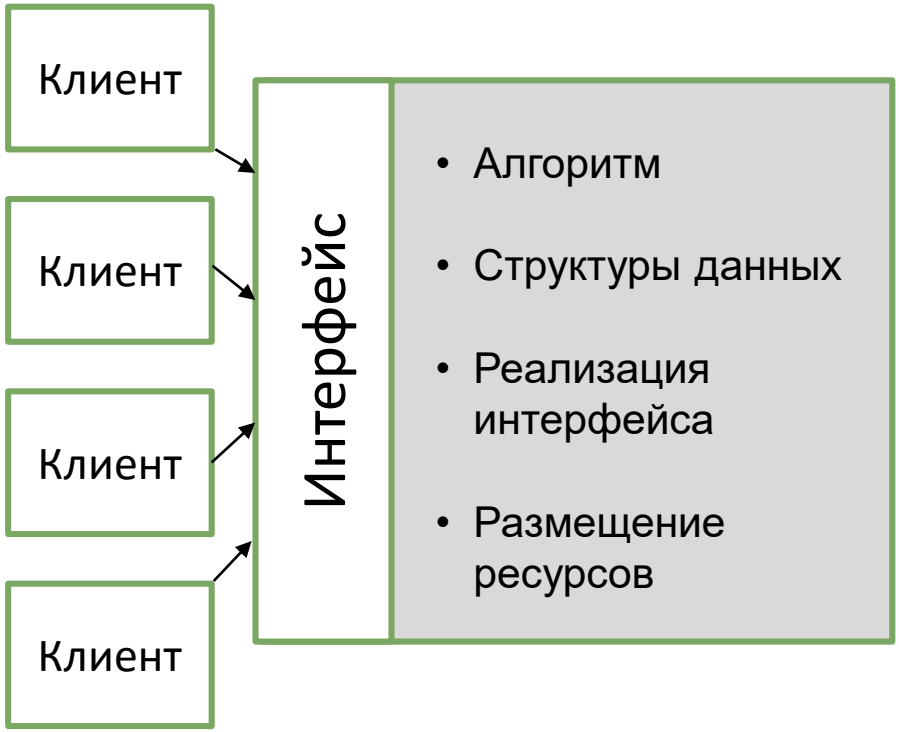
ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Модульность ПО.

