自己紹介

- なまえ： k.inaba (けーいなば)
- なにもの？： 珍妙言語機能妄想日記書き
  - http://www.kmonos.net/wlog/
  - 「タグストラクチャル型」
    「お気楽非決定性」「return[f→g]」
    「物知りなぬるぽ」「Я９Я∽」
    「手続型リアクティブ」「Int./¥d+/」
    「名前推論」「Aspect指向Yコンビネータ」
    …

プログラミング言語の過去

- 1954　FORTRAN
- 1958　LISP, ALGOL
- 1959　COBOL
- …

プログラミング言語の過去

- BC7000~4000　印欧祖語
- BC5000~0　日本語
- AD410~500　英語
- 1954　FORTRAN
- 1958　LISP, ALGOL
- 1959　COBOL
- …

これら数千年の歴史をもつ言語を
「プログラミング言語」と思って学ぼう！

FLTV - 2009/8/30

Rhetorical Programming

真・自然言語プログラミング

k.inaba

http://www.kmonos.net/wlog/

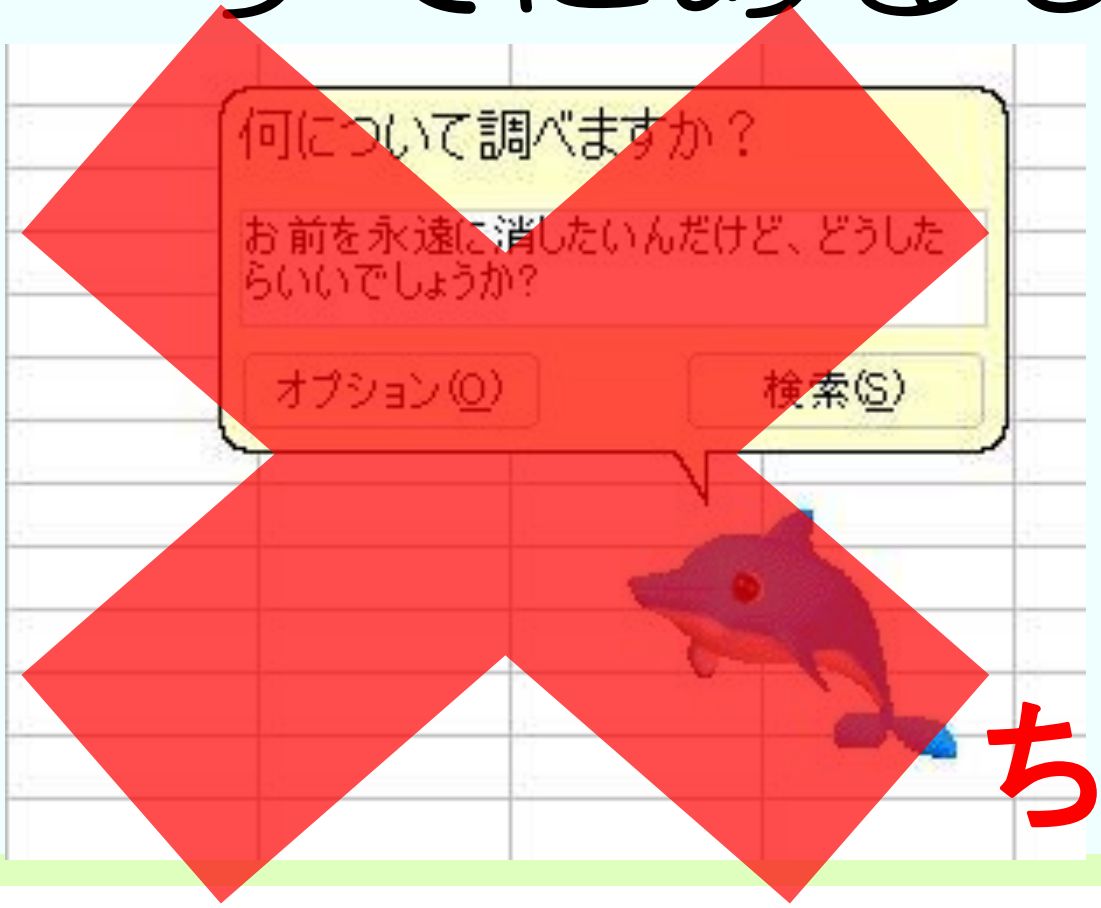
よくある突っ込み

- 「すでにあるじゃん」
  - 日本語G-Basic（ぴゅう太）
  - Mind
  - なでしこ
  - AppleScript, …

ちがいます

よくある突っ込み

・「すでにあるじゃん」



ちがいます

これらの言語は

- 自然言語の構文を取り入れてる
  - もし～ならば～違えば～ / ～とは～
  - *“「何をどうする」の語順はオブジェクト指向的”*
  - *“スタックマシンは日本語の語順と同じ”*
  - *“話し言葉のような人間に自然な文章で検索”*

