ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

ЛЕКЦИЯ №1

ПОНЯТИЕ ПАРАДИГМА

- **Парадигма** («пример, модель, образец» — «сравниваю») в философии науки — означает совокупность явных и неявных (и часто не осознаваемых) предпосылок, определяющих научные исследования и признаваемых на данном этапе развития науки, а также универсальный метод принятия эволюционных решений.
- **Парадигма** — совокупность фундаментальных научных установок, представлений и терминов, принимаемая и разделяемая научным сообществом и объединяющая большинство его членов. Обеспечивает преемственность развития науки и научного творчества.
- **Парадигма программирования** — это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию). Это способ концептуализации, определяющий организацию вычислений и структурирование работы, выполняемой компьютером.

ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

- Все они - всего лишь различные инструменты.
- Каждый из этих инструментов по-своему хорош.
- Различные методики программирования дают разный выигрыш для решения задач разных классов:

1. эффективность программного обеспечения на современных ЭВМ
2. общие затраты на разработку программного обеспечения

ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ

- **Императивное программирование**
- **Паралелизм. Паралельное и событийно-управляемое программирование**
- **Объектно-ориентированное программирование**
- **Функциональное программирование**
- **Логическое программирование**
- **Констрэйтное программирование**
- **Алгебраическое программирование**
- **Инсерционное моделирование**
- **и другие (WiKi Programming paradigms)**

ИМПЕРАТИВНОЕ ПРОГРАММИРОВАНИЕ

- Императивное программирование выигрывает любой другой методологии в эффективности реализации. Причина: аппаратная организация ПК.
- Стоит ли говорить о том, почему императивное программирование - практически наиболее "популярное"?
- Одна из характерных черт императивного программирования - наличие переменных с операцией "разрушающего присвоения«
- Про наш мир можно сказать, что он локально императивен.
- Императивное программирование наиболее пригодно для реализации небольших подзадач, где очень важна скорость исполнения на современных компьютерах (+работа с внешними устройствами).

Описывает процесс вычисления в виде инструкций, изменяющих состояние программы. Императивная программа очень похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть это последовательность команд, которые должен выполнить компьютер.

Особенности:

- в исходном коде программы записываются инструкции (команды);
- инструкции должны выполняться последовательно;
- при выполнении инструкции данные, полученные при выполнении предыдущих инструкций, могут читаться из памяти;
- данные, полученные при выполнении инструкции, могут записываться в память.

ИМПЕРАТИВНОЕ ПРОГРАММИРОВАНИЕ

Языки поддерживающие данную парадигму:

- Языки ассемблера
- Fortran
- Algol
- Cobol
- Pascal
- C
- C++
- Ada

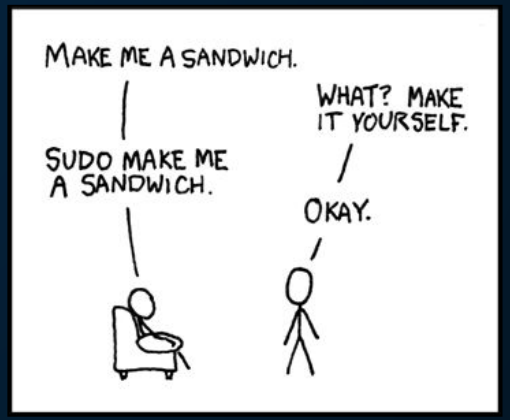
Декларативная парадигма программирования

Декларативное программирование — это парадигма программирования, в которой задаётся спецификация решения задачи, то есть описывается, что представляет собой проблема и ожидаемый результат. В общем и целом, декларативное программирование идёт от человека к машине, тогда как императивное — от машины к человеку. Как следствие, декларативные программы не используют понятия состояния, то есть не содержат переменных и операторов присваивания.

С декларативным программированием мы говорим компьютеру «что», но не «как». Мы описываем результат, который мы хотим, а детали того, как выполнить его, оставлены интерпретатору языка.

