CS201- Data Structures
Week 02

Muhammad Rafi
September 06, 2018

Agenda

- Inheritance
- Access Modifier
- Multi-level Inheritance
- Multiple Inheritance
- Virtual Base Class
- Polymorphism
- Virtual Functions
- Pure Virtual Functions
- Abstract Base Class

Agenda

- Vtable and Virtual Functions
- RTTI mechanism
- Conclusion

Inheritance

- Inheritance gives provision to derive a new class from an existing one. A general purpose class functionality for the domain is generally provided in a base class.
- Why use inheritance ?
  - The base class may be used by other programs and you want it original behavior to remain intact.
  - The source code of the base class Is not available to you.
  - Base class may be an abstract base class, which is design to be a base class only.

Base Class Public Inheritance

Accessibility in Public Inheritance

Accessibility	private variables	protected variables	public variables
Accessible from own class?	yes	yes	yes
Accessible from derived class?	no	yes	yes
Accessible from 2nd derived class?	no	yes	yes

Base Class Protected Inheritance

Accessibility in Protected Inheritance

Accessibility	private variables	protected variables	public variables
Accessible from own class?	yes	yes	yes
Accessible from derived class?	no	yes	yes (inherited as protected variables)
Accessible from 2nd derived class?	no	yes	yes

Base Class Private Inheritance

Accessibility in Private Inheritance

Accessibility	private variables	protected variables	public variables
Accessible from own class?	yes	yes	yes
Accessible from derived class?	no	yes (inherited as private variables)	yes (inherited as private variables)
Accessible from 2nd derived class?	no	no	no

Inheritance (Constructor/Destructor)

- The order of constructor and destructor is an important concept in OOP.

Multi-Level Inheritance

- In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as multilevel inheritance.

```
class A
{  ... .. ... };
class B: public A
{ ... .. ... };
class C: public B
{ ... ... ... };
```

- Virtual Destructor (see CS201-ExCodeOOP-7)

Multiple Base Classes

- A class can inherit from more than one class, the order of inheritance is important to object creation process.
- Constructors are executed in their order of derivation.
- Destructors are executed in reverse order of derivation. (see CS201-ExCodeOOP-6)
- Multiple Inheritance:
  - Constructors are called in order of derivation, left to right
  - Destructors are called in reverse order, right to left.

Deadly Diamond of Death (DDD)

- In cases where the diamond is not avoidable, using virtual inheritance. The biggest caveat, however, with virtual bases, is that the constructor for the virtual base must be called by the most derived class, meaning that a class that derives virtually has no control over the constructor parameters.
- Always use the diamond if you are explicit about which base you want to use.
- The solution is virtual inheritance, the presence of a virtual base tends to incur a performance/space penalty on casting through the chain

Virtual Base Classes

- When two or more classes are derived from a common base class, you can prevent multiple copies of the base class from being present in an object derived from those classes by declaring the base class as virtual when it is inherited.
- Ambiguity is main issue of such inheritance.
- Deriving a class as virtual base class can only maintain a single base memory for the derived object.

Polymorphism

- Polymorphism is an important attribute of OOP, specially in C++ it is quite evolved and rigor.
- Compile-time polymorphism is achieved by overloading functions and operators.
- Run-time polymorphism is accomplished by using inheritance and virtual functions.

Early Binding Vs. Late Binding

- Early Binding
  - Early binding refers to events that occur at compile time. In essence, early binding occurs when all information needed to call a function is known at compile time.
  - Examples of early binding include normal function calls (including standard library functions), overloaded function calls, and overloaded operators.
  - The main advantage to early binding is efficiency. Because all information necessary to call a function is determined at compile time, these types of function calls are very fast.

Early Binding Vs. Late Binding

- Late Binding
  - Late binding refers to function calls that are not resolved until run time. Virtual functions are used to achieve late binding.
  - RTTI is a mechanism to handle late binding.
  - The main advantage to late binding is flexibility. Unlike early binding, late binding allows you to create programs that can respond to events occurring while the program executes without having to create a large amount of "contingency code."
  - Late binding can make for somewhat slower execution times.

Virtual Function

- A virtual function is a member function that is declared within a base class and redefined by a derived class.
- To create a virtual function, precede the function's declaration in the base class with the keyword virtual.
- When a class containing a virtual function is inherited, the derived class redefines the virtual function to fit its own needs.
- In essence, virtual functions implement the "one interface, multiple methods" philosophy that underlies polymorphism.

Virtual Function Hierarchical

- When a function is declared as virtual by a base class, it may be overridden by a derived class. However, the function does not have to be overridden.
- When a derived class fails to override a virtual function, then when an object of that derived class accesses that function, the function defined by any of the base class is used.

Pure Virtual Function

- A pure virtual function is a virtual function that has no definition within the base class.

```
virtual type func-name(parameter-list) = 0;
```

- When a virtual function is made pure, any derived class must provide its own definition.

Abstract Base Class

- A class that contains at least one pure virtual function is said to be abstract.
- No objects of an abstract class can be created.
- Instead, an abstract class constitutes an incomplete type that is used as a foundation for derived classes.
- It can only provide interfaces to all derived classes.

VTable and Virtual Functions

- The late binding and run-time polymorphism requires to decide a lot of things on the fly.
- In any class if there is a pure virtual function it will be ABC and a VTable is associated with such code to determine the actual implementation of the virtual function.

Conclusion

- Inheritance and Polymorphism is quite complex and requires better understanding to utilize it effectively.
- Practice is required with OOA&D