小≪ ○ ○ ○ ◉ ≫大 サイズ変更

未来の言語は「APL」？ Rubyのまつもと氏が講演
NewsInsight

言語進化のダイナミズムとは
未来の言語は「APL」？ Rubyのまつもと氏が講演

2009/02/13

「今日はRubyの話はしません」。プログラミング言語「Ruby」の生みの親で開発コアメンバーでもある、まつもとゆきひろ氏は冒頭でそう話すと、自身のプログラミング経歴や半世紀に及ぶプログラミング言語の歴史を外観しつつ、未来のプログラミング言語へ向けた構想について語った。

書籍だけでPascalを習得した高校生

2009年2月12日、翔泳社主催で東京・目黒で行われた「Developers Summit 2009」でまつもと氏は「未来へつながる言語〜ある言語おたくの視点から」と題した講演を行った。立ち見が出るほど詰めかけた観衆に向かって、“最も有名なプログラミング言語オタク”として自身のプログラミング言語観を披露した。


ネットワーク応用通信研究所フェロー／楽天 技術研究所フェローのまつもとゆきひろ氏

1980年代の高校生時代からプログラミング言語が好きだったというまつもと氏だが、一番最初に使った言語はBASICだったという。ところが、ローカル変数や構造化プログラミングの手法が使えないBASICでは、大きなプログラムを書こうと思うと、すぐに限界に突き当たった。雑誌や書籍でPascalの存在を知り「実際に動かしたこともないのに、本を頭から最後まで読んで分かった気になっていた（笑）」と自嘲気味に振り返る。周囲にコンピュータのことを聞ける相手がいるわけでも、インターネットやCD-ROMがある環境でもなく、書籍だけでプログラミングを学んだ高校生のまつもと氏は特殊だったと言えそうだが、もっと特殊なのは「どうしてプログラミング言語を選ばなきゃいけないんだ、もしかして自分にもできるんじゃないかと思った」（まつもと氏）ことだろう。「周りに、そういう話をできる人がいなかったので、プログラミング言語を設計することが普通のことなのかヘンなことなのか、全然分からなかったんですよ」。

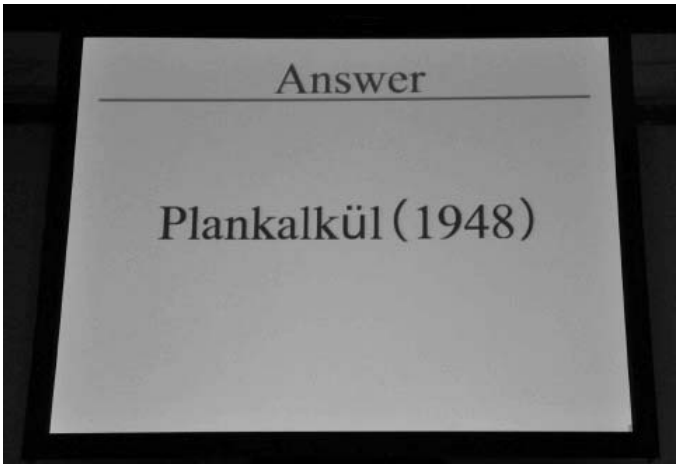
その後、実際にさまざまなプログラミング言語を使い、自分の言語を設計・実装するには大学4年生になるまで待たなければならなかったという。


まつもと氏の講演には立ち見が出るほどたくさん聴衆が詰めかけた

世界初の言語は？ Fortran!? ブブッー

無類のプログラミング言語好きのまつもと氏は、自身の言語遍歴を語り終わると、おもむろに会場に向かって“プログラミング言語トリビア・クイズ”を出した。その問題とは「世界最初のプログラミング言語は何でしょう？」というものだ。

ほとんどの人の頭に浮かぶ答えは1950年代に開発された「Fortran」（1954年）、「Lisp」（1958年）、「COBOL」（1959年）辺りだが、まつもと氏が用意していた答えは違った。正解は歴史に埋もれてしまっていたドイツ発の「Plankalkul」（プランカルキュル：英語で言うとPlan Calculus。実際のスペリングはuにウムラウトが付く）というプログラミング言語で、それは1948年にまでさかのぼるのだという。



世界初の言語、それは「Plankalkul」

商業的に成功し、今なおHPC分野では現役のFortranが単純なジャンプ命令程度しかなかったのに対して、Plankalkulには制御構造と呼べるものがあったり、例外処理まで存在していたという。そうしたトリビア知識だけでも、まつもと氏の言語おたくぶりに会場は沸いたが、さらにまつもと氏が「ただ、Plankalkulは仕様が複雑すぎて実装できなかったんですよ。なんと最初の実装が動いたのは2001年だったんですね」というと会場は笑いに包まれた。「作者のコンラート・ツーゼは1995年に亡くなっているので、生前に自分のプログラミング言語が実際に動くのを見ることができなかったというのは不幸です」。

温故知新、1950年代に生まれた言語たち

Fortran、COBOL、Lispはいずれも1950年代に生まれたプログラミング言語だが、半世紀を経た現在も使われている言語でもある。「OSなどに比べると、プログラミング言語の寿命は非常に長いのです」（まつもと氏）。過去の遺物として切り捨てる論調もあるが、まつもと氏がこれらの言語を挙げてプログラミング言語の歴史を語り始めた理由は、それらが現場利用において相変わらず重要な意味を持っていることを示すためと、言語進化のダイナミズムを説明するためだ。