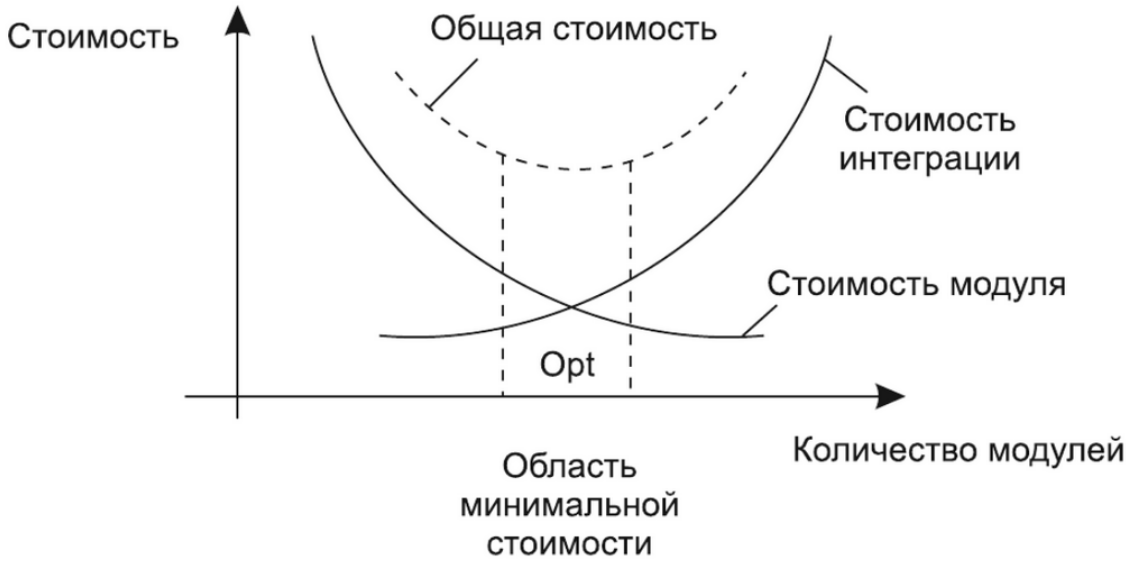
Модульность

- Программная система делится *на именуемые и адресуемые компоненты*, часто называемые *модулями*, которые затем интегрируются для совместного решения проблемы.
- **Модуль** - фрагмент программного текста, являющийся строительным блоком для физической структуры системы.
- Как правило, модуль состоит из интерфейсной части и части-реализации.



Модульность

- Модульность — свойство ПО, обеспечивающее интеллектуальную возможность создания сколь угодно сложной программы.
- Но необходимо учитывать затраты на дальнейшую интеграцию модулей.



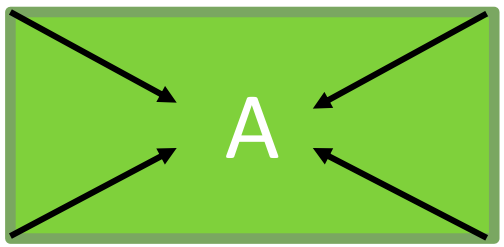
Информационная закрытость

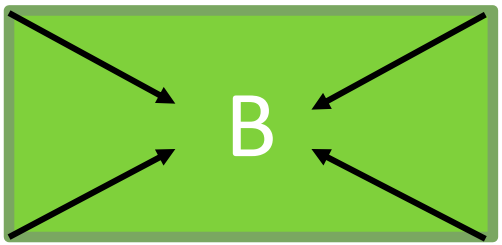
- Информационная закрытость означает, что:
  - Все модули независимы и обмениваются информацией только необходимой для работы
  - Доступ к операциям и структурам данных модуля ограничен.
- Это позволяет:
  - Обеспечить разработку модулей различными независимыми коллективами;
  - Обеспечить легкую модификацию системы
- Идеальный модуль – это черный ящик, содержимое которого не видно клиенту.

Cohesion (Внутренняя связность)

Связность модуля

- **Cohesion (Связность модуля)** – это мера зависимости частей модуля друг от друга.
- Чем больше связность – тем лучше скрыта внутренняя реализация модуля от внешнего мира.
- Выделяют 7 типов связности.





Некорректные типы связности

1. **Связность по совпадению (ВнутрСв = 0).** В модуле отсутствуют явно выраженные внутренние связи.
2. **Логическая связность (ВнутрСв = 1).** Части модуля объединены по принципу функционального подобия. Например, модуль состоит из разных подпрограмм обработки ошибок. *Недостатки:*
    1. Сложное сопряжение;
    2. Большая вероятность внесения ошибок при изменении сопряжения ради одной из функций.
3. **Временная связность (ВнутрСв = 3)** Части модуля не связаны, но необходимы в один и тот же период работы системы. Например, в модуле «Утро» могут быть элементы «умыться», «одеться», «позавтракать». *Недостаток:* сильная взаимная связь с другими модулями, отсюда — сильная чувствительность к внесению изменений.

Корректные типы связности

4. **Процедурная связность (ВнутрСв = 5).** Части модуля связаны порядком выполняемых ими действий, реализующих некоторый сценарий поведения (признак небрежного проектирования).

5. **Коммуникативная связность (ВнутрСв = 7).** Части модуля связаны по данным (работают с одной и той же структурой данных).

6. **Последовательная связность (ВнутрСв = 9).** Выходные данные одной части используются как входные данные в другой части модуля.

7. **Функциональная связность (ВнутрСв = 10).** Части модуля вместе реализуют одну функцию. Функция может быть предельно простой, может быть сложной, то есть распадаться на многие части, но с точки зрения внешнего клиента - это всегда единое действие.

