Разработка структуры программы. Модульное программирование.

Отвагин Алексей Владимирович,
доцент каф. ЭВМ, к.т.н., а. 505-5

Содержание

- Понятие модульного программирования
- Основные характеристики модуля
- Методы разработки структуры программы
- Контроль структуры

Цель модульного программирования

- Упрощение структуры ПС
- Отображение архитектуры ПС
- Повышение качества разработки
- Сокращение пространства поиска ошибок

Понятие модульного программирования

- Модуль – любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт
- Модуль может входить во многие процессы, если он должным образом документирован
- Модульное программирование – разработка ПС в виде совокупности модулей

Процессы модульного программирования

- Разделение – выделение отдельных модулей на основе их характеристик (отношения между элементами, интенсивности взаимодействия и т.д.)
- Программирование – разработка модулей индивидуальными программистами или группами (проходит параллельно)
- Интеграция – проверка взаимодействия модулей в реальной среде

Схема модульного программирования

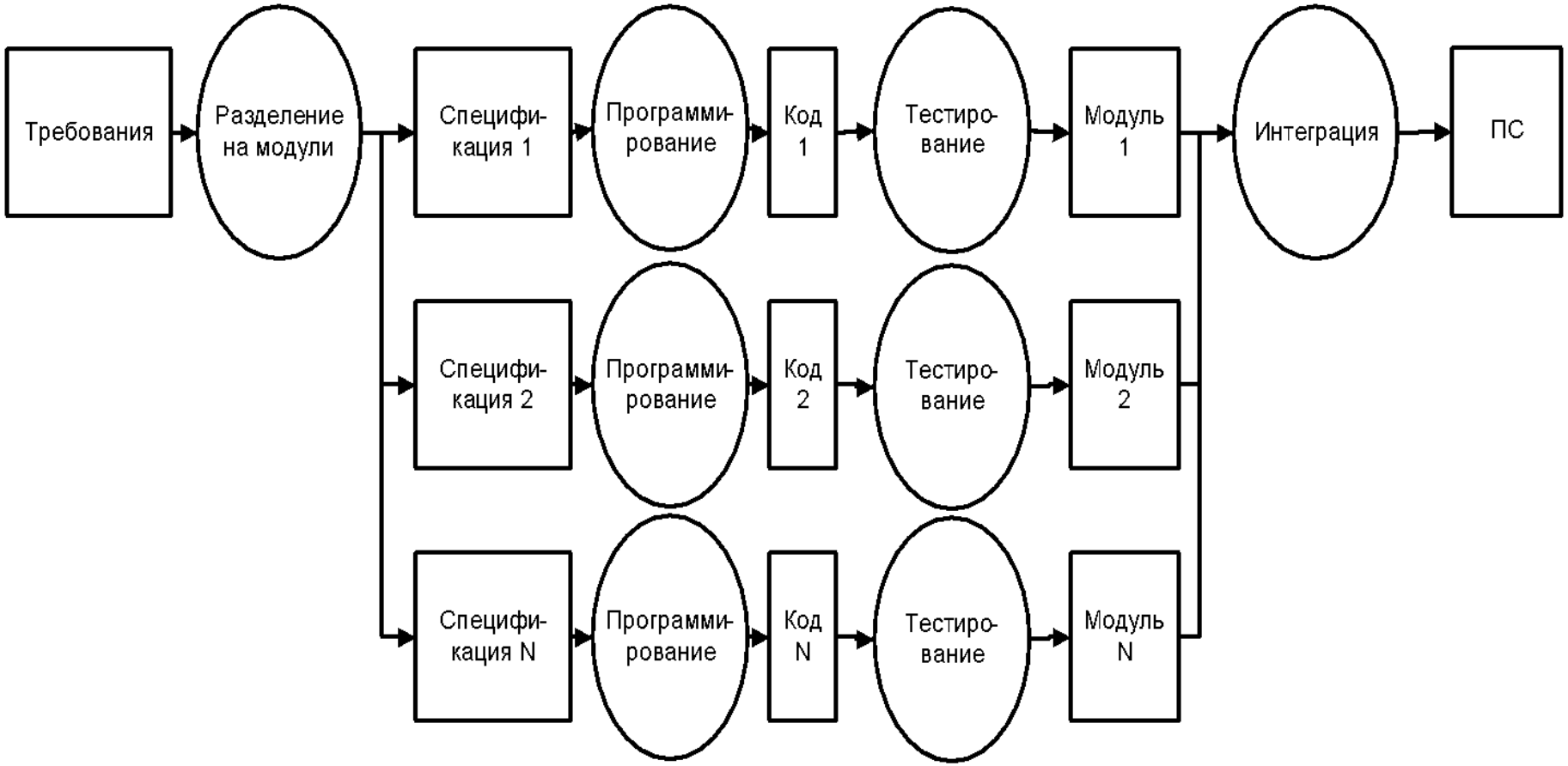


Схема реализации модуля

Свойства модулей

- Являются критериями оценки качества модуля
- Часто выражаются количественно
- Основные свойства:
  - Размер
  - Прочность
  - Сцепление
  - Рутинность