FortranはFormula Translating Systemという名前が示すとおり、数式・計算処理のために開発された言語だ。「Fortranはすばらしいツール。過去のソフトウェア資産を使って、最速に結果がほしい人にはほかに代えられないもの」。Fortranよりも、もっと新しいパラダイムを取り入れたプログラミング言語を使ったほうがいいという主張に対してまつもと氏は、「例えば天気予報をやるような人はコンピュータサイエンスなどどうでもいい」のだと指摘する。Fortranは1つの命令で並列に処理を走らせるSIMD系の処理を得意とし、ベクトル型コンピュータとの相性が良いという面もあるという。

まつもと氏が勤務するネットワーク応用通信研究所（NaCl）は、RubyやRuby on Railsの開発・コンサルティングで知られるが、COBOL案件も同じぐらい多いというのは意外に知られていない。まつもと氏によれば、全国約6000の病院に入っている保険料の点数計算システム「レセプト」は、NaClがオープンソースで実装したシステムを使っているという。既存のシステムを作っていた技術者はCOBOLエンジニア。医療システムや実務の知識に精通していて「彼らの知識を取りこぼすのはあまりに惜しい」という判断から、NaClではLinux上にCOBOL環境を実装したのだという。汎用機のトランザクションモニタやCOBOLコンパイラは独自実装で、X Window System上のGUIライブラリ「GTK」とリンクするライブラリも自作。その上にCOBOLで書いたソフトウェアを載せて全国の病院に納入したという。

COBOLには人的資産、レコード単位での処理などで優れた点がある。「COBOL技術者と話すことが良くあるんですが、右から左に帳票を流すシステムであればCOBOLの生産性はRailsにも負けないと言われることがあるんですよ。ぼくにはよく分かりませんが、ハイと答えるしかないですよね」（まつもと氏）。

言語進化のダイナミズム

FortranもCOBOLも現代に生きている。では、1950年代に生まれたもう1つの言語「Lisp」についてはどうか。Lispは人工知能研究ではデ・ファクト・スタンダードとなったし、アカデミックな世界では使われているが、一般に広く普及した言語とは言いがたい。しかし、Lispに含まれていたアイデアや機能は、現代的な言語にも取り入れられているという。

「VMやGC、例外処理など、Lispで当然だったものがJavaで紹介されて一般化した」（まつもと氏）。

BLISS、C、C++などのシステム記述に適した言語は別として、これらの機能はJavaだけではなく、Ruby、Python、Perlなど現代的なプログラミング言語にも、すべて、あるいは一部が取り入れられている。

LispやSmalltalkといった言語は、それ自身が広く普及することはなかったが、ほかの言語にそのエッセンスが取り込まれていくという流れがあった。まつもと氏は、言語進化は生物の進化にも似ているという。プログラミング言語は、発生や分岐を繰り返し「力」のあるものが生き残って進化を続けていく。力となるような機能やパラダイムは幅広い言語に取り入れられることで生き残っていく。こうした流れには「冒険→改良→普及」というサイクルがあるという。

第1フェーズでは冒険的で革新的な機能を実装するが、普及はしない。JavaのVMがそうであったように「遅くてダメじゃんという話になる」（まつもと氏）ことがある。第2フェーズは洗練のフェーズで、「けっこういいじゃんと、一部に受け入れられる」。そして、普通の人が使えるようになっていく。

講演するまつもとゆきひろ氏

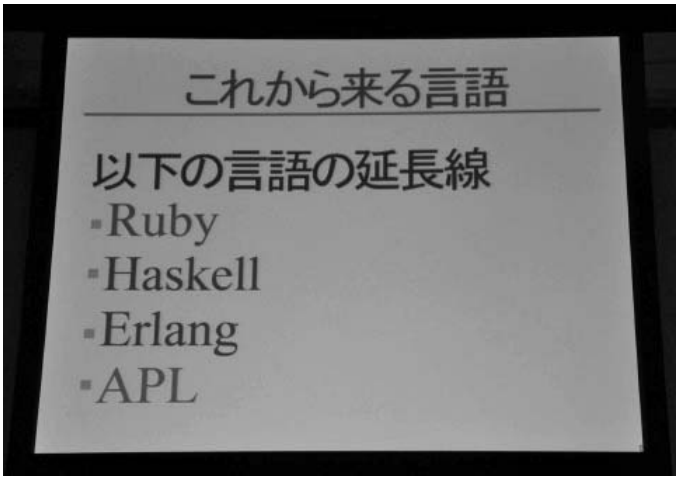
歴史に埋もれたロスト・テクノロジ

現代的言語が似通った機能を取り入れる一方、FortanやCOBOL、あるいはLispの例から分かるのは、まだ現代の言語に入っていない“ロスト・テクノロジ”があるのではないか、というのがまつもと氏の指摘だ。

言語は新しいパラダイムを取り入れる形で進化してきた。「最近は構造型プログラミングという言葉を聞かなくなりましたよね。それは当たり前になったからです」（まつもと氏）。同様にオブジェクト指向も、今やほとんどすべての言語に何らかの形で取り込まれている。