- Языки поддерживающие данную парадигму:

- Lisp
- Prolog
- Sql
- Html



ОБЪЕКТНО-ОРИЕНТИРОВАНАЯ ПАРАДИГМА ПРОГРАММИРОВАНИЯ

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

Разделяется на

- Программирование, основанное на классах
- Программирование, основанное на прототипах
- Субъектно-ориентированное программирование

ОБЪЕКТНО-ОРИЕНТИРОВАНАЯ ПАРАДИГМА ПРОГРАММИРОВАНИЯ

Языки поддерживающие данную парадигму:

- JAVA
- SCALA
- Python
- C++
- Delphi
- C#
- JavaScript

РЕАКТИВНАЯ ПАРАДИГМА ПРОГРАММИРОВАНИЯ

Реактивное программирование — парадигма программирования, ориентированная на потоки данных и распространение изменений. Это означает, что должна существовать возможность легко выражать статические и динамические потоки данных, а также то, что нижележащая модель исполнения должна автоматически распространять изменения благодаря потоку данных.

Объектно-ориентированное реактивное программирование (ООРП) — это комбинация объектно-ориентированного подхода с реактивным. Вероятно, наиболее естественный способ сделать это состоит в том, что вместо методов и полей, у объектов есть реакции, которые автоматически пересчитывают значения и другие реакции зависят от изменений этих значений.

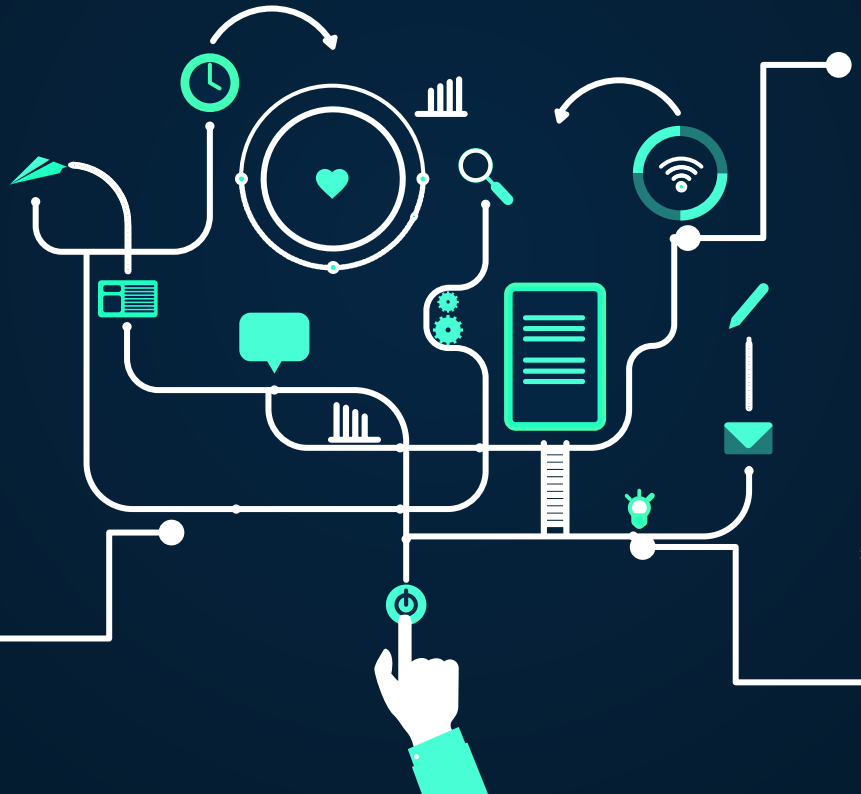
РЕАКТИВНАЯ ПАРАДИГМА ПРОГРАММИРОВАНИЯ

Функциональное программирование является наиболее естественным базисом для реализации реактивной архитектуры, хорошо сочетаясь с параллелизмом.

Реализации:

- React, созданная в Facebook JavaScript-библиотека разработки пользовательских интерфейсов.
- Elm, функциональный реактивный язык программирования, компилирующийся в HTML, CSS и JavaScript
- Flapjax, событийно-реактивный язык для программирования веб-приложений
- Reactive.jl

