



**Сошников Дмитрий Валерьевич**

к.ф.-м.н., доцент

dmitryso@microsoft.com

# Логическое программирование



Факультет Прикладной математики и физики

Кафедра Вычислительной математики и программирования

Московский авиационный институт (государственный технический университет)

# Обо мне



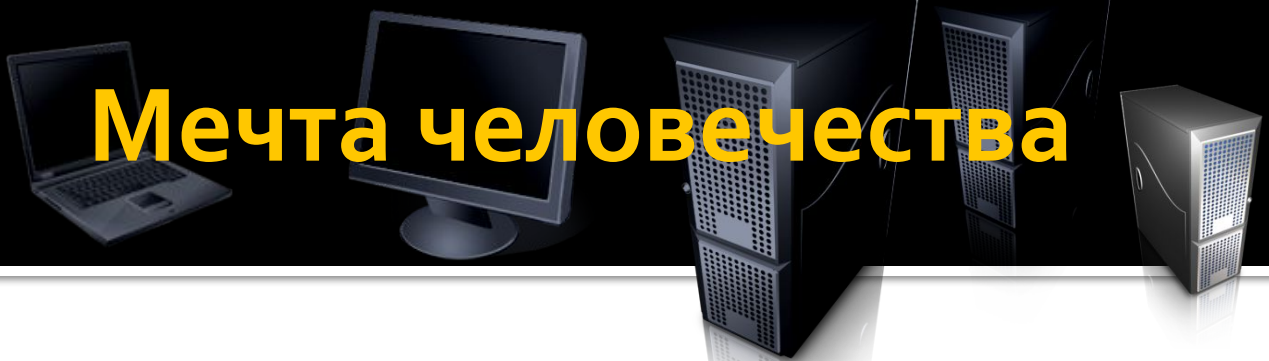
- Майкрософт Россия, академический евангелист
- Кандидат физ.-мат. наук
  - Распределенные интеллектуальные системы с явным представлением знаний
  - Интеллектуальная реструктуризация социальных сетей на основе онтологий
  - Семантически-ориентированные системы (Semantic Wiki)
- Кафедра Вычислительной математики и программирования МАИ (доцент)
  - Логическое программирование
  - Искусственный интеллект
  - Студенческая лаборатория MAILabs ([www.mailabs.ru](http://www.mailabs.ru))
- ФИВТ

<http://blogs.gotdotnet.ru/personal/sos>

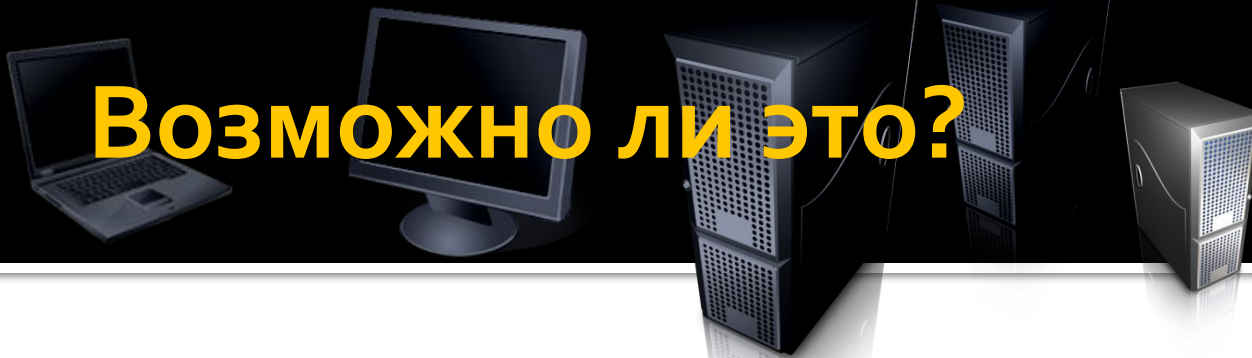
# Лекция 1

Что такое логическое программирование?

