



Парадигмы ООП



Что такое ООП?

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса или структуры.



Парадигмы ООП

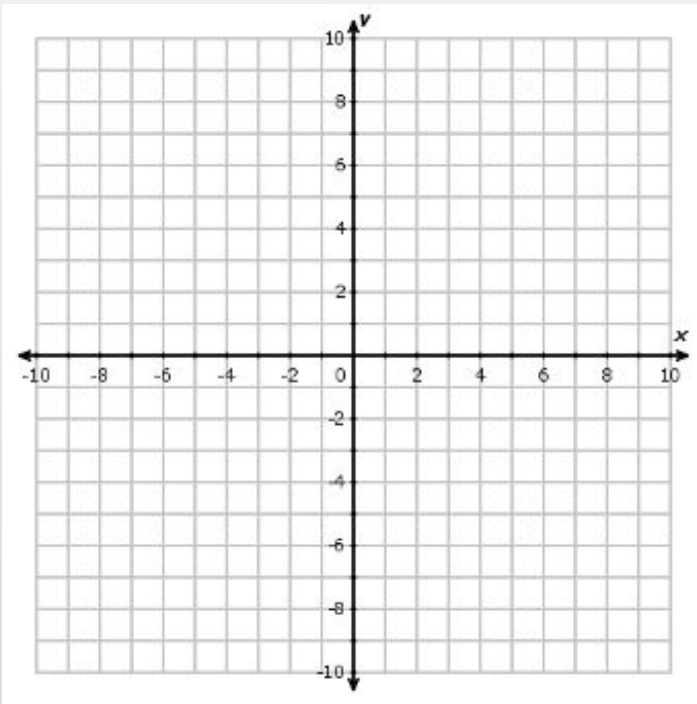
Как уже говорилось ранее, ООП подразумевает такие понятия, как:

- **Абстракция** – выделение значимой информации и исключение из рассмотрения незначимой.
- **Инкапсуляция** – свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе.
- **Наследование** – свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью.
- **Полиморфизм** – свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Структуры

Структура – некоторая совокупность полей и методов работы с ними.

Например, с помощью структуры мы можем задать точку на полкости – у точки 2 параметра: смещение по оси X и по оси Y .



```
1  #include <iostream>
2
3  struct point {
4      int X;
5      int Y;
6  };
7
8  int main() {
9      point A;
10     std::cin >> A.X >> A.Y;
11
12     return 0;
13 }
```

Категории доступа

Основной способ достижения **абстракции** данных.

Для разграничения доступа к **полям** и **методам** (например, если какие-то вычисления должны производиться только внутри структуры) существуют **модификаторы доступа**. К полям и методам, объявленным в этой зоне, нельзя обратиться извне – только из методов самой структуры. Пока что.

Модификатор действует на все поля и методы, идущие после него.

У структур по умолчанию **public**.

```
1  #include <iostream>
2
3  struct point {
4      // public:
5      int X;
6      int Y;
7      private:
8      int sum = X + Y;
9  };
10
11 int main() {
12     point A;
13     std::cin >> A.X >> A.Y;
14     std::cout << A.sum;
15     return 0;
16 }
```

int point::sum
член "point::sum" (объявлено в строке 8) недоступно

Конструкторы и деструкторы

При создании **объекта структуры** или **класса** вызывается специальный метод – **конструктор**. Он может быть:

1. По умолчанию
2. Копирования
3. Перемещения
4. Параметрический

При удалении объекта вызывается **деструктор**.

В то время как конструкторов может быть несколько, **деструктор может быть только один**.

Идиома RAII – получение ресурса есть инициализация.

```
1  #include <iostream>
2
3  struct nums {
4      int *a, *b;
5      nums(int a_n, int b_n) {
6          a = new int(a_n);
7          b = new int(b_n);
8      }
9      ~nums() {
10         delete a;
11         delete b;
12     }
13 };
14
15 int main() {
16     nums d(5, 5);
17
18     return 0;
19 }
```

Методы

```
1  #include <iostream>
2
3  struct nums {
4      int *a, *b;
5      nums(int a_n, int b_n) {
6          a = new int(a_n);
7          b = new int(b_n);
8      }
9      ~nums() {
10         delete a;
11         delete b;
12     }
13
14     void mod() {
15         *a = *a / *b;
16     }
17     int sum() {
18         return *a + *b;
19     }
20 };
```

Метод – это функция, описанная внутри структуры/класса, которые работают с полями класса.

```
22 int main() {
23     nums d(5, 5);
24     std::cout << d.sum() << std::endl;
25     // Вывод 10
26     d.mod();
27     std::cout << d.a << std::endl;
28     // Вывод 1
29     return 0;
30 }
```

Классы

Класс отличается от структуры только лишь тем, что по умолчанию права доступа у него **private**. Классы используют для того, чтобы абстрагировать данные от пользователя и позволить взаимодействовать с ними (если требуется) через так называемые **get-** и **set-методы (getters and setters)**. Таким образом, у программиста есть только определенный интерфейс, задающий все возможные действия над классом из области **public**.

```
1  #include <iostream>
2
3  class nums {
4      //private:
5      int* a, *b;
6  public:
7      nums(int a_n, int b_n) {
8          a = new int(a_n);
9          b = new int(b_n);
10     }
11     ~nums() {
12         delete a;
13         delete b;
14     }
15     int get_a() {
16         return *a;
17     }
```

```
18     int get_b() {
19         return *b;
20     }
21     void set_a(int a_n) {
22         *a = a_n;
23     }
24     void set_n(int b_n) {
25         *b = b_n;
26     }
27     void mod() {
28         *a = *a / *b;
29     }
30     int sum() {
31         return *a + *b;
32     }
33 };
34
```

```
35 int main() {
36
37     nums d(5, 5);
38     std::cout << d.sum() << std::endl;
39     //Вывод 10
40     d.mod();
41     std::cout << d.get_a() << std::endl;
42     //Вывод 1
43     std::cout << d.a << std::endl;
44     return 0;
45 }
46
```

int *nums::a
private:
член "nums::a" (объявлено в строке 5) недоступно