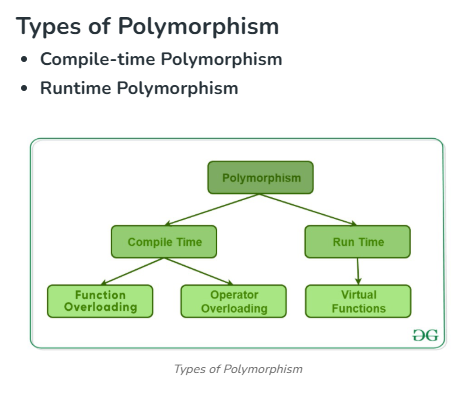
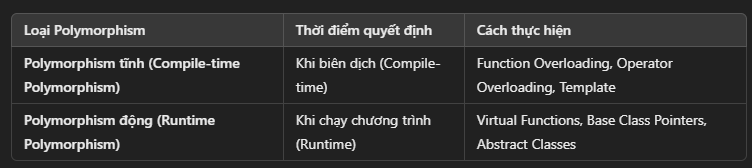
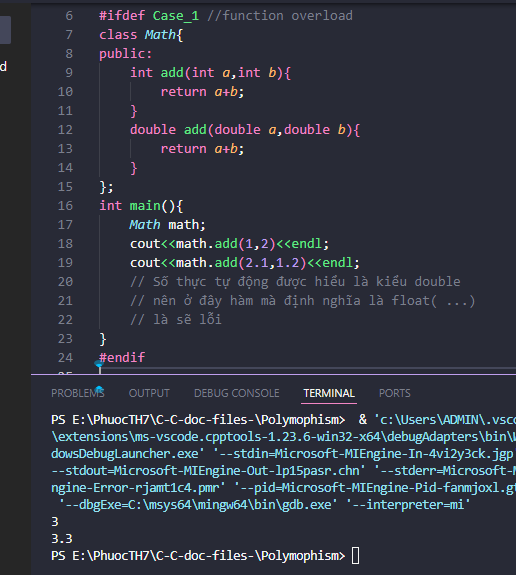
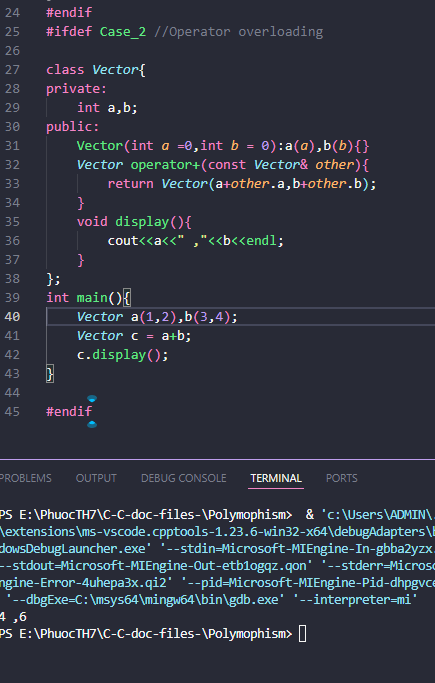
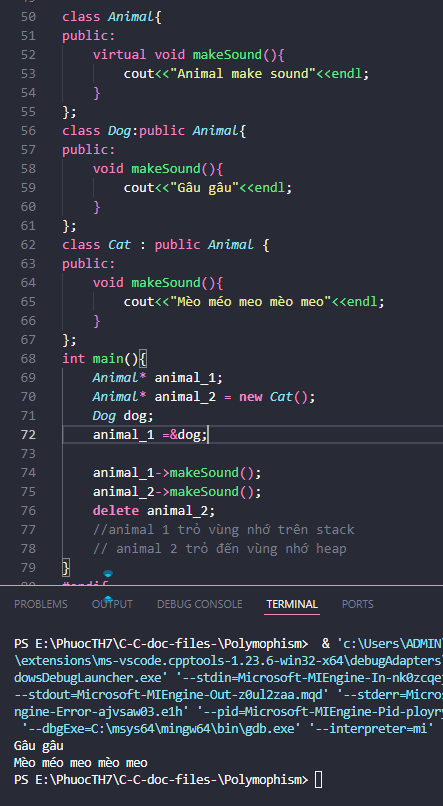
1. Polymorphism:

* The word “polymorphism” means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. A real-life example of polymorphism is a person who at the same time can have different characteristics. A man at the same time is a father, a husband, and an employee. So the same person exhibits different behavior in different situations. This is called polymorphism. Polymorphism is considered one of the important features of Object-Oriented Programming..



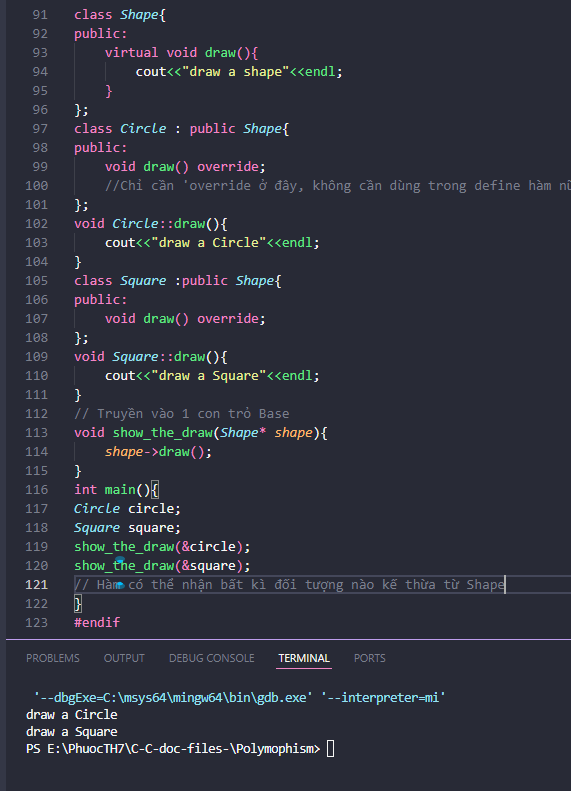
* Có 2 kiểu đa hình trong C++ là **đa hình compile time (tĩnh)**  và **đa hình Runtime (Động)**



* Đa hình tĩnh giúp chương trình chạy nhanh hơn vì nó được quyết định tại thời điểm biên dịch.
* Đa hình động linh hoat hơn, cho phép gọi phương thức lớp con thông qua con trỏ hoặc tham chiếu lớp cha, tốc độ chậm hơn vì phải tìm phương thức phù hợp khi chạy.
* Đa hình tĩnh sử dụng cách hình thức **overloading : function, operator, template**,..
* Đa hình động sử dụng **Virtual functions, base class pointers, abstract classes** \
* Function overload: cùng 1 hàm add nhưng có thể hoạt động với nhiều kiểu dữ liệu khác nhau.
* 
* Operator overloading: toán tử + hoạt động giữa 2 object chứ không + như bình thường.
* 
* Virtual Function( là 1 method của lớp có thể bị ghi đè) Đa hình động. Không có virtual thì chỉ gọi được method lớp cha mà thôi.
* 

1. Base class Pointer:

* Một câu hỏi cũ là tại sao lại dùng con trỏ lớp cơ sở, đơn giản là vì với cách này ta chỉ cần dùng 1 con trỏ ta có thể gọi đến method của nhiều class con khác nhau.
* Dùng 1 con trỏ để quản lý nhiều đối tượng khác nhau.
* Ví dụ bên trên ta cũng đã dùng con trỏ lớp cơ sở để gọi 2 method ở 2 lớp con khác nhau.
* Nói đơn giản thì lớp cha định nghĩa 1 virtual function, các lớp con kế thừa lớp cha và định override lại các method này thì ta dùng con trỏ lớp base để gọi các method ở lớp con.
* Có 1 cách sử dụng con trỏ base hiệu quả là **Dùng con trỏ base với tham sô hàm**