# Мечта человечества

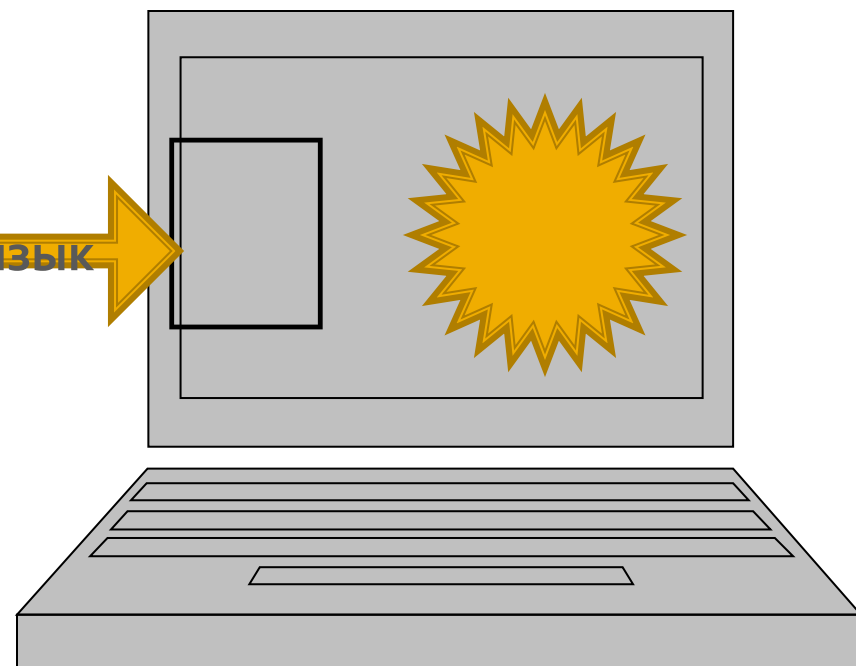


# Возможно ли это?



- Тест Тьюринга – подробнее в курсе ИИ
- Проблемы:
  - Неоднозначность человеческого языка
  - При коммуникации мы полагаемся на картину мира, которая есть у нас в голове (common knowledge)

