Объектно-ориентированное программирование

Введение в концепции и принципы

Александра Волосова,
к.т.н., доцент кафедры
ИУ5

Презентация разработана в рамках гранта «Грант на обучение студентов по образовательным программам высшего образования для топ-специалистов в сфере информационных технологий. Договор № 70-2025-000850 с АНО «Аналитический центр при Правительстве Российской Федерации»

Объектно-ориентированное программирование:

- Объектная модель как ответ на сложность ПО
- От алгоритмической к объектной декомпозиции
- ООА, ООД и ООП как единый подход

Ключевая идея: моделирование реального мира через взаимодействующие объекты

Проблема сложности программного обеспечения

Промышленные системы превышают возможности человеческого интеллекта:

- Сложность предметной области
- Трудности управления процессом разработки
- Гибкость программного обеспечения
- Проблемы описания дискретных систем

Примеры сложных систем в природе

1. Персональный компьютер:
   - Иерархия компонентов
   - Разные уровни абстракции

2. Растения:
   - Корни, стебли, листья
   - Клетки как базовые блоки

3. Животные:
   - Клетки → ткани → органы → системы

4. Социальные институты:
   - Иерархия организаций

5. Вещество:
   - Галактики → звезды → атомы → кварки

Пять признаков сложной системы

1. Иерархическая структура
2. Относительность элементарных компонентов
3. Сильные внутренние и слабые внешние связи
4. Ограниченное число типов компонентов
5. Развитие из простых работающих систем

Декомпозиция систем: два подхода

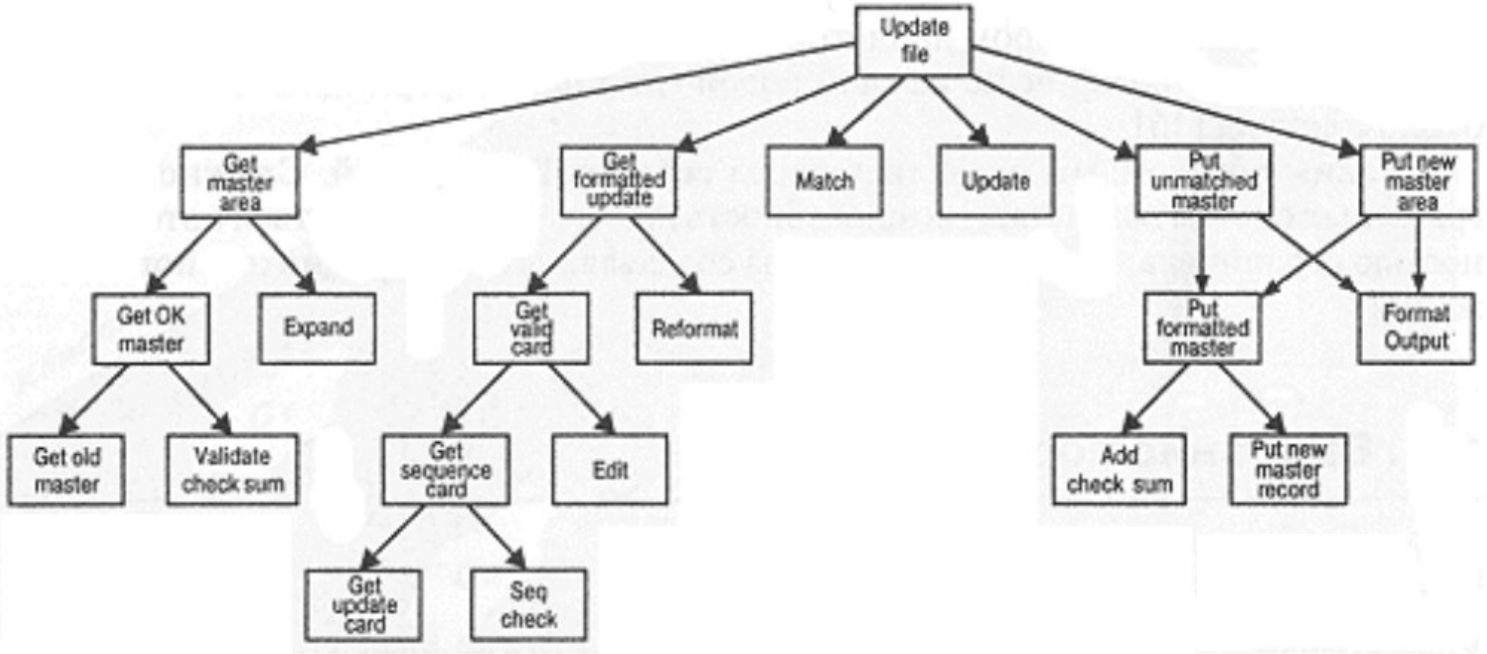
Алгоритмическая декомпозиция:

