C++. Уровень 3

Урок 1

Введение в ООП. Инкапсуляция

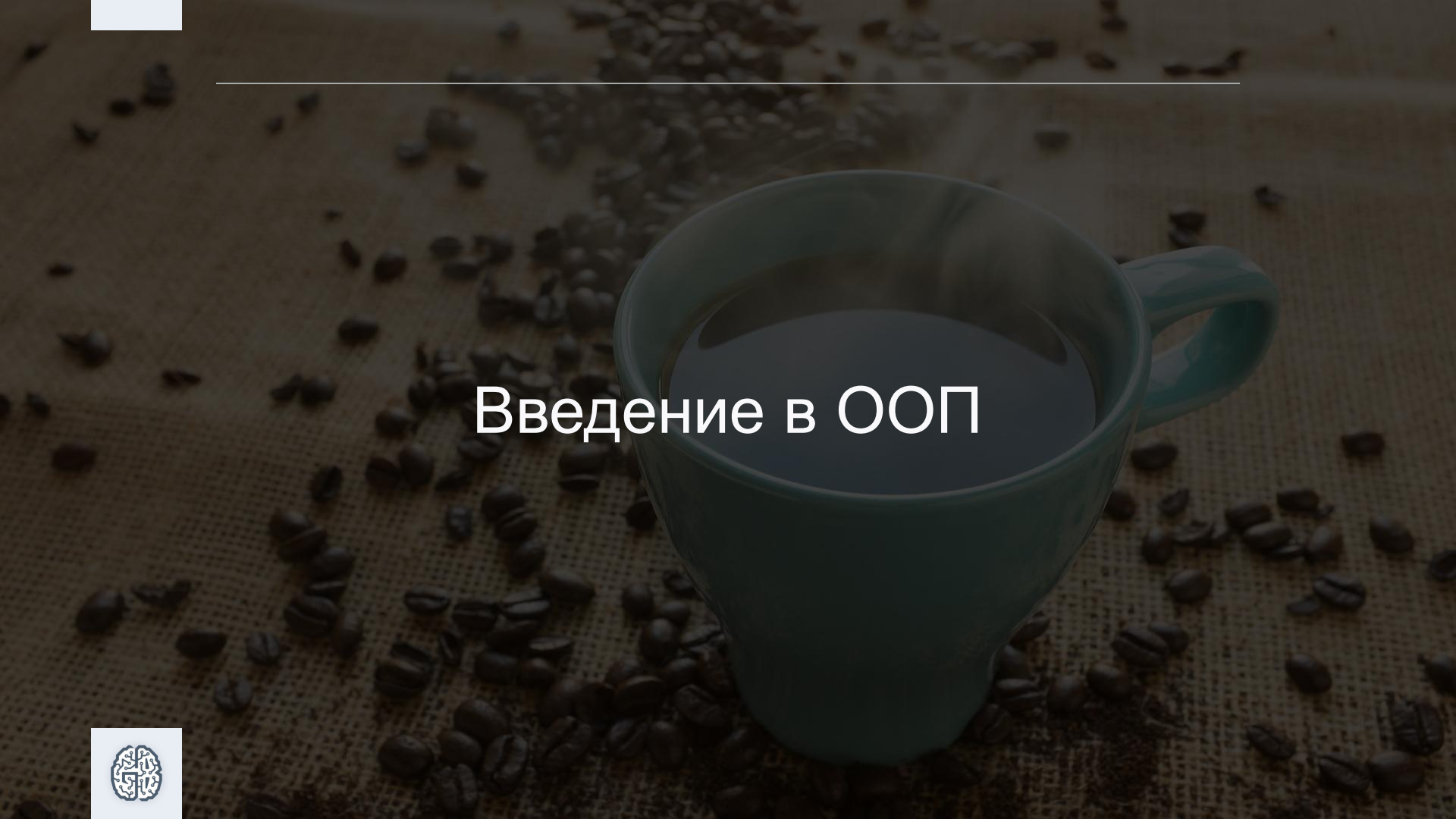
Классы и объекты. Свойства и методы. Конструкторы и деструкторы.

