Câu 1: Phân biệt các phạm vi truy cập private, protected, public và cho ví dụ minh họa.

Phạm vi truy cập	Lớp hiện tại	Lớp con	Lớp khác
Private	Có	Không	Không
Protected	Có	Có	Không
Public	Có	Có	Có

- Phạm vi truy cập public: có thể được truy cập ở tất cả các phần của chương trình. Các thành phần public tạo nên giao diện của một Lớp.
- Phạm vi truy cập private: chỉ có thể được truy cập bên trong phạm vi của lớp, bởi các hàm thành viên (phương thức) của một lớp và không thể được truy cập từ bên ngoài lớp. Phần private ẩn đi phần thực thi.
- Phạm vi truy cập protected: Là sự kết hợp giữa public và private, có thể được truy cập ở lớp dẫn xuất trong khi vẫn giới hạn quyền truy cập từ người dùng.

Câu 2: Nêu khái niệm Constructor và Destructor. Phân biệt Constructor mặc định và Constructor khác.

- **Constructor** (hàm tạo) là một phương thức đặc biệt được tự động gọi khi tạo ra một đối tượng mới trong lập trình hướng đối tượng. Mục đích chính của Constructor là **khởi tạo các thuộc tính** của đối tượng với giá trị ban đầu phù hợp.
- **Destructor** (hàm hủy) là một phương thức đặc biệt được tự động gọi khi hủy một đối tượng trong lập trình hướng đối tượng. Mục đích chính của Destructor là **giải phóng các tài nguyên được sử dụng bởi đối tượng**, chẳng hạn như bộ nhớ được cấp phát.

Loại constructor	Đặc điểm	Mục đích	Ví dụ
Constructor mặc định	- Không có tham số. - Được gọi tự động khi tạo đối tượng mà không truyền tham số cụ thể.	-Khởi tạo đối tượng với các giá trị mặc định.	Person()
Constructor có tham số	- Có thể có một hoặc nhiều tham số. - Được gọi khi tạo đối tượng với	Mục đích là khởi tạo các thuộc tính của đối tượng với giá trị cụ thể được	Person("Alice", 30)

Loại constructor	Đặc điểm	Mục đích	Ví dụ
	truyền tham số cụ thể.	truyền vào.	
Constructor sao chép	- Có một tham số là tham chiếu đến đối tượng hiện có.	- Được sử dụng để tạo một bản sao của một đối tượng hiện có.	person3 = Person(person2)

Câu 3: Ý nghĩa và mục đích của các hàm get/set trong một lớp.

- **Hàm get** và **hàm set** là những phương thức (method) đặc biệt được sử dụng trong lập trình hướng đối tượng (OOP) để truy cập và thao tác với các thuộc tính (attribute) của một lớp.
- Ý nghĩa:
  + Hàm get và hàm set đóng vai trò như một "cổng" để truy cập và thao tác với các thuộc tính của lớp.
  + Việc sử dụng hàm get và hàm set giúp đảm bảo tính an toàn và bảo mật cho dữ liệu của lớp.
  + Thay vì truy cập trực tiếp vào thuộc tính, lập trình viên nên sử dụng hàm get và hàm set để kiểm soát việc truy cập và thay đổi dữ liệu.
- Mục đích:
  + Hàm **get**: **Lấy giá trị** của một thuộc tính.
  + Hàm **set**: **Thay đổi giá trị** của một thuộc tính.
  + **Tăng tính an toàn:** Hàm get và hàm set cho phép kiểm tra và xác thực giá trị trước khi gán cho thuộc tính, giúp ngăn chặn lỗi và dữ liệu không hợp lệ.
  + **Tăng tính bảo mật:** Hàm get và hàm set có thể được sử dụng để ẩn đi các chi tiết thực thi của lớp, chỉ cung cấp cho người dùng giao diện truy cập và thao tác đơn giản.
  + **Tăng tính linh hoạt:** Hàm get và hàm set có thể được sử dụng để bổ sung logic xử lý bổ sung khi truy cập hoặc thay đổi giá trị thuộc tính.

Câu 4: Anh/Chị trình bày sự hiểu biết và cho ví dụ minh hoạ về khái niệm class (lớp) và object (đối tượng) trong lập trình hướng đối tượng. (HK1 2022 2023)

- Class (lớp): là một mô tả trừu tượng của nhóm các đối tượng cùng bản chất.
  + Một lớp bao gồm các thành phần dữ liệu (thuộc tính) và các phương thức (hàm thành phần).