Связанность по совпадению

- Элементы-действия модуля не связаны ни потоком управления, ни потоком данных. Они относятся к разным категориям и разным предметным областям и объединены исключительно т.к. содержат общий код (общие методы сортировки данных и др.)
- Фактически, модуль представляет собой «белый ящик», реализация которого доступна для вызова извне.
- Такой модуль значительно усложняет процесс разработки и поддержки системы.
- Обычно, такие модули не создаются намеренно, а появляются в результате необоснованного изменения модулей с плохой связностью.

Логическая связность

- Элементы логически-связного модуля принадлежат к одной категории, но клиент должен сам выбрать выполняемое действие в зависимости от контекста.

```
Модуль Пересылка сообщения
        переслать по E-mail
        переслать по факсу
        послать в телеконференцию
        послать по FTP
Конец модуля
```

- Фактически, модуль представляет собой «белый ящик».
- Это приводит к плохому интерфейсу модуля, с различными параметрами и методами для выполнения одного и того-же действия

Временная связность

- Элементы такого модуля должны выполняться в примерно в одно время.

```
Модуль Инициализировать систему
        Инициализация жесткого диска
        Инициализация сетевого интерфейса
        Включить индикатор Num Lock
        Инициализация гибких дисков
        Проверить системное время
Конец модуля
```

- Модуль представляет собой «белый ящик» или «серый ящик» в зависимости от качества реализации методов.

Процедурная связность

- Пограничное качество проектирования, может повлечь за собой плохую сопровождаемость кода.
- Модуль состоит из элементов, реализующих независимые действия, но для которых важен порядок передачи управления.

```
Модуль А
        ОбработкаДанныхА
        СчитатьДанные
        ОбработатьДанныеА
        СохранитьДанные
Конец модуля
```

```
Модуль Б
        ОбработкаДанныхБ
        СчитатьДанные
        ОбработатьДанныеБ
        СохранитьДанные
Конец модуля
```

- При реализации может возникнуть дублирование кода, если 2 модуля работают с разными данными.

Коммуникативная связность

- Элементы-обработчики используют одни и те же (может быть внешние) данные или участвуют в формировании общей структуры данных.

```
Модуль Успеваемость студентов
        Сгенерировать отчет по успеваемости
        Выдать отстающих студентов
        Выдать список отличников
Конец модуля
```

- При использовании, клиенту может быть предоставлено избыточное количество данных. Почти всегда разделение коммуникативно-связанного модуля на функционально-связанные приводит к улучшению сопровождаемости.

Последовательная связность

- При последовательной связности элементы-обработчики образуют конвейер для обработки данных – результаты одного являются исходными данными для другого.

```
Модуль Прием и проверка записи
        Прочитать запись из файла
        Проверить контрольные данные
        Удалить контрольные поля
        Вернуть обработанную запись
Конец модуля
```

- Хороший уровень связности. Но возможности повторного использования кода ограничены.

Функциональная связность

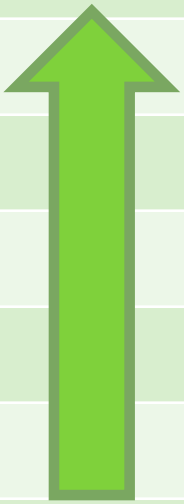
- Функционально связанный модуль содержит элементы, участвующие в выполнении одной и только одной проблемной задачи.

```
Модуль Шифрование данных
        + Зашифровать данные
        + Расшифровать данные
        - Сформировать набор ключей
        ...
Конец модуля
```

- Системы, сформированные из функционально-связанных модулей легче всего сопровождать.

Типы связности

Тип связности	Степень внутренней связности
Функциональная	Самая высокая (лучше всего)
Последовательная	
Коммуникативная	
Процедурная	
Временная	
Логическая	
По совпадению	Самая низкая (хуже всего)

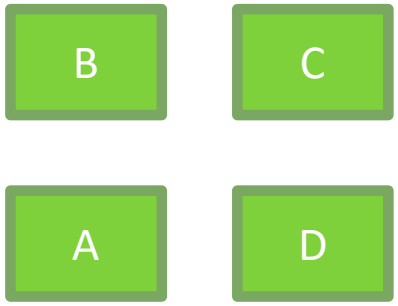


Coupling (Внешняя связанность)