План урока

- Введение в ООП.
- Классы и объекты. Методы и свойства.
- Спецификаторы доступа.

План урока

- Конструкторы.
- Список инициализации членов.
- Деструкторы.



Введение в ООП

Введение в ООП

У любого объекта есть:

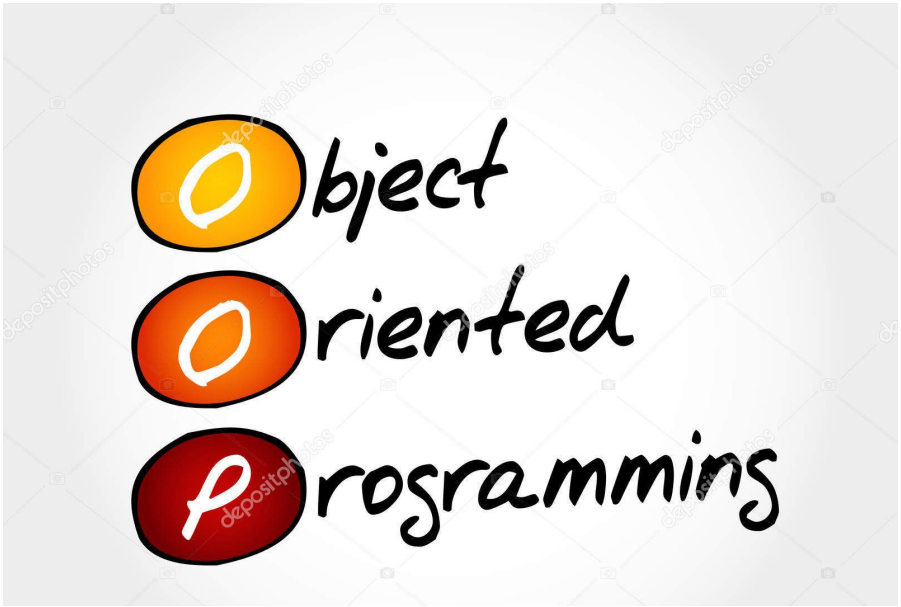
- *Список свойств*

(цвет, размер, вес и другие);

- *Список поведений*

(делать, открывать что-то и т.д.).

