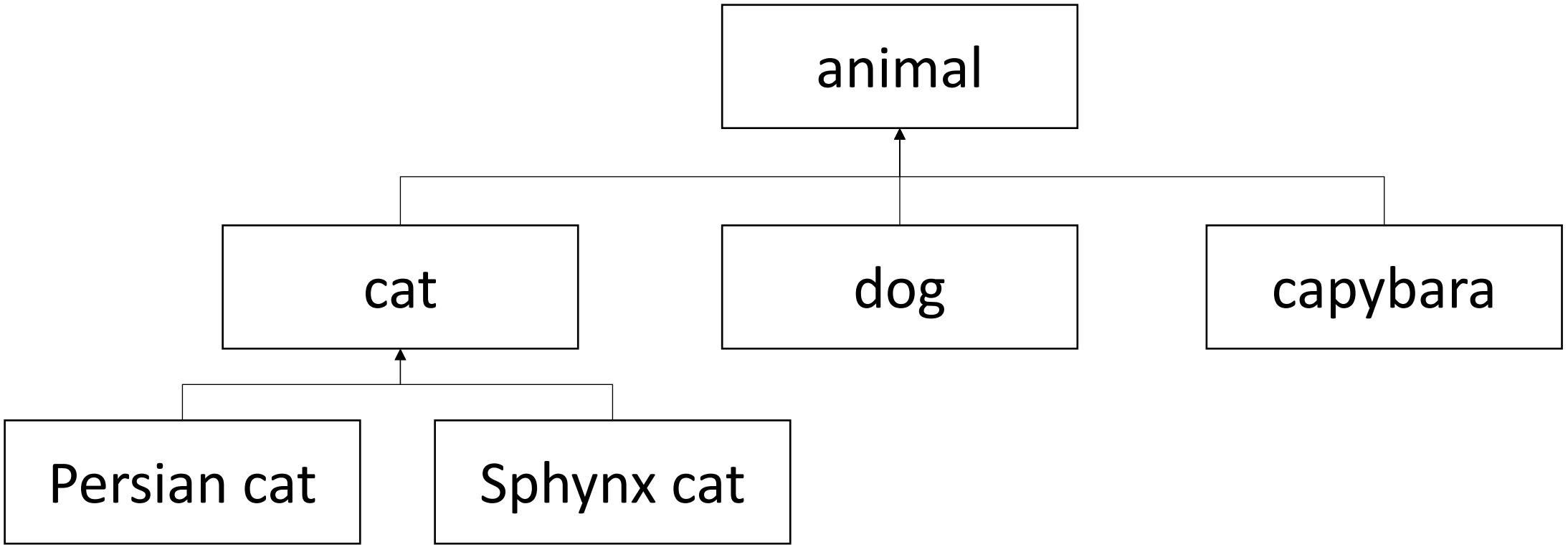
Наследование

Наследование



Наследование (inheritance) представляет один из ключевых аспектов объектно-ориентированного программирования, который позволяет наследовать функциональность одного класса или базового класса (base class) в другом - производном классе (derived class).

