

---

# Crystal紹介とRubyとの比較

---

2016/02/03

OSS推進フォーラム アプリケーション部会

富田昌宏



Japan OSS Promotion Forum

## プログラミング言語

### ■ オブジェクト指向

- (ほぼ)すべてがオブジェクト
- 演算子もメソッド

### ■ スクリプト

- コンパイル不要
- 実行環境に Ruby が必要

## ■ 動的

- クラス/メソッド定義が動的
- 実行時にメソッドが定義されてればOK

## ■ 型なし

- ダックタイピング
- オブジェクトがメソッドを持っていればいい

```
def hoge(x)
  p 123 * x  # 数値を期待
end
hoge("abc") # 文字列を渡しちゃった
```

```
`*': String can't be coerced into Fixnum (TypeError)
```

```
begin
  # 何かの処理
rescue => error
  # めったに起きないエラー
  logger.error error.message
end
```

undefined method `message'



**Yukihiro Matsumoto**

@yukihiro\_matz

フォローする

これ、Rubyでもそのまま動くんじゃないか？ Crystalすげーつ。  
[twitter.com/Linda\\_pp/statu...](https://twitter.com/Linda_pp/status/604111111111111111)

2015年6月17日 01:14



28



40



## CRYSTAL

The Programming Language

### Language Goals

- ✓ Ruby-inspired syntax.
- ✓ Statically type-checked but without having to specify the type of variables or method arguments.
- ✓ Be able to call C code by writing bindings to it in Crystal.
- ✓ Have compile-time evaluation and generation of code, to avoid boilerplate code.

<http://crystal-lang.org>

- Ruby風の静的型ありコンパイル言語
- Linux / OS X
- バージョン 0.11.1 (2016/1/26)
- まだ正式リリース前
- CrystalはCrystal自身で書かれている
- Apache License 2.0



```
class Hoge
  def foo
    p "abc"
  end
end
Hoge.new.foo
```

```
% crystal hoge.cr
"abc"
```

# Ruby風？

| Crystal | Ruby    |
|---------|---------|
| 静的      | 動的      |
| 型あり     | 型なし     |
| コンパイル言語 | スクリプト言語 |

スクリプト風の実行

```
% crystal hoge.cr
```

コンパイルして実行ファイル作成

```
% crystal build hoge.cr  
% ./hoge
```

```
def hoge(x)
  p 123 * x      # 数値を期待
end
hoge("abc")    # 文字列を渡しちゃった
```

```
% crystal build hoge.cr  
Error in ./hoge.cr:4: instantiating 'hoge(String)'
```

```
hoge("abc")  
^~~
```

```
in ./hoge.cr:2: no overload matches 'Int32#*' with types String
```

```
Overloads are:
```

- Int32#\*(other : Int8)
- Int32#\*(other : Int16)
- Int32#\*(other : Int32)
- Int32#\*(other : Int64)
- Int32#\*(other : UInt8)
- Int32#\*(other : UInt16)
- Int32#\*(other : UInt32)
- Int32#\*(other : UInt64)
- Int32#\*(other : Float32)
- Int32#\*(other : Float64)

```
p 123 * x  
    ^
```

```
# フィボナッチ数列のN番目の値
def fib(n)
  if n < 2
    n
  else
    fib(n-1) + fib(n-2)
  end
end
p fib(35)
```

# 速い!

```
% time ruby fib.rb
9227465
real    0m2.395s
user    0m2.396s
sys     0m0.000s
```

```
% time crystal fib.rb
9227465
real    0m0.396s
user    0m0.344s
sys     0m0.088s
```

# コンパイルするとさらに速い！

```
% time ruby fib.rb
9227465
real    0m2.395s
user    0m2.396s
sys     0m0.000s
```

```
% crystal build fib.rb --release
% time ./fib
9227465
real    0m0.087s
user    0m0.084s
sys     0m0.000s
```



RubyスクリプトをCrystalで実行すれば速くなる！

…なんてうまい話はない

CrystalとRubyは結構違う

- "A" - 文字列
- 'A' - 文字 (Ruby では文字列)

```
"A"+"B" #=> "AB"
```

```
'A'+ 'B' #=> no overload matches 'Char#+' with types Char
```

