

# Обзор парадигм программирования

## Часть 2

Юрий Литвинов  
yurii.litvinov@gmail.com

07.12.2018

# Логическое программирование

- ▶ Программа представляет собой набор фактов и правил, система сама строит решение с использованием правил логики
  - ▶ Использует логику предикатов как математическую формализацию
- ▶ Создавалось в 60-х для решения задач искусственного интеллекта и экспертных систем
  - ▶ Автоматическое доказательство теорем
- ▶ Могут использоваться разные стратегии доказательства
  - ▶ В общем случае, программа — это набор фактов и правил + стратегия вывода, которая управляет тем, как новые факты получаются из существующих
    - ▶ В формальной логике стратегия вывода обычно не важна, для компьютеров это критично
- ▶ Дедуктивные базы данных — хранят факты и правила вывода

# Пролог

- ▶ Появился в 1972 г. как научная разработка
- ▶ Реализации:
  - ▶ SWI-Prolog (<http://www.swi-prolog.org/>)
  - ▶ Amzi Prolog (<http://www.amzi.com/>)
  - ▶ Turbo Prolog
- ▶ Использует метод резолюций – последовательно перебирая правила и факты, пытается подобрать такой набор переменных, которые бы им удовлетворяли
  - ▶ Пример:
    - ▶ cat(tom)
    - ▶ ?- cat(tom).
    - Yes
    - ▶ ?- cat(X).
    - X = tom

# Пример программы

```
sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y).  
parent_child(X, Y) :- father_child(X, Y).  
parent_child(X, Y) :- mother_child(X, Y).  
mother_child(trude, sally).  
father_child(tom, sally).  
father_child(tom, erica).  
father_child(mike, tom).  
  
?- sibling(sally, erica).  
Yes  
?- father_child(Father, Child).
```

# Императивное программирование

```
?- write('Hello world!'), nl.
```

```
Hello world!
```

```
true.
```

```
program_optimized(Prog0, Prog) :-  
    optimization_pass_1(Prog0, Prog1),  
    optimization_pass_2(Prog1, Prog2),  
    optimization_pass_3(Prog2, Prog).
```

# QSort

```
quicksort(Xs, Ys) :- quicksort_1(Xs, Ys, []).
```

```
quicksort_1([], Ys, Ys).
```

```
quicksort_1([X|Xs], Ys, Zs) :-
```

```
    partition(Xs, X, Ms, Ns),
```

```
    quicksort_1(Ns, Ws, Zs),
```

```
    quicksort_1(Ms, Ys, [X|Ws]).
```

```
partition([K|L], X, M, [K|M]) :-
```

```
    X < K, !,
```

```
    partition(L, X, M, N).
```

```
partition([K|L], X, [K|M], N) :-
```

```
    partition(L, X, M, N).
```

```
partition([], _, [], []).
```

# Рекурсивное программирование, РЕФАЛ

- ▶ РЕкурсивных Функций Алгоритмический
  - ▶ В. Турчин, 1966г.
- ▶ Ориентирован на символьные вычисления
  - ▶ ИИ, перевод, манипуляции с формальными системами (лямбда-исчисление, например)
- ▶ Использует нормальные алгорифмы Маркова в качестве математической формализации
- ▶ Программа записывается в виде набора функций
  - ▶ Функция — упорядоченный набор предложений
  - ▶ Предложение состоит из шаблона и того, на что надо заменить шаблон
  - ▶ Выражения в угловых скобках (активные выражения)
  - ▶ Переменные
- ▶ Вычисление продолжается, пока в «поле зрения» Рефал-машины не окажется выражение без угловых скобок

# Рефал, пример

Hello, world:

```
$ENTRY Go { = <Hello>;}
Hello {
    = <Prout 'Hello world'>;
}
```

Палиндром:

```
Palindrom {
    s.1 e.2 s.1 = <Palindrom e.2>;
    s.1 = True;
    = True;
    e.1 = False;
}
```