Размер модулей

- Измеряется числом содержащихся операторов или строк (в зависимости от языка)
- Маленький модуль – большие потери на документирование
- Большой модуль – неудобно изменять и повторно транслировать

Прочность модулей

- Определяет меру внутренних связей между элементами модуля
- Различают несколько степеней прочности
- Прочность по совпадению – модуль, между элементами которого нет осмысленных связей

Прочность модулей

- Функционально прочный модуль – выполняет одну определенную функцию
- Может содержать вспомогательные функции
- Рекомендуется к использованию в структурном проектировании

Прочность модулей

- Информационно прочный модуль – выполняет операции над определенной структурой данных, известной только внутри модуля
- Высшая степень прочности
- Рекомендуется к использованию в ООП

Сцепление модулей

- Определяет меру зависимости по данным от других модулей и характеризует способ передачи данных
- Сцепление по содержимому – прямые ссылки на содержимое другого модуля (константы)
- Сцепление по общей области – использование общей области памяти для взаимодействия
- Параметрическое сцепление – передача данных или их возврат в виде параметров при обращении

Рутинность

- Определяет независимость модуля от предыстории обращений к нему
- Рутинный модуль – эффект обращения зависит только от переданных параметров
- Зависящий от предыстории модуль – эффект обращения определяется внутренним состоянием модуля

Рекомендации по обеспечению рутинности

- Желательно использовать только рутинные модули
- Зависящие от предыстории модули используются только для обеспечения параметрического сцепления
- Зависимости от предыстории должны четко отражаться в спецификации

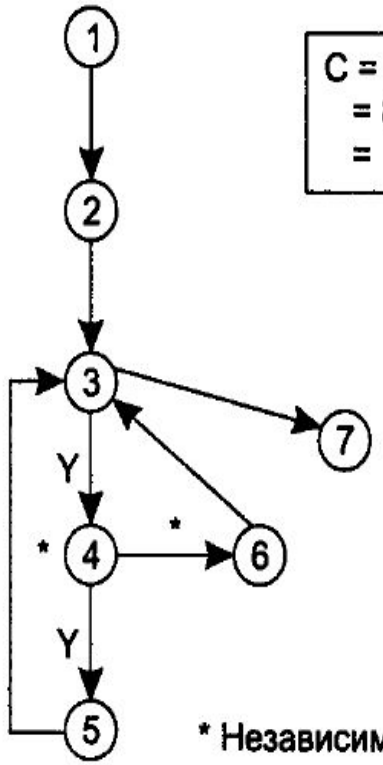
Метрики IEEE для оценки модуля

- Пусть s – это число отдельных операторов в программе (например, *,+,-)
- Пусть v – число отдельных операндов в программе
- S – общее число операторов, V – общее число операндов
- Оценки:
  - Предполагаемая длина программы
    s*(log S) + v*(log V)
  - Сложность программы
    s*V/(2*v)

Цикломатическая метрика

- Пусть N – это число операторов в программе
- Пусть E – число переходов между операторами
- Цикломатическая сложность: E-N+1

```
1 int x=anX;
2 int y=aY;
3 while(!(x==y)){
4     if(x>y)
5           x=x-y;
    else
6           y=y-x;
7 }
8 ...println(x);
```



