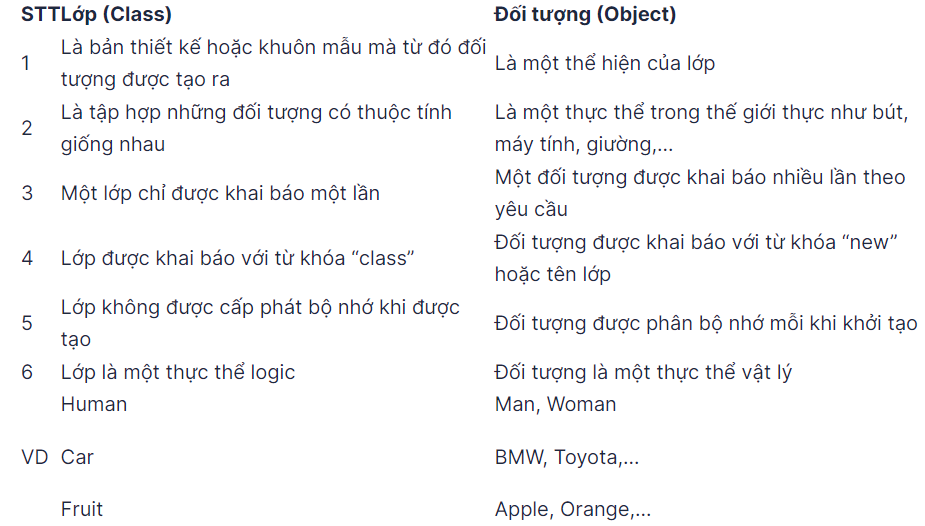
**Câu 13:** Phân biệt khái niệm class và object, cho ví dụ minh họa?

***Trả lời:***

**- Sự khác nhau giữa đối tượng và lớp:**

+ Lớp như là một khuôn mẫu, đối tượng là một thực thể thể hiện dựa trên khuôn mẫu đó

+ Ví dụ: Lớp Car (các thuộc tính: màu, biển số, hãng,… Phương thức: chạy, dừng,…) thì các đối tượng của lớp Car này là xe Aufi, Volvo,…



**Câu 14:**

Phân biệt các khái niệm và sự khác nhau giữa nạp chồng (overload) và ghi đè (override).

***Trả lời:***

|  |  |
| --- | --- |
| **Overloading** | **Overriding** |
| Thể hiện đa hình tại **compile time** | Thể hiện đa hình tại **runtime** |
| Thêm hành vi cho phương thức | Thay đổi hành vi hiện tại của phương thức |
| Có thể khác nhau về số lượng và kiểu dữ liệu của tham số | Số lượng và kiểu dữ liệu của tham số phải giống nhau |
| Xảy ra trong cùng một class | Xảy ra ở 2 class có quan hệ kế thừa |

**Câu 15:**

Hàm bạn được sử dụng trong trường hợp nào? Ví dụ minh họa?

- ***Một hàm friend*** của một lớp được khai báo bên trong lớp và được định  
nghĩa bên ngoài phạm vi của lớp đó, có quyền truy cập đến các thành viên  
private hoặc protected của một lớp. Tuy nhiên nó không phải là hàm thành  
viên của lớp.

- ***Một lớp friend*** có thể truy cập các thành viên private hoặc protected của  
lớp khác mà lớp đó được khai báo là friend.

**-**Nếu lớp A là bạn của lớp B thì B không tự động trở thành bạn của lớp A

**-**Không được kế thừa

-Muốn truy cập đến thuộc tính private bên ngoài class mà không dùng hàm  
bạn thì dùng hàm get

-Muốn truy cập đến phương thức khai báo private bên ngoài class bằng cách  
sử dụng phương thức ảo

-Các trường hợp bắt buộc sử dụng hàm bạn: nạp chồng toán tử nhập, xuất;  
nạp chồng toán tử hai ngôi với hai tham số truyền vào khác kiểu. VD: công một  
phân số với một số nguyên

VD: PhanSo , Fraction....