# Стековое программирование

- ▶ Язык Форт (Forth)
  - ▶ Разработан в 60-х Чарльзом Муром «для себя»
  - ▶ Был широко распространён для программирования встроенных систем и задач, естественным образом выражавшихся в терминах стеков
    - ▶ Синтаксический анализ
    - ▶ Анализ естественных языков

# Форт, подробнее

- ▶ Основной элемент программы: слово
- ▶ Форт-система состоит из словаря (набора слов) и стеков — арифметического и командного (с их помощью производятся вычисления)
- ▶ Используется обратная польская нотация

# Примеры

▶ `25 10 * 50 + .`

Вывод: 300 ok

▶ `: FLOOR5 ( n -- n' ) DUP 6 < IF DROP 5 ELSE 1 - THEN ;`

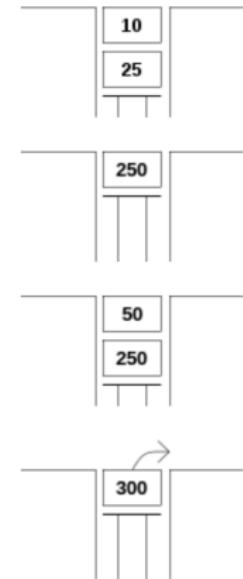
▶ то же самое на С:

```
int floor5(int v) { return v < 6 ? 5 : v - 1; }
```

▶ более красиво на Форте:

`: FLOOR5 ( n -- n' ) 1- 5 MAX ;`

▶ `: HELLO ( -- ) CR ." Hello, world!" ;`



# Форт, пример

\ Напечатать знак числа

: .SIGN ( n -- )

?DUP 0= IF

." НОЛЬ"

ELSE

0> IF

." ПОЛОЖИТЕЛЬНОЕ ЧИСЛО" ELSE

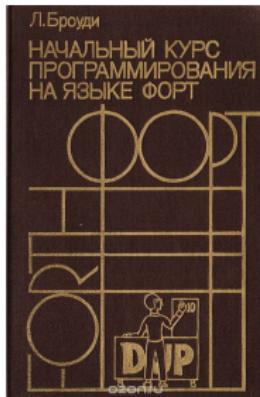
." ОТРИЦАТЕЛЬНОЕ ЧИСЛО" THEN

THEN

;

# Реализации

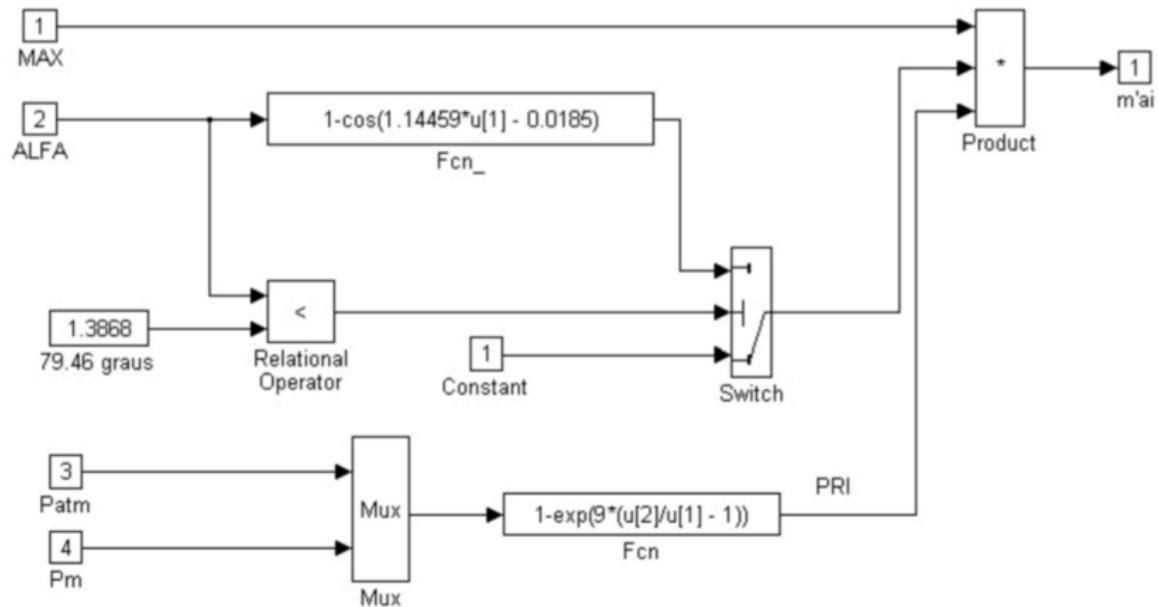
- ▶ SwiftForth
  - ▶ <https://www.forth.com/swiftforth/>
- ▶ Gforth
  - ▶ <http://www.gnu.org/software/gforth/>
- ▶ Десятки других реализаций
  - ▶ <http://www.forth.org/commercial.html>
- ▶ Книжка
  - ▶ Броуди Л. «Начальный курс программирования на Форте»



