

1) Первые языки программирования были процедурными, при этом программа состояла из одной или нескольких процедур. Процедура — это независимая именованная часть программы, которую после однократного описания можно многократно вызвать по имени из последующих частей программы для выполнения определенных действий. Процедура может рассматриваться просто как функция, которая выполняет определенную задачу, такую как сбор вводимых пользователем данных, выполнение вычислений, чтение и запись файлов, вывод результатов и т. д.

2) Процедура и функция имеют общее название — подпрограмма. В более ранних языках программирования функцию от процедуры отличало то, что функция всегда имела возвращаемое значение, а процедура нет.

3) В объектно-ориентированном программировании атрибуты и методы размещаются в рамках одного объекта, в то время как в процедурном программировании атрибуты и методы обычно разделяются.

ООП решает проблему разделения программного кода и данных посредством инкапсуляции и сокрытия данных. Инкапсуляция обозначает объединение данных и процедур в одном объекте. Сокрытие данных связано со способностью объекта скрывать свои атрибуты от программного кода, который находится за пределами объекта. Только методы объекта могут непосредственно получать доступ и вносить изменения в атрибуты объекта. Объект, как правило, скрывает свои данные, но позволяет внешнему коду получать доступ к своим методам. Как показано на рисунке ниже, методы объекта предоставляют программному коду за пределами объекта косвенный доступ к атрибутам объекта.

4) Из минусов ООП: дольше обрабатывается и больше весит.

5) Первый объектно-ориентированный язык SmallTalk был разработан фирмой Xerox в 1970-х годах.

6) Принцип абстракции позволяет нам скрывать детали и раскрывать только основные черты объекта. **Абстракция** - принцип ООП, позволяющий работать с объектами, не вдаваясь в особенности их реализации

7) **Инкапсуляция** с одной стороны, она объединяет атрибуты и методы в одном объекте. С другой стороны, инкапсуляция обозначает сокрытие данных, то есть невозможность напрямую получить доступ к внутренней структуре объекта, так как это может быть небезопасно.

8) **Наследование** — способ создания класса на основе уже существующего, при котором дочерний класс заимствует атрибуты и методы родительского класса, а также добавляет собственные.

9) Наследование может быть одиночным, а может быть множественным. При множественном наследовании, у класса может быть более одного родителя. В этом случае класс наследует атрибуты и методы всех родительских классов. Достоинства такого подхода в большей гибкости. В то же время множественное наследование — потенциальный источник ошибок, которые могут возникнуть из-за наличия одинаковых имён атрибутов и методов в родительских классах.

10) **Полиморфизм** - объекты разных классов, с разной внутренней реализацией, могут иметь одинаковые интерфейсы. Полиморфизм позволяет выполнять одно действие разными способами, другими словами, он позволяет определять один интерфейс и иметь множество реализаций.