ゴルフ出題するときに考えていたこと

My thoughts on creating code golf challenges.

pixiv Inc.
USAMI Kenta

pixiv

2025-04-23 #phpstudy
第175回 PHP勉強会@東京

お前誰よ

- うさみけんた（@tadsan）/ **Zonu.EXE** / **にゃんだーすわん**
- ピクシブ株式会社 pixiv事業本部 Webエンジニアリングチーム PHPer
  - 2012年末から現職、APIとかCIとかいろいろなところを見つめてきました
- Emacs PHP Modeを開発しています（2017年-）
- プログラミング言語にちょっとこだわりのある素人（spcamp2010）

さて

pixiv

コードゴルフを
ご存じですか？

コードゴルフはe-スポーツ

- 普通のゴルフ：
  - ボールを叩いてホールまで運び、1打でも少ない方が勝ち

コードゴルフはe-スポーツ

- 普通のゴルフ：
  - ボールを叩いてホールまで運び、1打でも少ない方が勝ち
- コードゴルフ：
  - キーを叩いて仕様を満たすコードを書き、1文字でも少ない方が勝ち

コードゴルフはe-スポーツ

- 普通のゴルフ:
  - ボールを叩いてホールまで運び、1打でも少ない方が勝ち
- コードゴルフ:
  - キーを叩いて仕様を満たすコードを書き、1文字でも少ない方が勝ち
- Vimゴルフ:
  - キーを叩いてテキストをゴールまで編集し、手順が少ない方が勝ち

← **Post**

にゃんだーすわん
@tadsan

PHP大好きなみなさん、これは解きましたか returntrue.win

Translate post

1:32 AM · Feb 16, 2018 from Shibuya-ku, Tokyo

View post engagements

1

1

pixiv

@ProjectICKX (ICKX Project) in Project ICKX

十分に時間経ったし、そろそろreturntrue.winのネタバレしてもいいか

PHP PHP7

最終更新日 2019年05月16日 投稿日 2018年03月05日

return true to winとは

2018/02/15にphpusers-ja.slack.comでバカウケした、「**関数の引数に任意の値を入れ、関数の実行結果がbool trueになれば勝ち**」というコードパズルです。
Level 1からLevel 11までの11問が配備されています。
開催会場はこちら => https://returntrue.win/

pixiv

2020年

1. PHPer Code Golf by pixiv

pixivさまご提供の企画、PHPer Code Golf by pixiv の中にPHPerトークンが隠されています。

PHPer Code Golf by pixiv は「与えられた出力をするPHPコードを書く」「書いたコードが一番短いプレイヤーが勝ち」というルールです。短かい方が勝ち、というあたりがゴルフぽいのでコードゴルフ、と呼ばれたりします。

さて、そのコードゴルフ、コードを書いたら専用サイトにそのコードを入力する訳ですが、そのサイトにPHPerトークンが隠されています。

PHPer Code Golf by pixiv は単体でも成績優秀者に商品がありますが、さらにPHPerトークンもゲットできてしまうというすばらしい企画です。是非ご参加ください！

2024年 (昨年)

pixiv

2025年 (今回)

2025年3月16日 ▪ 未分類

コードバトルのオンライン予選を開始しました

投稿者: nsfisis

はじめに

PHPerKaigi 2025 では、3/21 の day0 に track A で「PHPer コードバトル」という対戦型企画を実施予定です。

このたびオンライン予選のエントリーを開始しましたので、企画趣旨や参加方法について紹介しようと思います。

PHPer コードバトルとは

PHPer コードバトルは、指示された動作をする PHP コードをより短く書けた方が勝ちという 1 対 1 の対戦コンテンツです。3/21（金）day0 に track A で、予選を勝ち上がったプレイヤー6名で、トーナメント形式でバトルを実施します。

pixiv

コードゴルフ作問体制

- 2020年:「PHPer Code Golf by pixiv」
  - tadsan(私)が単独でゴルフ場建設・作問
- 2024年:「コードゴルフ企画」
  - いまむらさん(nsfisis)が単独でゴルフ場建設・作問
- 2025年:「PHPerコードバトル」
  - いまむらさんがゴルフ場建設・tadsan(私)が作問を担当

ゴルフ場 (playground)