- Crystal の文字列は変更不可能
- 文字/文字列のエンコーディングは UTF-8 のみ
- バイナリデータは文字列としては扱えない

```
0x7FFFFFFFF.class #=> Int32
0x80000000.class #=> Int64
0x7FFFFFFFF      #=> 2147483647
0x7FFFFFFFF+1    #=> -2147483648
0x7FFFFFFFFi64+1 #=> 2147483648
```

(Rubyでは整数の桁の制限はない)

- eval がない
- require はコンパイル時
- クラス/メソッド定義はコンパイル時

# 実行時エラーは発生箇所がわからなくて厳しい

## 正式版に期待

```
% crystal eval 'a=[1,2,3]; a[4]'  
Index out of bounds (IndexError)  
[4336327] *CallStack::unwind:Array(Pointer(Void)) +87  
[4336218] *CallStack#initialize<CallStack>:Array(Pointer(Void)) +10  
[4336170] *CallStack::new:CallStack +42  
[4344159] *Exception +31  
[4344093] *IndexError#initialize<IndexError, String>:CallStack +29  
[4344033] *IndexError::new<String>:IndexError +97  
[4343925] *IndexError::new:IndexError +21  
[4332456] *Array(Int32) +120  
[4332326] *Array(Int32) +6  
[4323255] ???  
[4330281] main +41  
[140456278723136] __libc_start_main +240  
[4320409] _start +41  
[0] ???
```

## ☆ Ruby脳にはCrystalつらい Advent Calendar 2015

タイプ別



1



0



★ 24



Tweet



13

作成者: tmtms

Ruby脳のままCrystal使うとつらい

| 日   | 月   | 火   | 水   | 木   | 金   | 土   |
|---|---|---|---|---|---|---|
| 29  | 30  | 1<br>tmtms<br>Crystal は配列が自動拡張<br>されなくてつらい                            | 2<br>tmtms<br>Crystal は実行時エラーで<br>ソース位置がわからなく<br>てつらい | 3<br>tmtms<br>Crystal は深さが不定の配<br>列が作れなくてつらい            | 4<br>tmtms<br>Crystal でも深さが不定の<br>配列が作れてつらくな<br>い...? | 5<br>tmtms<br>Crystal は日本語のドキュ<br>メントがあってつらくな<br>い!       |
| 6<br>tmtms<br>Crystal は配列の範囲外の<br>参照がエラーになってつ<br>らい | 7<br>tmtms<br>Crystal があちこちに<br>.crystal というディレク<br>トリを作ってつらい | 8<br>tmtms<br>Crystal は配列内要素に対<br>してすべての要素が持つ<br>メソッドしか呼び出せな<br>くてつらい | 9<br>tmtms<br>Crystal にはタブルがあっ<br>てつらくない              | 10<br>tmtms<br>Crystal は配列を渡した<br>引数をメソッドに渡せな<br>くてつらい? | 11<br>tmtms<br>Crystal は Ruby にあるメ<br>ソッド名がなくてつらい     | 12<br>tmtms<br>Crystal のインスタンス変<br>数が nilable になってつ<br>らい |
| 13<br>tmtms   | 14<br>tmtms   | 15<br>tmtms   | 16<br>tmtms   | 17<br>tmtms   | 18<br>tmtms   | 19<br>tmtms   |

# 型



“ もしもそれがアヒルのように歩  
き、アヒルのように鳴くのな  
ら、それはアヒルである

”

クラスが異なってもメソッドが同じように動く  
メソッドがあればいい

# Rubyと同様のダックタイピング

```
class A
  def bar() end
end
class B
  def bar() end
end

def foo(obj)
  obj.bar() # objにbarメソッドがあればよい
end
foo(A.new)
foo(B.new)
```

```
def hoge(a, b)
  a + b
end
```

```
hoge("abc", "xyz") #=> "abcxyz"
```

```
hoge(1, 2) #=> 3
```

```
hoge("abc", 2) #=> コンパイル時エラー
```

```
def hoge(a, b)
  a + b
end
```

```
def hoge(a : String, b : Int)
  a * b
end
```

```
hoge("abc", "xyz") #=> "abcxyz"
```

```
hoge(1, 2) #=> 3
```

```
hoge("abc", 2) #=> "abcabc"
```

```
var = rand() < 0.5 ? 123 : "abc"  
var.size # Int でエラー  
var + 1 # String でエラー
```

```
var = rand() < 0.5 ? 123 : "abc"  
if var.is_a? Int  
  var + 1  
else  
  var.size  
end
```

```
# 配列には特定の型しか入らない
a = [1, 2, 3] #=> Array(Int32)
a.push 4      #=> OK
a.push "a"
  #=> no overload matches 'Array(Int32)#push'
  #   with types String

# 空配列リテラルは型指定が必要
a = []      #=> for empty arrays use '[] of ElementType
a = [] of Int32  #=> OK
```

