Web ブラウザ上のプログラミング学習環境 WaPEN の改良

中西渉

watayan@meigaku.ac.jp
名古屋高等学校

SSS2019
2019 年 8 月 18 日

1. はじめに
2. 先行研究
   - DNCL の実行環境
   - 自動採点
3. WaPEN について
   - 概要
   - 改良点
4. 考察
   - フローチャート
   - 自動採点用データに関して
   - バックエンドとの連携
5. おわりに

今回は PyPEN の話はしません

（たぶん CE で近々...）

1 はじめに

2 先行研究
- DNCL の実行環境
- 自動採点

3 WaPEN について
- 概要
- 改良点

4 考察
- フローチャート
- 自動採点用データに関して
- バックエンドとの連携

5 おわりに

はじめに

次期学習指導要領（2022～）

「情報 I」…必履修科目

1. 情報社会の問題解決
2. コミュニケーションと情報デザイン
3. コンピュータとプログラミング
4. 情報通信ネットワークとデータの活用

「情報 B」「情報の科学」はあまり選択されてなかった...

どんな言語を使う？

「『情報 I』教員研修用教材」では...

- Python
- JavaScript
- VBA
- Swift
- ドリトル

→ 特定の言語ではない？
→DNCL にも目はアリ？

本稿の概要

WaPEN（DNCL のプログラミング学習環境）の改良

- 自作関数・手続きの実装
- 自動採点機能
- サンプルプログラムの差し替えを容易にするプログラムの開発
- その他

1 はじめに

2 先行研究
- DNCL の実行環境
- 自動採点

3 WaPEN について
- 概要
- 改良点

4 考察
- フローチャート
- 自動採点用データに関して
- バックエンドとの連携

5 おわりに

DNCL の実行環境

DNCL について

DNCL…センター試験「情報関係基礎」出題用擬似言語

DNCL のプログラム例

```
a ← 1
b を 1 から 10 まで 1 ずつ増やしながら,
  | a ← a*b
  | a を表示する
を繰り返す
```

