Polymorphism...

Monday, October 14, 2019  5:19 PM

many-shapes

• Static vs. Dynamic type.

    class A
    {...
    
    class B : public A
    {...

    A* p = &bob;

    What is the type of the object pointed by p?
    
    Statically: A
    
    Dynamically: B

Polymorphism: the ability of an object of a static type, to take "many shapes" i.e. to behave differently, according to its dynamic type. Also known as "Dynamic dispatch"

EXAMPLE:

class FarmAnimal
{
    virtual void speak() { cout "...";}
}

class Cow : public FarmAnimal
{
    
}

void main
{
    FarmAnimal* Farm[4];
    Farm[0] = new Cow;
    Farm[1] = new Chicken;
    Farm[2] = new Pig;
    Farm[3] = new Fox;
    ...
class Cow : public FarmAnimal
{
    void speak()
    { cout << "Moo"; }
}
class Chicken : public FarmAnimal
{
    void speak()
    { cout << "Cluck"; }
}
class Pig : public FarmAnimal
{
    void speak()
    { cout << "Oink!"; }
}

Farm[0] = new Cow;
Farm[1] = new Chicken;
Farm[2] = new Pig;
Farm[3] = new Fox;
for(int k=0; k<4; k++)
    Farm[k]->speak();

// version 2
class FarmAnimal
{
    virtual void speak() = 0;
};

FarmAnimal Bob;  
Fox Dan;  

void main
{
    FarmAnimal* Farm[4];
    Farm[0] = new Cow;
    Farm[1] = new Chicken;
    Farm[2] = new Pig;
    Farm[3] = new Fox;
    for(int k=0; k<4; k++)
        Farm[k]->speak();
}
The virtual quality of a function is inherited.

by convention write 'virtual'
on all virtual functions.

void main
{
    FarmAnimal* Farm[3];
    Farm[0] = new Cow;
    Farm[1] = new Chicken;
    Farm[2] = new Pig;

    for(int k=0; k<3; k++)
    {
        Farm[k]->speak();
    }

    for(int k=0; k<3; k++)
    {
        delete Farm[k];
    }
}
FarmAnimal *p = new Pig;

Pig* q = p;  // Not a good practice

Pig* q = dynamic_cast<Pig>( p ); // NULL if not successful

Cow* r = static_cast< Cow >( p ); // works but dangerous..