C++

POLYMORPHISM

Polymorphism

- The term "Polymorphism" is the combination of "poly" + "morphs" which means many forms.

- A real-life example of polymorphism: A lady behaves like a teacher in a classroom, mother or daughter in a home and customer in a market. Here, a single person is behaving differently according to the situations.

- Another example is: A shape has different implementation of area function depending on whether the shape is circle or rectangle or triangle etc. Here area() function is called on shape variable according to the what type of shape it is.

Polymorphism is achieved using -

- Function binding
- Generic pointers and Virtual keyword
- V-table and V-pointer
- Pure virtual function
- Abstract class
- Virtual Destructor

Function binding: Process of deciding which function to be called at the time of execution of code.

▪ Two types of function binding
    1. Compile time binding: Static binding
    2. Run-time binding: Dynamic binding

▪ Static binding happens when all information needed to call a function is available at the compile-time.
▪ Dynamic binding happens when all information needed for a function call cannot be determined at compile-time.

Static binding means Compile time binding

- Compiler decides, at the time of compilation, which function to be called at the point of invocation.

- Compiler decisions are based on types of objects or variables only

- Example:

  Function overloading: Best match on the basis of types of arguments
  Invocations using objects: Based on type of object
  Invocations using pointers: Based on type of pointer
  Invocations using references: Based on type of reference

Dynamic binding means Run-time binding

▪ Compiler does NOT decide the actual function to be called at the point of invocation.

▪ Compiler simply substitutes a code at the point of invocation. This code when executed decides the actual function to be called

▪ Such calls are called as polymorphic function calls

Polymorphic function call :

▪ A call polymorphic only if it satisfies following two conditions

1. Functions should be declared as virtual

2. Call should be made using either a generic pointer or a generic reference

▪ Polymorphic calls are not resolved by compiler. It simply substitutes a code at the point of invocation

▪ This code when executed, decides the actual function to be called.

▪ The actual function to be called is decided by the type of object pointed by the pointer and NOT by the type of pointer itself. The type of object is decided at run-time. This is polymorphism

Note that…

▪ Polymorphism is expensive because memory requirement for vTable per class and vPtr per object.

▪ It also may reduce the performance, but is very negligible.

▪ Constructors cant be virtual, because the vPtr gets initialized at the time of construction.

▪ Virtual function should be non static member function of the base class.

▪ Virtual functions cannot be used as a friend function.

▪ If a function is declared as virtual in the base class then ,it will be taken as virtual in the derived class even if the keyword virtual is not used in derived.

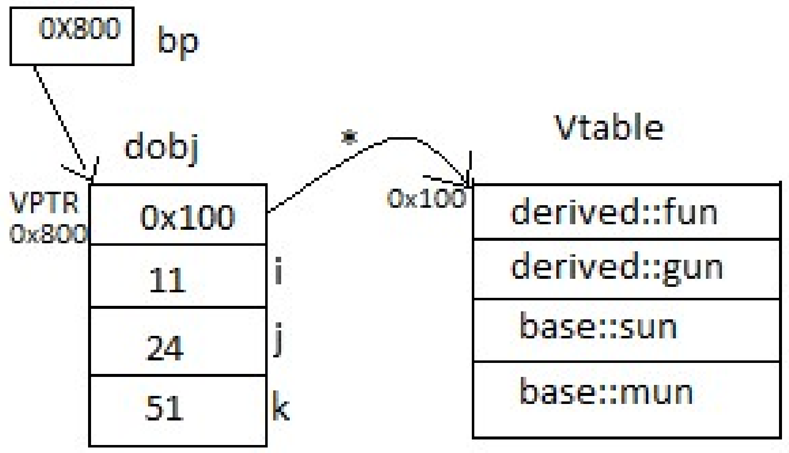
▪ Polymorphism can also be achieved using references.

Pure virtual function, Abstract Class

- Virtual function without definition is pure virtual function
- Class containing at least one pure virtual function is abstract class
- Abstract class cant be instantiated, its pointers and references can be created
- Class containing only pure virtual functions is pure abstract class, also called as interface
- Derived classes must override pure virtual functions else they are also abstract

Virtual Destructor

- Needed for base class pointer pointing to a dynamically allocated derived class object
- Results into resource leak of derived class if not made virtual
- Use only if needed, else it results into memory overheads in terms of vTable and vPt


0X800
bp
dobj
VPTR
0x800
0x100
11
i
24
j
51
k
*
Vtable
0x100
derived::fun
derived::gun
base::sun
base::mun