「センター試験用手順記述標準言語（DNCL）の説明」が仕様（？）

- 「入力」がない
- 関数の定義方法が不明
- 「DO～UNTIL」型ループがないのに，センター試験には出題

→ 拡張が必要

ローカルマシンで実行できる環境

- PEN
- PenFlowchart

PEN

初学者向けプログラミング学習環境

PenFlowchart

フローチャートでプログラムを生成できる機能を PEN に追加

※フローチャートは自作関数・手続きに非対応

PEN，PenFlowchart は Java アプリケーション

利点　いろんな OS で使える

欠点　最近は避けられがち

ソフトウェアのインストールに自由がない学校も

→Web ブラウザ上の学習環境に移行すべきか？

Web ブラウザで実行できる環境

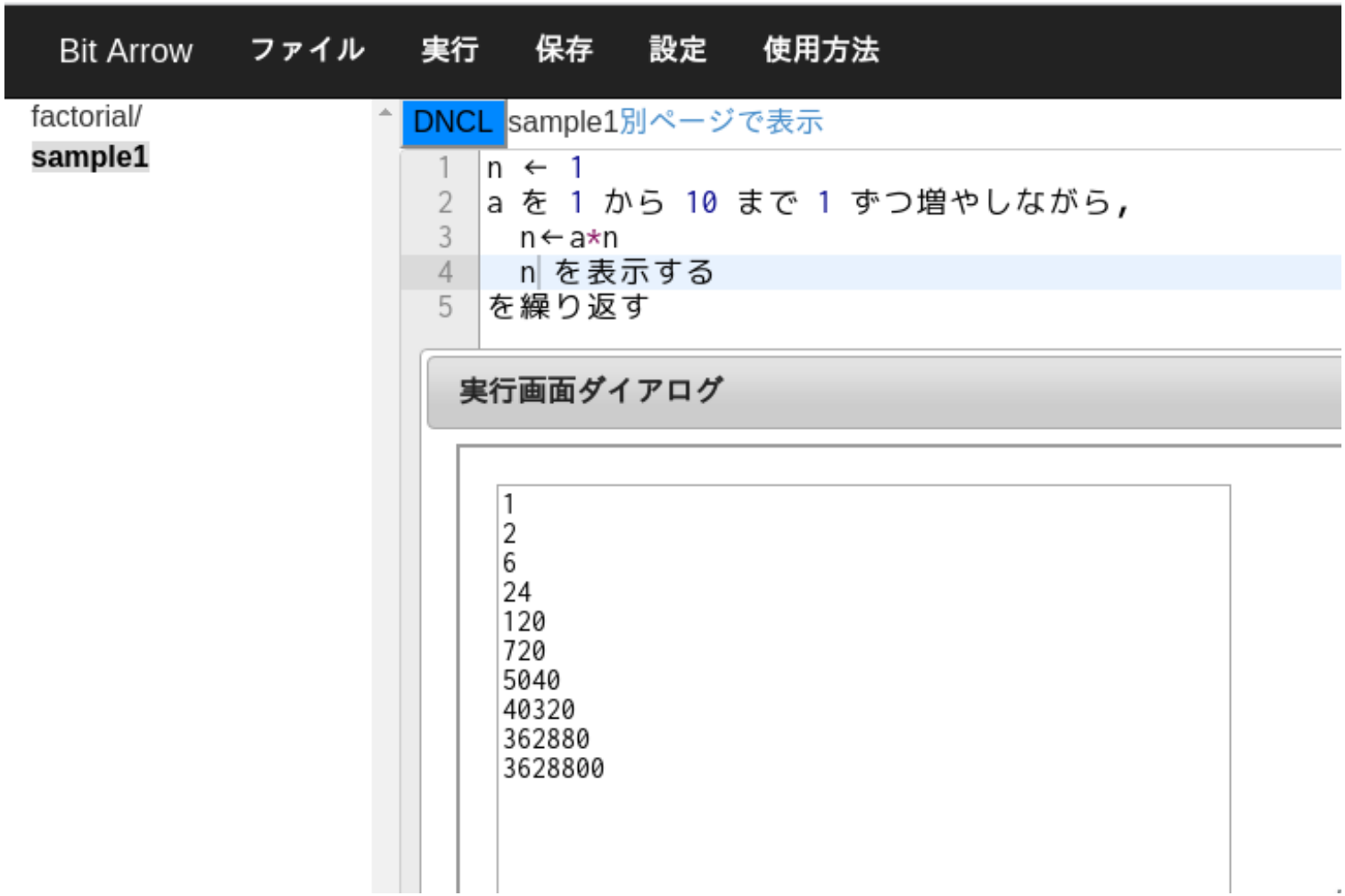
- WaPEN
- どんくり
- wPen

WaPEN

PenFlowchart を Web ブラウザ上に（ほぼ）実現
詳細は後ほど

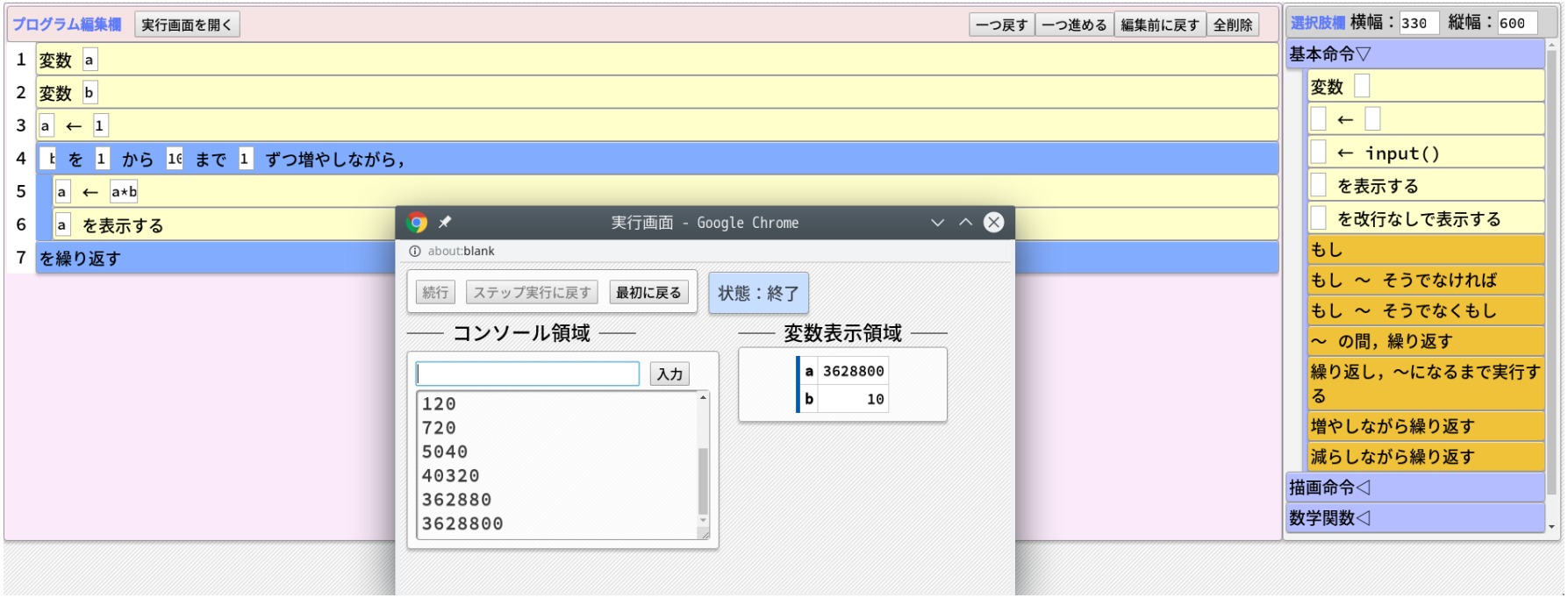
どんくり

Bit Arrow が DNCL 対応
DNCL の「仕様」に忠実



wPen

短冊でプログラミング
DNCL → JavaScript に変換
出題機能



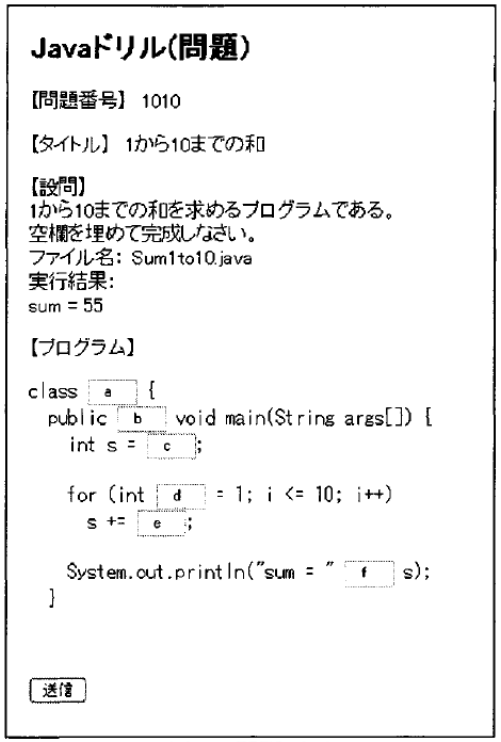
自動採点

ホワイトボックステスト ソースコードを評価

ブラックボックステスト 実行結果を評価

ホワイトボックステスト

- 穴埋めになりがち？
- 単純な文字列比較ではダメ
  → 実行結果の評価も必要？



Javaドリル(問題)

【問題番号】 1010

【タイトル】 1から10までの和