では、次の世代のプログラミング言語が取り入れる可能性があるパラダイムとは何か？ 次に来るロスト・テクノロジとは何か？

まつもと氏は、「まだメインストリームになっていない言語で、パラダイムシフトがある言語、これから来る言語は以下の言語の延長線にあるのではないか」と述べて、Ruby、Haskell、Erlang、APLの4つを挙げた。



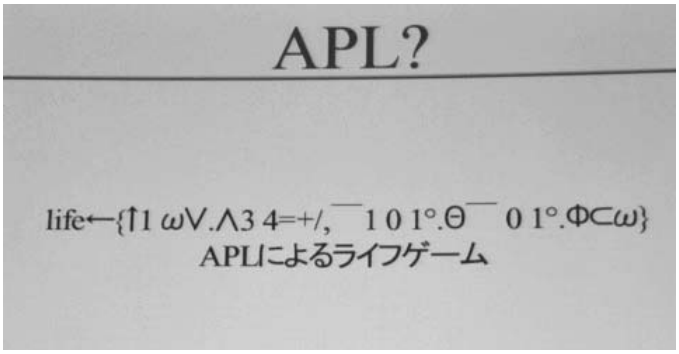
Rubyを除くこれらの言語は誕生年代は古いが、次にメジャーになる言語に取り込まれるロスト・テクノロジが含まれているかもしれない。そういう意味では、まつもと氏がいう「冒険」のフェーズにあるかもしれない言語たちだ。「古くからあって忘れられているのだけど、今の状況に適用すると結構面白かった、という技術はあるかもしれない」（まつもと氏）。

Haskell、OCaml、Standard ML、あるいはErlangは、関数型言語として近年注目を集めている。関数型言語は、より抽象度やモジュール性の高いプログラミングが可能で、並列処理に有利な特徴を多く備えている。プロセッサのコア数が増加しつつある現在、並列処理をソフトウェアでサポートするニーズは高まっており、こうした面からも普及が見込まれる。

すでにRubyにも、関数型の特徴が含まれている。ブロックと呼ばれる構造や手続きオブジェクトを使い、高階関数の扱いができることが1つだし、変数の中身を変更してしまうメソッドを「破壊的」と呼ぶのは、関数型言語の特徴の1つとされる「参照透明性」を意識してのことだろう。さらに、先日初の安定版リリースが出たばかりのRuby 1.9では関数型らしさを増す文法やメソッドが追加されているという。「すでに存在する言語、あるいはこれから出てくる言語に対して関数型言語が影響を与えるということは十分にあるのではないか」（まつもと氏）。

Erlangは関数型であると同時にProlog、KL/1などに連なる論理型プログラミングと呼ばれるマイナーなパラダイムの上にもある。論理型言語は、組み合わせの数が多いものの中から条件に合致したものを探すゴールシーキングと呼ばれる問題を解くのに向く、という特徴があるという。

“A Programming Language”を略した「APL」という身もフタもない名前を持つ1960年代生まれのプログラミング言語は、現代的プログラミング言語とは、かなり異質だ。まつもと氏がAPLで書かれたライフゲームのソースコードを示すと、あまりの字面の異様さに会場からは笑いがこぼれた。



APLが登場した当時には、特殊な記号を入力するために専用の入力デバイスまで必要だったという。そんなプログラミング言語トリビアを紹介して会場の笑いを誘いつつも、まつもと氏は、APLが持つ特徴が、今後のプログラミング言語に影響を与える可能性を示唆する。APLは、SIMD系命令の実行に適したようなベクトル演算を基本としている。コア数が増えるプロセッサや、より多くのマルチメディア処理を行うアプリケーションの存在により現在、こうした並列処理に向く言語が見直される可能性があるというわけだ。

APLで使われる記号類は、すでにUnicodeにすべて含まれている。ただ、「さすがにアルファベットの範囲内でやろうということで、APLに似た言語としてKとかJというものが出てきている」（まつもと氏）のだという。さらに、最近データ処理や統計処理に適した言語としてにわかに注目を集めている「R」など、APLのように配列やベクトルの処理を得意とする言語もある。

世界初のプログラミング言語「Plankalkul」から始まり、APLというロスト・テクノロジの話で終わった、まつもと氏の講演。該博なプログラミング言語進化史の知識に裏打ちされた、今後のプログラミング言語パラダイムの展望は、多くの示唆に富んでいる。

※記事初出時、APLを1950年代生まれと記述してありましたが正しくは1960年代です。訂正してお詫びいたします。

関連リンク

- Developers Summit 2009

関連記事

- 高速化したRuby 1.9系、初の安定版リリース（@ITNews）
- Ruby普及でNaClなど3社が提携、CodeGearのIDEを推奨（@ITNews）
- Rubyは10年前のJava（@ITNews）
- まつもと×笹田、Ruby 1.9を語る（@ITNews）
- Rubyが抱える課題、NaClの前田氏が講演（@ITNews）

（@IT 西村賢）

情報をお寄せください：tokuho@ml.itmedia.co.jp

---

recompile.net

前の記事 | このブログのトップへ | 次の記事

May 21, 2007

J言語でFizzBuzz

酔っぱらった勢いで、J言語でFizzBuzzを書くと言ってしまったらしい高井です。くだんのJ言語ですが、http://jsoftware.com/から処理系をダウンロードすることができます。jconsoleを実行すると、対話式のインタプリタが起動します。パスを間違えると、JMXの監視ツールが起動してしまいますので、ご注意ください。

