応用プログラム言語II / 演習II #15

Enjoy Programming!

情報システム学科
坂本政祐
sakapon@sit.ac.jp

はじめに

- この講義では，モダン（今風）な言語を使えばプログラミングは決して面倒なものではない，だから楽しんでプログラミングしましょうという視点を柱に，様々なトピックについて学んできました。
- ですので，特定の言語をマスターして欲しいというよりは，言語は好きなもので良いから，
  - モダンな言語なら楽ができるデータ構造や，
  - モダンなプログラミングパラダイム（オブジェクト指向など）や，
  - モダンなライブラリがあるよ，

  という流れの内容にしてきました。

はじめに

- というわけで今回は，そういった視点において語り足りなかった部分（特定のトピックに入れづらかった部分）や，言語の種類について語ろうと思います。

様々なプログラミング言語

言語の歴史

- そもそもプログラミング言語はどのように離合集散してきたのか。
- http://www.levenez.com/lang/ に良い年表がある。これで色々なことがわかる。主なものは:
  - 最初の言語は FORTRAN(1954)。
  - 関数型言語の雄 Lisp は1958年には出ていた。
  - 8ビットコンピュータで主流だったBASICも1964年には出ていた。
  - Cは1971年。（その原型のBCPLやBはもっと前）
  - オブジェクト指向の始祖 Smalltalk も実は1971年。
  - 1977年にApple II。1979年にPC-8001。

言語の歴史

- C++は1980年（年表では1983年だがStroustrupが提案した元々のがC with Classesなので）。意外と古い。
- 1982年にPC-9801（16ビット）
- Objective-C は1983年。実に20数年間マイナーだった言語が，iOS用開発言語として突然メジャー言語になった。今では世界3位（後述）。
  - そもそもなぜiOSはObjective-C?: ジョブズがAppleを追い出されて作ったNeXTのOS，NeXTSTEP用の開発言語がObective-C だったため。
- 1985年頃比較的多くの離合集散がある。コンパイラの技術も加速度的に伸びている時代で，パラダイムそのものも多く提案される。

言語の歴史

- 1985年に80386DX（32ビット）。
- LLの元祖Perlは1987年。Pythonは1991年，Rubyは1993年。
- Javaは1995年。ちなみにSun MicrosystemsはOracleに買収されてしまった。
- JavaScriptは1995年。PHPも1995年。Web黎明期。
- 1990年代後半にまた離合集散が多くなっているのは，パーソナルなコンピュータが誰でも手に入るようになったとか，Webとか，ワークステーションも色々なアーキテクチャが百花繚乱だったこととかで非常に活気のある時代だったのかも。

言語の歴史

- 2000年代後半は割と落ち着いているように見えるが，以下のような新しいトピックもたくさんある。
  - SystemCなど，ハードウェア記述言語とプログラミング言語の境目がなくなる。
  - 関数型言語が見直される。
  - Haskell, Lua, Scala, Clojure, Dなどの新世代言語が登場。
  - Google が Dart 言語や Go 言語をリリース。
- また，
  - GPUで計算を行うGPGPU用の技術であるCUDAやOpenCL
  - メニーコア用並列プログラミングはどうしたら良いか

  など，ハードウェアの進化に伴うプログラミングの変化も出てきている。

言語の人気度

applied-progII #15

- このように色々な言語がある中で，では世界のトレンドはどうなのか?
- サーチエンジンの検索結果からプログラミング言語の人気度を評価する TIOBE Index が便利。
  - http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

