C++. Уровень 3

Урок 2

Наследование

Конструкторы и наследование.
Многоуровневая иерархия классов.
Простое и множественное наследование.

План урока

- Константные объекты классов.
- Определение методов вне класса.
- Скрытый указатель this.
- Базовое наследование.

План урока

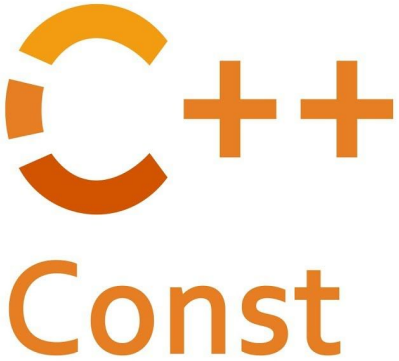
- Дружественные функции.
- Спецификатор доступа protected.
- Типы наследования.
- Указатели и ссылки на производные классы.
- Множественное наследование.

Подготовка к наследованию

Константные объекты

Константный метод – это такой метод, который гарантирует, что не будет изменять объект.

Правило: делайте все ваши методы, которые не изменяют данные объекта класса, константными.



Определение методов вне класса

```
void student :: setdata (char *f, char *n, int a)
{     age = a; strcpy(fam, f);
      strcpy(name, n);
 }
```

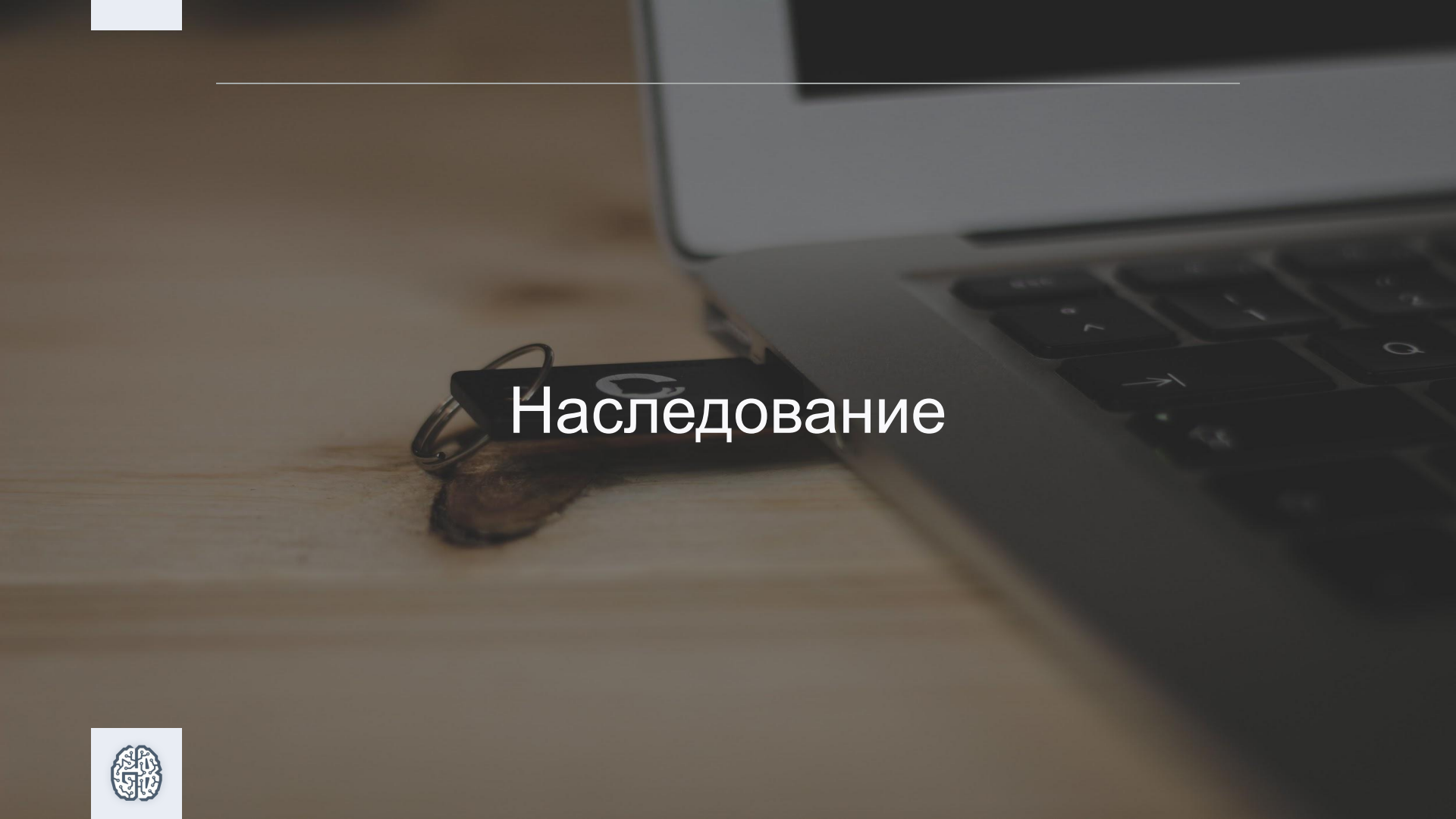
::