さて、J言語でやってみるといっても、大学2年の頃にやったことなど、すっかり覚えていないわけです。まずは整数列をつくるところから手をつけてみました。ぐぐってやると、i.という演算子が0から引数までの整数列をつくる演算子だということがわかりました。

```
   i. 10
0 1 2 3 4 5 6 7 8 9
```

今回欲しい数値は、1から始まる数値なわけです。そこで、この集合の要素それぞれに1を足してみましょう。

```
   (i. 10) + 1
1 2 3 4 5 6 7 8 9 10
```

とりあえず、1から100までの整数列をつくることができました。演算子の優先順位の都合から、表現をかえます。

```
   1 + i. 100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
```

ここで、いったんAを1から100の整数列とおきます。

```
  A =. 1 + i. 100
```

剰余は|という演算子を利用します。

```
   15 | A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0
```

さて、1から100までの数値のうち、15の倍数の集合を考えてみましょう。剰余が0の場合が割り切れるということですから、剰余が0のときだけ、1というフラグをたてるようにします。ここは、もう聞かないでください。

```
   0 E. 15 | A
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

フラグがたっている部分だけをコピーするには#を利用します。

```
   (0 E. 15 | A) # A
15 30 45 60 75 90
```

X m } Yという式(?)は、インデクスmで指定したYの要素をXにします。例えば、0から9までのうち、3、5、7を0にしたい場合は、次のようにできます。

```
  0 (3 5 7) } i. 10
0 1 2 0 4 0 6 0 8 9
```

ここまで、くれば後の考え方は単純です。1から100までの数列のうち、15の倍数をFizzに、3の倍数をBuzzに、5の倍数へと修正してあげればいいわけです。というわけで、とりあえず15の倍数から手をつけてみましょう。

```
 'Fizz' (((0 E. 15 | A) # A) - 1) } A
|domain error
|   'Fizz'    (((0 E.15|A)#A)-1)}A
```

ううー。数値と文字の変換あたりでエラーがでてしまうようです。うまく文字列と数値を置換してあげる方法については、時間切れでした……。心より恥じる。どなたかJ言語ハッカーの方、お助けを！！

-1をFizzBuzz、-2をFizz、-3をBuzzとでも読み替えてあげてください。

```
   A =. 1 + i. 100
   A =. _1 (((0 E. 15 | A) # A) - 1) } A
   A =. _2 (((0 E. 3 | A) # A) - 1 ) } A
   A =. _3 (((0 E. 5 | A) # A) - 1 ) } A
   A
1 2 _2 4 _3 _2 7 8 _2 _3 11 _2 13 14 _1 16 17 _2 19 _3 _2 22 23 _2 _3 26 _2 28 29 _1 31 32 _2 34 _3 _
```

記事検索
検索

最新記事
エンジニアの未来サミット0905で登壇しました
livedoor Blogへの引越し
QCon Tokyo 2009へ行ってきました（続き）
QCon Tokyo 2009へ行ってきました
『プログラミング言語Ruby』出版記念トーク懇親会
『プログラミング言語Ruby』出版記念トーク懇親会締切間近！
『プログラミングRuby』出版トーク懇親会のお知らせ
仙台Ruby会議01でライトニングトーク発表をしました
ジュンク堂池袋本店トークセッションにお邪魔します
『Javaプログラマのための Ruby入門』が出版されます

Archives
May 2009
April 2009
February 2009
January 2009
November 2008
October 2008
August 2008
July 2008
June 2008
May 2008
April 2008
March 2008
February 2008
January 2008
December 2007
November 2007
October 2007
September 2007
August 2007
July 2007
June 2007
May 2007
April 2007
March 2007
February 2007
January 2007
December 2006
November 2006
October 2006
September 2006
August 2006
July 2006
June 2006
May 2006
March 2006
February 2006
November 2005

QRコード

at 13:44 | コメント(5) | トラックバック(0)

トラックバックURL

http://trackback.blogsys.jp/livedoor/takai_naoto/51148807 クイック

コメント一覧

1. Posted by satze October 12, 2007 01:50
   Jを使って研究をしていた通りすがりです.
   知り合いからこのBlogの記事を教えられ, 作ってみました.
   例えばこんなのどうでしょうか？
   ' ' (('|' E. B) # i.#B) } B=.}.}:1{":(<'FizzBuzz') (<:(0 E. 15 | A) # A) } (<'Buzz') (<:(0 E. 5 | A) # A) } (<'Fizz') (<:(0 E. 3 | A) # A) } <"0 A=.>:i.100
   (1行)
   もしくは,
   FizzBuzz=: 3 : 0
   ' ' (('|' E. B) # i.#B) } B=.}.}:1{":(<'FizzBuzz') (<:(0 E. 15 | A) # A) } (<'Buzz') (<:(0 E. 5 | A) # A) } (<'Fizz') (<:(0 E. 3 | A) # A) } <"0 A=.>:i.y.
)
   として,
   FizzBuzz 100
   でしょうか.（これは1行で書くのは僕の能力では無理でした･･･）
   いかがでしょうか？

2. Posted by takai October 12, 2007 06:15
   J言語ハッカー！！ 一見して、さっぱりわからないのがすごいです！！

3. Posted by satze October 12, 2007 06:59
   >J言語ハッカー！！
   いえいえ. 通りすがりのJを使っていた学生です･･･. あくまで所属している研究室の先生がJ使いという影響でJを使っていただけで･･･.
   プログラムの簡単な解説を付け忘れていました.
   takeiさんのプログラムですと,
   3の倍数をFizzにしたりするところで, 数値の列に文字を代入しようとしてエラーが起きていると思われます.
   まず, 数値列をいったんボックス（と呼ばれる1つずつの区切り）に入れ, 3, 5, 15の倍数をFizz等に変換します.
   その後ボックスを解きたいのですが, どうもうまくいかなかったので, 強引に文字の変換をして答えを導いています.
   もっとうまいやり方があるような気がするのですが, 今の僕の実力ではこんな感じです･･･.

4. Posted by satze October 12, 2007 08:42
   たびたびすいません.
   上記のプログラムだと, JのVersion6（の少なくともWindows版）だとうまくいかないみたいです･･･.

5. Posted by takai October 12, 2007 10:01
   いえいえ、情報ありがとうございます！

コメントする

このブログにコメントするにはログインが必要です。

niming538の日記

<前の日 | 次の日>

2009-03-09 APL/J言語でFizzBuzz

FizzBuzz問題というのがあって、1から100までの数字で3で割り切れる場合は'Fizz'、5で割り切れる場合は'Buzz',15で割り切れる場合は'FizzBuzz'、その他はそのままの数字をアウトプットせよ、というのだったと思う。APLやJ言語で解いた例もないわけではないのですが、それを読む前に今の知識と道具でチャレンジしてみました。

Jでやろうと思うのですが、ひとつずつ処理していくというのがJ言語らしくないので、配列で考えなければいけない。あと数字と文字列を配列の中に混合できませんので、すべてを文字列にするか、混合を許すボックス(Box)にしなければいけません。結構ハードル高いです。

1から7まででイメージしてみます。