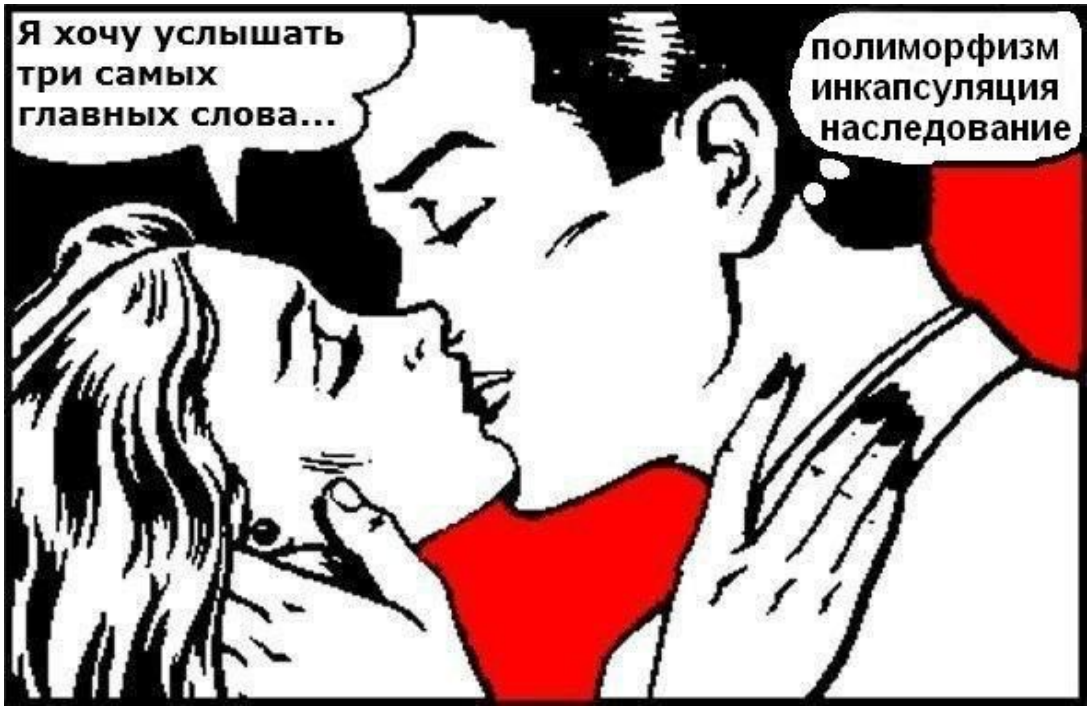
Введение в ООП



Причины популярности ООП

- Увеличение производительности.
- Улучшения надежности.
- Многократное использование кода.
- Упрощение написания и понимания кода.

Концепция ООП



Классы и объекты. Свойства и методы

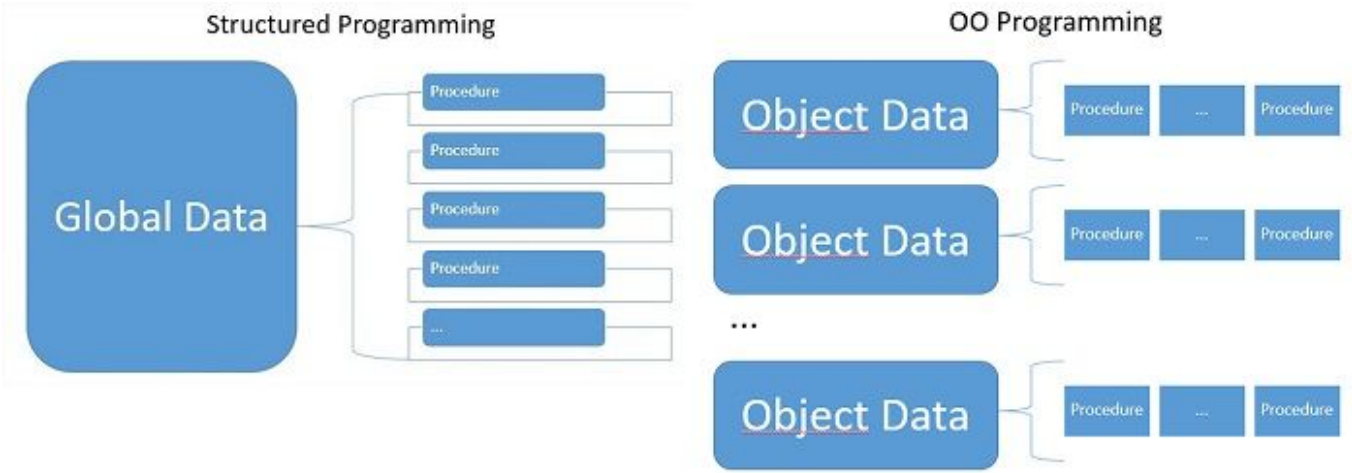
Процедурный подход

```
struct DateStruct
{
    int day;
    int month;
    int year;
} today = {12, 12, 2018};

void print(DateStruct &date)
{
    std::cout << date.day<< "/" << date.month << "/" << date.year;
}

int main()
{
    today.day = 18; // используем оператор выбора члена для выбора члена структуры
    print(today);

    return 0;
}
```