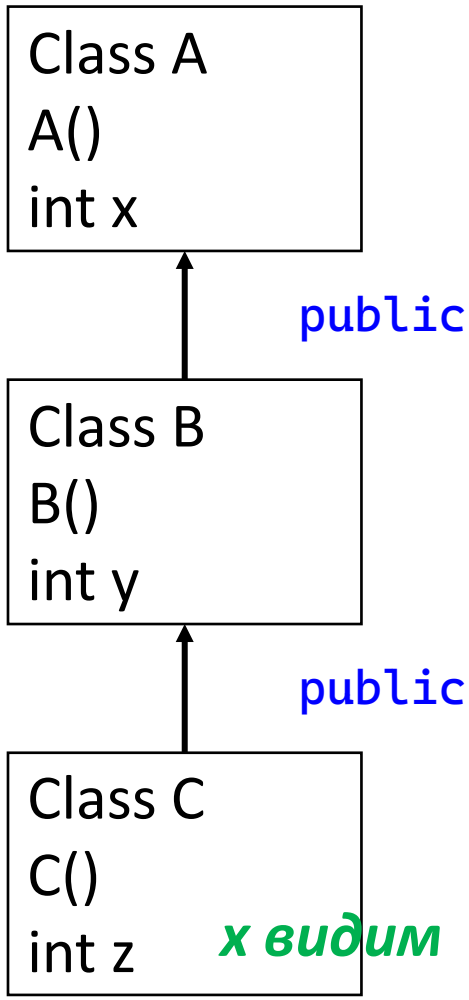
Общий вид наследования

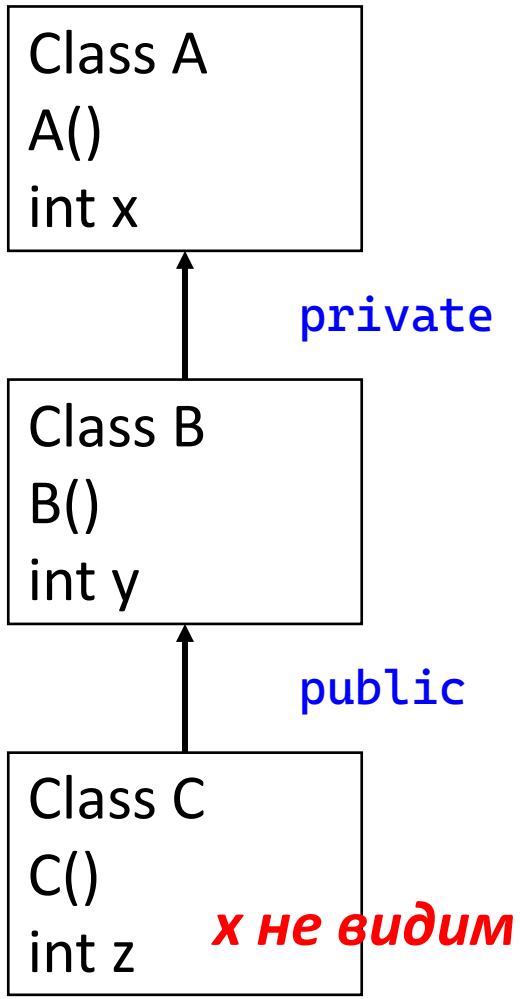
```
class B :{public
         protected} A{
         private

};
```

Модификаторы доступа при наследовании

Наследование классов






```
class B : public
          protected
          private    A{

};
```

Пример наследования

```
#include <iostream>

class A {
    private:
        int x;
    public:
        A() {
            std::cout << "A constructor" << std::endl;
        }
};

class B : A{
    public:
        B() {
            std::cout << "B constructor" << std::endl;
        }
};

int main()
{
    A a;
    B b;
}
```

1. Конструкторы не наследуются
2. При создании дочернего класса сначала создается экземпляр базового класса

```
A constructor
A constructor
B constructor
```

Пример наследования

```
class A {
public:
    int x;
    A(int _x) {
        x = _x;
        std::cout << "A constructor" << std::endl;
    }
};

class B : public A{
public:
    int y = 15;
    B() : A(1) {};
    B(int x):A(x) {
        std::cout << "B constructor" << std::endl;
    }
    void printX() {
        std::cout << x << std::endl;
    }
};
```

1. Конструкторы не наследуются
2. При создании дочернего класса сначала создается экземпляр базового класса
3. Если у базового класса нет конструктора по умолчанию, то необходимо напрямую передать параметр

Абстрактные классы

```
class A {
public:
    int x;
    A(int _x) {
        x = _x;
        std::cout << "A constructor" << std::endl;
    }
    virtual void printX() = 0;
};
```

Использование ключевого слова virtual и указание, что функция является «чистой» виртуальной функцией приводит к тому, что класс становится абстрактным и невозможно создавать его экземпляры

Дружественные функции

Дружественные функции

```
class A {
public:
    friend void f(A&);
    void print_x();
private:
    int x;
};

void f(A& a) {
    a.x = 10;
}

int main()
{
    A a;
    f(a);
}
```

Дружественные функции - это функции, которые не являются членами класса, однако имеют доступ к его закрытым членам - переменным и функциям, которые имеют спецификатор private.

В качестве дружественной функции могут выступать методы другого класса

Возможное применение – одна функция работающая со многими классами.

Дружественные классы

```
class A {
    friend class B;
private:
    int x;
};

class B {

public:
    void print_x(A& a) {
        std::cout << "X = " << a.x << "\n";
    }
    void print_y() {
        std::cout << "Y = " << y << "\n";
    }
private:
    int y;
};
```

При объявлении класса можно объявить сразу все функции-члены другого класса дружественными одним объявлением. Таким образом создается **дружественный класс**.