```
   1;2;'Fizz';4;'Buzz';6;7
+-+-+----+-+----+-+-+
|1|2|Fizz|4|Buzz|6|7|
+-+-+----+-+----+-+-+
   '1';'2';'Fizz';'4';'Buzz';'6';'7'
+-+-+----+-+----+-+-+
|1|2|Fizz|4|Buzz|6|7|
+-+-+----+-+----+-+-+
    7 4$'   1   2Fizz   4Buzz   6   7'
   1
   2
Fizz
   4
Buzz
   6
   7
```

最終形のイメージが固まらないとあとで苦労しますよねー。

1から100までの数字をボックスに入れて、101番目と102番目103番目にそれぞれ'Fizz','Buzz','FizzBuzz'をいれたもの、つまり103個の箱からなるボックスから配列で100個取り出すというので行けるかな？

```
   ]a=:<"0(1+i.7)
+-+-+-+-+-+-+-+
|1|2|3|4|5|6|7|
+-+-+-+-+-+-+-+
   ]a=:a,'Fizz';'Buzz';'FizzBuzz'
+-+-+-+-+-+-+-+----+----+--------+
|1|2|3|4|5|6|7|Fizz|Buzz|FizzBuzz|
+-+-+-+-+-+-+-+----+----+--------+
   (1 2 8 4 9 6 10 - 1) { a
+-+-+----+-+----+-+--------+
|1|2|Fizz|4|Buzz|6|FizzBuzz|
+-+-+----+-+----+-+--------+
```

これは行けそうですね。となると、次は

```
   1 2 101 4 102 6 7 8 101 102 ...
```

という配列を作ることになります。

この辺は力ずくでなんとかなりそう。

```
   ]a=:<"0 (1+i.100)
+-+-+-+-+-+-+-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31|32|33|34|35|36|37|
+-+-+-+-+-+-+-+-+-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+ ...
   boxall=:a,'Fizz';'Buzz';'FizzBuzz'
   11 10$boxall
+----+----+--------+--+--+--+--+--+--+---+
|1   |2   |3       |4 |5 |6 |7 |8 |9 |10 |
+----+----+--------+--+--+--+--+--+--+---+
|11  |12  |13      |14|15|16|17|18|19|20 |
+----+----+--------+--+--+--+--+--+--+---+
|21  |22  |23      |24|25|26|27|28|29|30 |
+----+----+--------+--+--+--+--+--+--+---+
|31  |32  |33      |34|35|36|37|38|39|40 |
```