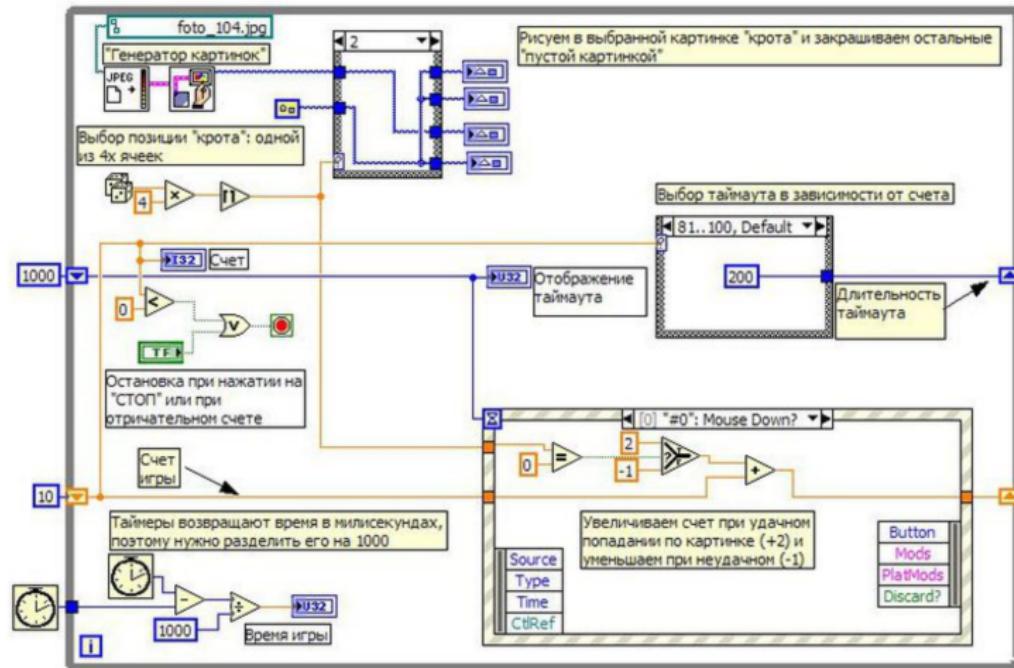
# Визуальное программирование

- ▶ Визуальные языки появились ещё до компьютеров
  - ▶ Диаграммы потоков данных
  - ▶ Сети Петри
- ▶ Применяются прежде всего для моделирования, а не для программирования
  - ▶ Описание архитектуры системы (UML, SysML, IDEFx)
  - ▶ Описание бизнес-процессов (UML, BPMN)
  - ▶ Описание схем баз данных (ER, ORM)
- ▶ Есть и языки программирования: G (LabVIEW), Simulink, ДРАКОН, Scratch
- ▶ Предметно-ориентированные визуальные языки
  - ▶ TRIK Studio