+ Thực chất là 1 kiểu dữ liệu do người sử dụng định nghĩa
+ Có thể kế thừa từ các lớp khác
+ Chỉ được khai báo 1 lần
+ Được sử dụng để tạo ra các đối tượng cụ thể.
+ Ví dụ: Lớp Animal chứa các thuộc tính name, age và các phương thức eat(), sleep(), move()

- Object (đối tượng): là một thể hiện cụ thể cho những mô tả trừu tượng đó.
  + Được tạo ra từ một lớp.
  + Có trạng thái riêng biệt được xác định bởi các giá trị của thuộc tính.
  + Có thể khai báo nhiều đối tượng
  + Ví dụ: Đối tượng con chó được tạo ra từ lớp Animal. Đối tượng này có các thuộc tính như name(Rex), age(5), và có thể thực hiện các hành bi như eat(), sleep(), move()...

Câu 5: Phân biệt khái niệm overload và override. (HK3 2021 2022)

	Overload	Override
Mục đích	Cung cấp nhiều cách thức thực thi cho cùng một chức năng	Thay đổi hành vi của phương thức được kế thừa
Phạm vi	Trong cùng một lớp	Giữa lớp con và lớp cha
Thay đổi	Số lượng tham số hoặc kiểu dữ liệu tham số	Hành vi của phương thức
Điều kiện	Không có	Phương thức trong lớp con phải cùng tên, kiểu trả về và kiểu dữ liệu tham số như phương thức được kế thừa trong một lớp cha.

Ví dụ	Int add(int a, int b);
	Int add(int a, int b, int c);
	--- 	
	class Meo {
	public:
	void Keu()
	{
	cout << “meo meo”;
	}
	
	void Keu(int n)
	{
	for(int i = 0; i < n; i++)
		Lớp Animal: void sound()
		Lớp Dog: void sound()
		--- 	
		class Dongvat
		{
		public:
		Virtual void keu() = 0;
		};
		
		class Meo : public Dongvat
		{
		public:
		void keu()

	cout << “Meo Meo”;
	};
		{
		cout << “meo meo”;
		}
		};

Câu 6: Phân biệt phạm vi truy xuất các thành phần theo chiều dọc và chiều ngang trong kế thừa của lập trình hướng đối tượng. (HK3 2021 2022)

		Truy xuất các thành phần theo chiều dọc	Truy xuất các thành phần theo chiều ngang
Khái niệm		Liên quan đến mức độ truy cập các thành phần **bên trong cùng một lớp** hoặc **lớp con** của nó.	Liên quan đến mức độ truy cập các thành phần **giữa các lớp không có mối quan hệ kế thừa**.
Mức độ truy cập	Public	Thành phần có thể được truy cập từ bất kỳ nơi nào trong chương trình.	Thành phần có thể được truy cập từ bất kỳ lớp nào trong chương trình.
	Protected	Thành phần chỉ có thể được truy cập từ bên trong lớp hoặc các lớp con kế thừa nó.	Thành phần chỉ có thể được truy cập từ lớp khai báo nó và các lớp con của nó, **không** bao gồm các lớp không liên quan.
	Private	Thành phần chỉ có thể được truy cập từ bên trong lớp mà nó được khai báo.	Thành phần chỉ có thể được truy cập từ bên trong lớp mà nó được khai báo, **không** truy cập được từ các lớp khác.

Câu 7: Phân biệt chuyển kiểu bằng phương thức thiết lập và chuyển kiểu bằng toán tử chuyển kiểu (HK2 - 2022 2023)

	Chuyển kiểu bằng phương thức thiết lập	Chuyển kiểu bằng toán tử chuyển kiểu
Mục đích	Khởi tạo giá trị cho thuộc tính của đối tượng.	Ép buộc chuyển đổi giá trị của một biến.

Phạm vi	Bên trong phương thức thiết lập của lớp.	Có thể sử dụng ở bất kỳ nơi nào trong chương trình.
Kiểm soát	Cao hơn, có thể thực hiện các thao tác bổ sung.	Thấp hơn, chỉ thực hiện chuyển đổi đơn giản.
Đọc code	Dễ hiểu hơn, thể hiện rõ ràng ý định chuyển đổi.	Khó hiểu hơn, có thể gây nhầm lẫn.
Hiệu suất	Có thể chậm hơn do thực hiện các thao tác bổ sung.	Nhanh hơn do chỉ thực hiện chuyển đổi đơn giản.