# CrystalとRubyの性能比較



- The Ruby Benchmark Suite
- <https://github.com/acangiano/ruby-benchmark-suite.git>
- Crystal で動作するようにテストを改変

## Crystal

Crystal 0.11.0 [ed88d5f] (Sat Jan 23 17:52:40  
UTC 2016)

## Ruby

ruby 2.3.0p0 (2015-12-25 revision 53290)  
[x86\_64-linux]

## PC

---

ThinkPad X220

Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz

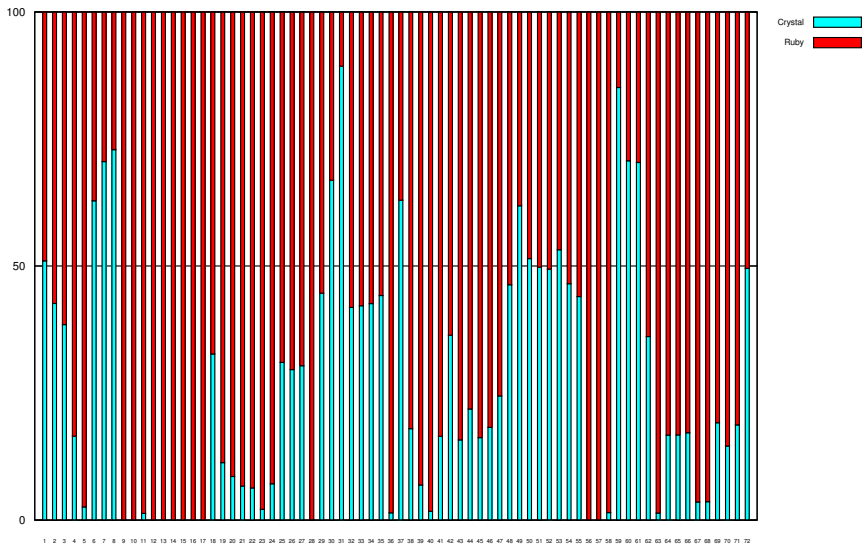
Memory 8GB

## OS

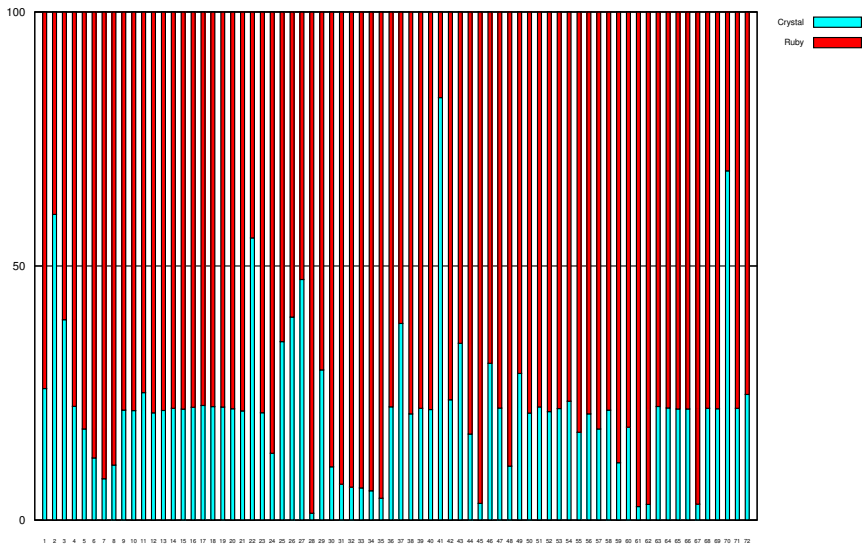
---

Ubuntu 15.10 (Kernel 4.2.0-25)

# 測定結果：時間



# 測定結果: メモリ



- Ruby風の静的型付きコンパイル言語
- コンパイル時にエラーを検出できる
- 生成された実行ファイルはCrystalが無い環境でも実行可能
- Ruby風だけどRubyとはかなり異なる
- Rubyに比べて概ね速くて小さい

