# Полиморфизм и виртуальные функции

Допускается, что указатель и объект могут иметь разные типы.

Animal\* myDog = new Dog();

Указатель **базового** класса может указывать на объект **производного** класса.

При этом указатель **базового** класса может вызывать методы **только** **базового** класса (без ключевого слова virtual)

class Animal {

…

void speak() { cout << "I am an animal"; }

};

class Dog : **public Animal** {

…

void speak() { cout << “Woof!!!"; }

};

int main() {

Animal\* p\_animal1 = new Animal;

Animal\* p\_animal2 = new Dog;

p\_animal1->speak(); // I am an animal

p\_animal2->speak(); // I am an animal

}

# Ключевое слово virtual

class Animal {

…

**virtual** void speak() { cout << "I am an animal"; }

};

class Dog : public Animal {

…

**virtual** void speak() { cout << “Woof!!!"; }

};

int main() {

Animal\* p\_animal1 = new Animal;

Animal\* p\_animal2 = new Dog;

p\_animal1->speak(); // I am an animal

p\_animal2->speak(); // Woof!!!

}

# Виртуальный деструктор

class Animal {

…

**virtual** ~Animal();

};

class Dog : public Animal {

…

**virtual** ~ Dog ();

};

Animal\* p\_animal [] = {new Animal, new Dog, new Cat};

…

for (int i =0; i < 3; i++)

delete p\_animal[i];

При наличии виртуальных деструкторов оператор delete вызывает деструктор производного класса, а затем базового

# Абстрактные базовые классы

class Animal {

…

virtual void speak() = 0 // **=0** делает функцию чисто виртуальной.

};

int main() {

Animal an; // Ошибка

Animal\* p\_an [] = {new Dog, new Cat, …} //Ок

}