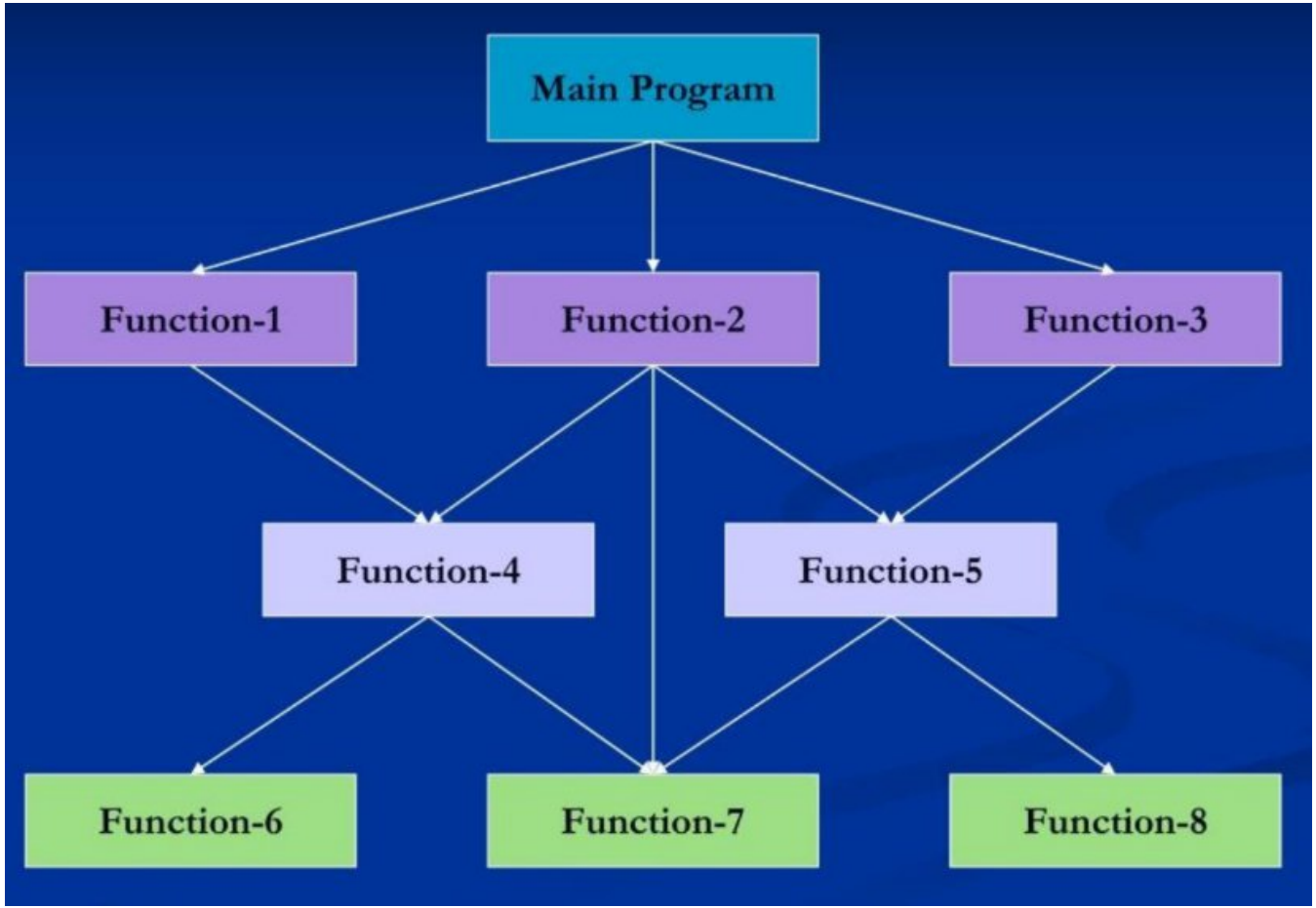
- Структурное проектирование
- Разделение по шагам выполнения
- Последовательность действий

