

Cプログラマのための  
カッコつけないプログラミング  
の勧め

光成滋生(サイボウズ・ラボ)

# 自己紹介

## 好きな言語

- ▶ C++, Xbyak (自作x86/x64用C++用JITアセンブラ)
  - ・ C++WG小委員会エキスパート

## まあまあ使う言語

- ▶ Python, JavaScript, Haskell

## サイボウズ・ラボで最近やってること

- ▶ 暗号の高速実装研究
- ▶ 自然言語処理 (まだ始めたばかり)

# 昨今の言語

## 多機能, 高度な概念

- ▶ C++0x
  - ・ 型推論, ラムダ式, move semanticsなどが追加される
- ▶ Haskell
  - ・ 高階関数, 参照透過性, モナド, 非正格評価…

# 難しい概念は理解も使用も大変

人にものを教えるには, その7倍の理解が必要  
(要出典)

プログラマが難しい概念を使うと間違えやすい  
▶ 簡単な概念しか使わないでプログラムを書こう!

# 例題

引数に与えられたnまでの階乗を表示する

- ▶ 引数がないければn = 1とする

<http://e-arrows.sakura.ne.jp/2010/08/is-lisp-really-has-too-many-parenthesis.html>

```
>. /fac  
1! = 1
```

```
>. /fac 9  
1! = 1  
2! = 2  
3! = 6  
4! = 24  
5! = 120  
6! = 720  
7! = 5040  
8! = 40320  
9! = 362880
```

# 普通の回答

```
#include <stdio.h>
#include <stdlib.h>

int fac(int n)
{
    return n == 1 ? 1 : fac(n - 1) * n;
}

int main(int argc, char *argv[])
{
    int n = argc == 1 ? 1 : atoi(argv[1]);
    for (int i = 1; i <= n; i++)
        printf("%d! = %d¥n", i, fac(i));
}
```

# 普通の回答の駄目なところ(1/7)

関数の定義に再帰を使ってる

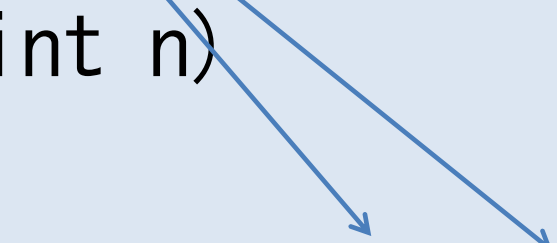
▸ 再帰なんて難しい

```
int fac(int n)
{
    return n == 1 ? 1 : fac(n - 1) * n;
}
```

# 普通の回答の駄目なところ(2/7)

条件演算子は分かりにくい

```
int fac(int n)
{
    return n == 1 ? 1 : fac(n - 1) * n;
}
```





# 普通の回答の駄目なところ(3/7)

ループなんて時代遅れ

```
for (int i = 1; i <= n; i++)  
    printf("%d! = %d¥n", i, fac(i));
```

# 普通の回答の駄目なところ(4/7)

ぶらさがりfor

- ▶ インデントが崩れてると読み間違える

```
for (int i = 1; i <= n; i++)  
    printf("%d! = %d¥n", i, fac(i));
```

# 普通の回答の駄目なところ(5/7)

副作用のあるコードなんて論外

▶ 時代は参照透過性

```
for (int i = 1; i <= n; i++)  
    printf("%d! = %d¥n", i, fac(i));
```

# 普通の回答の駄目なところ(6/7)

## 型安全でないprintf

- ▶ %dの代わりに%sを使うと落ちるかも

```
for (int i = 1; i <= n; i++)  
    printf("%d! = %d¥n", i, fac(i));
```

# 言語による解決法

## ループ

- ▶ foreach, xrangeなどの専用構文, リスト内包表記  
ぶら下がりfor
- ▶ Pythonではインデントで明示化

## 副作用のある式

- ▶ Haskellでは副作用のあるコードは書けない
- ## 型安全でないprintf
- ▶ いくつかの言語で型推論による安全なprintf

でも難しそうな概念は習得も大変そう

# ところで先程のblogの(本当の)目的

括弧((), [], {})の数の少なさを競う

- ▶ Perl, Haskell, Ruby, Pythonでは0も可能

言語	Java	Scheme	Java Script	Clojure	Common Lisp	Perl
括弧の数	24	20	16(6)	18	12	4(0)

- ▶ 我等がCでは...