оператором разрешения области видимости

Скрытый указатель this

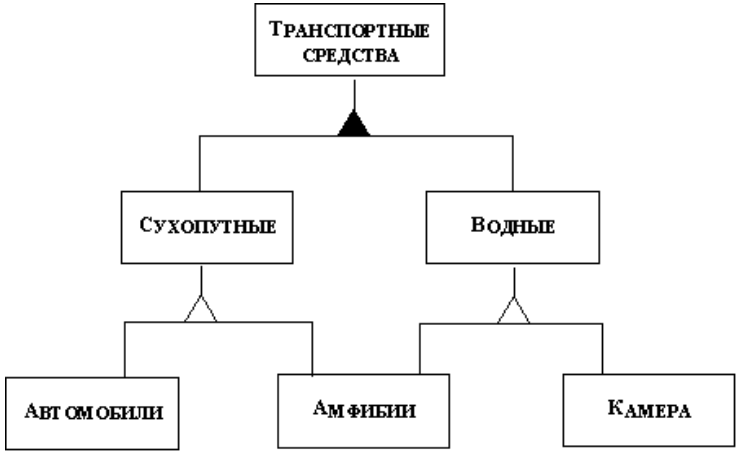
Указатель this – это скрытый константный указатель, который содержит адрес объекта, вызывающего метод класса.

```
void setNumber(SomeClass* const this, int
number)
{
    this->m_number = number;
}
```