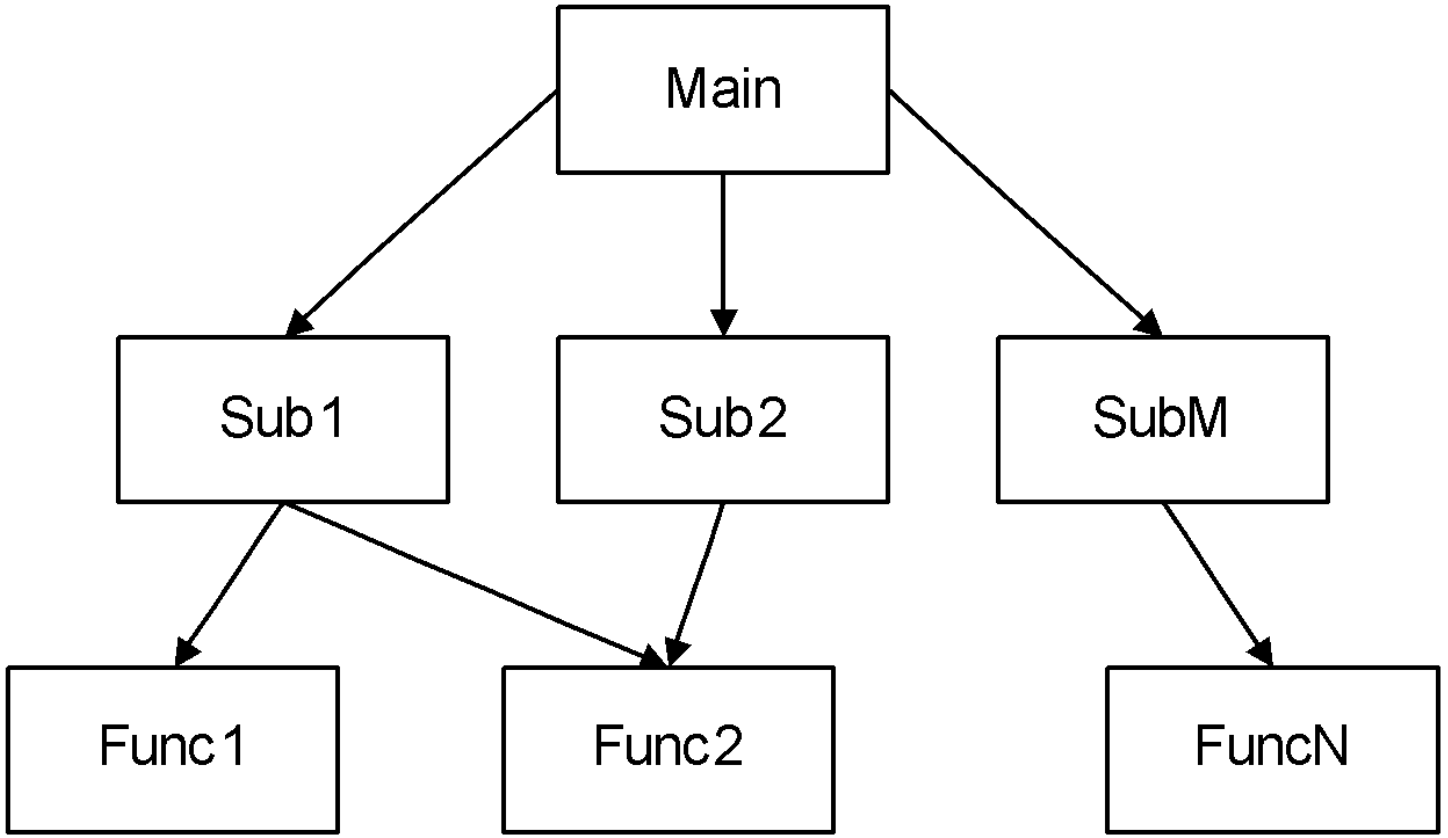
C = E – N + 1
= 8 – 7 + 1
= 2

Цикломатическая сложность

Разработка структуры программы

- Структура представляется в виде дерева, где каждый модуль может использовать или использоваться другими
- Деревья могут иметь сросшиеся ветви

Пример структуры программы



Спецификация модуля

- Синтаксическая
- Функциональная или семантическая
- Представляет модуль законченной единицей
- Может быть формализована

Методы разработки структуры ПС

- Определяются направлением и способом обхода дерева
- Восходящая и нисходящая разработка
- Комбинированный метод
- Направленная разработка

Восходящая разработка

- Строится структура в виде дерева
- Движение начинается от наименьших модулей
- Тестирование в том же порядке
- Порядок разработки кажется естественным

Недостатки восходящей разработки

- Для программирования модуля не обязательна готовность всех используемых им модулей (можно заменить заглушками)
- Часто возникает проблема перепроектирования модулей высшего уровня
- Для отладки модуля создается специальный стенд (программа, моделирующая внешнюю среду)

Нисходящая разработка

- Строится структура в виде дерева
- Движение начинается от головного модуля
- Модуль начинает разрабатываться, если уже готов модуль, обращающийся к нему
- Тестирование в том же порядке

Преимущества нисходящей разработки

- Использование имитаторов или заглушек
- Естественное формирование внешней среды
- Возможность реализовать сложные тесты
- Меньшая вероятность перепроектирования

Другие подходы

- Восходящая и нисходящая разработка – классические подходы, требующие наличия готовой структуры программы
- Структура может формироваться в ходе разработки:
  - Конструктивный подход
  - Архитектурный подход

Конструктивный подход

- Является модификацией нисходящей разработки
- Выделяются направления разработки и формируются ветви дерева
- Отсутствующие модули и ветви заменяются имитаторами

Архитектурный подход

- Является модификацией восходящей разработки
- Основная цель – повышение уровня языка разработки за счет создания крупных абстракций
- Готовые модули параметризуются и допускают повторное использование

Комбинированный метод (сэндвич)

- Дерево обходят в двух направлениях – сверху и снизу
- Сочетает достоинства и недостатки классических подходов
- Достаточно широко применяется

Классификация методов разработки структуры