Объектно-ориентированная декомпозиция:

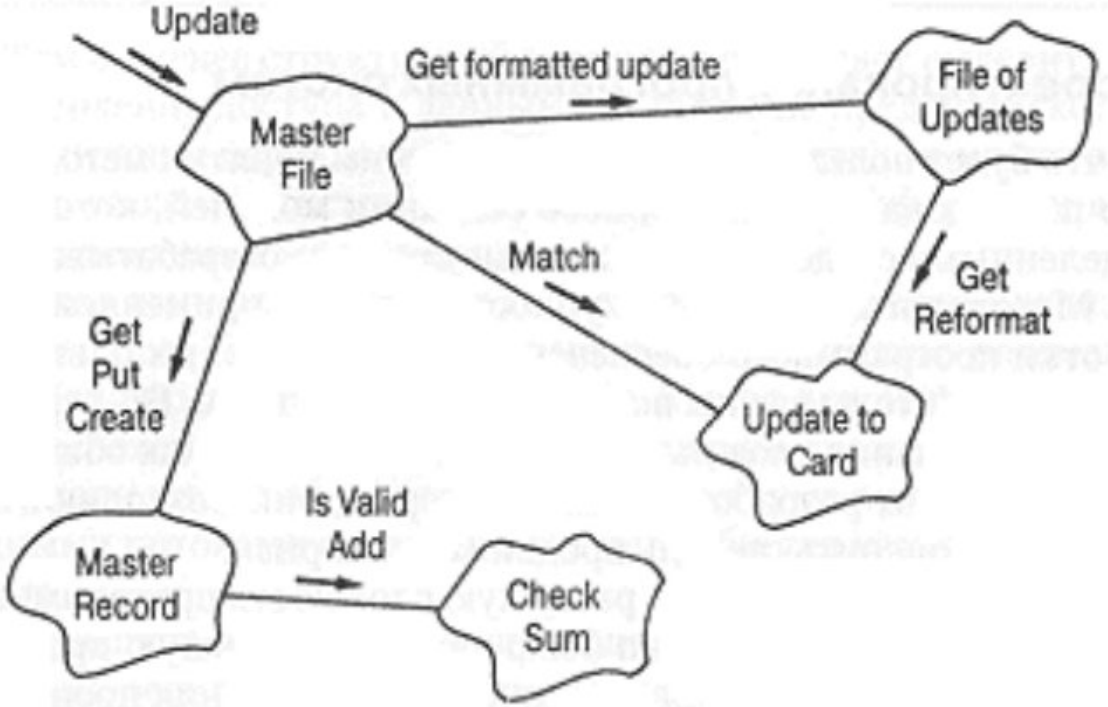
- Разделение по сущностям предметной области
- Взаимодействующие агенты