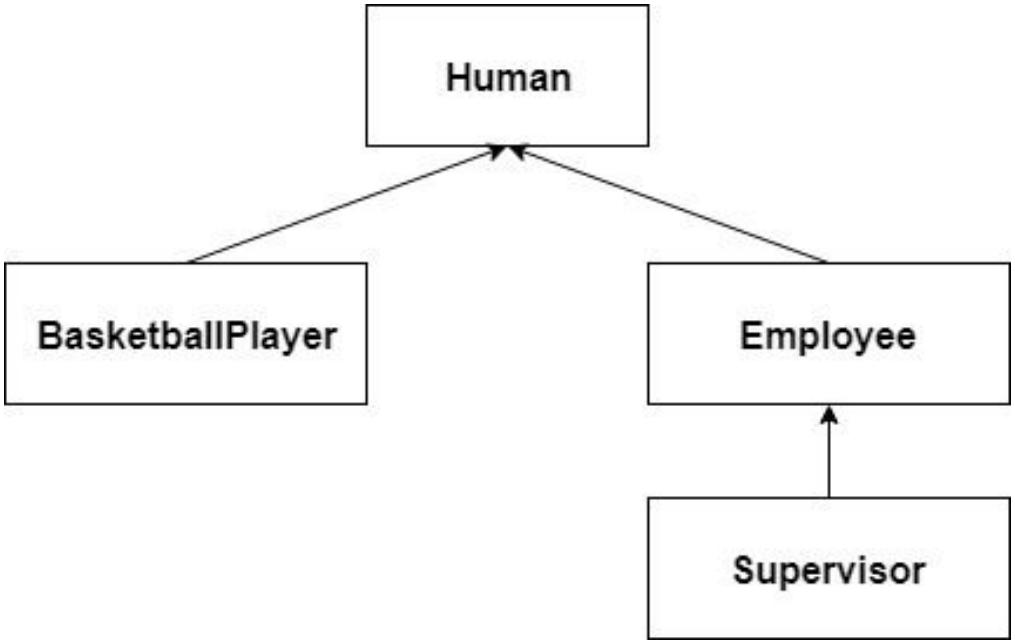
Наследование

Примеры наследования



Наследование в C++


```
class BasketballPlayer : public Human
```


Human
BasketballPlayer
Employee
Supervisor

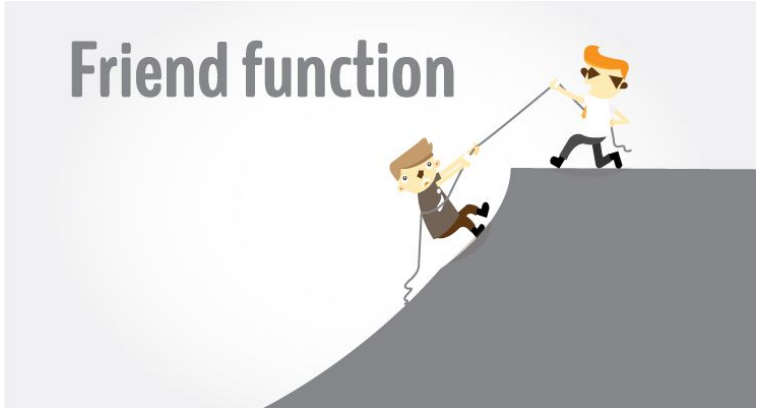
Класс 3 уровня	Класс 2 уровня	Класс 1 уровня	Переменные-члены
Супервайзер	Работник	Человек	Имя
			Возраст
			Зарплата
			Рабочий ID
			ID подчиненных

Класс 2 уровня	Класс 1 уровня	Переменные-члены
Баскетболист	Человек	Имя
		Возраст
		Количество игр
		Количество очков


Для зачатия ребёнка нужны мальчик и девочка.
Сторонники ООП считают что достаточно одного существа.

