

# Модульное программирование

*Модульность в программировании  
подобна честности в политике:  
каждый утверждает, что она - одно  
из его достоинств, но кажется,  
никто не знает, что она собой  
представляет ...*

*Йодан Э.*

# Основные понятия

- Модульное программирование – это такой способ программирования, при котором вся программа разбивается на группу компонентов, называемых модулями, причем каждый из них имеет свой контролируемый размер, четкое назначение и детально проработанный интерфейс с внешней средой.
- Альтернатива модульности – монолитная программа

# Основные концепции:

- В основе модульного программирования лежат три основные концепции:
  - *Принцип утаивания информации Парнаса*  
При разработке программ формируется список проектных решений, которые особенно трудно понять или которые, скорее всего, будут меняться. Затем определяются отдельные модули, каждый из которых реализует одно из указанных решений. Большие программы должны использовать модули без каких-либо предварительных знаний об их внутренней структуре. Примерами удачных модулей могут служить программы ППП (пакетов прикладных программ) и стандартные процедуры.
  - *Сборочное программирование Цейтина.*  
Модули – это программные «кирпичи», из которых строится программа.



# Основные концепции:

- *Аксиома модульности Коуэна*  
Модуль – независимая программная единица, служащая для выполнения некоторой определенной функции программы и для связи с остальной частью программы. Программная единица должна удовлетворять следующим условиям:
  - Блочность организации
  - Синтаксическая обособленность
  - Семантическая независимость
  - Общность данных
  - Полнота определения

# Характеристики модуля (Майерс, 1980)

- Размер модуля
- Связность (прочность) модуля
- Сцепление модуля с другими модулями
- Рутинность (независимость от предыдущих обращений) модуля



# Размер модуля

- Модуль не должен превышать 60 строк  
В результате его можно поместить на одну страницу распечатки или легко просмотреть на экране монитора

# Связность модуля

- Связность модуля – мера независимости его частей.

Чем выше связность, тем больше связей он «упрятывает» в себе

- Типы связности:

- **Функциональная**

Модуль с функциональной связностью реализует одну какую-нибудь функцию и не может быть разбит на два модуля с теми же типами связности

- **Последовательная**

# Связность модуля

- Типы связности:

- Последовательная

Модуль с такой связностью может быть разбит на последовательные части, выполняющие независимые функции, но реализующие совместно единственную функцию (например, оценка , а затем обработка данных)



# Связность модуля

- Типы связности:

- Информационная

Модуль с информационной связностью – это модуль, выполняющий несколько операций или функций над одной и той же структурой данных, которая считается неизвестной вне этого модуля (применяется для реализации, например, абстрактных типов данных таких как стек, очередь и др.)

# Связность модуля: Следует избегать

- **Временной связности**
  - когда объединяются действия, связанные со временем (например, действия, которые должны быть выполнены в один и тот же момент времени)
- **Логической связности**
  - когда в модуль объединяются действия по признаку их некоторого подобия (например, функции для проверки корректности входных данных для всей программы)
- **Случайной связности**
  - когда действия объединяются произвольным образом
- **Процедурной связности**
  - когда действия сгруппированы вместе только потому, что они выполняются в течение одной и той же части процесса



# Сцепление модулей

- Сцепление – мера относительной независимости модулей от других модулей.

Независимые модули могут быть модифицированы без переделки других модулей.

Чем слабее сцепление модуля, тем лучше.



# Типы сцепления

- *Независимые модули* – идеальный случай.  
В этом случае модули ничего не знают друг о друге.

Взаимодействие модулей организуется через их интерфейсы, когда выходные данные одного модуля передаются на вход другого.

Достичь такого сцепления очень сложно, и в большинстве случаев не нужно.

# Типы сцепления

- *Сцепление по данным (параметрическое)* – это сцепление, когда данные передаются модулю как значения его параметров или как результат его обращения к другому модулю для вычисления некоторой функции (Этот тип сцепления реализуется в языках программирования при обращении к функциям)
- **Разновидности этого сцепления:**
  - Сцепление по простым элементам данных
  - Сцепление по структуре данных (оба модуля при этом должны знать о внутренней структуре данных)



# Типы сцепления

## Не рекомендуется использовать:

- *Сцепление по управлению* – это сцепление, в котором один модуль управляет решениями внутри другого с помощью передачи флагов, переключателей и т. п.  
В этом случае один модуль должен достаточно хорошо знать структуру вызывающего модуля
- *Сцепление по внешним ссылкам* – возникает, когда у одного модуля есть доступ к данным другого
- *Сцепление по кодам* – возникает, когда коды инструкций модулей перемежаются друг с другом (внутренняя область одного модуля доступна другому)



# Рутинность модуля

- *Рутинность модуля* – это независимость модуля от предыдущих обращений к нему.

Будем называть модуль *рутинным*, если результат его работы зависит только от количества переданных параметров (а не от количества обращений)

# Рутинность модуля

- В некоторых случаях возникает необходимость в создании модулей, которые должны сохранять предысторию (не рутинные)
- В выборе степени рутинности пользуются тремя рекомендациями:
  - В большинстве случаев делаем модуль рутинным
  - Зависящие от предыстории модули следует использовать только в том случае, когда необходимо сцепление по данным
  - В спецификации зависящего от предыстории модуля должна быть сформулирована эта зависимость



# Свойства модуля

- *На модуль можно ссылаться с помощью имени модуля.*
- *Модуль должен иметь один вход и один выход*
- *Модуль должен быть сравнительно невелик*
- *Возможность сепаратной компиляции*
- *Модуль может вызвать другой модуль или сам себя*
- *Модуль должен возвращать управление тому, кто его вызвал*
- *Модуль не должен сохранять историю своих вызовов для управления своим функционированием*



# Преимущества модульного программирования:

- *Функциональные компоненты модульной программы могут быть написаны и отлажены порознь*
- *Модульную программу проще проектировать, легче сопровождать и модифицировать*
- *Становится проще процедура загрузки в оперативную память большой программы, требующей сегментации*

# Недостатки модульного программирования:

- *Может увеличиться время компиляции и загрузки.*
- *Может увеличиться время исполнения программы.*
- *Может возрасти объем требуемой памяти.*
- *Организация межмодульного взаимодействия может оказаться довольно сложной.*

*Для современных компьютеров первые три недостатка несущественны.*



# Стандартные модули

- Разработка и использование стандартных библиотечных программ является одним из путей построения модульного программирования
- Преимущества стандартных модулей:
  - 1) стандартные модули экономят время программирования;
  - 2) они также могут экономить память компьютера и выполняться максимально быстро;
  - 3) использование стандартных модулей защищает от ошибок программирования.



# Стандартные модули

- Недостатки:
  - Нужный стандартный модуль иногда бывает трудно найти. Еще труднее – подробную документацию к нему
  - Стандартный модуль может оказаться более универсальным, чем это нужно пользователю
  - Стандартный модуль может быть написан на другом языке

Каждый программист решает самостоятельно использовать ему стандартные модули или разрабатывать свой собственный.

## Подпрограммы (функции)

- Подпрограммы также являются средством для построения модульных программ
- Не всякая подпрограмма является модулем. Модуль должен удовлетворять перечисленным выше характеристикам и свойствам.