# Потенциальный способ реализации



# Какие языки программирования вы знаете?

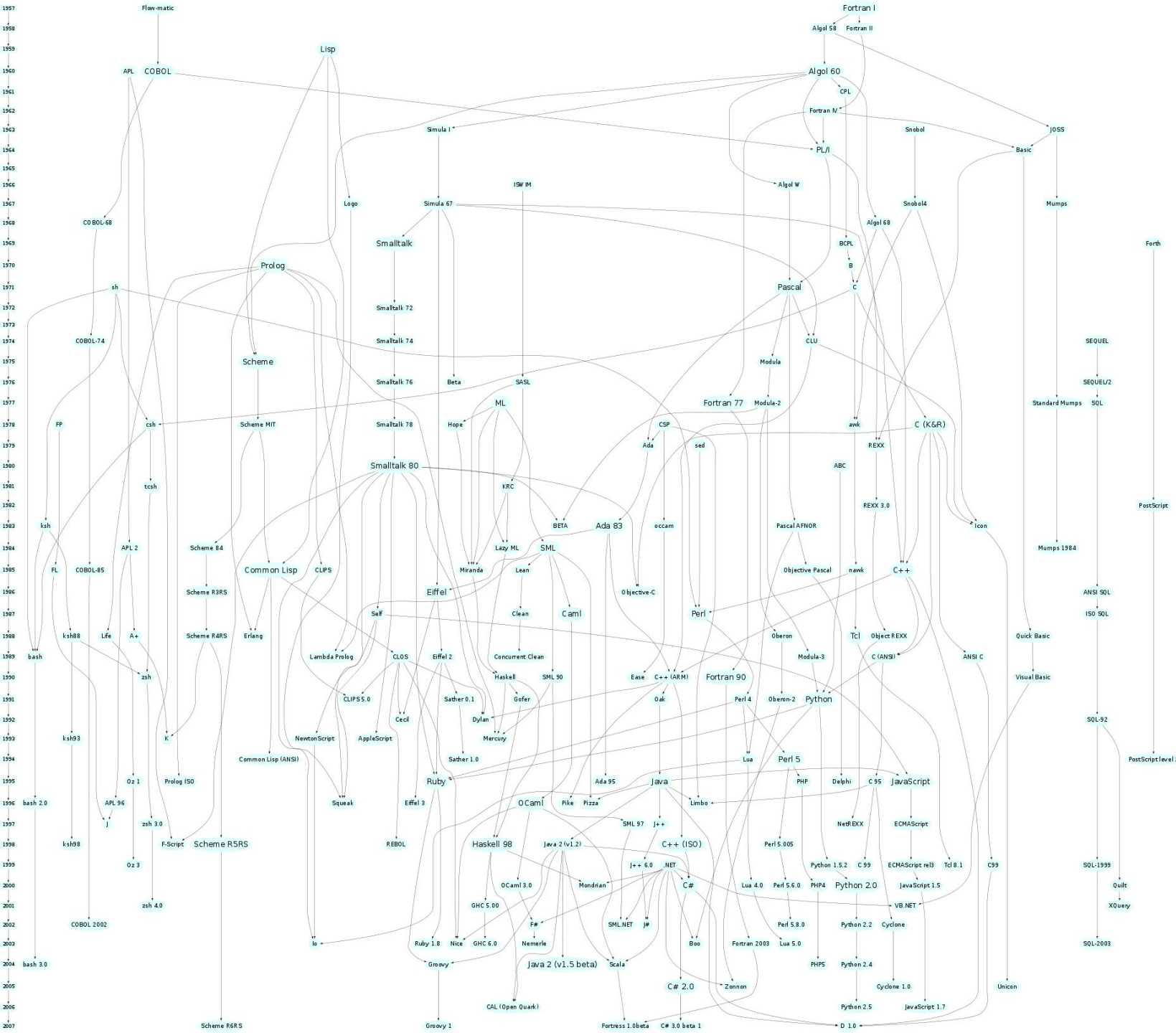


- Assembler (x86, ...)
- C, C++, C#, Java
- Pascal
- ...
  
- Brainfuck?
- FORTH?
- LISP, FP, ML, Haskell, OCaml, F#, ...
- Prolog, Mercury, Datalog, ...



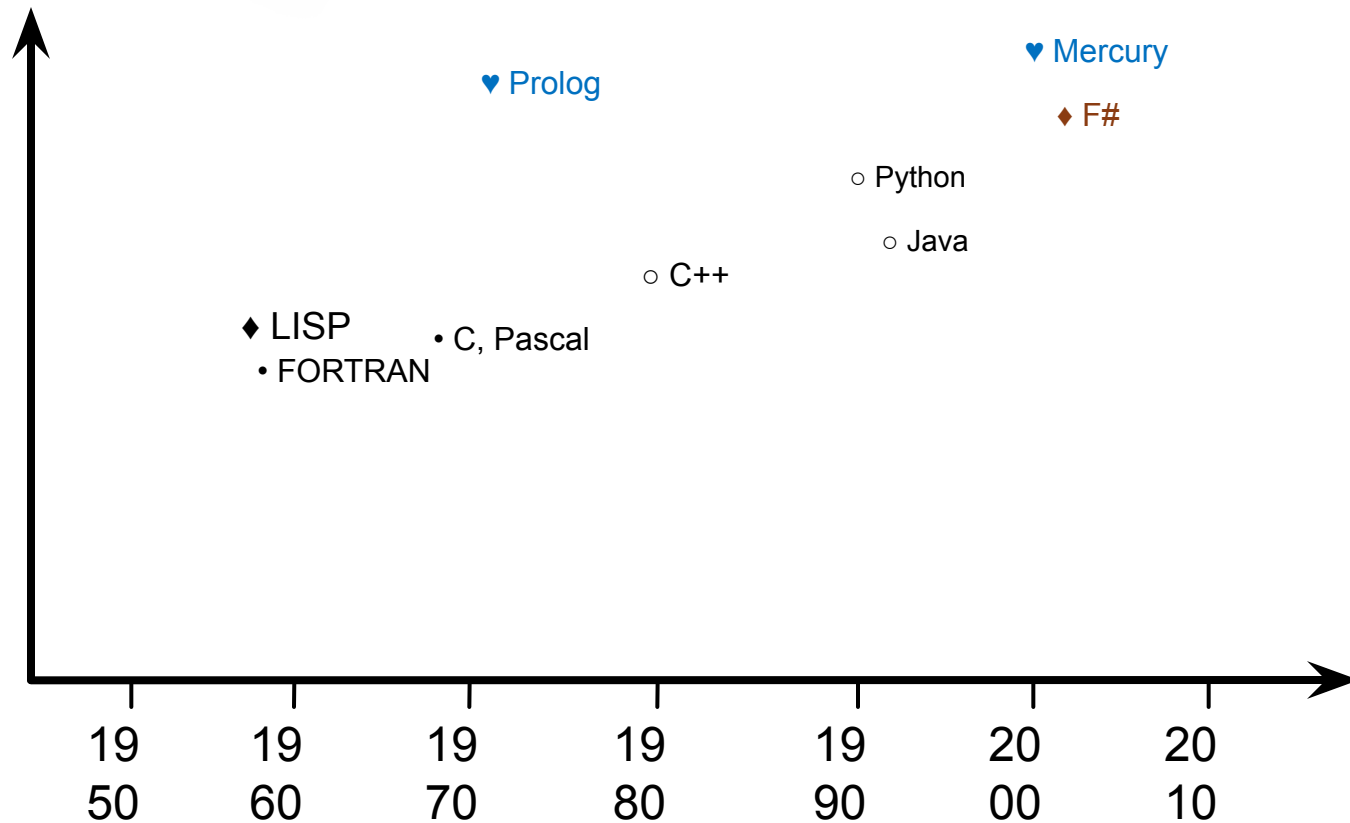
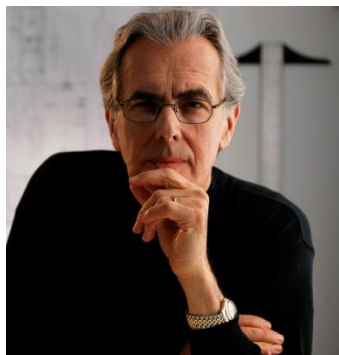
# ЯЗЫКИ ПРОГРАММИРОВАНИЯ

