```
#include <iostream>                // inh4dyncast.cpp
using namespace std;               // Ersatz fuer virtual mittels Nutzung des
                                   // dynamic_cast<derived *>(pb)
class base {
  public:  base(int i=0):a(i){ cout<<"Konstruktor base\n"; };
           ~base(){ cout<<"Destruktor base\n"; };
           void print(){ cout<<"base print a = "<<a<<endl; };
           void give(){  cout<<"base give  a = "<<a<<endl; };
           int geta(){ return a; }
           virtual void dummy(){}; //notw. wegen dynamic_cast
  private: int a;
};

class derived : public base {
  public: derived(int i=0):base(i){ cout<<"Konstr. derived\n";};
          ~derived(){ cout<<"Destruktor derived\n"; };
          void print(){cout<<"derived print a = "<<geta()<<endl;}//override
          void give(){cout<<"derived give a = "<<geta()<<endl;}  //override
};

void main(){
     base b1(1), *pb=&b1;
     pb->print();   // base print(), Zeiger auf base-Objekt, 1
     //Ersatz fuer late binding: Testen des dynamic_cast<derived *>(pb)
     dynamic_cast<derived *>(pb)?(dynamic_cast<derived *>(pb))->give():
                                 (dynamic_cast<base *>(pb))->give();
     derived d1(2);
     pb = &d1;
     pb->print();   // base print(), Zeiger auf derived-Objekt, 2
     //Ersatz fuer late binding: Testen des dynamic_cast<derived *>(pb)
     if(dynamic_cast<derived *>(pb))((derived *)pb)->give(); else pb->give();
}
```