9

言語の人気度

applied-progII #15

- TOP20の表

Position Dec 2012	Position Dec 2011	Delta in Position	Programming Language	Ratings Dec 2012	Delta Dec 2011	Status
1	2	↑	C	18.696%	+1.64%	A
2	1	↓	Java	17.567%	+0.01%	A
3	5	↑↑	Objective-C	11.116%	+4.31%	A
4	3	↓	C++	9.203%	+0.95%	A
5	4	↓	C#	5.547%	-2.66%	A
6	6	=	PHP	5.541%	-0.46%	A
7	7	=	(Visual) Basic	5.174%	+0.42%	A
8	8	=	Python	3.848%	+0.36%	A
9	9	=	Perl	2.174%	-0.30%	A
10	11	↑	Ruby	1.728%	+0.23%	A
11	10	↓	JavaScript	1.321%	-0.88%	A
12	12	=	Delphi/Object Pascal	0.977%	-0.27%	A
13	13	=	Lisp	0.949%	-0.23%	A
14	16	↑↑	Pascal	0.894%	+0.16%	A
15	35	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	0.889%	+0.53%	A
16	17	↑	Ada	0.648%	+0.02%	B
17	22	↑↑↑↑↑	MATLAB	0.608%	+0.07%	B
18	21	↑↑↑	Lua	0.601%	+0.05%	A-
19	19	=	Assembly	0.580%	+0.02%	B
20	14	↓↓↓↓↓↓	PL/SQL	0.574%	-0.23%	B

10

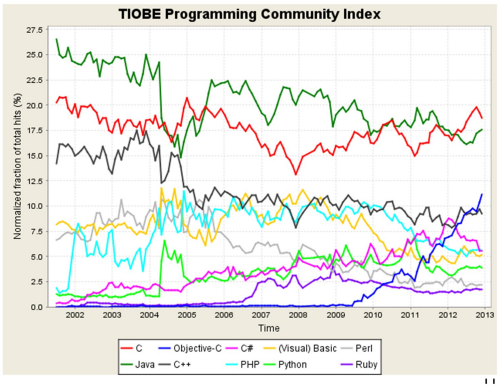
言語の人気度

applied-progII #15

- 長期傾向のグラフ

Long term trends

The long term trends for the top 10 programming languages can be found in the line diagram below.



11

言語の人気度

applied-progII #15

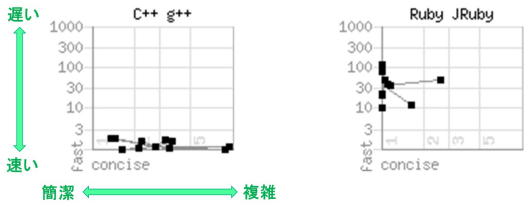
- これらの図表の他に以下のようなデータも公開されている。
  - 超長期的人気傾向(1987, 1997, 2007, 2012)
  - Programming Language Hall of Fame(2003～2011)
    - 2003: C++
    - 2004: PHP
    - 2005: Java
    - 2006: Ruby
    - 2007: Python
    - 2008: C
    - 2009: Go
    - 2010: Python
    - 2011: Objective-C
  - カテゴリー別人気分布
    - OOPL 58.5%，手続き型36.9%，関数型3.2%
    - 静的型付け71.4%，動的型付け28.6%

12

言語の実行速度比較

applied-progII #15

- また，各言語を速さと簡潔さの両視点から比較したのが以下のページにあり。
  - http://shootout.alioth.debian.org/u32/code-used-time-used-shapes.php



13

言語と日本

applied-progII #15

- 言語と日本
  - 言語はやはり欧米発が多い。
  - 日本発の言語:
    - ぴゅう太(1982)のG-BASIC `10 カケ Hello, World`
    - Ruby
    - ひまわり/なでしこ `「Hello, World」と表示。`
    - Hot Soup Processor `mes "Hello, World"`
    - Xtal `println("Hello, World!");`

※写真はWikipediaより引用，以下同

14

言語と日本

applied-progII #15

- 日本で最初のハッカーと呼ばれている方:
  - 和田英一 東京大学名誉教授
    - パラメトロンコンピュータのプログラミング
    - 和田研フォント
    - Happy Hacking Keyboard