```
+----+----+--------+--+--+--+--+--+--+---+
|41  |42  |43      |44|45|46|47|48|49|50 |
+----+----+--------+--+--+--+--+--+--+---+
|51  |52  |53      |54|55|56|57|58|59|60 |
+----+----+--------+--+--+--+--+--+--+---+
|61  |62  |63      |64|65|66|67|68|69|70 |
+----+----+--------+--+--+--+--+--+--+---+
|71  |72  |73      |74|75|76|77|78|79|80 |
+----+----+--------+--+--+--+--+--+--+---+
|81  |82  |83      |84|85|86|87|88|89|90 |
+----+----+--------+--+--+--+--+--+--+---+
|91  |92  |93      |94|95|96|97|98|99|100|
+----+----+--------+--+--+--+--+--+--+---+
|Fizz|Buzz|FizzBuzz|1 |2 |3 |4 |5 |6 |7  |
+----+----+--------+--+--+--+--+--+--+---+
  0 1 2 3 100 101 102 { boxall
+-+-+-+-+----+----+--------+
|1|2|3|4|Fizz|Buzz|FizzBuzz|
+-+-+-+-+----+----+--------+
  ]a=:1+i.100
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 ...
  3 | a
1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 ...
  0 E. 3|a
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 ...
  (0 E. 3|a) # a
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99 ...
  ]fizz=:101 (((0 E. 3|a) # a) - 1)} a
1 2 101 4 5 101 7 8 101 10 11 101 13 14 101 16 17 101 19 20 101 22 23 101 25 26 101 28 29 101 31 32 101 ...
  ]buzz=:102 (((0 E. 5|a) # a) - 1) } fizz
1 2 101 4 102 101 7 8 101 102 11 101 13 14 102 16 17 101 19 102 101 22 23 101 102 26 101 28 29 102 31 32 101 ...
  ]fizzbuzz=:103 (((0 E. 15|a) #a) - 1) } buzz
1 2 101 4 102 101 7 8 101 102 11 101 13 14 103 16 17 101 19 102 101 22 23 101 102 26 101 28 29 103 31 32 101 ...
  10 10$result=:(fizzbuzz - 1) { boxall
+----+----+----+----+--------+----+----+----+----+--------+
|1   |2   |Fizz|4   |Buzz    |Fizz|7   |8   |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|11  |Fizz|13  |14  |FizzBuzz|16  |17  |Fizz|19  |Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|Fizz|22  |23  |Fizz|Buzz    |26  |Fizz|28  |29  |FizzBuzz|
+----+----+----+----+--------+----+----+----+----+--------+
|31  |32  |Fizz|34  |Buzz    |Fizz|37  |38  |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|41  |Fizz|43  |44  |FizzBuzz|46  |47  |Fizz|49  |Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|Fizz|52  |53  |Fizz|Buzz    |56  |Fizz|58  |59  |FizzBuzz|
+----+----+----+----+--------+----+----+----+----+--------+
|61  |62  |Fizz|64  |Buzz    |Fizz|67  |68  |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|71  |Fizz|73  |74  |FizzBuzz|76  |77  |Fizz|79  |Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|Fizz|82  |83  |Fizz|Buzz    |86  |Fizz|88  |89  |FizzBuzz|
+----+----+----+----+--------+----+----+----+----+--------+
|91  |92  |Fizz|94  |Buzz    |Fizz|97  |98  |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
```

合っていると思うけど、いかがでしょうか？

コメントを書く

トラックバック － http://d.hatena.ne.jp/niming538/20090309

〈前の日 | 次の日〉

niming538の日記

niming538のブックマーク
TED.comは、最高レベルの英語学習コンテンツでは？ - livedoor Blog（ブログ）
最新情報を知りたい人が抑えておくべきたった一つのサイト: 304 Not Modified
実開発で分かったGoogle App Engine for Javaの“すごさ”：ITpro
無題のドキュメント クラブミュージックの代表的なジャンルと曲
【連載】セカイ系ウェブツール考 (72) 文章を効率的に書くために － 執筆作業サポートツール | ネット | マイコミジャーナル

日記の検索
検索
詳細 一覧

最新タイトル
日本人の起源とY染色体
APL/J言語：数え上げ
APL/J言語：樹形図
APL/J言語：コイントス
APL/J言語:円周率の計算（マチンの公式）
APL/J言語：連立一次方程式
APL/J言語：オイラーの公式
APL/J言語：1001の素因数分解
APL/J言語：スクリーンへのアウトプット(質問)
中国語基本音節表

カウンター
60509

リンク集
niming538's fotolife

カレンダー
<< 2009/03 >>
日 月 火 水 木 金 土
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

〈前の日 | 次の日〉

2009-03-10 APL/J言語：FizzBuzz問題の解説と感想

昨日の記事でJ言語でFizzBuzzをやりました。はっきり言って難しかった。配列で考えること自体は慣れの問題だと思うのですが、ボックスがやっかい。じつはずいぶん試行錯誤をしました。

```
   1;2;'Fizz';4;'Buzz';6;7
 +-+-+----+-+----+-+-+
 |1|2|Fizz|4|Buzz|6|7|
 +-+-+----+-+----+-+-+
```

というようにセミコロンでつないでいけばボックスになるのですが、数列の後ろに'Fizz','Buzz','FizzBuzz'の箱を3つ続けるのがなかなかできませんでした。

```
   ]a=:<"0(1+i.7)
 +-+-+-+-+-+-+-+
 |1|2|3|4|5|6|7|
 +-+-+-+-+-+-+-+
   ]a=:a,'Fizz';'Buzz';'FizzBuzz'
 +-+-+-+-+-+-+-+----+----+--------+
 |1|2|3|4|5|6|7|Fizz|Buzz|FizzBuzz|
 +-+-+-+-+-+-+-+----+----+--------+
```

というように、一旦aに代入してからカンマでつなげましたが、これがふつうのやりかたかどうかはしりません。（追記：(<"0(1+i.7)),('Fizz';'Buzz';'FizzBuzz')で出来ました。）

取り出すのは{(左ブレイス)は知っていた。

次に配列の作成ですが、3で割ったあまりは|(パイプ、たてぼう)で、それが0(ゼロ)のときにフラグを立てるというのが知りませんでした。