Алгоритмическая декомпозиция

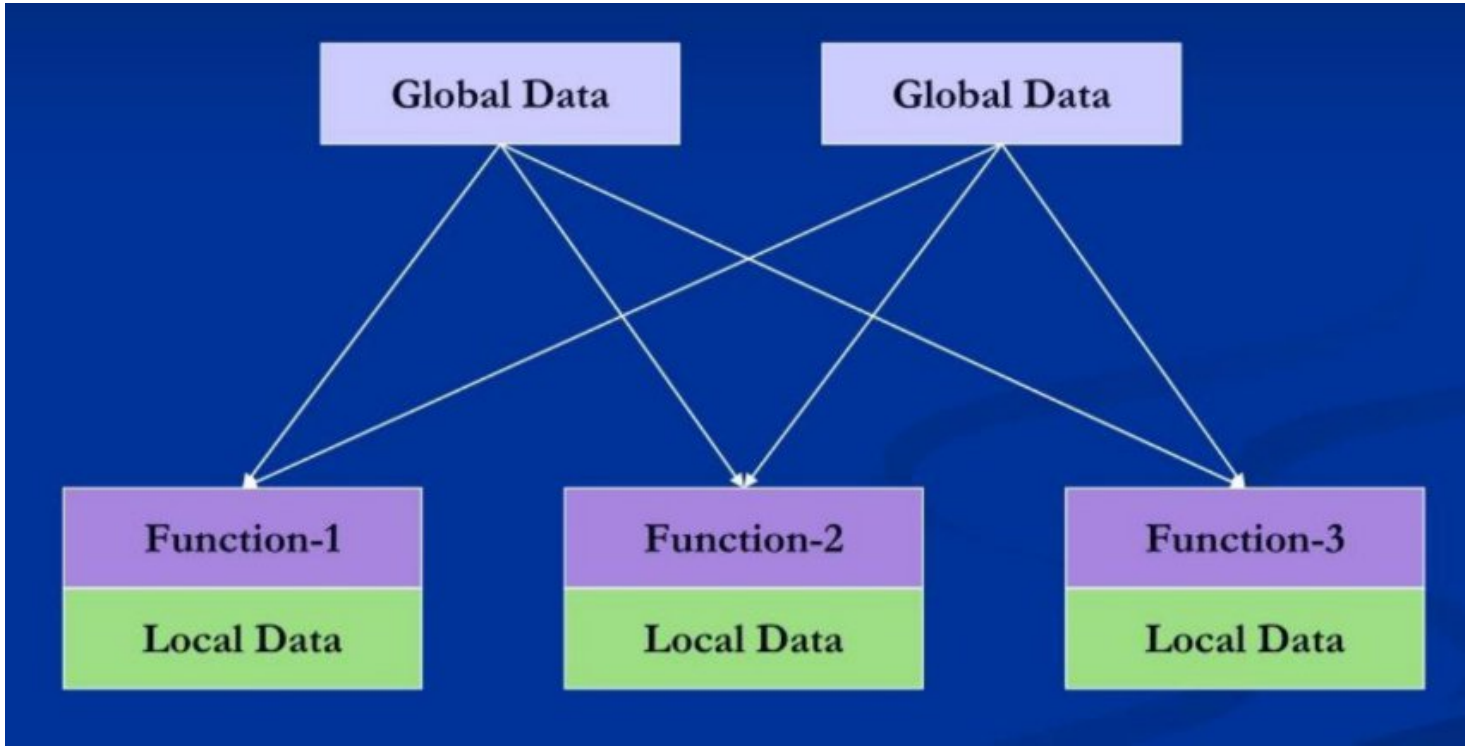



Main Program
Function-1
Function-2
Function-3
Function-4
Function-5
Function-6
Function-7
Function-8

Объектно-ориентированная композиция



Объектно-ориентированная композиция



Роль абстракции в управлении сложностью

Абстрагирование — игнорирование несущественных деталей

Принципы:
- Работа с обобщенными моделями
- Пример: изучение фотосинтеза
- Объекты как информационные единицы

Ограничения восприятия:
- Рабочая память: 7±2 элемента
- Абстракция позволяет оперировать более крупными единицами

Роль иерархии в организации сложных систем

Структура классов:
- Иерархия «is-a» (является)
- Наследование
- Пример: FruitGrowingPlan является GrowingPlan

Структура объектов:
- Иерархия «part-of» (часть целого)
- Агрегация
- Пример: Garden содержит Plant

Преимущества:
- Разделение общих и уникальных свойств
- Упрощение понимания системы

Парадигмы программирования

1. Процедурно-ориентированная:
   - Алгоритмы как основной элемент
   - Языки: C, Pascal, FORTRAN

2. Объектно-ориентированная:
   - Классы и объекты
   - Языки: C++, Java, Smalltalk