【設問】
1から10までの和を求めるプログラムである。
空欄を埋めて完成しなさい。
ファイル名: Sum1to10.java
実行結果:
sum = 55

【プログラム】

```
class [a] {
  public [b] void main(String args[]) {
    int s = [c];

    for (int [d] = 1; i <= 10; i++)
      s += [e];

    System.out.println("sum = " [f] s);
  }
```

送信

図 2　問題画面

JavaDrill

step	プログラム
	//キーボードからユーザが整数を入力します。その値が4ならば「秀です」,
	//3ならば「優です」, 2ならば「良です」, 1ならば「可です」, それ以外ならば
	//「不可です」と表示しなさい。
	//ヒント:この問題でif文を使うと分岐が増えて大変になります。
	# include < iostream >
	using namespace std ;
	int main ( ) {
1	int input_num ;
2	cin >> input_num ;
3	[ ]
3.1	[ ] cout << " 秀です " << endl ;
3.2	break ;
3.3	[ ] cout << " 優です " << endl ;
3.4	break ;
3.5	[ ] cout << " 良です " << endl ;
3.6	break ;
3.7	[ ] cout << " 可です " << endl ;
3.8	break ;
3.9	[ ] cout << " 不可です " << endl ;
3.10	break ;
	[ ]
4	return 0 ;
	}