- 参加者が自由に入力したプログラムが実行できる(≒任意コード実行)
- 悪意を持った利用者が悪用できないようにする必要がある
  - ファイル書き込みでサーバのディスクを埋めて利用不能にするとか…
  - 無限ループでプログラムが永遠に終わらないとか…
  - 外部サービス攻撃の踏み台にされるとか、ビットコイン採掘されるとか…
- いかに安全に実行できるようにするかは創意工夫の余地がある

http://ja.doukaku.org/ Go

117 captures

1 Jul 2007 – 25 Jun 2023

2

どう書く?org β

「どう書く?org」へようこそ！このサイトは出されたお題をいかに解くか競い合う、プログラマのためのコロシアムです。投稿を試してみたい方はテスト、とりあえず眺めてみたい方は言語の一覧 がおすすめです。

おしらせ

どう書く?orgはGoogle App Engineへ移行しました。

最新のトピック10件

- 数字混じり文字列ソート
- ネットワークアドレスを求める
- 階層的なキーの連想配列化
- 階層的なキーの連想配列化
- 文字列で+を表示する
- 年賀はがきの当せん番号
- 箱詰めパズルの判定
- 回文日付
- 関数やメソッドのソースの平均行数
- コレクションの実装

[もっと...]

pixiv

出題

今回の要件

- 標準入出力だけを使う・デフォルト有効化されていないモジュールは不可
- 競技プログラミング文脈に頼りすぎない
- 模範解答を開示する
  - 参加者は問題解答だけでなく、文字数を縮めることに集中できる
- （非機能要件）コンテンツとして観戦して楽しい・わかりやすくする

予選問題

- コードゴルフに不慣れな人も参戦するので、コードの難易度としては
  プログラミングの初年度授業の発展課題くらいを「意識」
  - ただし、ワンパターンにならないように難易度にこだわりすぎない

pixiv

作問方法

PHP Playground

PHP Playground let you to execute basic PHP code in real time using WebAssembly technology.

About PHP Playground?

< Request and Report

Donate

Version: 8.4

```php
<?php

if (!defined('STDIN')) {
    define('STDIN', fopen('php://memory', 'rw'));
    fwrite(STDIN, <<<'EOT'
9 9

EOT);
    rewind(STDIN);
}

fscanf(STDIN, "%d %d", $c, $r);
$w = strlen($c * $r);
for(;@++$i<=$r;)vprintf(str_repeat(" %{$w}d",$c)."\n",array_map(fn($j)=
```

```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81
```


pixiv

php-play.devを使ったコーディング

- 完全にブラウザ上で動くPHPランタイム
- ここはstdioが実装されてないが…
  - これを逆手にとって自分で定義してやる
- PHPで定数にオブジェクトを代入できるようになったのはPHP 8.1からだが…

```php
if (!defined('STDIN')) {
    define('STDIN', fopen('php://memory', 'rw'));
    fwrite(STDIN, <<<'EOT'
9 9

EOT);
    rewind(STDIN);
}

fscanf(STDIN, "%d %d", $c, $r);
$w = strlen($c * $r);
for(;@++$i<=$r;)vprintf(str_repeat(" %{$w}d",$c)."\
```


pixiv

php-play.devを使ったコーディング

- 完全にブラウザ上で動くPHPランタイム
- ここはstdioが実装されてないが…
  - これを逆手にとって自分で定義してやる
- PHPで定数にオブジェクトを代入できるようになったのはPHP 8.1からだが…
  - 実はfwrite()の戻り値はリソースなので昔から許されてた

```php
if (!defined('STDIN')) {
    define('STDIN', fopen('php://memory', 'rw'));
    fwrite(STDIN, <<<'EOT'
9 9

EOT);
    rewind(STDIN);
}

fscanf(STDIN, "%d %d", $c, $r);
$w = strlen($c * $r);
for(;@++$i<=$r;)vprintf(str_repeat(" %{$w}d",$c)."\
```


pixiv

php-play.devのデメリット

- タイムアウトが実装されていない
- うっかりコードに while(true) とか書いてしまうとブラウザが固まる
  - 後で脱出条件を書こうとするとひっかかる
  - しばらく待ってるとChromeが処理を打ち切ってくれるが、それまで待つ
- php-playはキーボードを打つたびにhistory.pushStateしてくれるので冷静に戻るボタンを1回押せば無限ループ直前まで帰ってこれる