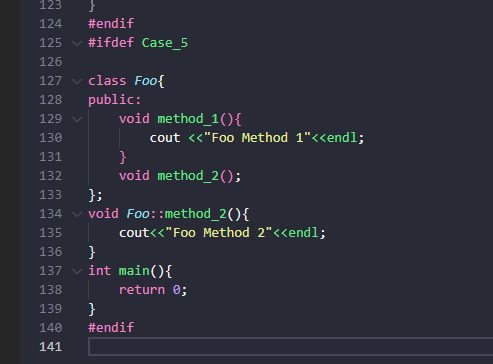


1. **Virtual Function:**

* Hàm ảo là 1 method trong lớp cơ sở, được khai báo với từ khóa virtual và cho phép lớp dẫn xuất có thể ghi đề override nó.
* Mục đích: Đảm bảo đa hình động.
* Khi có hàm ảo, nếu ta gọi phương thức thông qua con trỏ base thì sẽ gọi đúng method lớp dẫn xuất thay vì lớp cơ sở.

1. Virtual Table – Vtable:

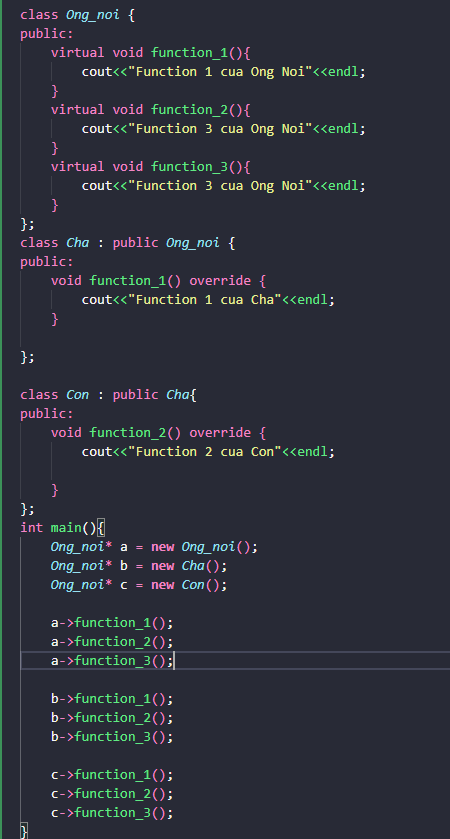
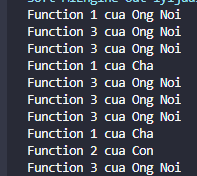
* Để hiểu về Vtable, trước hết ta phải hiểu về khái niệm “**static dispatch**” :

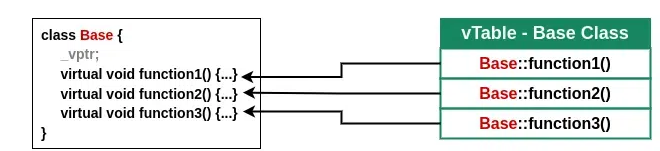


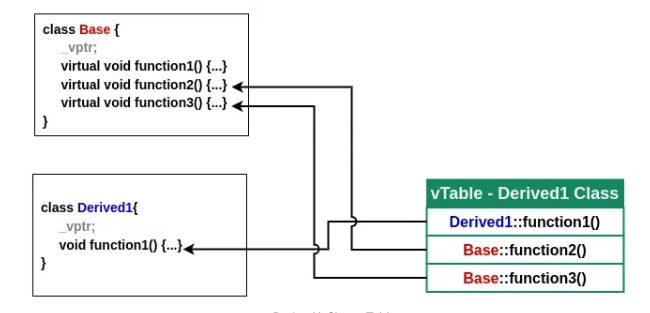
* + Khi không dùng hàm ảo (virtual), thì tại thời điểm biên dịch compiler sẽ lưu địa chỉ của các method trong class và jump đến khi hàm được gọi.
  + Ta phải hiểu rằng các hàm method\_1() và method\_2() chỉ có 1 địa chỉ tương ứng duy nhất và được sử dụng chung cho tất cả các đối tượng của class Foo.