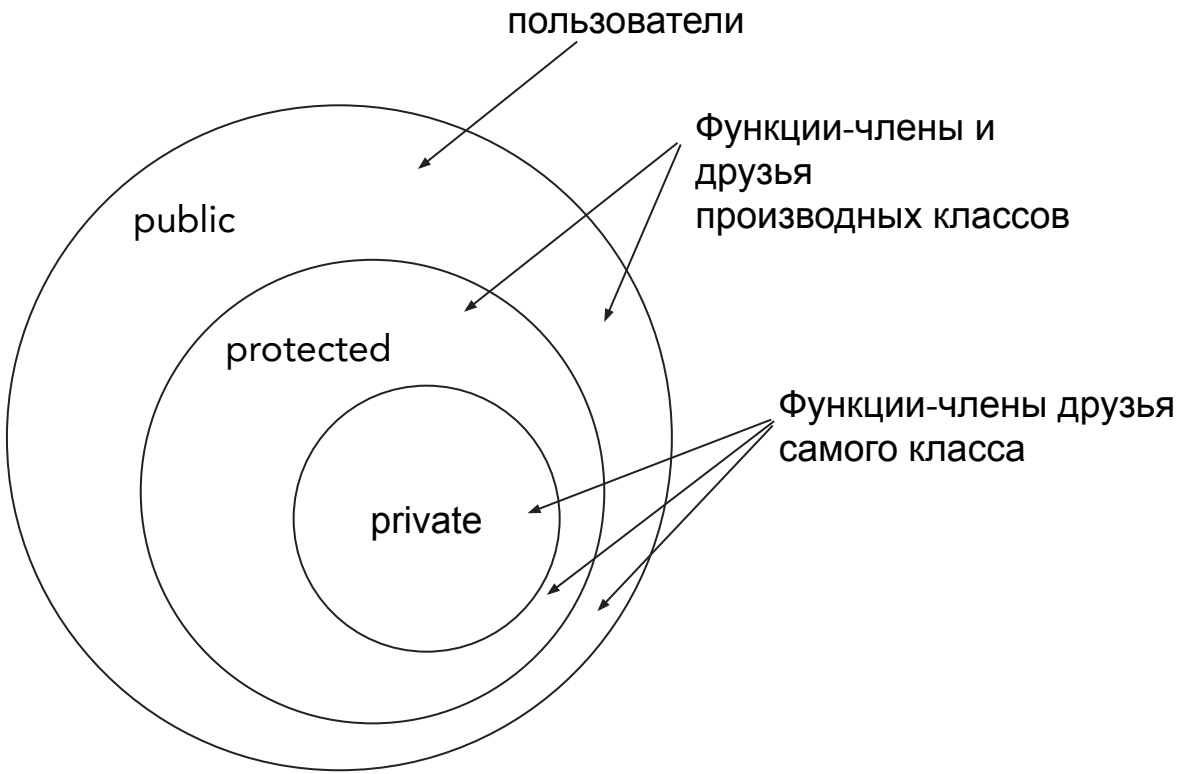
Дружественные функции

Дружественные функции имеют доступ к закрытым членам класса.




```
friend void resetAge(Cat &cat);
```

Спецификаторы доступа



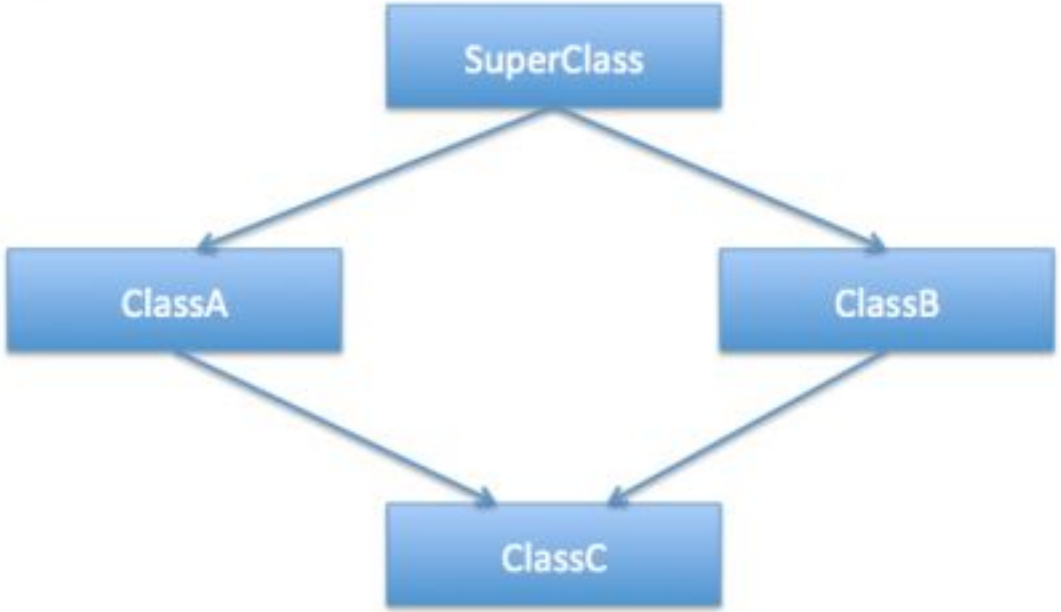
Типы наследования

	Публичное (CDerived :public CBase)	**Защищенное** (CDerived :protected CBase)	**Приватное** (CDerived :private CBase)
public	public	protected	private
protected	protected		
private	недоступно	недоступно	недоступно
	public, private, protected	public, private, protected	public, private, protected

