1. Making private member inheritable: Use protected modifier
2. Hybrid Inheritance:
3. Virtual base classes:
4. Abstract classes:
5. Constructors in derived classes:
6. Operators overloading: (), [], \*, new, <<, >>, &, |
7. Virtual functions (need, syntax and use):

Need:

Without virtual compile time/early binding is done.

To call the correct function while using pointers/references for derived class objects.

If the base class is Base, and a derived class is Der, you can have a Base \*p pointer which points to an instance of Der. When you call p->foo();, if foo is not virtual, then Base's version of it executes, ignoring the fact that p actually points to a Der. If foo is virtual, p->foo() executes the "leafmost" override of foo, fully taking into account the actual class of the pointed-to item.

Syntax:

1. Virtual functions cannot be static.
2. They can be friend function of some other class.
3. Prototype must be same for base and derived class.
4. Function call is resolved in runtime (late binding or dynamic linkage) for virtual functions. Also called runtime polymorphism.
5. Virtual constructor cannot be created but virtual destructor can be.
6. Pointers to base and derived classes:
7. Overloading through friend functions:
8. Data conversion using objects:
9. Conversion of one object to another:
10. Virtual constructor and destructor:

Constructor cannot be virtual.