ルーチン	step	main input_num	出力
main	1	?	
main	2	3	
main	3	3	
main	[ ]	3	[ ]
main	4	3	

図 1　穴埋め問題の例

pgtracer

（画像は参考文献 [10][11] より引用）

ブラックボックステスト

- 実行可能でないとダメ
- 「悪さ」をさせないための工夫（SandBox など）

1 はじめに

2 先行研究
- DNCL の実行環境
- 自動採点

3 WaPEN について
- 概要
- 改良点

4 考察
- フローチャート
- 自動採点用データに関して
- バックエンドとの連携

5 おわりに

概要

- DNCL によるプログラミングを Web ブラウザ上で行える学習環境
- JavaScript のみで作成 → ローカルでも実行可能
- 多くのブラウザで動作確認（IE11 も）
- Jison で生成したパーサで構文解釈
  → フローチャート生成を楽に
  → 一種の SandBox？
- ~~PEN にある変数表示画面がない~~
  → 代替策を実装（どんくりの真似）

勤務校で使用（2017〜）
情報処理学会教員免許更新講習で使用（2018〜）

WaPEN の実行画面

新規
実行
ステップ実行
リセット
フローチャート
コード→フローチャート
Upload
ファイルを選択
選択されていません
Download
ファイル名：
問題選択
採点
整数 a,b,c[5]
「サイコロを60回振って出た目の回数を数えるシ
aを1から60まで1ずつ増やしながら,
| b←random(5)
| c[b]←c[b]+1
を繰り返す
bを0から5まで1ずつ増やしながら,
| c[b]を表示する
を繰り返す
サイコロを60回振って出た目の回数を数えるシミュレーション
7
7
16
13
9
8

整数
実数
文字列
入力
出力
改行無出力
代入
もし
もし～そうでなければ
～の間，繰り返す
～になるまで実行する
増やしながら
減らしながら
関数
手続き
サンプル1
サンプル2
サンプル3
サンプル4
サンプル5
サンプル6
サンプル7
サンプル8
マニュアル
整数
a,b,c[5]
サイズ:↑
実数
サイズ:0
文字列
サイズ:↓
真偽
はじめ
「サイコロを60回振って出た目の回数を数えるシミュレーション」
a:1→60
1ずつ増
b←random(5)
c[b]←c[b] + 1
b:0→5
1ずつ増
c[b]
おわり

改良点

1. 関数・手続きの実装
2. 自動採点機能
3. MakeSample の作成
4. その他
5. その他の他

改良点１...関数・手続きの実装

次期学習指導要領「情報 I」　(3) コンピュータとプログラミング

関数の定義・使用によりプログラムの構造を理解するとともに，性能を改善する工夫の必要性についても触れるようにする。

自作関数・手続きの実装状況：

PEN	OK
どんくり	OK
PenFlowchart	フローチャートが対応せず
WaPEN	実装せず

改良点１...関数・手続きの実装

次期学習指導要領「情報 I」　(3) コンピュータとプログラミング

関数の定義・使用によりプログラムの構造を理解するとともに，性能を改善する工夫の必要性についても触れるようにする。

自作関数・手続きの実装状況：

PEN	OK
どんくり	OK
PenFlowchart	フローチャートが対応せず
WaPEN	~~実装せず~~ → 実装した（フローチャートは未対応）

関数の定義

```
関数　《関数名》（《引数》）
｜…
｜《値》を返す
関数終了
```

値を返さないとエラー

手続きの定義

```
手続き　《手続き名》（《引数》）
｜…
｜手続きを抜ける
｜…
手続き終了
```

DNCL でいうところの「値を返さない関数」

呼び出すときは（引数がなくても）カッコをつける

関数の使用例

```
1  整数 x,y
2
3  x を入力する
4  y を入力する
5  x を表示する
6  y を表示する
7  gcd(x,y) を表示する
8
9  関数 gcd(a,b)
10 |整数 c
11 |c ← a%b
12 |もし c=0 ならば
13 | |bを返す
14 |を実行し，そうでなければ
15 | |gcd(b,c)を返す
16 |を実行する
17 関数終了
18
19
20
21
```

```
19661220
20190624
19661220
12
---
```