# 普通の回答の駄目なところ(7/7)

括弧の数が22個もある!

```
#include <stdio.h>
#include <stdlib.h>
int fac(int n) {
    return n == 1 ? 1 : fac(n - 1) * n;
}
int main(int argc, char *argv[]) {
    int n = argc == 1 ? 1 : atoi(argv[1]);
    for (int i = 1; i <= n; i++)
        printf("%d! = %d¥n", i, fac(i));
}
```



# やさしいCプログラムを書こう

そのためには

- ▶ 副作用のあるコードを書かない
- ▶ ループなんて書かない
- ▶ 再帰はもちろんしない
- ▶ printfも使わない
- ▶ 条件演算子も使わない

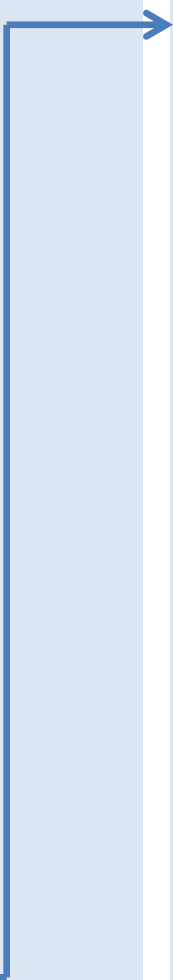
加えて

- ▶ 括弧も使わない

# カッコつけないプログラム

```
#include <stdio.h>
#include <stdlib.h>
```

```
a = atoi;
b = printf;
c = 0x20216425;
d = 0x7525203d;
e = 0x0000000a;
main = 0x55535756;
f = 0x000000e8;
g = 0x6c8b5e00;
h = 0xfd831424;
i = 0x8b127401;
j = 0x8b18246c;
```



```
k = 0x8b55046d;
l = 0xd5ffe36e;
m = 0x8904c483;
n = 0x0cec83c5;
o = 0x89eb468d;
p = 0x768b2404;
q = 0x43db31e7;
r = 0xaf0fdf89;
s = 0x245c89fb;
t = 0x247c8904;
u = 0x43d6ff08;
v = 0xee76eb39;
w = 0x5d0cc483;
x = 0xc35e5f5b;
```

# 解説

- ▶ 型名が無い
- ▶ 変数の宣言のみ
- ▶ 副作用が無い
- ▶ ループが無い
- ▶ 条件演算子もifも無い
- ▶ もちろん再帰も無い
- ▶ というか関数が無い

とても易しい

```
#include <stdio.h>
#include <stdlib.h>
a = atoi; b = printf;
c = 0x20216425; d = 0x7525203d;
e = 0x0000000a; main = 0x55535756;
f = 0x000000e8; g = 0x6c8b5e00;
h = 0xfd831424; i = 0x8b127401;
j = 0x8b18246c; k = 0x8b55046d;
l = 0xd5ffe36e; m = 0x8904c483;
n = 0x0cec83c5; o = 0x89eb468d;
p = 0x768b2404; q = 0x43db31e7;
r = 0xaf0fdf89; s = 0x245c89fb;
t = 0x247c8904; u = 0x43d6ff08;
v = 0xee76eb39; w = 0x5d0cc483;
x = 0xc35e5f5b;
```

# デモ

x86 (32bit) のWindows, Linux, Mac上で動作

- ▶ Visual Studioなら>cl fac.c
  - ・ OSによってはDEPをoffにする必要あり
- ▶ gccなら>gcc fac.c -m32 -z execstack
- ▶ 古いMac OSだと何もしなくても動く
  - ・ 新しいMacは?