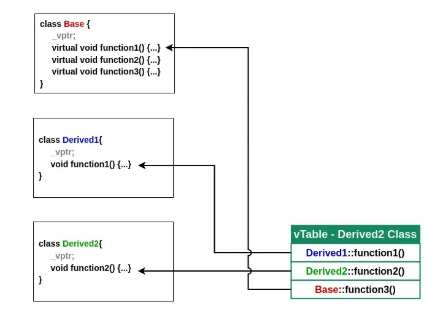
**What is VTable ?**

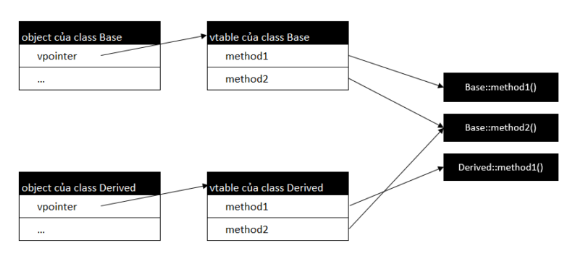
* Là 1 bảng chứa Function pointers được tạo bởi compiler để hỗ trợ đa hình động. Khi một class có virtual function, compiler sẽ tạo 1 Vtable cho class đó ( lưu lý virtual table thuộc về class chứ không thuộc về object) và trong Vtable là địa chỉ của các Virtual function.
* Còn riêng mỗi object của các class chứa virtual function hoặc kế thừa virtual function sẽ có 1 pointer ẩn là **Vptr** trỏ đến Virtual Table của class.
* The Vtable has one entry for each virtual function accessible by the class. These entries are pointers to the **most derived function** that the current object should call.
* Vtable có 1 mục cho mỗi virtual function trong class, mỗi mục này là con trỏ trỏ đến hàm ghi đè ở class kế thừa gần nhất. Xem ví dụ sau là hiểu.
* Vtable của Class Ông nội sẽ có địa chỉ của 3 function 1 2 3 của lớp ông nội, cái này ko cần bàn nhiều
* Class Cha ghi đè function 1 của Ông Nội, Nghĩa là VTable của lớp Cha sẽ 1 con trỏ trỏ đến Function 1 của class Cha, còn 2 con trỏ trỏ đến function 2 3 của Class ông Nội. -> Most derived function.
* Class Con kế thừa từ Cha và ghi đề Function 2 của ông nội, lúc này Vtable của Con sẽ gồm: 1 con trỏ trỏ đến function\_1 của class Cha, con trỏ Function 2 trỏ đến function 2 Class Con, 1 con trỏ trỏ đến function 3 của lớp ông Nội.