手続きの使用例

```
1  整数 a[9],b[5]
2  a ← {59,34,24,99,38,10,29,89,81,6}
3  b ← {72,18,49,23,90,5}
4  summary(a)
5  summary(b)
6
7  手続き summary(x)
8  |整数 s, i
9  |実数 a
10 |s←0
11 |i を 0 から length(x)-1 まで 1 ずつ増やし
12 | |s ← s+x[i]
13 |を繰り返す
14 |a ← (s + 0.0)/length(x)
15 |「合計：」とs を表示する
16 |「平均：」とa を表示する
17 手続き終了
18
19
20
21
```

```
合計：469
平均：46.9
合計：257
平均：42.833333333333336
---
```

改良点２. . . 自動採点機能

用意した入力に対し，正しい出力が得られるかを判定する機能
（ブラックボックステスト）

次ページのように answer.js に記述
Quiz の引数は

1. タイトル
2. 問題
3. 入力値（値の配列）の配列
4. 期待される出力結果（文字列）の配列
5. 制限時間（ミリ秒。省略時は 10000）

answer.js の例

```
"use strict";
var Quizzes=[
new Quiz(’敬称’,
’文字列を 1 つ受け取って，後ろに「さん」をつけて表示しなさい’,
[[’47’],[’さかなクン’],[’外宮’]],
[’47 さん’,’さかなクンさん’,’外宮さん’],
1000),
new Quiz(’大小比較’,
’整数を 2 つ受け取って，大きい方を表示しなさい’,
[[23,21],[21,23],[10,10],[-13,12],[100,20]],
[’23’,’23’,’10’,’12’,’100’]),
…];
```

実は answer.js を読めばテストケースがわかってしまう...

自動採点の正解例

Q6:曜日の計算 ▾　採点

3つの整数（西暦年，月，日）を受け取って，
その日の曜日を番号（0:日曜，1:月曜，…，6:土曜）で表示しなさい。

```
1  整数 y,m,d,j,k,h
2  y を入力する
3  m を入力する
4  d を入力する
5  もし m<3 ならば
6  |y ← y-1
7  |m ← m+12
8  を実行する
9  j ← y/100
10 k ← y%100
11 h ← d+(13*m+8)/5+k+k/4+j/4+5*j
12 h%7 を表示する
13
14
15
16
17
18
19
20
21
```

```
*** 採点開始 ***
ケース1...成功
ケース2...成功
ケース3...成功
ケース4...成功
ケース5...成功
ケース6...成功
ケース7...成功
ケース8...成功
ケース9...成功
*** 合格 ***
```

自動採点の不正解例

Q6:曜日の計算 ▾　採点

3つの整数（西暦年，月，日）を受け取って，
その日の曜日を番号（0:日曜，1:月曜，…，6:土曜）で表示しなさい。

```
1  整数 y,m,d,j,k,h
2  y を入力する
3  m を入力する
4  d を入力する
5  j ← y/100
6  k ← y%100
7  h ← d+(13*m+8)/5+k+k/4+j/4+5*j
8  h%7 を表示する
9
10
11
12
13
14
15
16
17
18
19
20
21
```

```
*** 採点開始 ***
ケース1...成功
ケース2...成功
ケース3...失敗
結果が違います。
ケース4...失敗
結果が違います。
ケース5...成功
ケース6...失敗
結果が違います。
ケース7...成功
ケース8...成功
ケース9...失敗
結果が違います。
--- 不合格 ---
```

