OSDN 2013.

Велосипедостроение:

Старые «новые» идеи,
которые стоит иметь в виду

ruslan@shevchenko.kiev.ua
https://github.com/rssh/
@rssh1

Старые «новые» идеи

Language research
Mainstream — то что очевидно

Не-мейнстрим, но золотой фонд

j: lol where r u romeo

r: wana come over?

j: cant

r: y?

j: idk parents

r: lets kill ourself

j: lol k

r: swag

j: swag

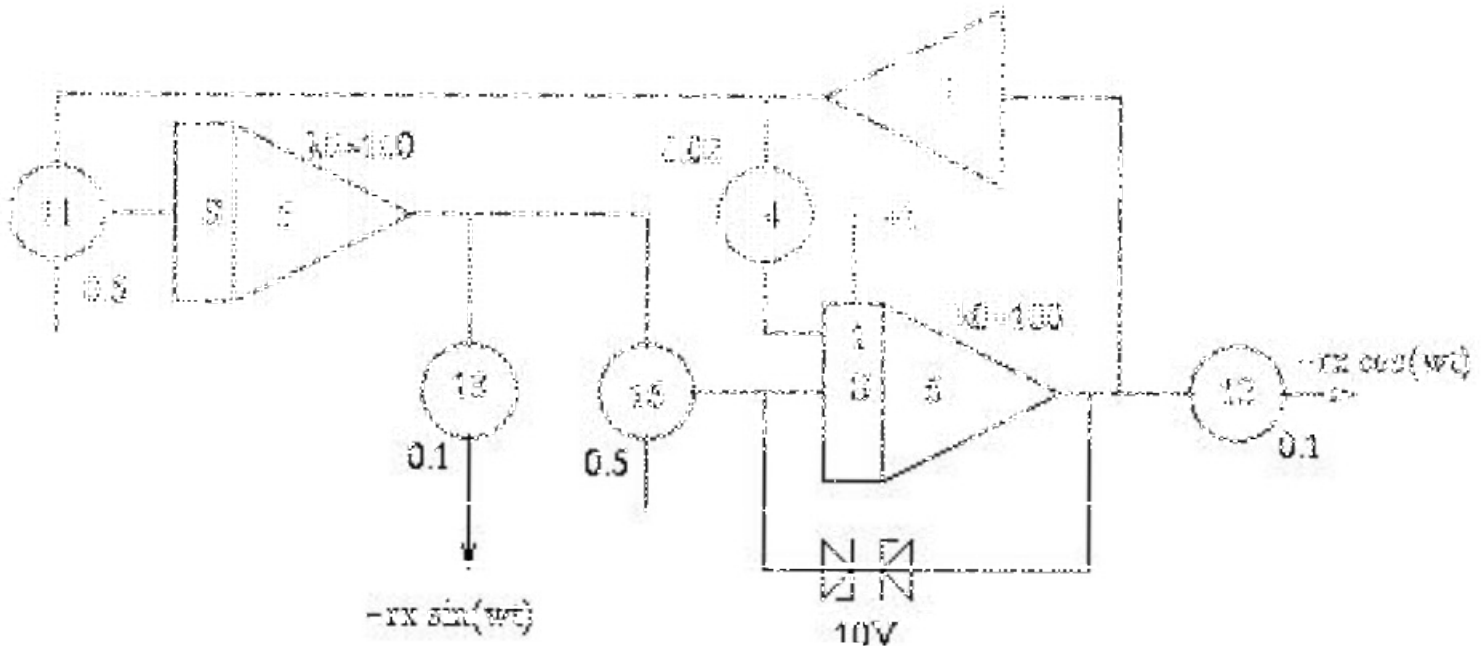
Rome & Julietta. Twitter Style.

Shakespeare ?