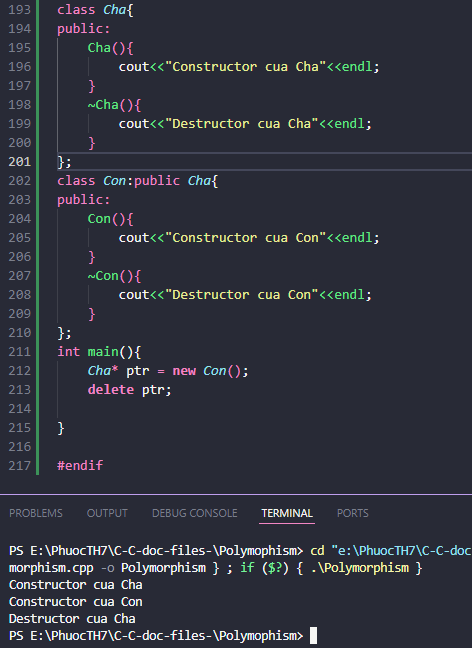
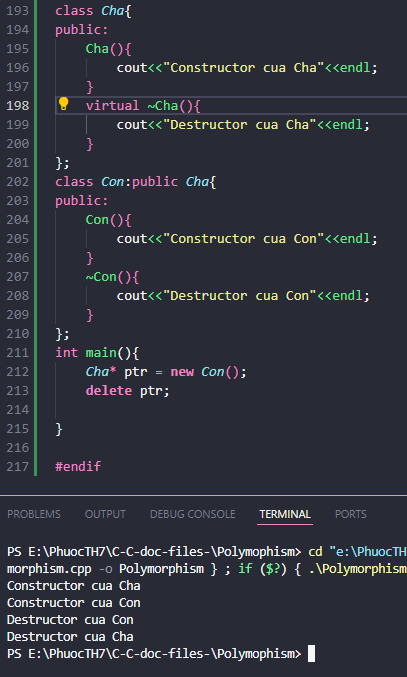




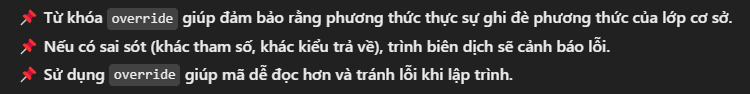


Còn VPointer như ta đã nói ở trên, nó thuộc về object, con trỏ này trỏ đến Vtable của Class, và mỗi con trỏ của Vtable sẽ trỏ đến function ở class gần nhất nó được kế thừa.

**Virtual Destructors**

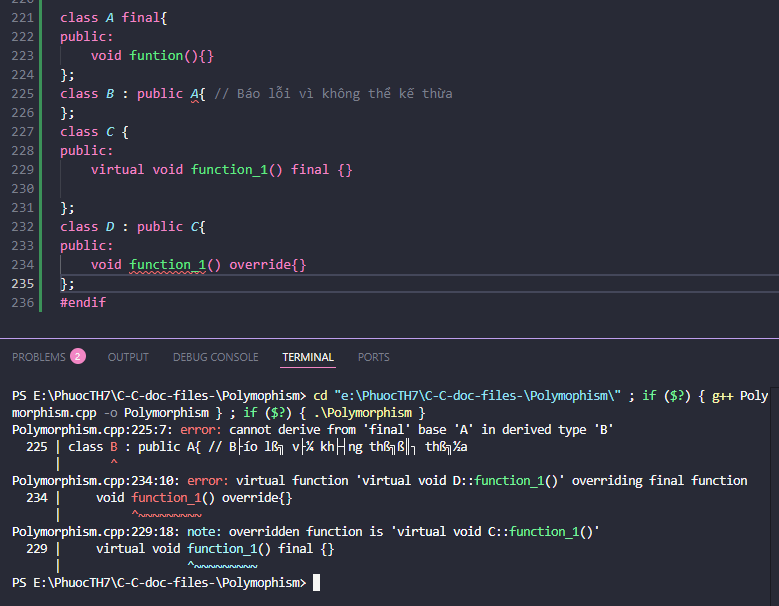
* Như đã bàn ở phần kế thừa, Virtual destructor là một destructor trong lớp base nhằm đảm baorawngf khi xóa đối tượng thông qua con trỏ base thì destructor lớp con sẽ được gọi chính xác.
* Nếu không có virtual, destructor lớp con sẽ không được gọi -> rò rỉ bộ nhớ.
* ****Bản chất ở đây là đẻ ra VTable cho lớp Con, khi delete obj; chương trình sẽ tra cứu và gọi destructor của lớp Con trước ;v

**Override specficer:**

****

**Final Specifier:**

* Từ khóa **final** trong C++ có thể dùng trong 2 trường hợp:
  + Dùng với virtual Method: Chống ghi đè
  + Dùng với Class: Chống kế thừa.

****