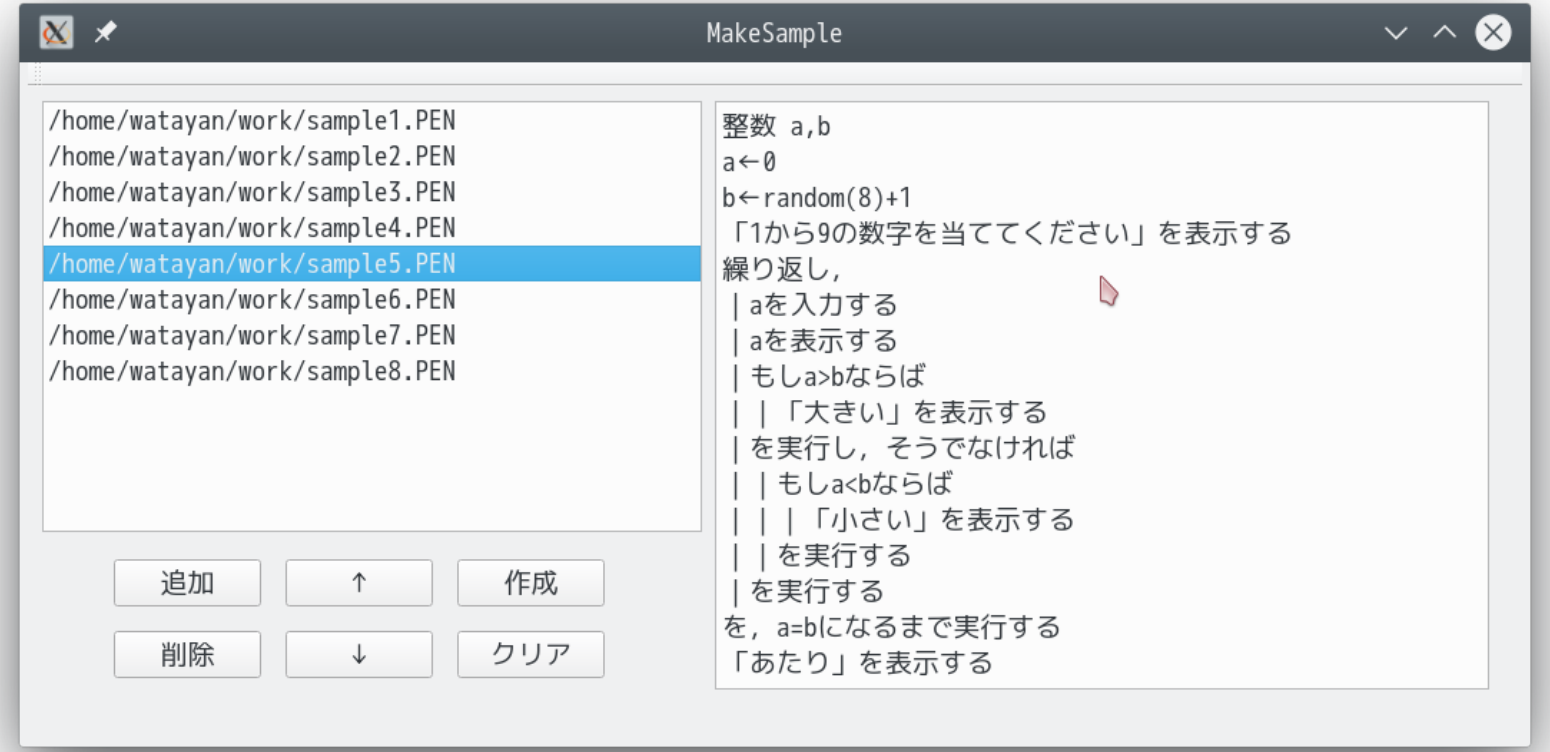
改良点３. . . MakeSample の作成

sample.js にサンプルプログラムを記述
→UTF-8 でないと困る
→JavaScript の文法に従わないとエラー

- ”” でくくる → 必要に応じてエスケープ処理
- 行末に \n
- 文字列結合の ‘+’
- 区切りの ‘,’

→GUI のプログラムを作成
（Qt なのでいろいろな OS で使える）
.PEN ファイルをまとめて sample.js を生成する

MakeSample の実行画面



改良点その他

全角・半角のミスを起こさせないオプション
. . . 実は賛否両論

- アルゴリズムに専念するためにエラーを回避すべし
- 間違いを直す能力をつけるならエラーにすべし
- 警告でいいのでは
- 簡単に切り替えられるといい
- 半角全角の違いが意味を持たないように（手本はドリトル）

→ 設定ファイルで選択

設定ファイル setting.js

```
1  var setting =
2  {
3      var_declaration: 0,
4      // 変数の宣言が必要か否か
5      // 0 宣言なしだとエラー
6      // 1 宣言なしだとその場で生成
7
8      array_origin: 0,
9      // a[n]と宣言した配列の添字の範囲
10     // 0 a[0]～a[n]
11     // 1 a[0]～a[n-1]
12     // 2 a[1]～a[n]
13
14     div_mode: 0,
15     // 整数の割り算における/と÷の区別
16     // 0 同じ（商の整数部分）
17     // 1 /は商，÷は商の整数部分
18
19     zenkaku_mode: 0,
20     // 変数名や数値，数値の入力で全角が使われた場合の処理
21     // 0 内部的に強制的に半角に直してエラーを出さない
22     // 1 エラーを出して停止する
23
24     quiz_mode: 1,
25     // 問題を表示するかどうか
26     // 0 表示しない
27     // 1 表示する
28 };
```