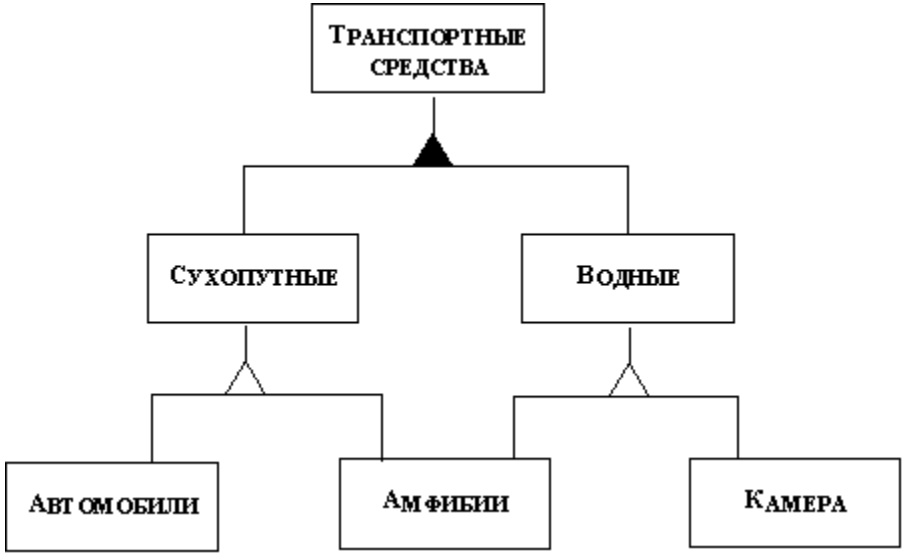
Типы наследования

Спецификатор доступа в родительском классе	Спецификатор доступа при наследовании типа public в дочернем классе	Спецификатор доступа при наследовании типа private в дочернем классе	Спецификатор доступа при наследовании типа protected в дочернем классе
Public	Public	Private	Protected
Private	Недоступен	Недоступен	Недоступен
Protected	Protected	Private	Protected

Множественное наследование



Множественное наследование



Решите задачи

Что будет выведено на экран?

1

```
#include <iostream>
using namespace std;


class Parent
{
public:
    Parent()
    {
        cout << "Parent()\n";
    }
    ~Parent()
    {
        cout << "~Parent()\n";
    }
};
```

Продолжение

```
class Child: public Parent
{
public:
    Child()
    {
        cout << "Child()\n";
    }
    ~Child()
    {
        cout << "~Child()\n";
    }
};


int main()
{
    Child ch;
}
```

Что будет выведено на экран?

2

```
#include <iostream>


class Parent
{
public:
    Parent()
    {
        std::cout << "Parent()\n";
    }
    ~Parent()
    {
        std::cout << "~Parent()\n";
    }
};
```

Продолжение

```
class Child: public Parent
{
public:
    Child()
    {
        std::cout << "Child()\n";
    }
    ~Child()
    {
        std::cout << "~Child()\n";
    }
};


int main()
{
    Child ch;
    Parent p;
}
```

Что будет выведено на экран?