3. Логико-ориентированная:
   - Цели, предикаты
   - Языки: Prolog

4. Ориентированная на правила:
   - Правила «если-то»

5. Ориентированная на ограничения:
   - Инвариантные соотношения

Процедурно-ориентированная парадигма

Основные характеристики:
- Основа: алгоритмы и подпрограммы
- Декомпозиция по шагам выполнения
- Последовательное выполнение

Исторические языки:
- FORTRAN (1957)
- COBOL (1959)
- Pascal (1970)
- C (1972)

Ограничения:
- Сложность больших систем
- Проблемы с повторным использованием
- Слабые механизмы абстракции

Объектно-ориентированная парадигма

Основные характеристики:
• Основа: классы и объекты
• Декомпозиция по сущностям
• Взаимодействие объектов

Ключевые принципы:
1. Инкапсуляция
2. Наследование
3. Полиморфизм

Преимущества:
• Естественность моделирования
• Управление сложностью
• Повторное использование

Исторический контекст развития ООП

1960-е:
- Simula 67 (1967) - первый ОО-язык

1970-е:
- Smalltalk (1972)
- CLU (1974)
- Ada (1979)

1980-е:
- C++ (1983)
- Objective-C (1984)
- Eiffel (1986)

1990-е - …:
- Java (1995)
- Python (1991)

Эволюция языков: Поколения 1-2

Первое поколение (1954-1958):
- FORTRAN I, ALGOL-58
- Математические формулы
- Освобождение от ассемблера

Второе поколение (1959-1961):
- ALGOL-60, COBOL, Lisp
- Подпрограммы, типы данных
- Алгоритмические абстракции

Эволюция языков: Поколения 3-4

Третье поколение (1962-1970):
• Pascal, Simula, PL/I
• Классы, абстрактные данные
• Поддержка абстракции данных

Потерянное поколение (1970-1980):
• ~2000 языков создано
• Немногие выжили

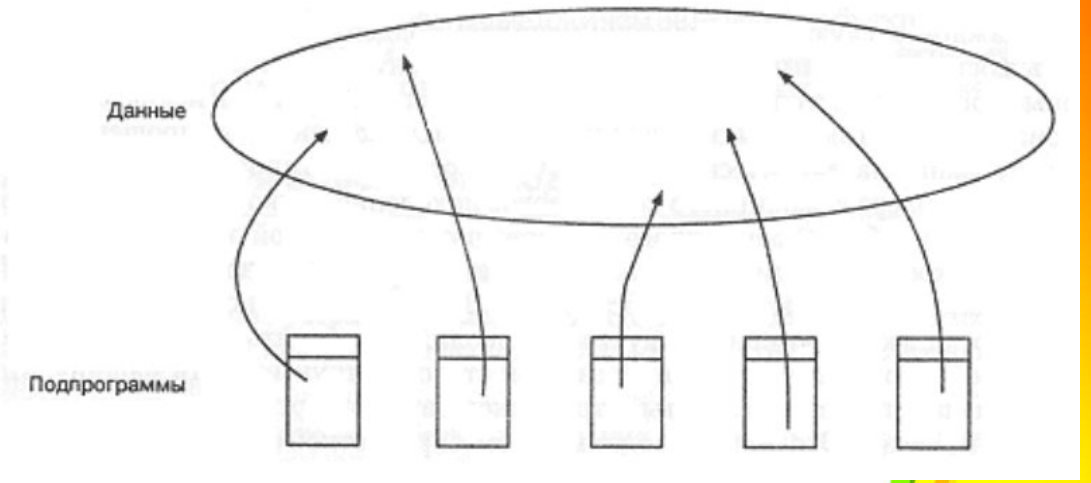
Современные ОО-языки:
Java, C#, C++, Python, Kotlin, Swift, TypeScript, Ruby, PHP (с версии 5+), Dart и многие другие.
Большинство из них — мультипарадигмальные

