- *虚函数*: 定义一个函数为虚函数，不代表函数为不被实现的函数。定义他为虚函数是为了允许用基类的指针来调用子类的这个函数。
- *纯虚函数*: 定义一个函数为纯虚函数，才代表函数没有被实现。定义纯虚函数是为了实现一个接口，起到一个规范的作用，规范继承这个类的程序员必须实现这个函数。

虚函数

```
class A
{
public:
    virtual void foo()
    {
        cout<<"A::foo() is called"<<endl;
    }
};
class B:public A
{
public:
    void foo()
    {
        cout<<"B::foo() is called"<<endl;
    }
};
int main(void)
{
    A *a = new B();
    a->foo();   // 在这里，a虽然是指向A的指针，但是被调用的函数(foo)却是B的!
    return 0;
}
```

- 这个例子是虚函数的一个典型应用，通过这个例子，也许你就对虚函数有了一些概念。它虚就虚在所谓"推迟联编"或者"动态联编"上，一个类函数的调用并不是在编译时刻被确定的，而是在运行时刻被确定的。由于编写代码的时候并不能确定被调用的是基类的函数还是哪个派生类的函数，所以被成为"虚"函数。

  虚函数只能借助于指针或者引用来达到多态的效果。

纯虚函数

- 定义： 纯虚函数是在基类中*声明*的虚函数，它在基类中没有定义，但要求任何派生类都要定义自己的实现方法。在基类中实现纯虚函数的方法是在函数原型后加 =0:

```
    virtual void funtion1()=0;
```

- 引入原因

1. 为了方便使用多态特性，我们常常需要在基类中定义虚拟函数。

2. 在很多情况下，基类本身生成对象是不合情理的。例如，动物作为一个基类可以派生出老虎. 孔雀等子类，但动物本身生成对象明显不合常理。

为了解决上述问题，引入了纯虚函数的概念，将函数定义为纯虚函数（方法：`virtual ReturnType Function()= 0;`），则编译器要求在派生类中必须予以重写以实现多态性。同时含有纯虚拟函数的类称为**抽象类**，它**不能生成对象**。这样就很好地解决了上述两个问题。

声明了纯虚函数的类是一个抽象类。所以，用户不能创建类的实例，只能创建它的派生类的实例。

定义纯虚函数的目的在于，**使派生类仅仅只是继承函数的接口。**

纯虚函数的意义，让所有的类对象（主要是派生类对象）都可以执行纯虚函数的动作，但类无法为纯虚函数提供一个合理的默认实现。所以类纯虚函数的声明就是在告诉子类的设计者，"你必须提供一个纯虚函数的实现，但我不知道你会怎样实现它"。

抽象类的介绍

- 抽象类是一种特殊的类，它是为了抽象和设计的目的为建立的，它处于继承层次结构的较上层。

1. 抽象类的定义： 称带有纯虚函数的类为抽象类。

2. 抽象类的作用： 抽象类的主要作用是将有关的操作作为结果接口组织在一个继承层次结构中，由它来为派生类提供一个公共的根，派生类将具体实现在其基类中作为接口的操作。所以派生类实际上刻画了一组子类的操作接口的通用语义，这些语义也传给子类，子类可以具体实现这些语义，也可以再将这些语义传给自己的子类。

3. 使用抽象类时注意：

   抽象类只能作为基类来使用，其纯虚函数的实现由派生类给出。如果派生类中没有重新定义纯虚函数，而只是继承基类的纯虚函数，则这个派生类仍然还是一个抽象类。如果派生类中给出了基类纯虚函数的实现，则该派生类就不再是抽象类了，它是一个可以建立对象的具体的类。 抽象类是不能定义对象的。

总结：

1. 纯虚函数声明如下： `virtual void funtion1()=0;` 纯虚函数一定没有定义，纯虚函数用来规范派生类的行为，即接口。包含纯虚函数的类是抽象类，抽象类不能定义实例，但可以声明指向实现该抽象类的具体类的指针或引用。

2. 虚函数声明如下：virtual ReturnType FunctionName(Parameter) 虚函数必须实现，如果不实现，编译器将报错，错误提示为：

   ```
   error LNK****: unresolved external symbol "public: virtual void __thiscall
   ClassName::virtualFunctionName(void)"
   ```

3. 对于虚函数来说，父类和子类都有各自的版本。由多态方式调用的时候动态绑定。

4. 实现了纯虚函数的子类，该纯虚函数在子类中就编程了虚函数，子类的子类即孙子类可以覆盖该虚函数，由多态方式调用的时候动态绑定。

5. 虚函数是C++中用于实现**多态(polymorphism)**的机制。核心理念就是通过基类访问派生类定义的函数。

6. 在有动态分配堆上内存的时候，析构函数必须是虚函数，但没有必要是纯虚的。

7. 友元不是成员函数，只有成员函数才可以是虚拟的，因此友元不能是虚拟函数。但可以通过让友元函数调用虚拟成员函数来解决友元的虚拟问题。

8. 析构函数应当是虚函数，将调用相应对象类型的析构函数，因此，如果指针指向的是子类对象，将调用子类的析构函数，然后自动调用基类的析构函数。

   有纯虚函数的类是抽象类，不能生成对象，只能派生。他派生的类的纯虚函数没有被改写，那么，它的派生类还是个抽象类。

   定义纯虚函数就是为了让基类不可实例化化，因为实例化这样的抽象数据结构本身并没有意义，或者给出实现也没有意义。

   实际上我个人认为纯虚函数的引入，是出于两个目的：

9. 为了安全，因为避免任何需要明确但是因为不小心而导致的未知的结果，提醒子类去做应做的实现。

10. 为了效率，不是程序执行的效率，而是为了编码的效率。