これらの言語は

・自然言語の構文を
取り入れてる

普段書いてるプログラムは
使い慣れた構文で書ければ
もっと楽なはず! という発想

今回のおはなし

- 自然言語の機能を取り入れよう!!
  - 自然言語は「プログラミング言語」と見ると恐しくパワフル

今回のおはなし

・自然言語の機能を
取り入れよう!!

楽なのは慣れてるからじゃない,ヤツらがもの凄く強力な言語だから！という発想

ここからのお話

以下、いくつか
「強力な機能」
の具体例を紹介

“The”

型を１つ引数にとり現在のスコープに唯一存在するその型のオブジェクトを返す演算子

ネタ元：4年前の自分の日記

プログラマなら誰もが悩む変数名

変数名

ウェブ全体から検索　日本

ウェブ　検索ツールを表示

他のキーワード: java 変数名　php 変数名　変数名

プログラマなら誰もが悩む変数名や関数名

Google検索結果 "変数名"

プログラマなら誰もが悩む**変数名**や関数名の名

2009年2月25日 ... という事で、**変数名**や関数名にどんな名
みました。 ... **変数名**や関数名は、「こういうような処理をする
前を調べたい事も多いだろうと思うので、文章で検索 ...
d.hatena.ne.jp/stellaqua/20090225/1235565012 - キャッシ

変数名として「lst」や「dct」や「L」や「D」がと

2009年7月14日 ... じゃあもう完全に前提が違います
名を使ってたら、わかりにくいと思います("なんかリス
数名を使いたい)。Javaのような静的な言語と ...
d.hatena.ne.jp/kwatch/20090714/1247572630 - キ
d.hatena.ne.jp からの検索結果 »

変数名の力 - いいプログラムを書こう

特に、省略した変数は要注意です。省略した結果、似たよう
りします。発音が同じくなると、チーム内で会話したりするとき
上手に名前をつけるのも、目立たないですが、 ...
Top Of Page - いい名前 - 変数種による名前 - 命名規約
homepage1.nifty.com/~takaot/prprpr/varname.html - キャ

変数名はどう付けよう? - ホリデープログラミン

変数名に含めることができるのは英数字のみ・数字はお勧め

変数名のこだわりについて -OKWave

2007年2月3日 ... 私はC言語始めて10ヶ月ぐらいのものなので
んでいます。C言語で用いている**変数名**や関数名、クラスのオ
いたのですが、最近とてつもなくソースコードが読み ...
okwave.jp/qa2720993.html - キャッシュ - 類似ページ -

動的な**変数名**のつけ方について -OKWave

2009年8月14日 ... 環境OS:XP言語:C++コンパイラ:
された文字列を使用して変数を宣言することは可能でし
字列を取得しその文字列を**変数名**としてプログラミング
okwave.jp/jump.php3?q=5207509&t=a - キャッシュ -
okwave.jp からの検索結果 »

1-2-3.**変数名**の命名(Windows API Topics)

変数名に関しては検討を要する事項がたくさんあります。まず
言う、今では当たり前の命名法を使います。いわゆるハンガリー記法と

ふざけた**変数名**を使う奴は、ばかなの? 死ぬの?

アルファルファモザイクのふざけた**変数名**を使う奴は、ばかなの
alfalfa.livedoor.biz/archives/51364223.html - キャッシュ - 類似

Google検索結果 "変数名"

いいプログラムを書くには
変数名を
正しくつけないとダメ

「どう付けよう？」
「私のこだわり」
「検討を要する事項がたくさん」

一方自然言語は普通名詞を使った

- m_pMainWnd->close()
- 窓を閉めて
- Close the window

一方自然言語は普通名詞を使った

- function(int n)
  {return n*2}
- 受け取った整数を2倍する関数
- Function that doubles the given integer

英語では…（日本語だと暗黙）

“The 型の名前”
　　(例: The window)
という式で
その型の、現在のスコープに
唯一あるインスタンス
　　(例: 今いる部屋の窓)
を取得できる

というわけで

プログラミング言語にも
この機能を
入れてみよう！

作りました

- google://Nikes+名前推論
  - Jikes を改造

```
var _   = “Hello World”
var __  = new JFrame(String);
JFrame.setBounds(0,0,100,100);
JFrame.setVisible(true);
```