©2009 Сошников Д.В.





# Программирование вчера и сегодня



# Обратимся к истории



**1954-57 г., Дж.Бэкус**

- FORTRAN
- язык ассемблера
- машинные коды
- программирование переключателей

```
0000 0A 12 1F 4B C3 E0 EE F1
0008 C3 1D 23 17 F2 00 0C 0D
0010 ...
```

```
MOV AX, [ARG1]
ADD AX, [ARG2]
MOV [RES], AX
JMP NEXT
ARG1: DB 10
ARG2: DB 20
RES: DB 0
NEXT: ...
```

```
S = 0
DO 10 I=1,10
S = S + I*I
10 CONTINUE
```

1950 1960 1970 1980 1990 2000 2010

- Первый язык программирования высокого уровня – ФОРТРАН – был создан Дж.Бэкусом, чтобы математики могли программировать на уровне формул.

# Программирования для математиков



```
def fac = eq 0 → 1;  
*o[id, faco(-o [id, 1])]
```

1958 г., Дж.Маккарти      1977 г., Дж.Бэкус

◆ LISP

◆ FP

• FORTRAN

- язык ассемблера
- машинные коды
- программирование переключателей

```
(defun factorial (n)  
  (if (<= n 1) 1  
      (* n (factorial (- n 1)))))
```