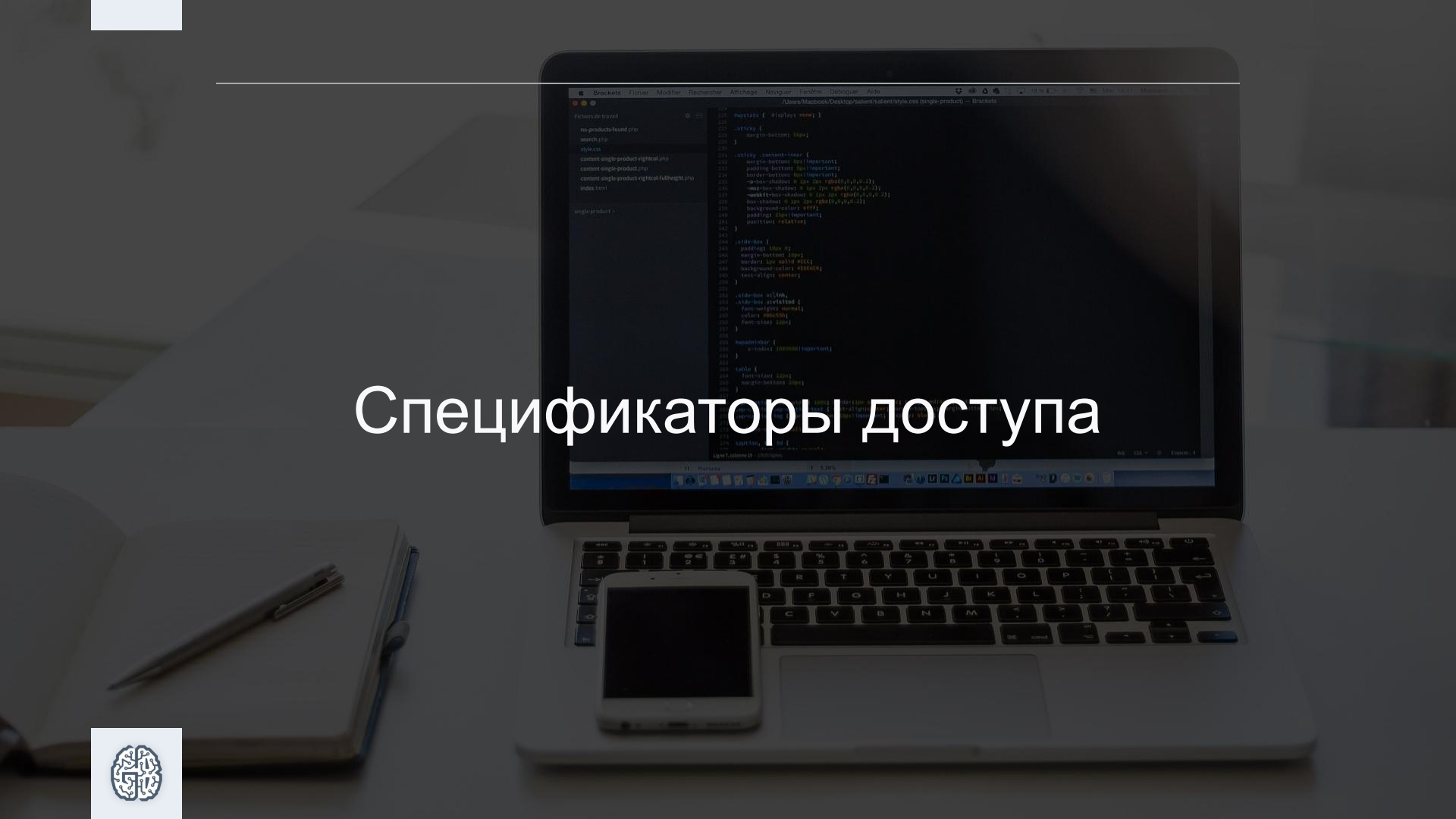
Объектно-ориентированный подход

```cpp
#include <iostream>

class DateClass
{
public:
    int m_day;
    int m_month;
    int m_year;

    void print()
    {
        std::cout << m_day << "/" << m_month << "/" << m_year;
    }
};

int main()
{
    DateClass today { 12, 12, 2018 };

    today.m_day = 18;
    today.print(); // используем оператор (.) для вызова метода объекта today класса DateClass

    return 0;
}
```