```
   0 E. 3|a
 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 ...
```

#(シャープ)は二項動詞としてはコピーですので、配列aと組み合わせると、位置が取り出せます。ホントは位置ではなく、その場所にある数字です。

```
   ]fizz=:101 (((0 E. 3|a) # a) - 1)} a
   ]buzz=:102 (((0 E. 5|a) # a) - 1) } fizz
   ]fizzbuzz=:103 (((0 E. 15|a) #a) - 1) } buzz
   10 10$result=:(fizzbuzz - 1) { boxall
```

}(右ブレイス)は不思議な動詞で、左側に引数を2つ取って、ある数字を配列の指定した場所に代入します。

最後にボックスから{(左ブレイス)で取り出して出来上がり。

宿題：ボックスについてもっと勉強する。(0 E. 3|a)はこれがふつうかわからないので覚えておく。数値のボックスの後ろに文字列のボックスをつなげたのはよいアイデアだが、課題の「数が変化したときに作り直す必要がある。つまりあまりかっこよい解決方法ではない。これも覚えておく。今回の解法をひとつの動詞(関数)にできるか考える。最後の4行を配列とボックスを使わずすべてボックスでできないか。数字の位置と数字を合わせるために1を引いたりしているがこの辺をもっとエレガントにできないか。

以上

コメントを書く
トラックバック － http://d.hatena.ne.jp/niming538/20090310

〈前の日 | 次の日〉

niming538の日記

<前の日 | 次の日>

2009-03-12 APL/J言語でFizzBuzz(つづき)

J言語でFizzBuzz(http://recompile.net/2007/05/jfizzbuzz.html)というところに、satzeさんが

例えばこんなのどうでしょうか？

```
' ' ((('|' E. B) # i.#B) } B=.}.}:1{'':(<'FizzBuzz') (<:(0 E. 15 | A) # A) } (<'Buzz') (<:(0 E. 5 | A) # A) } (<'Fizz') (<:(0 E. 3 | A) # A) } <'0 A=.>:i.100
```

（1行）

もしくは，

```
FizzBuzz=: 3 : 0
' ' ((('|' E. B) # i.#B) } B=.}.}:1{'':(<'FizzBuzz') (<:(0 E. 15 | A) # A) } (<'Buzz') (<:(0 E. 5 | A) # A) } (<'Fizz') (<:(0 E. 3 | A) # A) } <'0 A=.>:i.y.
)
```

として，

```
FizzBuzz 100
```

と書かれています。これを解析します。J言語ではワンライナーは後ろから読んでいきます。

```
    i.100       NB.これはご存知0..99の配列作成
    >:i.100   NB.'>:'はデクリメンとで各要素に1を足します。かっこいいですね。
    <"0           NB.これで要素別にボックス作成。"0がないとひとつのボックスになる。
```

ここまでをまとめると以下のようになります。

```
    <"0 A=.>:i.100
  +-+-+-+-+-+-+-+-+-+--+--+--+--+--+--+
  |1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|
  +-+-+-+-+-+-+-+-+-+--+--+--+--+--+--+ ...
```

次が大変参考になったところです。

```
    (<'Fizz') (<:(0 E. 3 | A) # A) }</pp>
```

}(みぎブレイス)は左辺の引数を二つとって、右辺の配列について左辺の二つ目で示す場所に左辺の一つ目の引数を代入します。この場合はAを3で割った結果ゼロになる場所にある数字の配列をおのおのデクリメントした場所に'Fizz'というボックスを入れる、という意味になる。同じように'Buzz'も、'FizzBuzz'も処理します。ここまでで、おおかた私の達成したところまでと同様なことをやっている。

次に、冒頭の部分でボックスから外に出しているようです。

```
    ' ' ((('|' E. B) # i.#B) } B=.}.}:1{'':
```

結果は一つの巨大な文字列になります。冒頭に$を置いてみたら412という数字が返って来ました。

```
    1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz ...
```

'':(クオートコロン)はボックスをボックス描画の線も中の文字列も数字もすべて文字列にしてしまう強力な動詞です。

```
    ]box=:1;2;'Fizz'
  +-+-+----+
  |1|2|Fizz|
  +-+-+----+
    $box
  3
    $'':box
  3 10
    1{'':box
  |1|2|Fizz|
```

{(左ブレイス)は左辺で指定した要素を右辺から取り出しますので、この例ではボックスの切れ目を示すたてぼうがついた状態の行がとりだせました。次に}.}:で前後の一文字を消したものをBに代入します。

```
    ]B=:}.}:1{'':box
 1|2|Fizz
```

この結果のBについて、たてぼうの位置を調べてそこにスペースを代入していくのが残りです。

```
    ' ' (('|' E. B) # i.#B) } B
 1 2 Fizz
```

うむ。すばらしい。

FizzBuzzという動詞を定義している例についてひと言。

```
   FizzBuzz 100
 1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 ...
```

となるような単項動詞の定義方法が最近変わっていて、上記のままではエラーになります。定義内の変数yとかに以前はドットをつけていましたが、現在はドットがあるとえらーになります。したがって、正しくは最後のドットを取って、以下のような定義になります。