1950      1960      1970      1980      1990      2000      2010

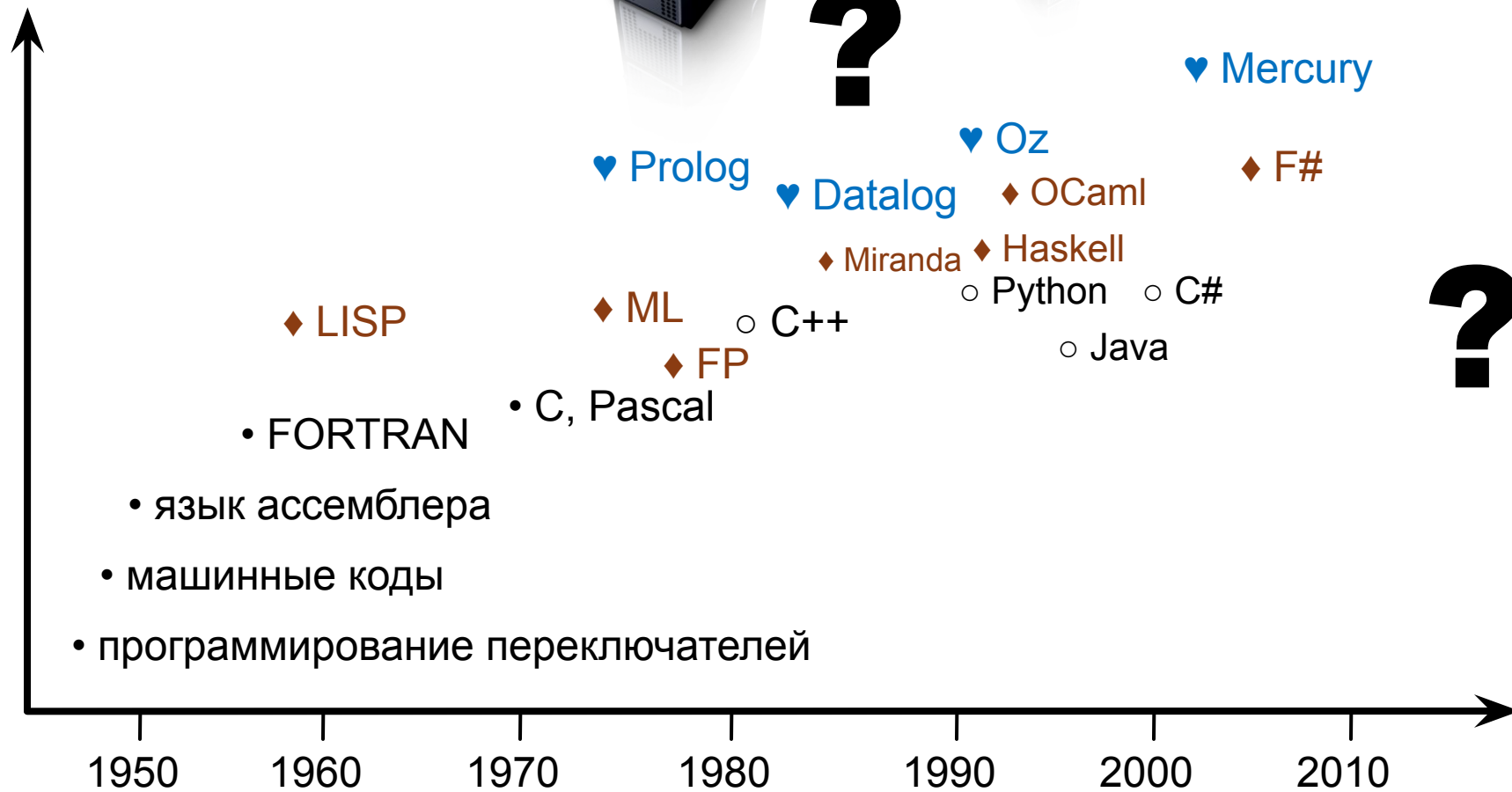
- Позже Дж.Бэкус пошел дальше и предложил язык FP, в котором формулы более соответствовали математическому понятию функции

# Что делать, чтобы приблизиться к человеческому языку?



- Надо пытаться формализовать человеческий язык!
- Основной инструмент формализации:
  - Формальные аксиоматические системы
  - Логика!

# Языки логического программирования



# Как это выглядит на практике?



## ■ Вычисление факториала:

```
fact(1)=1.  
fact(N)=N*fact(N-1).
```

*Логическое  
программирование (Mercury)*

*Императивное  
программирование  
(Pascal)*

```
function fact(x:integer):integer;  
var i, r : integer;  
begin  
  r:=1;  
  for i:=1 to x do r:=r*i;  
  fact:=r  
end;
```

```
fact(1,1).  
fact(N,F) :- N1 is N-1,  
             fact(N1,F1),  
             F is F1*N.
```

*Логическое  
программирование (Prolog)*

```
let rec fact = function  
  1 -> 1  
  | x -> x*fact(x-1);;
```

*Функциональное  
программирование (F#)*

# Декларативное программирование



- При декларативном программировании мы (на некотором формальном языке) описываем результат (его свойства), а не способ его достижения
  - Описание факториала
  - HTML – описание расположения объектов
  - SQL
  - LINQ
  - Функциональные, логические языки

# Императивное программирование



- Императивное – мы говорим компьютеру, как решать задачу (что делать)
- Основной акцент – манипулирование ячейками памяти
  - Оператор присваивания
- Явные операторы передачи управления
  - Циклы, условный оператор



# Декларативный стиль



```
function fact(x:integer):integer;  
begin  
  if x=1 then fact:=1  
  else fact:=x*fact(x-1)  
end;
```

- Это не «чистая» императивная программа.
  - В «чистых» императивных языках (ФОРТРАН) нет рекурсии
- Нет операторов присваивания
  - «:= » -это возврат результата из функции, а не присваивание

# Логическое программирование



- Парадигма декларативного программирования, в которой
  - программа представляет собой описание требуемого решения в терминах определенной логики
  - решение задачи строится в процессе логического вывода по заданному описанию
- Различные разновидности логического программирования: индуктивное, в ограничениях, ...
- Подход к программированию
- Языки программирования Prolog, Datalog, Mercury, Oz, ...