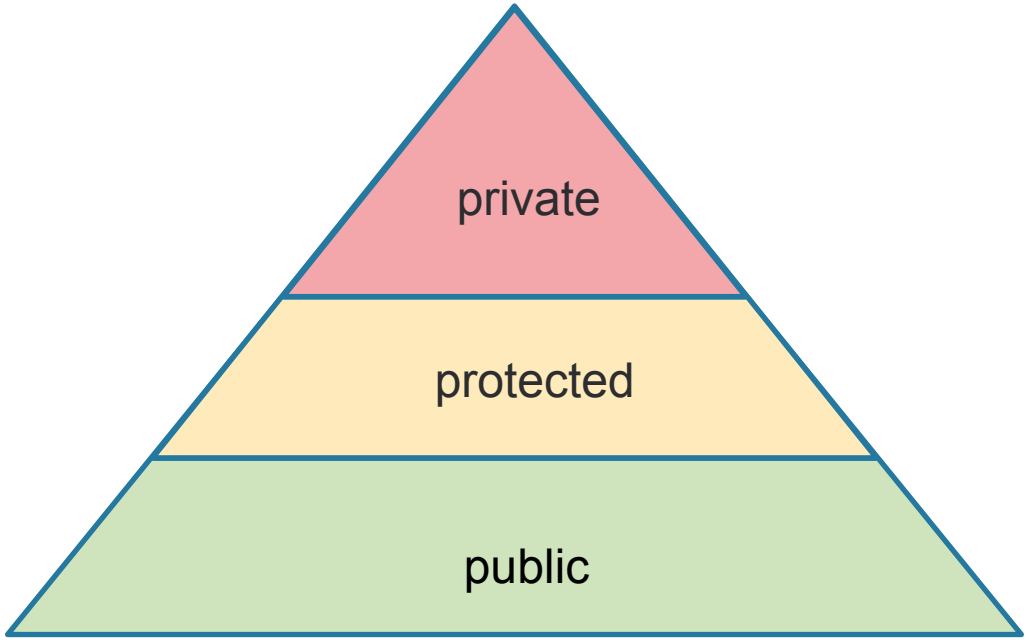
Процедурное программирование vs ООП



Спецификаторы доступа



Спецификаторы доступа

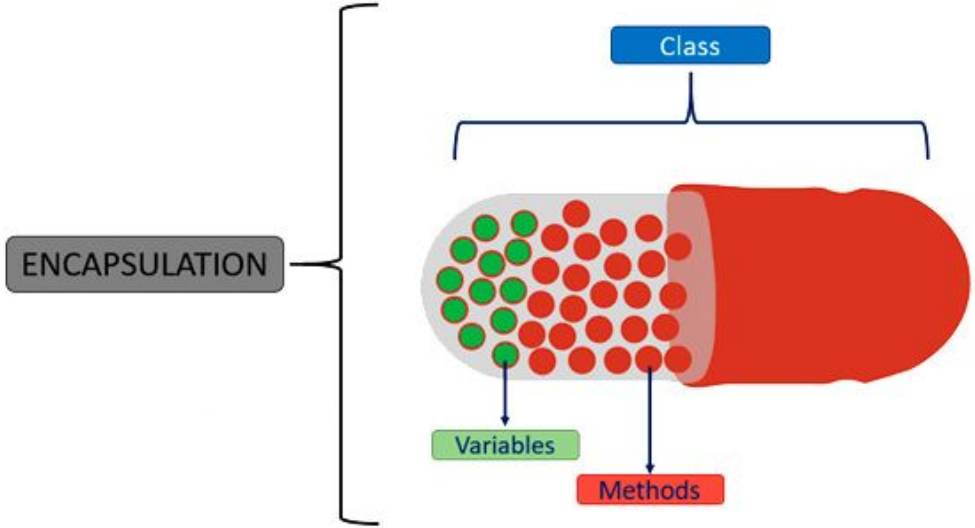


```cpp
#include <iostream>

class DateClass // члены класса являются закрытыми по умолчанию
{
private:
    int m_day; // закрыто по умолчанию, доступ имеют только другие члены класса
    int m_month; // закрыто по умолчанию, доступ имеют только другие члены класса
    int m_year; // закрыто по умолчанию, доступ имеют только другие члены класса

public:
    void setDate(int day, int month, int year) // открыто, доступ имеет любой объект
    {
        // метод setDate() имеет доступ к закрытым членам класса, так как сам является членом класса
        m_day = day;
        m_month = month;
        m_year = year;
    }

    void print() // открыто, доступ имеет любой объект
    {
        std::cout << m_day << "/" << m_month << "/" << m_year;
    }
};

int main()
{
    DateClass date;
    date.setDate(12, 12, 2018); // ок, так как setDate() имеет спецификатор доступа public
    date.print(); // ок, так как print() имеет спецификатор доступа public

    return 0;
}
```