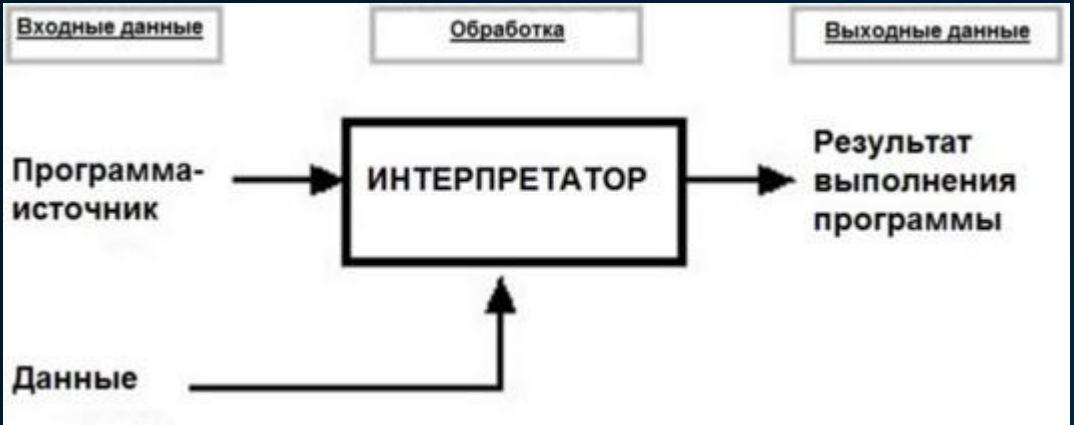
СПОСОБЫ РЕАЛИЗАЦИИ ЯЗЫКОВ



- Языки программирования могут быть реализованы как компилируемые и интерпретируемые.

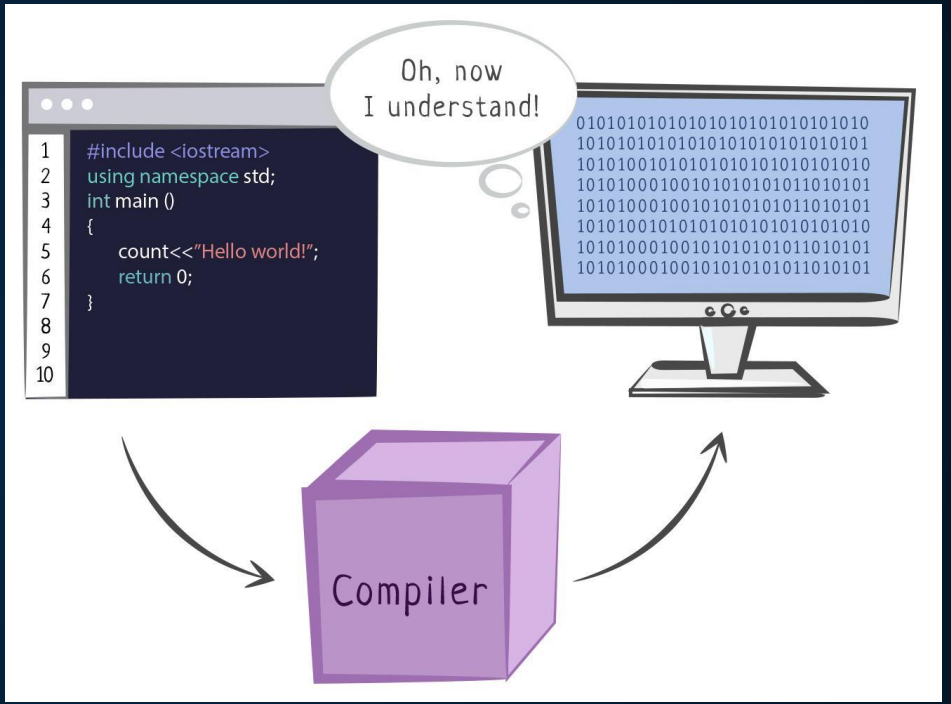
компилятор переводит исходный текст программы с языка программирования высокого уровня в двоичные коды инструкций процессора

процессор компьютера — это интерпретатор машинного кода



Интерпретатор — это компьютерная программа, которая преобразует каждый программный **оператор** высокого уровня в машинный код. Сюда входят исходный код, предварительно скомпилированный код и сценарии.
Интерпретатор представляет собой машинную программу, которая непосредственно выполняет набор инструкций без их компиляции.
Примерами интерпретируемых языков являются Perl, Python и Matlab.

Писать их было долго и сложно, поэтому инженеры стали создавать языки программирования, обозначая команды словами и знаками. Для того, чтобы процессор понимал, какие команды записаны в программе, программисты создали **компилятор** — программу, которая преобразует программный код в машинный.



- И **компилятор**, и **интерпретатор** выполняют одну и ту же работу — преобразовывают язык программирования высокого уровня в машинный код. Однако компилятор преобразовывает исходный материал в машинный код **перед запуском программы**. Интерпретатор выполняет эту функцию **при ее запуске**.

СПАСИБО ЗА ВНИМАНИЕ!