# С практической стороны:



- Найдем все комбинации  $\langle a, b, c \rangle$  чисел от 1 до 10, что  $a^2 + b^2 = c^2$

```
for a:=1 to 10 do
  for b:=1 to 10 do
    for c:=1 to 10 do
      if a*a+b*b=c*c then
        write(a,b,c) ;;
```

```
[ for a in 1..10
  for b in 1..10
  for c in 1..10
    when a*a+b*b=c*c ->
      (a,b,c)
];;
```

```
solve(A,B,C) :-
  for(A,1,10),
  for(B,1,10),
  for(C,1,10),
  A*A+B*B == C*C.
```

```
zip3 [1..10] [1..10]
[1..10] |>
filter (
  fun (a,b,c)->a*a+b*b=c*c
);;
```

# Практические преимущества



- **Функциональные языки**
  - Компактный синтаксис для списков, n-ок (tuples), вариантных типов
- **Логические языки**
  - Компактный синтаксис для списков, n-ок (tuples), вариантных типов
  - Возможность перебора и поиска различных решений, заложенная в язык

# Ещё пример



```
studied(petya,mathematics).  
studied(petya,compscience).  
studied(petya,english).
```

```
studied(vasya,german).  
studied(vasya,literature).
```

```
studied_technical(X) :- studied(X,mathematics).  
studied_technical(X) :- studied(X,compscience).  
studied_languages(X) :- studied(X,english).  
studied_languages(X) :- studied(X,german).
```

```
speciality(X,tech_translator) :- studied_languages(X),studied_technical(X).  
speciality(X,programmer) :-  
    studied(X,mathematics),studied(X,compscience).  
speciality(X,lit_translator) :- studied_languages(X),studied(X,literature).
```

```
?-specialty(vasya,X).  
?- specialty(X,lit_translator).
```

# Что особенного?



- Определения на логическом языке похожи на предложения математической логики
  - Логическое программирование имеет очень четкую математическую основу
  - Возможны рассуждения о программах: доказательство корректности, ...
- **Отсутствует оператор присваивания**
  - Есть знак  $=$ , но он имеет другую семантику – унификация, связывание имен
  - Переменные связываются неявно, в процессе логического вывода
  - Будучи один раз связанным, имя может менять свое значение только в процессе пересмотра решения (возврата)
  - А это значит – нет побочных эффектов!

# Парадигмы программирования

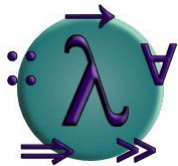
# Парадигмы программирования



Шифрующие роторные машины.  
Слева направо: ENIGMA (Германия), SIGABA (США),  
PURPLE (Япония).

## Императивное (алгоритмическое)

- Машина Тьюринга, Машина фон Неймана
- Pascal, C и т.д.



## Аппликативное (функциональное)

- $\lambda$ -исчисление, рекурсивные функции
- F#, LISP / Scheme, ML и друзья, Haskell



## Декларативное (логическое)

- Логика предикатов 1-го порядка
- Prolog, Mercury, Oz, ...

## Ситуационное (продукционное)

- Нормальные алгоритмы Маркова
- Рефал

## Объектное, компонентное, многоагентное (эмерджентное)

- Синергетика, теория сложных систем



# Семантика языков



## Императивные языки

- Оперировать состоянием памяти. Выполнение операторов изменяет состояние.

## Функциональные языки

- Оперировать данными. Применение функции к аргументам изменяет данные.
- Подход, ориентированный на данные.

## Логические языки

- Оперировать пространством поиска решений.
- Программа задаёт множество возможных переходов в пространстве поиска



# Мультипарадигмальные языки



- C# - императивный (ОО) + элементы функциональности
- F# - функциональный с элементами императивности
- Mercury – функционально-логический
- Oz
- Python
- ...

# Почему важно изучать логическое программирование?

# Главный вопрос



- Придется ли нам программировать на Прологе в реальной жизни?
- В лучшем случае – 1 человеку из группы
- Пролог удобен при быстром прототипировании определенного класса систем
- Принципы, заложенные в основу логического программирования, помогут при решении реальных задач

# Какие задачи хорошо решаются на логических языках?



- Задачи искусственного интеллекта
  - Экспертные системы
- Лингвистика, обработка естественного языка
- Задачи с неопределенностью
- Задачи, связанные с поиском решений
- Мета-программирование, построение специализированных языков

# Особенности логических языков



- Отсутствие операторов присваивания и побочных эффектов
- Декларативное программирование
- Естественная математическая модель вычислений
- Заложенная в язык возможность возвратов и перебора
- Заложенные в язык возможности по представлению списков, деревьев
- Развитые возможности мета-программирования и построения проблемно-ориентированных языков