USAMI Kenta @tadsan · 1 month ago

Maintainer

いろは順ソート

「いろはにほへとちりぬるを」の12種類の文字から構成される文字列が改行区切りで入力されます。標準入力の内容を全て読み取り、文字列を「いろは」順に並べて出力してください。

```
入力:
るりいろ
りぬる
いちにち
ほへとちに
いち
ほとはろ
りにへ
いい
いいい
とりほい
いろ
いろいろ
```

```
出力:
いい
いいい
いろ
いろいろ
いち
いちにち
るりいろ
ほへとちに
ほとはろ
とりほい
りにへ
りぬる
```


pixiv

USAMI Kenta @tadsan · 1 month ago

Maintainer

しりとりしましょう

標準入力から改行区切りでカタカナ文字列の単語が入力されます。全ての行を読み取り、最初に入力された行から開始して「しりとり」になるように単語を改行区切りで順番に出力してください。一度使った文字を再利用することはできません。出力した単語の最後の文字が「ン」で終わったときは"負けました\n"と出力して、プログラムを終了してください。最初の文字と最後の文字は重複しないように入力されます。

```
入力:
リンゴ
ライオン
ゴリラ

出力:
リンゴ
ゴリラ
ライオン
負けました
```


pixiv

USAMI Kenta @tadsan · 1 month ago

Maintainer

カレンダー

標準入力から「西暦年-月」の形式で 2025-02 のような行が入力されます。入力された月に合わせて「[X年Y月]」「 日 月 火 水 木 金 土」と出力してから、入力された月のカレンダーを日曜始まり形式で整列して出力してください。全ての日付は「3桁空白右寄せ」で表示します。

```
入力:
2025-02

出力
[2025年2月]
 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
```


pixiv

ボツ問題

USAMI Kenta @tadsan · 1 month ago

Maintainer

しりとりしましょう (ASCII版)

標準入力から改行区切りで英単語が入力されます。全ての行を読み取り、最初に入力された行から開始して「しりとり」になるように単語を改行区切りで順番に出力してください。一度使った文字を再利用することはできません。全ての単語を使い切ったら「おわり」最初の文字と最後の文字は重複しないように入力されます。与えられる文字列は半角英字のみで、の大文字小文字は同一視されます。

```
入力:
Austin
Egypt
Norway
Yosemite
Osaka
Tokyo

出力:
Austin
Norway
Yosemite
Egypt
Tokyo
Osaka
おわり
```

USAMI Kenta @tadsan · 1 month ago

Maintainer

UTF-8エンコーダー

標準入力で行区切りの文字列が入力される。`U\+[0-9A-F]+`形式のUnicodeコードポイントをUTF-8にエンコードして出力せよ。コードポイント以外の文字は無視し、行ごとに末尾に改行コード`\n`を付けて出力せよ。

```
入力:
U+0061 U+0062 U+0063 U+0064 U+0065
U+3042 U+3044 U+3046 U+3048 U+304A
U+1F9D1 U+200D U+1F9D1 U+200D U+1F9D2

出力:
abcde
あいうえお
👨‍👩‍👦
```


pixiv

USAMI Kenta @tadsan · 1 month ago

Maintainer

ロボットゴルファー

標準入力から、`<?php`から始まるPHPコードが入力されます。コードを末尾まで読み取って、仕様通りにコードを短縮してください。

- 変数名を短かい名前にします (出現順に`a`, `b`, `c`...を使用します)
- シンタックスエラーにならない範囲で空白を削除します

仕様外の人間ゴルファーの最適化テクニックを反映しても点数には反映されないので気をつけてください。

```
入力:
<?php

$fruits = ['apple', 'orange', 'banana'];
foreach ($fruits as $fruit) {
    echo strtoupper($fruit);
    echo PHP_EOL;
}

出力:
$a=['apple','orange','banana'];foreach($a as$b){echo strtoupper($b);echo PHP_EOL;}
```


pixiv

USAMI Kenta @tadsan · 1 month ago

Maintainer

リスト処理言語

標準入力で以下のような形式のJSONが入力される。以下の規則に従って処理せよ。出力は`echo json_encode(JSON_PRETTY_PRINT)`形式で出力する。