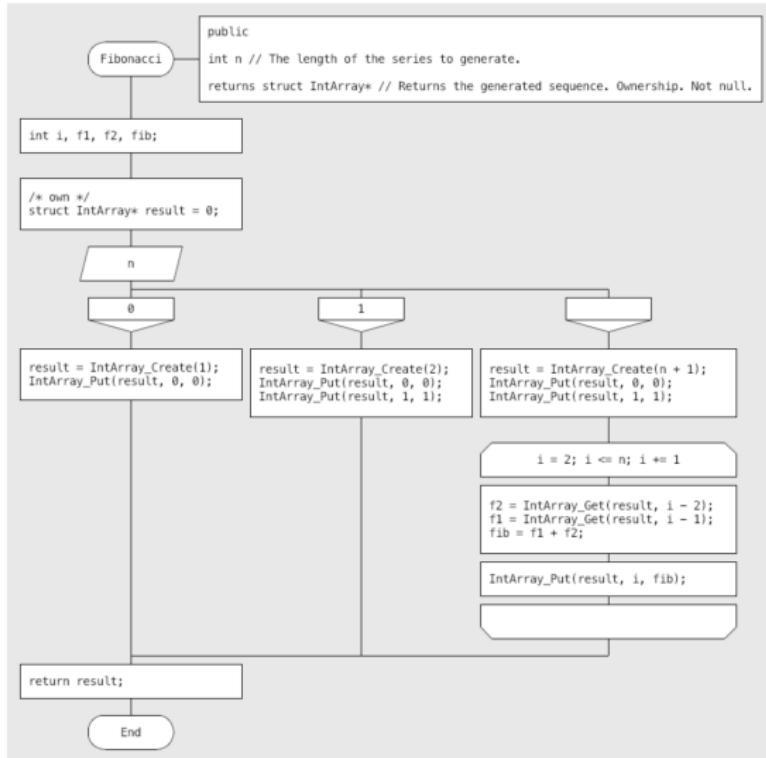
## Пример (Matlab/Simulink)