Топология языков первого поколения

Основные элементы:
- Подпрограммы
- Глобальные данные
- Простая структура

Примеры:
- FORTRAN I/II
- Ранний COBOL

Проблемы:
- Перекрестные связи
- Сложность управления
- Отсутствие инкапсуляции



Топология языков второго поколения

Новые возможности:
- Механизмы передачи параметров
- Вложенность подпрограмм
- Структурное программирование

Примеры:
- ALGOL-60
- Pascal

Развитие:
- Процедурная абстракция
- Структурное проектирование

Не решена проблема «в большом»

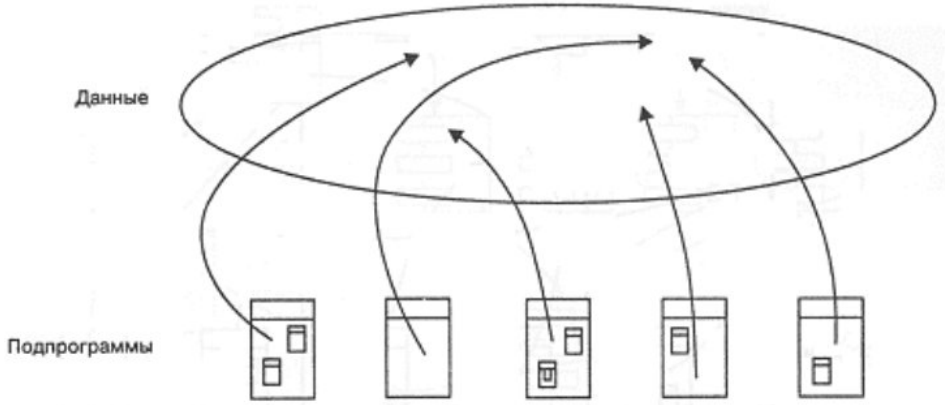
Топология языков третьего поколения

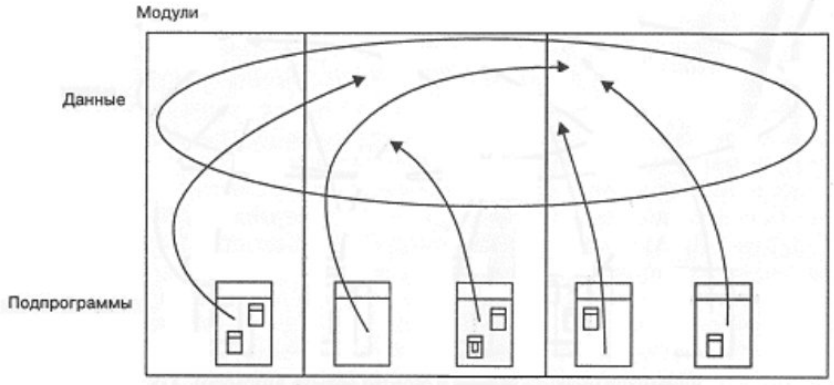
Введение модулей:
- Отдельная компиляция
- Физическое разделение
- Независимая разработка

Структура:
- Данные + подпрограммы
- Слабая поддержка интерфейсов

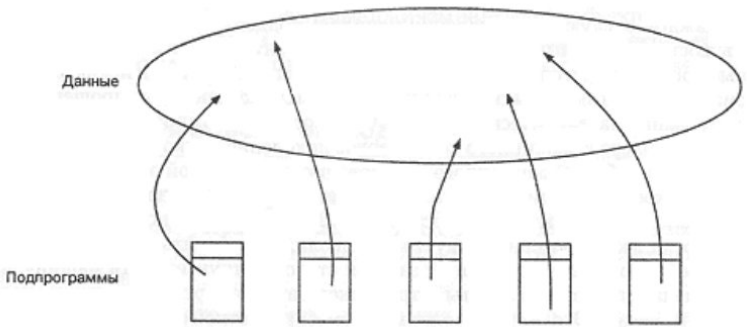
Примеры:
- Modula-2
- Ada (ранние версии)

Программирование «в большом»