Nikes

- 実装
  - 式っぽい場所に型名があったらその型の変数に置換。以上！

```
var _  = “Hello World”
var __ = new JFrame(String);
JFrame.setBounds(0,0,100,100);
JFrame.setVisible(true);
```

※デザインの考察

- 日本語風ではなく英語風に
  ″the″を明示する方がいいかも

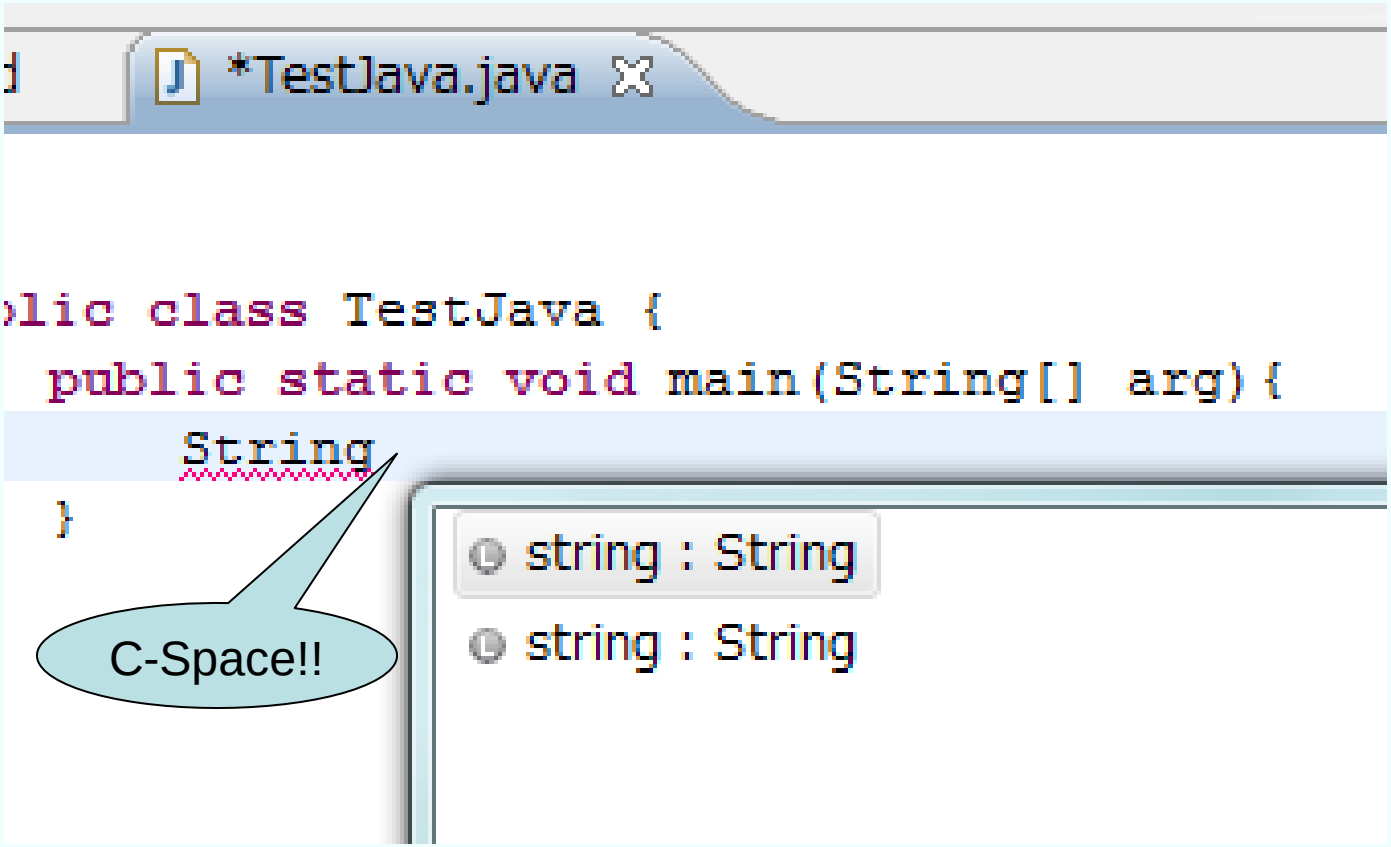
```
var _ = new JFrame($String);
$JFrame.setBounds(0,0,100,100);
```

- 注意！自然言語の語彙まで真似る
  必要はない（真似るのは機能！）

```
var _ = new JFrame(The String);
The JFrame.setBounds(0,0,100,100);
```