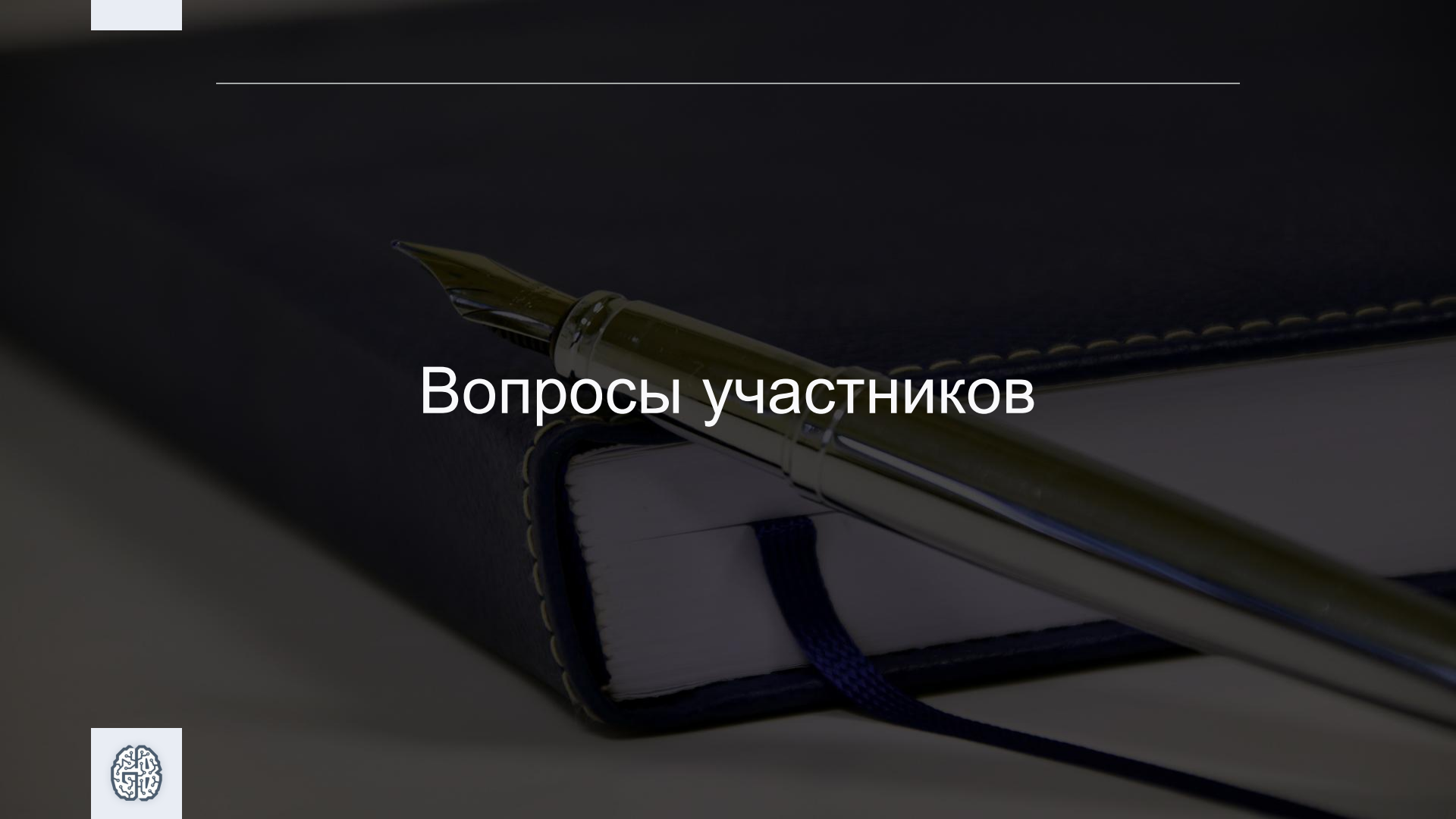
3

```
#include <iostream>
using namespace std;

class Parent {
public:
    void getValue() {count();}
private:
    virtual void count() {cout << 1;}
};

class Child : public Parent {
private:
    void count() {cout << 2;}
};

int main() {
    Parent* obj = new Child;
    obj->getValue();
    return 0;
}
```



Вопросы участников