# Пример (LabVIEW)

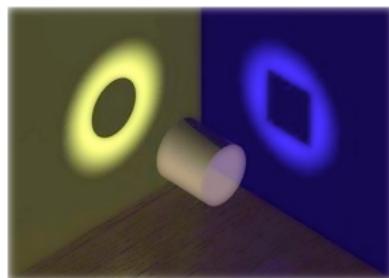


# Пример (ДРАКОН)

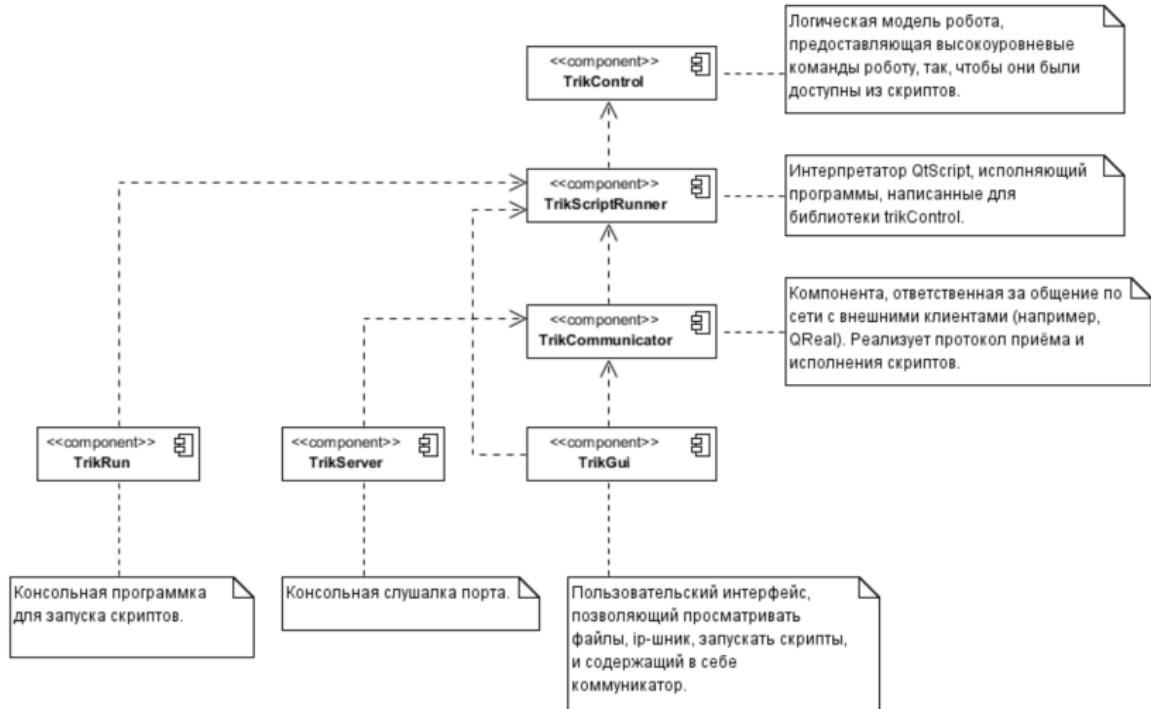


# Визуальное моделирование

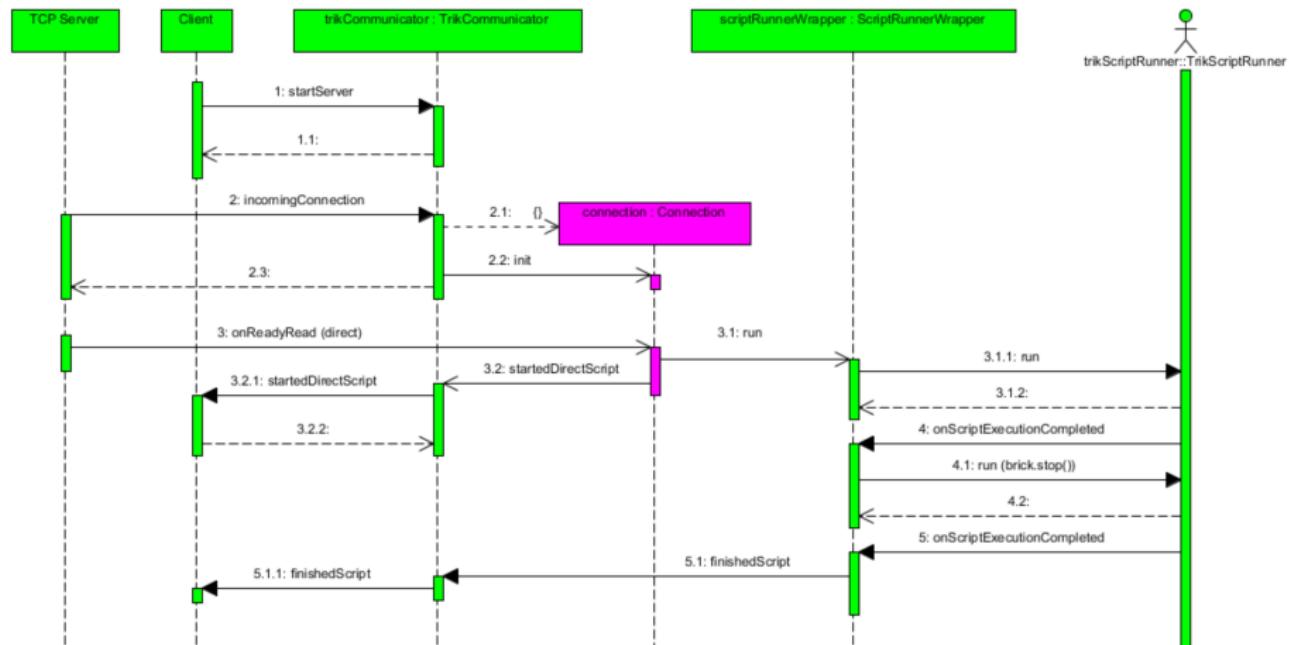
- ▶ Не ставит своей целью получить работающую программу
- ▶ Модель проще, чем нужно исполнителю
- ▶ Модель даже для сложной системы обозрима
- ▶ Система описывается с разных дополняющих друг друга точек зрения
  - ▶ При этом описание системы остаётся целостным, визуальная модель — это не просто картинка
- ▶ Модели можно анализировать до реализации
- ▶ Могут быть сгенерированы заглушки классов и иногда даже реализации методов



