```
#include <iostream>       // virt9.cpp
using namespace std;
// Ersatz fuer virtuelle Konstruktoren mit "clone()"-Methoden

class X {
    public:
         X(){ cout<<"\nKonstruktor X"<<endl; }

         virtual ~X(){ cout<<"Destruktor X\n"<<endl; }

         X(const X& x){ cout<<"Konstruktor X::X(const X&)"<<endl; }

         virtual X *clone(){ cout<<"\nclone() from X"<<endl;
                             return new X(* this);
         }
};

class Y : public X {
    public:
         Y(){ cout<<"Konstruktor Y"<<endl; }

         virtual ~Y(){ cout<<"Destruktor Y"<<endl; }

         virtual X *clone(){ cout<<"\nclone() from Y"<<endl;
                             return new Y(*this);
         }
};

void main(){
             X *x1 = new Y;          // Zeiger x1 auf Y
             X *x2 = new X;          // Zeiger x2 auf X
             X *x3 = x1->clone();    // typgerechtes Duplikat von *x1 (late)
             X *x5 = new X(*x1);     // nicht typgerecht bzgl. Y
             X *x4 = x2->clone();    // typgerechtes Duplikat von *x2 (late)
             delete x1; x1 = 0;
             delete x2; x2 = 0;
             delete x3; x3 = 0;
             delete x4; x4 = 0;
}

/*
Konstruktor X
Konstruktor Y

Konstruktor X

clone() from Y
Konstruktor X::X(const X&)
Konstruktor X::X(const X&)

clone() from X
Konstruktor X::X(const X&)
Destruktor Y
Destruktor X

Destruktor X

Destruktor Y
Destruktor X

Destruktor X
```