15

言語の数＝コンパイラ(or インタプリタ)の数

applied-progII #15

- さて，それだけ沢山言語があれば，その分，コンパイラ(あるいはインタプリタ)の数がある。
- だとすると，各言語をそれぞれ試そうと思っても，それぞれのコンパイラをそれぞれインストールしなければいけない。
- せっかく「ちょっと試してみるか」と知的好奇心を持っても，そのインストールが面倒くさい。

16

オンライン実行環境

applied-progII #15

- そんなときに，いまどきはWeb上で言語を試すことができる環境がいくつかある。
  - Ideone
    - http://ideone.com
    - 対応言語: Ada, Assembler, awk, Bash, bc, Brainfuck, C, C#, C++, CLIPS, Clojure, COBOL, Common Lisp, D, Erlang, F#, Factor, Falcon, Forth, Fortran, Go, Groovy, Haskell, Icon, Intercal, Java, Java7, JavaScript, Lua, Nemerle, Nice, Nimrod, Node.js, Objective-C, Ocaml, Oz, PARI/GP, Pascal, Perl, PHP, Pike, Prolog, Python, R, Ruby, Scala, Scheme, Smalltalk, SQL, Tcl, Text, Unlambda, VB.NET, Whitespace

17

オンライン実行環境

applied-progII #15

- lleval
  - http://colabv6.dan.co.jp/lleval.html
  - 対応言語: awk, BASIC, Brainfuck, C, Emacs Lisp, Grass, Haskell, io, JavaScript, Lazy K, Common Lisp, Lua, m4, OCaml, Perl, PHP, PostScript, Python, Ruby, Scheme, Tcl
- codepad
  - http://codepad.org
  - 対応言語: C, C++, D, Haskell, Lua, OCaml, PHP, Perl, Plain Text, Python, Ruby Scheme, Tcl
- この他，言語別にまとめてくれてあるのが http://joel.franusic.com/w/page/26128430/Online-REPs-and-REPLs にあり。

18

たったひとつではない，冴えたやりかた

19

TMTOWTDI

applied-progII #15

- Perl言語のコミュニティで言われるスローガンとして，TMTOWTDI（ティムトゥディ）というのがある。
  - There's More Than One Way To Do It. の略。
  - 「やり方はひとつじゃない」ということ。
  - つまり，あることをプログラムで書きたいときに，それを実現する方法は，難しく書いたり，簡単に書いたり，ライブラリを使ったり使わなかったり，いろいろに書けるよ，Perlは特にそれが顕著だよ，ということ。
  - Perlに限らず，LL(Lightweight Language)は割とそういう性質を多く持っている。

20

TMTOWTDI

applied-progII #15

- 例えばRubyで，「Stringオブジェクトを1文字ずつ分解してArrayオブジェクトにする（"foo"から["f", "o", "o"]を得たい）」というのを書くと，以下のように様々な方法がある。

```
ary = []
str = "foo"
str.length.times do |i|
  ary.push str[i].chr
end
```

```
ary = []
str = "foo"
(0...str.size).each do |i|
  /.{#{i}}(.)/ =~ str
  ary[i] = $1
end
```

```
ary = []
"foo".each_byte{|c| ary << c.chr}
```

```
"foo".split(//)
```

```
"foo".scan(/./)
```

21

コードゴルフ

applied-progII #15

- 今の中では，最後の2つが15バイトで書けているので一番短い。
- このように，あるお題を達成するための一番短いソースを書けた人が勝ち，というのを楽しんでいる人たちがいる。こういう遊びをコードゴルフと呼んでいる。
- コードゴルファー達の集まるゴルフ場:
  - http://codegolf.com
  - http://golf.shinh.org

22

コードゴルフ

applied-progII #15

- Rubyist Magazine で過去に出題された「標準入力から複数の行を受け取って，同じ内容の行は捨てつつ標準出力に出力する」というゴルフ問題。
  （http://jp.rubyist.net/magazine/?0026-RubiMaGolf）
  - 普通に書くとこんな感じ。

```
dict = {}
while line = STDIN.gets
  if !dict[line]
    puts line
  end
  dict[line] = 1
end
```

  - ゴルファーにかかるとこう。

```
puts$*|[*$<]
```

!?