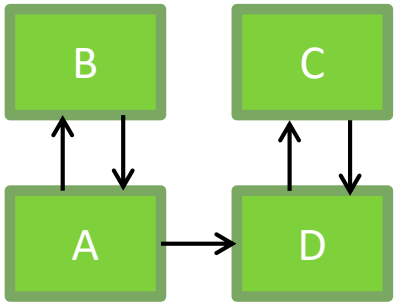
Связанность модулей

- **Связанность модулей** – это мера, определяющая степень независимости между модулями.
- Два модуля называются *сильносвязанными*, если сильно взаимодействуют друг с другом в процессе работы, таким образом, сильно зависят друг от друга.

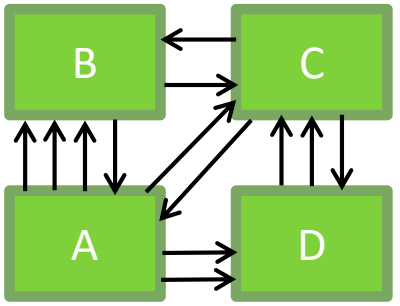
Несвязанные



Слабосвязанные

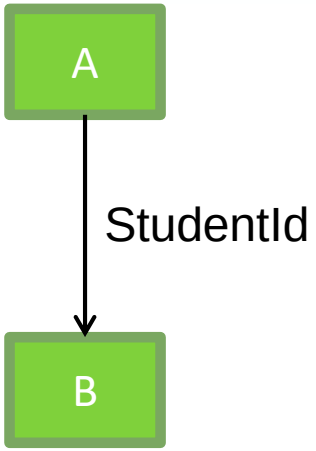


Сильносвязанные

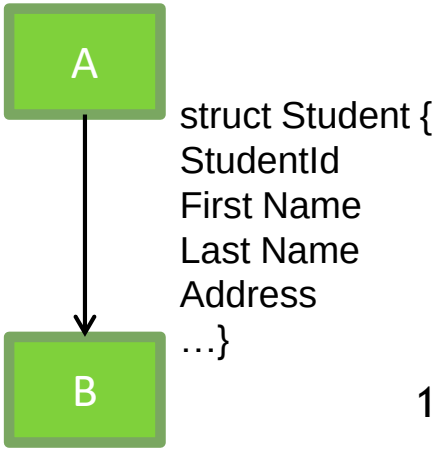


Типы связанности

1. **Связанность по данным.** Коммуникация между модулями происходит исключительно посредством обмена необходимыми элементами данных. При этом необходимо минимизировать объем передаваемых между модулями элементов.



2. **Связанность по образцу.** В качестве параметров коммуникации используются структуры данных. Т.к. часто в структуре могут быть лишние, не используемые данные, такой тип связности менее предпочтителен.



Типы связанности

3. **Связанность по управлению** Здесь начинаются проблемы со связанностью. Модуль А явно управляет функционированием модуля В, отправляя ему сигналы управления (устанавливая флаги или переключатели).
4. **Общая Связанность** Модули имеют возможность чтения и модификации некоторого внешнего ресурса (таблицы базы данных, глобальных переменных, файлов)
5. **Связанность по содержанию** возникает в том случае, если один модуль имеет возможность на прямую модифицировать данные другого модуля, или же есть прямая точка перехода из середины метода одного модуля в исполняемый код другого модуля (посредством Goto и метки в исходном коде).

Типы связанности

Тип связанности	Степень внешней связанности
По данным	Самая низкая (лучше всего)
По образцу	
По управлению	
Общая связанность	
Связанность по содержанию	Самая сильная (хуже всего)

Внутренняя связность и внешняя связанность.

Принцип проектирования модульной структуры ПО

- Если программная система некорректно разбита на модули, это может привести к значительным сложностям при модификации ПО, и, как следствие, ко смерти проекта.
- Необходимо соблюдать
  - **максимальную** степень **внутренней** связности
  - **минимальную** степень **внешней** связанности
- Это позволит спроектировать каждый модуль как «черный ящик», т.е. каждый модуль можно будет разрабатывать или модифицировать независимо от остальных компонентов системы.