- 入力はJSONのarrayであり、先頭が**命令**、以降の要素を引数とする
  - `["+", 1, 2, 3]` は、一般的な計算式の `1 + 2 + 3` を意味する
  - `["-", 1, 2, 3]` は、一般的な計算式の `1 - 2 - 3` を意味する
  - `["*", 1, 2, 3]` は、一般的な計算式の `1 * 2 * 3` を意味する
  - `["/", 1, 2, 3]` は、一般的な計算式の `1 / 2 / 3` を意味する
    - 計算結果は全て以上の式をPHPで処理したものとする
  - `["list", ...exprs]`
    - 引数を処理した結果を配列(リスト)として返す
    - `["list", 1, 2, ["-", 5, 2]]` → `[1, 2, 3]`
- 以下の式は特別に処理する
  - `"var"`: `var` という変数を参照する
  - `["quote", expr]`: 引数の式を評価しない
    - `["quote", "str"]`: 文字列の `"str"` を返す
    - `["quote", ["+", 1, 2, "foo"]]`: `["+", 1, 2, "foo"]` というリストを返す
    - TODO: ***`quote` と文字列の関連がめんどくさいので、たぶん両方とも仕様から消す***
  - `["let", [binds], ...body]`
    - **binds**: `[["a", 1], ["b", 2]]` → `...body` の部分でのみ有効なローカル変数 `a = 1; b = 2` を束縛する
    - **...body**: 複数の式を順番に評価し、最後の式を結果として返す
      - `["let", [["n", 1], ["m", 2]] ["+", "a", "b"]]`: `3` という整数を返す
      - `["let", [["n", 1], ["m", 2]] ["list", "a", "b"]]`: `[1, 2]` という配列を返す
  - `["lambda", [params], ...body]`: 無名関数を作成する
    - `["lambda", ["num"], ["*", 2, "n"]]`: 引数を二倍した値を返す関数
    - `["funcall", ["lambda", ["num"], ["*", 2, "num"]], 10]`: `20` を返す
  - `["map", lambda, list]`: listの要素をlambdaで呼び出した結果の新しいリストを返す
    - `["map", ["lambda", ["num"], ["*", 2, "num"]], [1, 2, 3]]`: `[2, 4, 6]` を返す

pixiv

入力:
```
["+", 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

出力:
```
55
```

入力:
```
["let", [["a", 2], ["b", 3]],
  ["setq", "c", 4],
  ["/", ["-", ["+", 1, ["*", "a" , "b"]], "c"], 10]]
```

出力:
```
0.3
```

入力:
```
["let", [["1+", ["lambda", ["n"], ["+", "n", 1]]]],
  ["map", "1+", ["list", 1, 2, 3, 4, 5]]]
```

出力:
```
[
    2,
    3,
    4,
    5,
    6
]
```


pixiv

```php
$prog = json_decode(stream_get_contents(STDIN), JSON_THROW_ON_ERROR);

echo json_encode(lisp($prog), JSON_PRETTY_PRINT), "\n";

function lisp(mixed $expr, array &$env = []): mixed
{
    //var_dump($expr);
    //var_dump(compact('env'));
    if (is_int($expr) || is_float($expr)) {
        return $expr;
    }
    if (is_string($expr)) {
        return $env[$expr];
    }
    if ($expr === null) {
        return null;
    }
    $op = array_shift($expr);
    //var_dump($op);
    switch ($op) {
        case "funcall":
            $func = $env[array_shift($expr)];
            $args = $expr;
            if ($func[0] === "lambda") {
                [$lambda, $params, $body] = $func;
                $new_env = $env;
                foreach ($params as $i => $param) {
                    $new_env[$param] = $args[$i];
                }
                return lisp($body, $new_env);
            }
        case "setq":
            $name = array_shift($expr);
            $result = lisp($expr[0], $env);
            $env[$name] = $result;
            foreach ($expr as $e) {
                $result += lisp($e, $env);
            }
            return $result;
        case "-":
            $result = lisp(array_shift($expr), $env);
            foreach ($expr as $e) {
                $result -= lisp($e, $env);
            }
            return $result;
        case "*":
            $result = 1;
            foreach ($expr as $e) {
                $result *= lisp($e, $env);
            }
            return $result;
        case "/":
            $result = lisp(array_shift($expr), $env);
            foreach ($expr as $e) {
                $result /= lisp($e, $env);
            }
            return $result;
        case "let":
            $bind = array_shift($expr);
            $new_env = $env;
            foreach ($bind as [$name, $value]) {
                $new_env[$name] = $value;
            }
            $result = null;
            foreach ($expr as $e) {
                $result = lisp($e, $new_env);
            }
            return $result;
        case "list":
            $result = [];
            foreach ($expr as $e) {
                $result[] = lisp($e, $env);
            }
            return $result;
        case "map":
            $result = [];
            $func = array_shift($expr);
            foreach (lisp($expr[0], $env) as $e) {
                $result[] = lisp(["funcall", $func, $e], $e
            }
            return $result;
        default:
            var_dump(compact('op', 'expr'));
            //var_dump(compact('env'));
    }
}
```