23

やり方はひとつじゃない

applied-progII #15

- 前スライドのコードは極端すぎるが，いろんな書き方ができるというのは，
  - より読みやすい
  - より速い
  - より短い
  - より「その言語らしい」

  など様々な視点で自分のコードやその言語の特徴を見つめ直すことにつながる，という利点がある。

24

まとめ

- というわけで半年間，プログラミングにまつわる色々なトピックについて学んできました。
- 題材は様々でしたが，一貫していたのは，プログラミングは面倒臭いものでは決してなく，モダンな言語や開発環境であれば，楽に，直感的に，楽しみながら書けるということを知って欲しいということでした。

まとめ

- プログラミングはやはり一種のものづくりです。コードがうまく動いたときの感動はひとしおです。
- それがあるからこそ，ずっと仕事としても趣味としても続けていけるものなのだと思います。
- プログラミングを楽しむマインドを是非これからもつちかって欲しいと思います。
- Enjoy Programming!

ハックの基本は「面倒感度を研ぎ澄ます」こと――大橋悦夫

最終レポート課題

最終レポート課題（再掲）

- 最終レポート課題:
  - この講義でこれまでに取り扱った内容に関係することなら何でも良いので，何か1つプログラムを作ってください。
    - 規模は問いません。意味のあるプログラムになっているなら，ワンライナー（1行プログラム）でも構わないということです。
    - オリジナリティを重視します。
    - #08「ライブラリで楽しよう」という回がありましたが，それに関係するという意味では，#08で紹介した以外のライブラリを使ったプログラムでも構いません。もちろん，紹介したライブラリを使ったプログラムでも構いません。

最終レポート課題（再掲）

- 最終レポート課題（続き）:
  - 提出〆切: 2013年1月28日（月）17:00まで。
  - 提出方法: 今回はE-mailに添付して提出して下さい。
    - To: sakapon@sit.ac.jp
    - Subject: 応用プログラム言語IIレポート
    - Visual Studio で作ったプログラムは，プロジェクトのフォルダー式をzip圧縮して添付してください。ただし，Gmail 経由だと exe ファイルを含んでいると送れない場合があります。その場合は，他のアカウントから送るか，もしくは lzh 圧縮でも構いません。
    - それ以外のシステムや言語で作ったプログラムは，ソースコードのみで構いません。（例: Ruby, Brainfuck）

最終レポート課題（再掲）

- 最終レポート課題（続き）:
  - 自作したものであるなら，既発表のものでも構いません（例えば，自身のブログで過去に発表したソースや，過去にネットで自分で公開した自作フリーソフトのソース等）。この講義の課題として作ったものは駄目です。
  - （追加）この講義で作ったソースに対しての追加は無しということで。

今週の落穂拾い

- 途中の回でやったアンケートには結局ほとんど応えられませんでした。ごめんなさい。
- ただ，私の思うに，自力でも勉強ができる範囲のことはあまりこの講義では扱わなくても良いと思うのです。
- 自分では始めにくい，とっつきにくい，何から手をつけていいかわからない，ググってもあまりピンと来ない，というようなトピックこそがこういう講義には適していると考えます。

今週の落穂拾い

- そういう意味でもっと時間があればやりたかったネタとしては:
  - VBAかC#でOfficeの自動操縦。
  - Androidプログラミング。（iOSでも良いのですが，環境を揃えるのに金がかかりすぎます…）
  - 関数型プログラミング。
  - ネットワークに関係するプログラミング。

  などです。
- 興味があったら是非自分でも調べたりあるいは坂本に聞いたりしてみて下さい。
- Once more, Enjoy Programming!