Set и get-функции

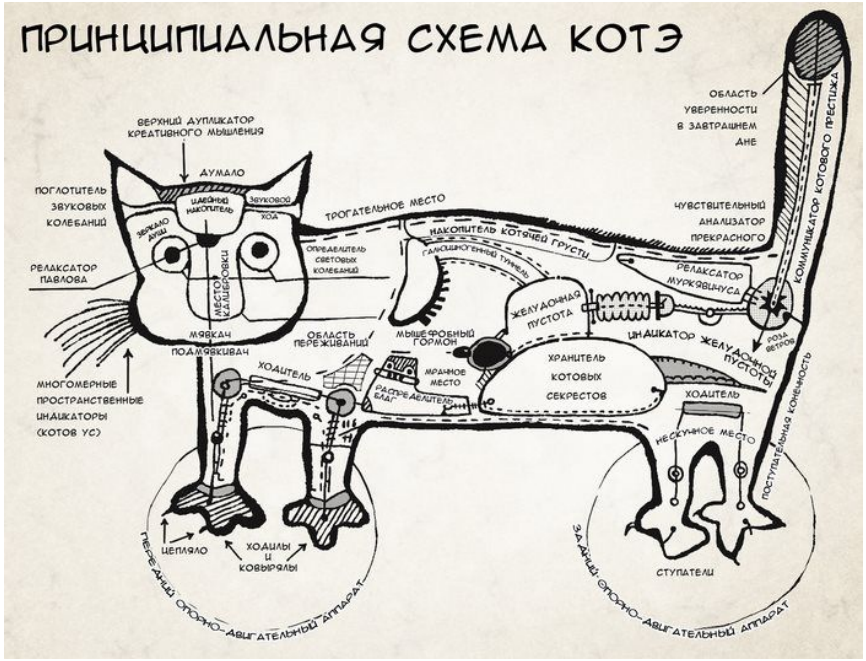
- Пишите set- и get-функции только для тех классов, для которых это необходимо.
- Get-функции должны возвращать значения по значению или по константной ссылке. Не используйте неконстантные ссылки.

Инкапсуляция



Инкапсуляция — это процесс скрытого хранения деталей реализации объекта.

Инкапсуляция



Конструкторы

Правила объявления

- Конструкторы всегда должны иметь то же имя, что и класс.
- Конструкторы не имеют типа возврата (даже void).

Типы конструкторов

- Конструктор по умолчанию.
- Конструктор с параметром.