```
FizzBuzz=: 3 : 0
 ' ' (('|' E. B) # i.#B) } B=.}.}:1{":(<'FizzBuzz') (<:(0 E. 15 | A) # A) } (<'Buzz') (<:(0 E. 5 | A) # A) } (<'Fizz') (<:(
)
```

以上

コメントを書く

トラックバック - http://d.hatena.ne.jp/niming538/20090312

〈前の日 | 次の日〉

niming538のブックマーク
TED.comは、最高レベルの英語学習コンテンツでは？ - livedoor Blog(ブログ)
最新情報を知りたい人が抑えておくべきたった一つのサイト: 304 Not Modified
実開発で分かったGoogle App Engine for Javaの“すごさ”:ITpro
無題のドキュメント クラブミュージックの代表的なジャンルと曲
【連載】セカイ系ウェブツール考 (72) 文章を効率的に書くために - 執筆作業サポートツール | ネット | マイコミジャーナル

日記の検索
検索
詳細 一覧

最新タイトル
日本人の起源とY染色体
APL/J言語:数え上げ
APL/J言語:樹形図
APL/J言語:コイントス
APL/J言語:円周率の計算(マチンの公式)
APL/J言語:連立一次方程式
APL/J言語:オイラーの公式
APL/J言語:1001の素因数分解
APL/J言語:スクリーンへのアウトプット(質問)
中国語基本音節表

カウンター
60511

リンク集
niming538's fotolife

カレンダー
<< 2009/03 >>
日 月 火 水 木 金 土
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

niming538の日記

niming538のブックマーク
TED.comは、最高レベルの英語学習コンテンツでは？ - livedoor Blog(ブログ)
最新情報を知りたい人が抑えておくべきたった一つのサイト: 304 Not Modified
実開発で分かったGoogle App Engine for Javaの“すごさ”:ITpro
無題のドキュメント クラブミュージックの代表的なジャンルと曲
【連載】セカイ系ウェブツール考 (72) 文章を効率的に書くために － 執筆作業サポートツール | ネット | マイコミジャーナル

日記の検索
検索
詳細 一覧

最新タイトル
日本人の起源とY染色体
APL/J言語：数え上げ
APL/J言語：樹形図
APL/J言語：コイントス
APL/J言語:円周率の計算(マチンの公式)
APL/J言語：連立一次方程式
APL/J言語：オイラーの公式
APL/J言語：1001の素因数分解
APL/J言語：スクリーンへのアウトプット(質問)
中国語基本音節表

カウンター
60512

リンク集
niming538's fotolife

カレンダー
<< 2009/03 >>
日 月 火 水 木 金 土
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

<前の日 | 次の日>

2009-03-13 APL/J言語：FizzBuzz完成品

わたしとしての完成品を書いておきます。

ボックスから出す必要はないと思うのでその分短くなっています。

結果の表現は10 x 10の表にします。

Windows版のJ602での結果です。

```
FizzBuzz=: 3 : 0
(<'FizzBuzz') (<:(0 E. 15 | A) # A) } (<'Buzz') (<:(0 E. 5 | A) # A) } (<'Fizz') (<:(0 E. 3 | A) # A) } <"0 A=.>:i.y
)

   10 10$FizzBuzz 100
+----+----+----+----+--------+----+----+----+----+--------+
|1   |2   |Fizz|4   |Buzz    |Fizz|7   |8   |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|11  |Fizz|13  |14  |FizzBuzz|16  |17  |Fizz|19  |Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|Fizz|22  |23  |Fizz|Buzz    |26  |Fizz|28  |29  |FizzBuzz|
+----+----+----+----+--------+----+----+----+----+--------+
|31  |32  |Fizz|34  |Buzz    |Fizz|37  |38  |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|41  |Fizz|43  |44  |FizzBuzz|46  |47  |Fizz|49  |Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|Fizz|52  |53  |Fizz|Buzz    |56  |Fizz|58  |59  |FizzBuzz|
+----+----+----+----+--------+----+----+----+----+--------+
|61  |62  |Fizz|64  |Buzz    |Fizz|67  |68  |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|71  |Fizz|73  |74  |FizzBuzz|76  |77  |Fizz|79  |Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
|Fizz|82  |83  |Fizz|Buzz    |86  |Fizz|88  |89  |FizzBuzz|
+----+----+----+----+--------+----+----+----+----+--------+
|91  |92  |Fizz|94  |Buzz    |Fizz|97  |98  |Fizz|Buzz    |
+----+----+----+----+--------+----+----+----+----+--------+
```

少しわかりやすく定義すると下記のような感じかな？

```
   FizzBuzz =: 3 : 0
   FizzBuzz =: 3 : 0
 array =. >:i.y  NB.1ベースの配列を作成する
 original =. <"0 array   NB.ボックスにする
 fizz =. (<'Fizz') (<:(0 E. 3 | array) # array) } original   NB.3で割り切れた数を'Fizz'とする</pp>
 buzz =. (<'Buzz') (<:(0 E. 5 | array) # array) } fizz   NB.5で割り切れた数を'Buzz'とする</pp>
 fizzbuzz =. (<'FizzBuzz') (<:(0 E. 15 | array) # array) } buzz   NB.15で割り切れた数を'FizzBuzz'とする</pp>
 )
```

以上

コメントを書く

トラックバック - http://d.hatena.ne.jp/niming538/20090313

- 主に言語とシステム開発に関して - プログラミング言語「APL」の入...

〈前の日 | 次の日〉