**1. Цель разработки модульной структуры**

Основная цель — **упрощение разработки, тестирования и сопровождения программного обеспечения**.  
Модульная структура позволяет:

* Разделять программу на независимые части (модули).
* Уменьшить сложность системы → легче анализировать и изменять.
* Повысить повторное использование кода (reuse).
* Сократить время разработки (разные программисты пишут разные модули).
* Обеспечить локализацию ошибок (ошибка в одном модуле не рушит всю систему).
* Улучшить масштабируемость и расширяемость проекта.

**2. Понятие программного модуля, передачи управления, организации связи по управлению и по данным**

* **Программный модуль** — относительно независимый компонент программы (функция, процедура, класс, пакет), выполняющий определённую задачу.
* **Передача управления** — механизм вызова одного модуля из другого (например, вызов функции/процедуры).
* **Связь по управлению (control coupling)** — один модуль передает другому **информацию о том, как выполнять задачу** (например, через флаги или управляющие параметры).
* **Связь по данным (data coupling)** — обмен **данными** между модулями через параметры, аргументы функций, возвращаемые значения.

**3. Виды связности модулей (Coupling)**

**Связность (coupling)** — это мера зависимости одного модуля от другого.  
Виды (от слабой к сильной, лучше — чем слабее):

1. **Связь по данным (Data Coupling)** — обмен только необходимыми данными (лучший вариант).
2. **Связь по управляющим данным (Control Coupling)** — передача управляющей информации (флагов, условий).
3. **Связь по общей области данных (Common Coupling)** — модули используют общие глобальные переменные.
4. **Связь по содержимому (Content Coupling)** — один модуль напрямую обращается к внутреннему коду или данным другого (самая сильная и плохая).

**4. Виды целостности модулей (Cohesion)**

**Целостность (cohesion)** — это степень того, насколько хорошо модули решают **одну определённую задачу**.  
Виды (от слабой к сильной, лучше — чем сильнее):

1. **Случайная (Coincidental cohesion)** — элементы связаны случайно.
2. **Логическая (Logical cohesion)** — задачи одного типа сгруппированы, выбор действия по условию.
3. **Временная (Temporal cohesion)** — задачи объединены по времени выполнения (например, инициализация).
4. **Процедурная (Procedural cohesion)** — задачи следуют в определенной последовательности.
5. **Коммуникационная (Communicational cohesion)** — модули используют одни и те же данные.
6. **Последовательная (Sequential cohesion)** — выход одного элемента является входом для следующего.
7. **Функциональная (Functional cohesion)** — модуль выполняет одну чётко определенную задачу (наилучший вариант).

**5. Типовые модульные структуры**

* **Иерархическая структура** (дерево модулей: главный модуль вызывает подмодули).
* **Каскадная структура** (цепочка последовательно вызываемых модулей).
* **Многоуровневая структура** (уровни абстракции: интерфейсный слой, бизнес-логика, работа с данными).
* **Сетевая структура** (модули взаимодействуют через сообщения или события).
* **Клиент-серверная структура** (одни модули — клиенты, другие — серверы).

**6. Проектирование модульной структуры с помощью структурных карт**

**Структурная карта** — графическая диаграмма, показывающая:

* иерархию модулей,
* связи управления и данных между ними.

Правила построения:

1. В верхней части карты — главный модуль (основная программа).
2. Ниже — подчинённые модули, вызываемые сверху.
3. Линии соединяют модули, показывая передачу управления или данных.
4. Для данных используют стрелки с подписями (какие данные передаются).
5. Для управляющих связей используют стрелки с параметрами-флагами.

**7. Построение структурных карт с помощью EasyCASE Professional Version 4.21.016**

EasyCASE — CASE-средство (Computer-Aided Software Engineering) для моделирования.  
В этой программе можно:

* Создавать структурные карты (hierarchical diagrams).
* Определять модули (блоки), связывать их стрелками.
* Разделять типы связей: **по управлению** и **по данным**.
* Использовать графические обозначения:
  + прямоугольники — модули,
  + стрелки — вызовы,
  + аннотации — описание параметров.
* Автоматически генерировать документацию по структуре.