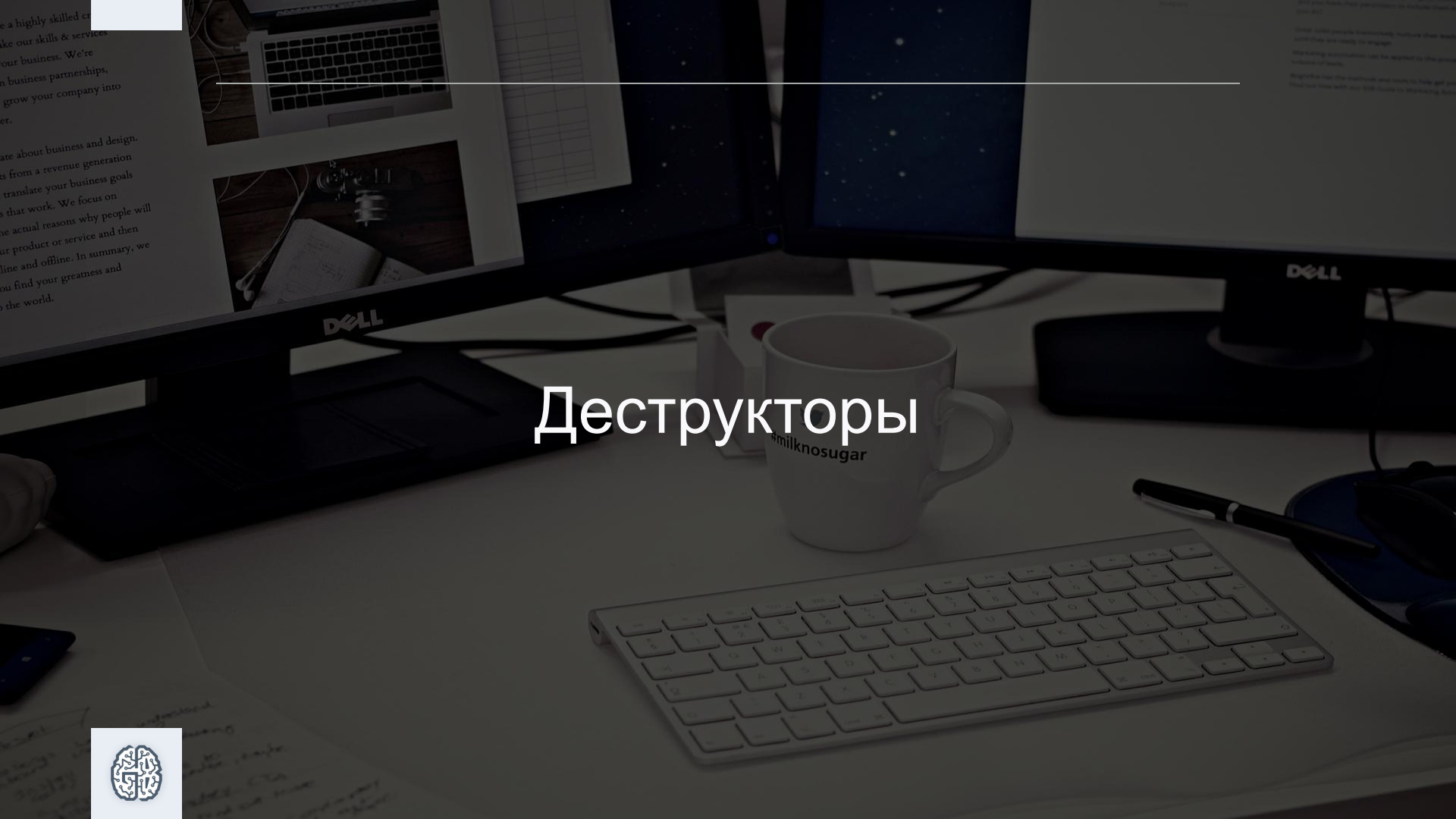
Список инициализации членов

Синтаксис

Конструктор (параметры) : переменная_1(значение_1), … , переменная_N(значение_N)

```
Date(int day, int month, char dayOfWeek='mon')
    : m_day(day), m_month(month), m_dayOfWeek(dayOfWeek)
// напрямую инициализируем переменные-члены класса
    {
        // Нет необходимости использовать присваивание
    }
```

Деструкторы



Правила объявления

- Деструктор должен иметь то же имя, что и класс, со знаком «тильда» (~) в самом начале.
- Деструктор не может принимать аргументы.
- Деструктор не имеет типа возврата.