※関連する話：Eclipse先生も

・型名と同じ変数名を補完



※関連する話

・それ（なでしこ）、$_(Perl)

「ｘｘと吠える」の「ｘｘ」を
「ワンワン」に置換して、
それを表示する。

or

「ｘｘと吠える」の「ｘｘ」を
「ワンワン」に置換して、
表示する。

※ついでに脱線

・既存の「日本語プログラミング」の
“自然言語の機能を真似る”
的にいいなーと思うところ

[相手]へ、[名前]を、紹介する手順
…
終わり
「次郎」を「太郎」へ紹介する

- 助詞の有効活用（例はプロデル）

“Every”

現在の評価コンテキストを部分継続として切出しコンテナの要素に適用し集計する演算

ネタ元：　“Wild Control Operators”
by Chris Barker, 他色々

しばらく前に見た発言

【急募】Javaプログラミングについて質問です。もし整数が２か３で割り切れるときのみ、その整数をプリントアウトする時は、int number = a; b=a%(2|3); if(b==0) System.out.println(″割れる！！″)でいいんでしょうか…。

分析する

- 自然言語
  2か3で割り切れる
  divisible by 2 or 3
- 直訳
  a%(2||3)==0

自然言語には
〝か〟という、||よりも
超強力な演算子がある

- プログラミング言語
  a%2==0 || a%3==0
- 直訳
  2で割り切れるか
  3で割り切れる

冗長
DRY則に反している
∵ 表現力が貧弱

というわけで

プログラミング言語にも
この機能を
入れてみよう！

実はすでにほぼある

- Icon
  http://www.cs.arizona.edu/icon/

```
every a%(2|3)=0 & write(“割れる！”)
```

- ambオペレータ

最初に 2、次に 3 を
生成するジェネレータ

```
(begin
  (require (= (modulo a (amb 2 3)) 0))
  (print “割れる！”))
```

要するに

- 「か」や「or」は
  継続(continuation) で作れる
- 逆に言うと
- 自然言語は継続を自由自在に
  使いこなすパワフルさ

でも&&っぽいのは直には難しい

- 自然言語
  - 2と3で割り切れる
- 直訳
  - a%(2 && 3)==0 ←？

他の似たような「機能」：every

- value = [1, 2, 3], A = 4
  - A は5より小さい
  - A is less than 5
  - A < 5
  - どの値も 5より小さい
  - Every value is less than 5
  - value.all?{ |x| x < 5 }　　…bad
  - every(value) < 5

他の似たような「機能」：some

- value = [1, 2, 3], A = 4
  - A は偶数
  - A is even
  - 　　isEven(A)
  - どれかの値は偶数
  - Some value is even
  - 　　value.any?{ |x| isEven(x) }　…bad
  - 　　isEven(some(value))

こういうのは全部

- 限定継続 (Delimited Continuation) で説明できる！
  - という研究が最近言語学の方で盛んらしいです

で、(限定)継続とは？

・一言で言うと

「俺、この処理が
終わったら
○○するんだ…」

のこと

で、(限定)継続とは？

```
function operator // (var x, var y)
{
  var f = rest_of_full_expression()
  var g = after_full_expression()
  g( f(x) || f(y) )
}

if( a%(2 // 3)==0 )
  {write(“割れる！”)}
write(…)
```

で、(限定)継続とは？

```
function operator // (var x, var y)
{
  var f = rest_of_full_expression()
  var g = after_full_expression()
  g( f(x) || f(y) )
}

if( a%(2 // 3)==0 )
  {write("割れる！")}
write(…)
```

俺、この // 演算子が
終わったら
fun x → (a%x==0)
を計算するんだ…

で、(限定)継続とは？

俺、if文の条件部が終わったら
fun b →
if(b){write(“割れる”)}write(…)
するんだ…

```
function operator // (var x, var y)
{
  var f = rest_of_full_expression()
  var g = after_full_expression()
  g( f(x) || f(y) )
}

if( a%(2 // 3)==0 )
  {write(“割れる！”)}
write(…)
```

俺、この // 演算子が
終わったら
fun x → (a%x==0)
を計算するんだ…

他の例

```
function every(var list)
{
  var f = rest_of_full_expression()
  var g = after_full_expression()
  foreach( e in list )
    if( !f(e) )
      g(false)
  g(true)
}
if( every([1,2,3]) < 5 ) { … }
```

