Тема лекции:

Парадигмы программирования

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.

ИМПЕРАТИВНАЯ ПАРАДИГМА ПРОГРАММИРОВАНИЯ

Парадигма программирования, которая 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.

Языки поддерживающие данную парадигму:

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

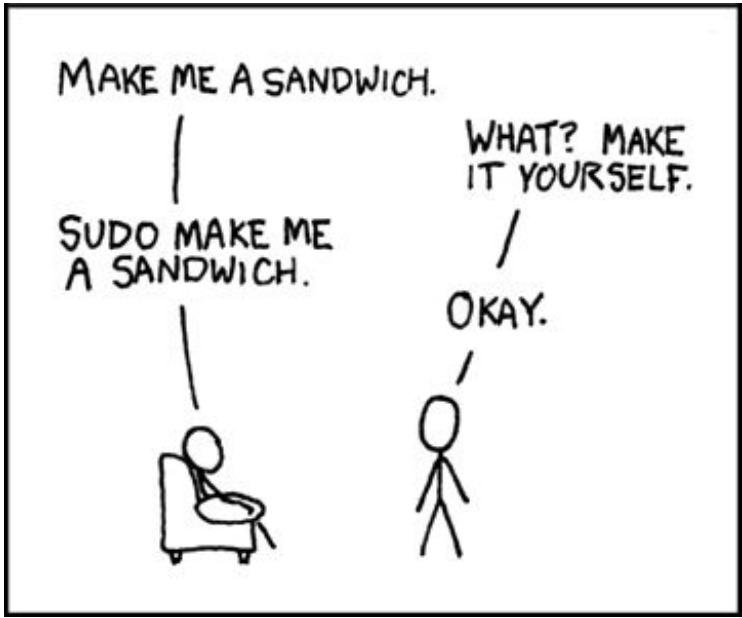
ДЕКЛАРАТИВНАЯ ПАРАДИГМА ПРОГРАММИРОВАНИЯ

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

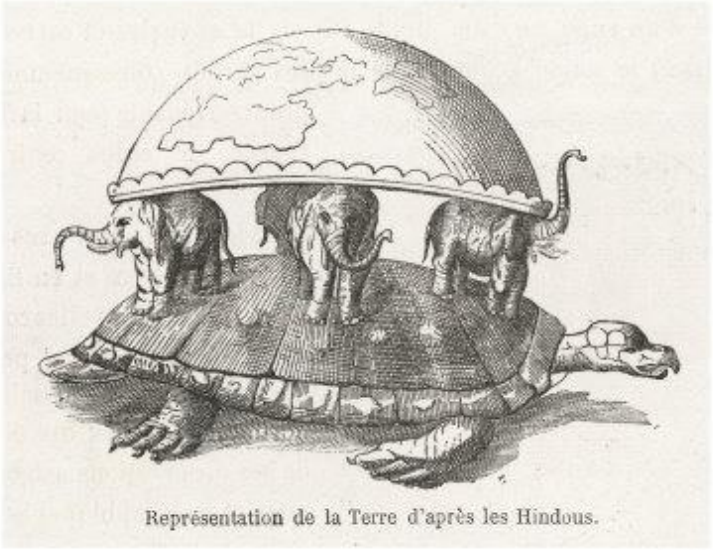
Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Разделяется на

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

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.

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.

Программа на компилируемом языке при помощи специальной программы компилятора преобразуется (компилируется) в набор инструкций для данного типа процессора (машинный код) и далее записывается в исполнимый модуль, который может быть запущен на выполнение как отдельная программа. Другими словами, компилятор переводит исходный текст программы с языка программирования высокого уровня в двоичные коды инструкций процессора.

Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет (интерпретирует) исходный текст без предварительного перевода. При этом программа остаётся на исходном языке и не может быть запущена без интерпретатора. Можно сказать, что процессор компьютера — это интерпретатор машинного кода.

Интерпретируемые языки обладают некоторыми специфическими дополнительными возможностями (см. выше), кроме того, программы на них можно запускать сразу же после изменения, что облегчает разработку. Программа на интерпретируемом языке может быть зачастую запущена на разных типах машин и операционных систем без дополнительных усилий. Однако интерпретируемые программы выполняются заметно медленнее, чем компилируемые, кроме того, они не могут выполняться без дополнительной программы-интерпретатора.

ТИПЫ ДАННЫХ

Современные цифровые компьютеры обычно являются двоичными и данные хранят в двоичном(бинарном) коде (хотя возможны реализации и в других системах счисления). Эти данные как правило отражают информацию из реального мира (имена, банковские счета, измерения и др.), представляющую высокоуровневые концепции.

Особая система, по которой данные организуются в программе, — это система типов языка программирования; разработка и изучение систем типов известна под названием теория типов.

Языки могут быть классифицированы как системы со статической типизацией и языки с динамической типизацией.

Статически-типизированные языки могут быть в дальнейшем подразделены на языки с обязательной декларацией, где каждая переменная и объявление функции имеет обязательное объявление типа, и языки с выводимыми типами.