改良点その他の他

- 変数表示画面の代用. . .「変数を確認する」命令
- 配列の比較

「変数を確認する」命令の例

```
整数 a,b,c
a を入力する
b を入力する
繰り返し,
|c ← a%b
|a ← b
|b ← c
|変数を確認する
を, c=0 になるまで実行する
a を表示する
```

```
49
*** 変数確認 ***
a:49
b:21
c:21
*** 変数確認 ***
a:21
b:7
c:7
*** 変数確認 ***
a:7
b:0
c:0
7
---
```

配列の比較の例

```
1  整数 a[5],b,c,n
2  n ← 0
3  繰り返し,
4  |a ← [0,0,0,0,0,0]
5  |n ← n + 1
6  |b を 1 から 60 まで 1 ずつ増やしながら,
7  | |c ← random(5)
8  | |a[c] ← a[c] + 1
9  |を繰り返す
10 |a と n を表示する
11 を, a = [10,10,10,10,10,10] になるまで実行
12
13
14
15
16
17
18
19
20
21
```

```
[16,12,6,10,9,7]55324
[13,5,4,12,11,15]55325
[14,12,8,7,11,8]55326
[15,10,8,7,11,9]55327
[14,9,8,9,12,8]55328
[9,11,8,8,9,15]55329
[13,7,11,9,8,12]55330
[13,9,11,9,9,9]55331
[9,7,13,13,8,10]55332
[4,10,9,20,10,7]55333
[11,7,11,13,8,10]55334
[7,7,14,10,14,8]55335
[10,5,11,7,14,13]55336
[7,9,15,13,7,9]55337
[6,15,12,13,6,8]55338
[10,12,10,5,7,16]55339
[9,10,10,6,16,9]55340
[10,10,10,10,10,10]55341
---
```

1 はじめに

2 先行研究
- DNCL の実行環境
- 自動採点

3 WaPEN について
- 概要
- 改良点

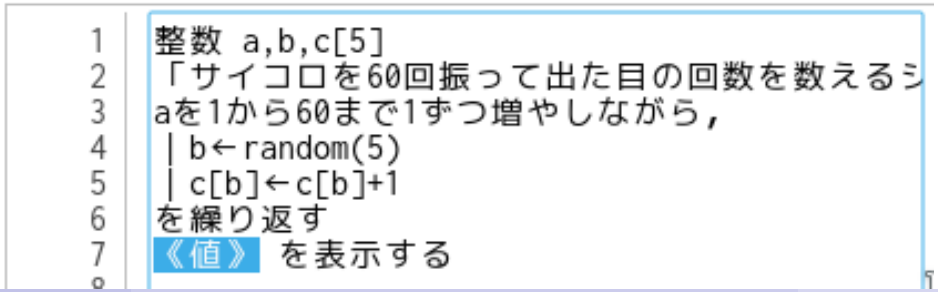
4 考察
- フローチャート
- 自動採点用データに関して
- バックエンドとの連携