他の大変マニアックな例

・C++の後置++演算子で
コピー要らなくできる気がする

```
class BigInt {
  BigInt&/<noreturn> operator++(int) {
    var f = rest_of_full_expression();
    var g = after_full_expression();
    var r = f(*this);
    this->incr();
    g(r);
  }}
```

Same

- さらに
  - All students are reading the same book
  - all(students).read(same(book))
- マルチホール継続というのが必要らしい…？

※関連する話

- http://d.hatena.ne.jp/ku-ma-me/20090312/p1
  - map が面倒なので DelegateMap

```
p [1,2,3].dmap + 1      # => [2,3,4]
p [1,2,3].map{|x| x+1} # => 上と等価
```

- http://www.tom.sfc.keio.ac.jp/~sakai/d/?date=20090316#p01
  - 部分継続でDelegateMap

※関連する話

- Chris Barker
  - http://homepages.nyu.edu/~cb125/
- Chung-chieh Shan
  - “Linguistic Side Effects”
  - http://www.cs.rutgers.edu/~ccshan/
- Oleg Kiselyov
  - “Delimited Continuations in Natural Language Semantics”
  - http://okmij.org/ftp/gengo/

この章のまとめ

- 自然言語の「2か3で割り切れる」はとてもパワフルです
- どのくらいパワフルかというと限定継続くらい
  - Or more …?

“▽は○が□”

関数適用の結果をメッセージ
としてオブジェクトへ飛ばす
ファーストクラスメッセージ

ネタ元：『象は鼻が長い - 日本文法入門』
by 三上章, 他色々

主語と述語とオブジェクトとメソッド

- 犬が鳴く
  - dog.cry()
- 猫が鳴く
  - cat.cry()
- というのをふまえて…

主語？　主語？

象は鼻が長い

・どういう文構造と解釈すれば??
・長らく日本語学者の論争の的

解釈A

象は鼻が長い

- 日本語の「は」は「主語」ではなく「主題」を表す

```
with(elephant)
   { trunk.isLong }
```

解釈B

象は鼻が長い

- 「主述述語文」という考え方
- 主語＋述語の組み合わせが再び述語になる

```
elephant.(trunk.isLong)
```

というわけで

プログラミング言語にも
この機能を
入れてみよう！

大雑把に考えて

- 主語＋述語 がまた述語

＝

- オブジェクト.メソッド が、またメソッドに

obj.methodがまたmethodに

- http://shinh.skr.jp/m/?date=20090301#p02
  - それっぽい話題があった！！

```
param2.(param1.(obj.msg))
```

全く美しくないな…
obj.msg は obj を保持したメッセージを返している、 クロージャ的なメッセージ

例1

- shinhさんによる
  - max{…}の部分をまとめて変数につっこみたい

```
even = evens.max{|a,b|
    a.priority <=> b.priority}
odd = odds.max{|a,b|
    a.priority <=> b.priority}
```

例1

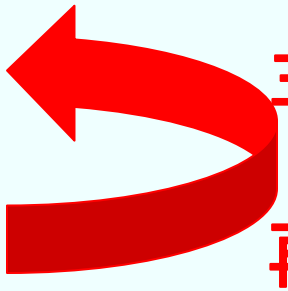
- shinhさんによる
  - max{…}の部分をまとめて変数につっこみたい

```
.maxPrio = .max{|a,b|
    a.priority <=> b.priority}
even = evens.maxPrio
odd  = odds.maxPrio
```

自然言語なら

♪ぞうさん ぞうさん
おはなが ながいのね
そうよ かあさんも ながいのよ

・象は鼻が長い。
・母さんもそう。



主述述語
の
再利用！

例2

・こういうのできたら
　嬉しいよね

```
puts n.rjust(8, “0”)
puts m.rjust(4, “0”)
puts k.rjust(6, “0”)
```

例2

・こういうのできたら
　嬉しいよね

```
.zeroPad = .rjust(_, “0”)
puts n.zeroPad(8)
puts m.zeroPad(4)
puts k.zeroPad(6)
```

まとめ

まとめ：未来言語は…

・自然言語からパクれ！

× 単語や語順や構文を真似る

× NLPの技術を駆使して
人間相手のように曖昧に
命令できる処理系を作る

○ 自然言語の**機能**をパクれ！

※追記：Q&A※

- 最後のって要は部分適用？
  - 機能としてはそうです
  - メソッド/メッセージという概念を崩さずそのままに部分適用を可能にするとどうなるか？的なことを考えてます
- 題の「レトリカル」って？
  - ホントは比喩とか転移修飾とかそういう技法をプログラミングに持って行くレベルの話までしたかったんですが、
    巧い読み替えが思いつかず…＞＜