Очевидности

```
int n = 10;
int s = 0;
for(int x = 1; x < n; ++x) {
  s+=x;
}
```

- состояние хранится в именованных переменных
- инструкции выполняются последовательно
- управляющие конструкции являются частью языка



Более естественное представление программы (60-ые годы прошлого века)

АВМ – Аналоговые вычислительные машины

Задача =
система уравнений в частных производных

Решение = моделирование
(электрическая схема, в которой происходят процессы, описывающиеся с так-же)

- *нет вопросов сложности расчетов*
- *машина действительно «думает»*

- *Не все задачи можно привести к такому виду*
- *Потери точности при масштабировании*
- *Надо перекоммутировать схему руками*

Очевидности

```
int n = 10;
int s = 0;
for(int x = 1; x < n; ++x) {
  s+=x;
}
```

- состояние хранится в именованных переменных

- можно ли представить себе программирование без них ?

```
int n = 10;
int s = 0;
for(int x = 1; x < n; ++x) {
  s+=x;
}
```

```
0  10  1  DO  I + LOOP
```

Forth

```
0  10  1  DO  I + LOOP
```

Нет переменных – все на стеке

```
5  3 +  .
```

```
: sqrt-closer  (square guess -- square guess adjustment)
                2dup / over - 2 / ;
: sqrt ( square -- root )
        1 begin sqrt-closer dup while + repeat drop nip ;
```

Forth

```
: sqrt-closer  ( square guess -- square guess adjustment )
                2dup / over - 2 / ;
: sqrt ( square -- root )
        1 begin sqrt-closer dup while + repeat drop nip ;
```

$x_1 = 1$

$x_{n+1} = (x_n + S/x_n)/2$

Forth

Чак Моррис

colorForth

http://www.greenarrays.com

Интерпретатор реализован в «железе»
На одной плате — 144 форт-ядра

Размер кода — 1% от аналогичного на C

Очевидности

```
int x = 1
int d = 1
while(d != 0) {
   d = (s/x - x)/2
   x = x+d
}
```

- инструкции выполняются последовательно
- управляющие конструкции являются частью языка

```
int x = 1
int d = 1
while(d != 0) {
   d = (s/x - x)/2
   x = x+d
}
```

- возможно ли программирование без последовательности исполнения ?

Переписывающие правила

sqrtd(s,x,0) -> x

sqrtd(s,x,d) -> sqrt(s, x+d, (s/x - x)/2)

sqrt(s) -> sqrtd(s,1,(s-1)/2)

- программирование без последовательности исполнения.

Переписывающие правила

Http://www.refal.org - исторический интерес

APS – институт кибернетики НАНУ
(сейчас следов в интернете нет)

Maude (лидер рынка): http://maude.cs.uiuc.edu/

Termware:
http://www.gradsoft.ua/products/termware_eng.htm

Stermware: https://github.com/rssh/stermware

Очевидности

```
int x = 1
int d = 1
while(d != 0) {
   d = (s/x - x)/2
   x = x+d
}
```

- управляющие конструкции являются частью языка

Tcl

```
set x 1
do {
   set d [expr  ($s/$x - $x)/2]
while { $d != 0 }
```

- управляющие конструкции можно определять в языке

```
proc do {body whileword condition} {
   global errorInfor errorCode
   if {![string equal $whileword while]} {
      error "should be \"do body while condition\""
   }
   while {1} {
      set code [catch {uplevel 1 $body} message]
       if { !ok( $code) } {
           return handleError($code, $body, $message)
         }
      }
      if {![uplevel 1 [list expr $condition]]} {break}
   }
}
```

Tcl

Интерпретатор предлоставляет программисту API для манипуляции исполнением текущего модуля

[uplevel n block] – исполняет блок в контексте функции, которая находится на n фреймов Выше чем текущая

Макросы

Кмпилятор предлоставляет программисту API для манипуляции компиляцией текущего модуля

<[block]> – выполняет блок кода при компиляции и вставляет результат

- Nemerle

- Scala

Scala

- в какой-то степени мейнстрим

- удачный аналог Algol68
  (давайте сбросим все что придумала наука
  в индустриальный язык)

- Макросы, передача по имени,
  зависимые типы …….

Итого:

Очевидность мейнстрима — обман

Хотите быть хорошим разработчикам
и сдвинуть «точку сборки»

- системотехника
- forth
- переписывающие правила
- tcl
- nemerle
- scala

(скорее как источник идей чем
прямое использование)

Спасибо за внимание

// Руслан Шевченко
<ruslan@shevchenko.kiev.ua>

- http://www.github.com/rssh

-R&D http://www.gosave.com