引用、变量、指针

1. 引用：

相当于一个别名。

1. 变量

相当于复制了一个对象。

1. 指针

相当于创建了一个新的指针指向之前的对象，新指针会占用4个字节的内存空间。

多态

1. 类的静态绑定
   1. 成员函数不会在对象内存空间中，在编译期就分配好了内存地址；
   2. 尽管指针指向null或者其他，指针依然可以调用成员函数；
   3. 如果指针指向null或者其他并且成员函数操作了成员变量（赋值或者读取），那么程序会报错。
2. 动态绑定
   1. 在运行时确定调用函数的绑定称为动态绑定，
   2. 实现动态绑定：虚函数

在函数前面加关键字 virtual

例：Virtual void func(){}

* 1. 如果一个类中定义了虚函数，那么对象中会保存虚函数表的地址，
  2. 子类重写父类的虚函数不需要写关键字virtual，编译器会自动识别为virtual。

1. 多态

同一个类型的指针（或引用）调用相同函数表现不一样。

虚析构函数、抽象类、全局变量

1. 一般基类的析构函数会定义为虚析构函数的形式，在实现多态时，防止删除一个new出来的子类对象时未调用子类的析构函数。
2. 类中如果有纯虚函数(虚函数=0)，那么该类是一个抽象类。C++中抽象类不能有new对象。如果子类会要创建实例，那么子类必须重写父类的纯虚函数。
3. 绑定全局变量：

在.cpp文件最开始部分int a=10;

四．使用全局变量

在使用地方使用extern int a;

全局变量是针对整个工程的。正确的应该是在一个CPP文件中定义如下：int   g\_Test;那么在使用的CPP文件中就应该使用：extern   int  g\_Test即可

操作符重载

1. 格式：

返回类型 operator 操作符(操作对象) {}

二．++重载

cInt operator++(int)---- 对后置++的重载

cInt operator++()---- 对前置++的重载

1. 友元函数 关键字friend：
   1. 实现类之间数据共享时，减少系统开销，提高效率
   2. 友元函数破环了封装机制，尽量不使用成员函数，除非不得已的情况下才使用友元函数。
   3. 友元函数没有this指针,要访问非static成员时，需要对象做参数。访问static成员或全局变量时，则不需要对象做参数
   4. 友元函数是类外的函数，所以它的声明可以放在类的私有段或公有段且没有区别
   5. 直接调用友元函数，不需要通过对象或指针

另一个类访问当前类中的数据（包括private和protecde）。

1. 内联函数 关键字 inline 要求函数尽量简单。
2. 转换构造函数:
   1. 可以通过一个参数来构造一个对象。
   2. 必须是const类型的引用或者是一个对象
   3. 其他类转换为本类
3. 转换函数
   1. 本类转换为其他类型