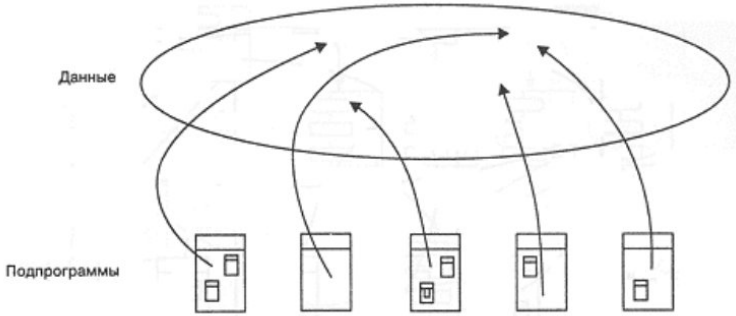




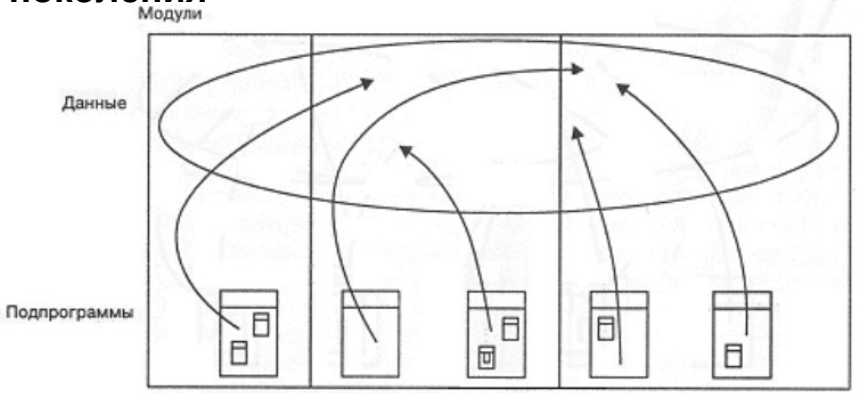
Топология языков первого и начала второго поколения



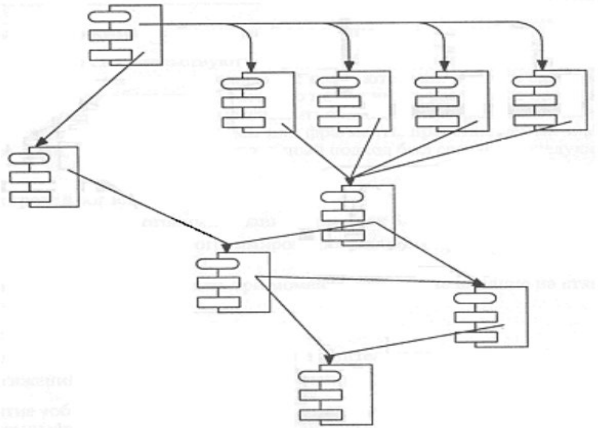
Топология языков позднего второго и раннего третьего поколения



Топология языков конца третьего поколения



Топология малых и средних приложений в объектных и объектно-ориентированных языках



Топология ОО-языков

Основные элементы:
- Классы
- Объекты
- Модули как контейнеры

Структура:
- Граф объектов
- Не дерево подпрограмм
- Минимизация глобальных данных

Аналогия:
- Процедурные = глаголы
- ОО = существительные

Примеры: Smalltalk, C++, Java

Ключевые понятия ООП

1. Объект:
• Экземпляр класса
• Состояние, поведение, идентичность

2. Класс:
• Шаблон для объектов
• Определяет структуру и поведение

3. Наследование:
• Отношение «is-a»
• Повторное использование

4. Инкапсуляция:
• Сокрытие реализации

5. Полиморфизм:
• Разные реализации интерфейса

Определение ООП

Объектно-ориентированное программирование — это методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования

Три обязательных условия:

1. Объекты как базовые элементы (иерархия "part-of")
2. Объекты — экземпляры классов
3. Иерархия классов (иерархия "is-a")

Основные концепции ООП

1. Сложность — основная проблема
   - Управляема через объектную модель

2. Историческая эволюция
   - От процедурного к объектному подходу

3. Многопарадигменность
   - ООП — одна из парадигм

4. Фундаментальные концепции
   - Объекты, классы, наследование, инкапсуляция, полиморфизм