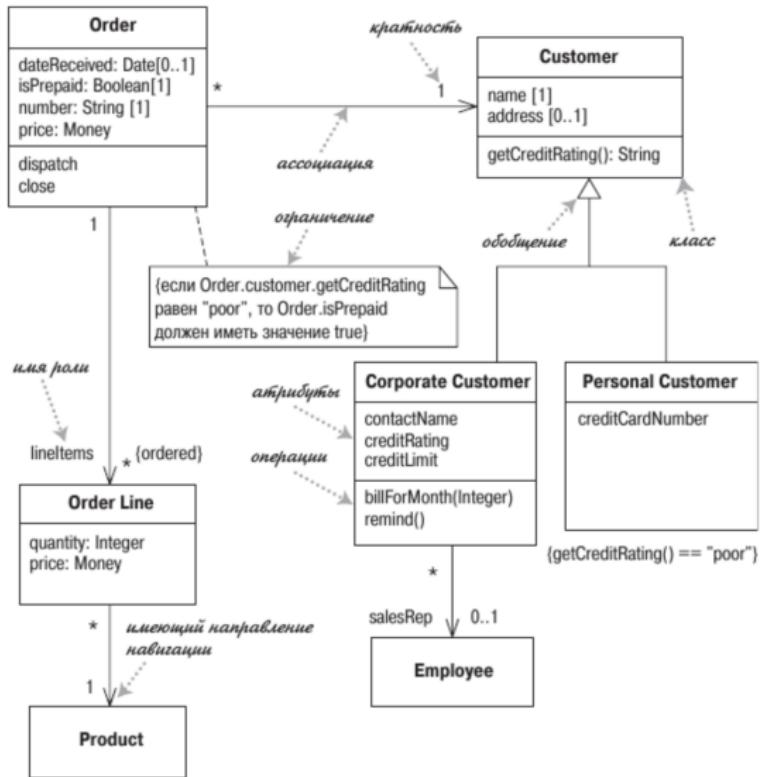
# Пример высокоуровневой модели (UML)



# Ещё пример (UML)



# UML, диаграммы классов



# Предметно-ориентированное моделирование

- ▶ Специальные языки и инструменты для конкретной задачи или группы похожих задач
- ▶ Благодаря узости предметной области можно генерировать полностью работающую программу по диаграмме
  - ▶ Зато оно работает только для этой предметной области
- ▶ Могут программировать даже непрограммисты

# Пример: TRIK Studio

- ▶ Среда программирования роботов
- ▶ Программа — набор элементарных команд
  - ▶ Исполняются на реальном роботе по WiFi, Bluetooth, USB
  - ▶ Генерируются в код на текстовом языке и загружаются на робот
  - ▶ Исполняются на двумерной модели
- ▶ Можно программировать только роботы
- ▶ Могут программировать даже дети, не умеющие ещё читать