Câu 8: Phân biệt các phạm vi truy cập private, protected, public trong tính kế thừa

Phạm vi truy cập các thuộc tính, phương thức ở lớp cha	Phạm vi thừa kế		
	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private	Không	Không	Không

- Thành phần private ở lớp cha thì không truy xuất được ở lớp con.
- Kế thừa public: Lớp con kế thừa public từ lớp cha thì các thành phần protected của lớp cha trở thành protected của lớp con, các thành phần public của lớp cha trở thành public của lớp con.
- Kế thừa private: Lớp con kế thừa private từ lớp cha thì các thành phần protected và public của lớp cha trở thành private của lớp con
- Kế thừa protected: Lớp con kế thừa protected từ lớp cha thì các thành phần protected và public của lớp cha trở thành protected của lớp con

Câu 9: Constructor là gì ? Có bao nhiêu loại ? Lấy ví dụ từng loại ? Hãy nêu 3 đặc điểm khác nhau giữa Constructor và Destructor (HK2 - 2022 2023)

- **Constructor** (hàm tạo) là một phương thức đặc biệt được tự động gọi khi tạo ra một đối tượng mới trong lập trình hướng đối tượng. Mục đích chính của

Constructor là **khởi tạo các thuộc tính** của đối tượng với giá trị ban đầu phù hợp.

- Có 3 loại constructor:
  + Constructor mặc định: PhanSo a()
  + Constructor có tham số: PhanSo b(1, 2)
  + Constructor sao chép: PhanSo c = PhanSo(b)
- 3 đặc điểm khác nhau
  + Constructor :
    - Mục đích: khởi tạo trạng thái
    - Có thể có tham số để truyền giá trị khởi tạo cho các thuộc tính đối tượng.
    - Có thể gọi trực tiếp
  + Destructor:
    - Mục đích: giải phóng tài nguyên
    - Không có tham số
    - Không thể gọi trực tiếp

Câu 10: Nêu khái niệm về sự kế thừa và những ưu điểm của kế thừa trong việc lập trình. Cho ví dụ minh họa.

- Khái niệm về thừa kế.
  + Tính thừa kế cho phép các lớp được xây dựng trên các lớp đã có.
  + Tính thừa kế được dùng để biểu diễn mối quan hệ đặc biệt hóa – tổng quát hóa giữa các lớp.Trong đó, lớp dẫn xuất thừa kế lớp cơ sở.
  + Lớp dẫn xuất thừa kế tất cả các thành phần(thuộc tính và phương thức) của lớp cơ sở, kể cả các thành phần mà lớp cơ sở được thừa kế.
- Những ưu điểm của kế thừa trong việc lập trình:
  + Có thể dùng tính thừa kế để phát triển khả năng của chương trình bằng cách xây dựng thêm các lớp dẫn xuất từ các lớp đã có, trong đó có thêm các thuộc tính và phương thức mới. Ngoài ra, ta cũng có thể xây dựng các lớp mới có thuộc tính là đối tượng của các lớp đã có. Như vậy, sẽ nhận được một dãy các lớp ngày càng hoàn thiện và có nhiều khả năng hơn.
  + Có thể dùng tính thừa kế để sửa đổi, bổ sung, nâng cấp chương trình. Bằng cách xây dựng thêm các lớp dẫn xuất để thực hiện các bổ sung sửa đổi thay vì phải sửa chữa trên các lớp đã có.
  + Tính thừa kế cũng được dùng để thiết kế bài toán theo hướng từ khái quát đến cụ thể, từ chung đến riêng. Đầu tiên đưa ra các lớp để mô tả những đối tượng chung, sau đó dẫn xuất tới các đối tượng ngày một cụ thể hơn.

+ Tính thừa kế cũng được dùng trong việc thiết kế bài toán chung và bài toán bộ phận. Khi đó ta có thể định nghĩa các lớp cho các bài toán bộ phận và lớp cho bài toán chung sẽ là lớp dẫn xuất của các lớp trên.

Câu 11: Hàm thuần ảo là gì? Lớp trừu tượng là gì? Cho ví dụ minh họa.

- Hàm thuần ảo:

- Là phương thức ảo không có nội dung.
- Phương thức thuần ảo có ý nghĩa cho việc tổ chức sơ đồ phân cấp các lớp, nó đóng vai trò chừa sẵn chỗ trống cho các lớp con điền vào với phiên bản phù hợp.

- Lớp trừu tượng:

- Là lớp cơ sở không có đối tượng nào thuộc chính nó.
- Khi lớp có phương thức thuần ảo, lớp trở thành lớp cơ sở trừu tượng.

Ví dụ minh họa:

```
class DongVat //lớp trừu tượng
{
public:
   virtual void Keu()=0; //hàm thuần ảo
};

class Meo:public DongVat
{ public:
   void Keu()
   {
      cout<<"meo meo";
   }
};

class Cho:public DongVat
{ public:
   void Keu()
   {
      cout<<"gau gau";
   }
};
```