Пример программы

Найдите ошибки в коде

1

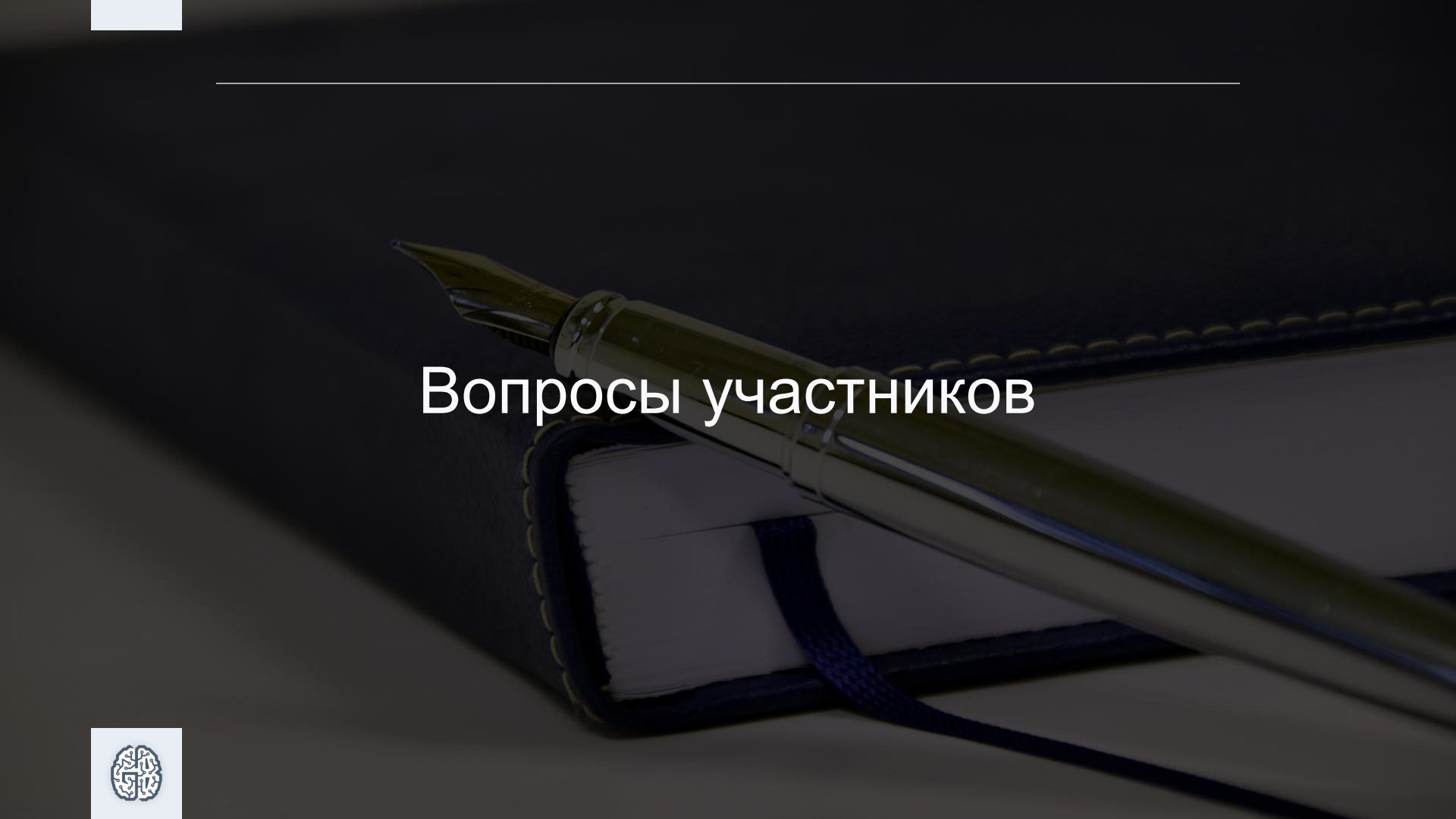
```
#include "iostream"
class A {
    int x=0;
    void print(int x) {
        std::cout << x;
    }
};
int main() {
    A a;
    a.print;
}
```

2

```
class A {
private:
    const char x[2];
public:
    void A() {
        x[0]='A';
        x[1]='B';
    }
};
```

3

```
class A {
    B b;
public:
    A() { b.set(1); }
};

class B {
    int x;
public:
    B() { x=0; }
    void set(int y) { x=y; }
};
```



Вопросы участников