5 おわりに

フローチャート

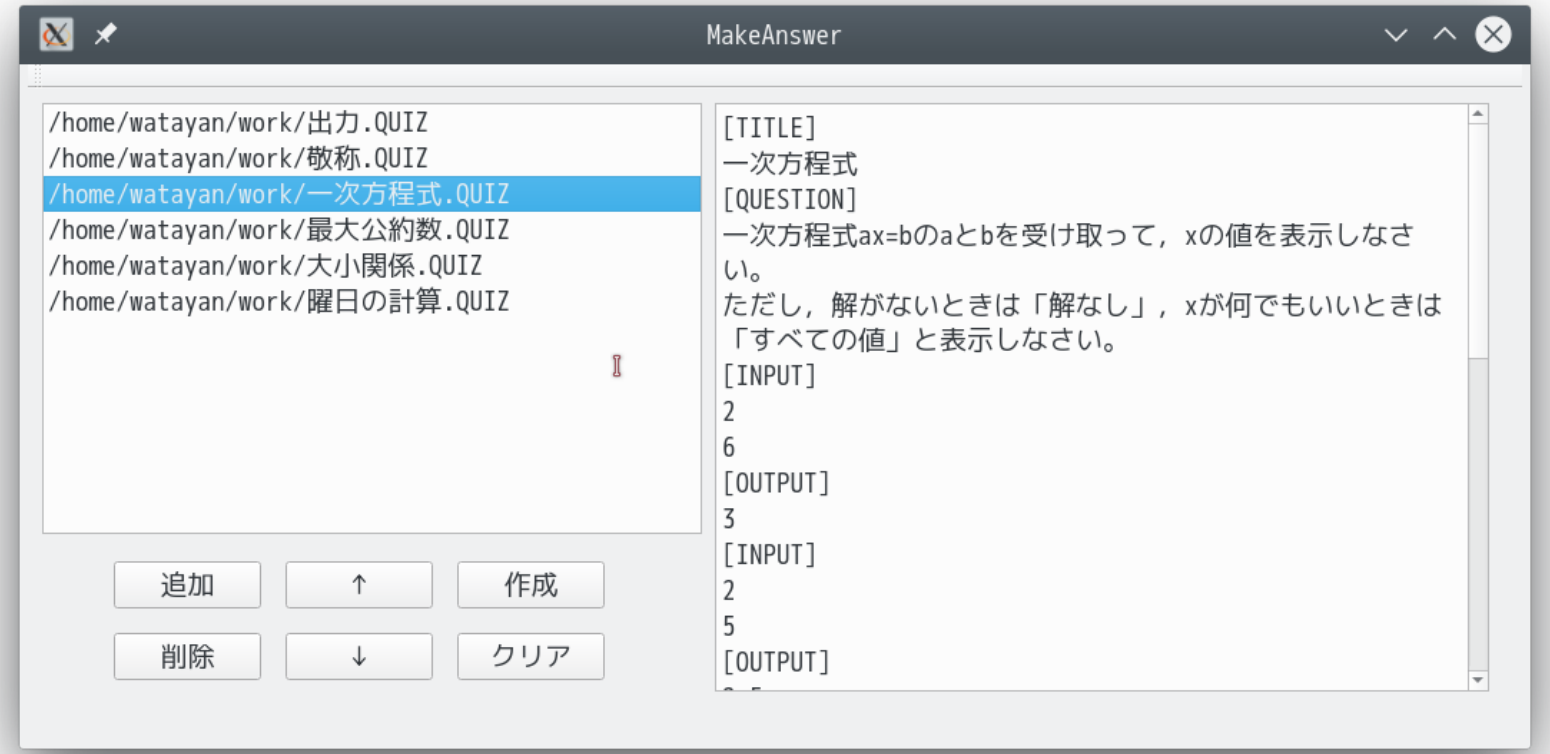
フローチャートは初学者には有効

- 上達したものには？
- 開発する上では？
  フローチャートでの表現が決まらずに実装しないものがある
  - 「そうでなくもし」（else-if に相当）
  - 複数の代入を 1 行で書く
  - 自作関数・手続き（のフローチャート）
- フローチャートの操作の説明が必要
  短時間での演習だと「入力支援ボタン」の活用の方が早いという実践報告
  《～》をクリックすると選択状態になる（PEN の真似）



自動採点用データに関して

MakeSample でサンプル用スクリプト sample.js を生成
→ 自動採点用データも同様にやればいい
→MakeAnswer を実装



バックエンドとの連携

本システムは JavaScript だけでできている
→ ローカルでも実行できる

バックエンドを作れば...

- 課題提出の自動化
- 課題の自動採点
- 自動採点のテストケース隠蔽

※設置のハードルは高くなる

1. はじめに
2. 先行研究
   - DNCL の実行環境
   - 自動採点
3. WaPEN について
   - 概要
   - 改良点
4. 考察
   - フローチャート
   - 自動採点用データに関して
   - バックエンドとの連携
5. おわりに

おわりに

WaPEN に改良を施した

- 関数・手続きの実装
- 自動採点機能
- MakeSample，MakeAnswer の作成
  →Windows 用はインストーラを筆者サイトで配布
- 半角・全角無視オプション
- 「変数を確認する」命令
- 配列の比較

WaPEN の配布先

- 筆者のサイト　`https://watayan.net/prog`
- GitHub　`https://github.com/watayan/WaPEN.git`

GitHub の方がオススメ（更新が簡単）

インストール方法

```
git clone https://github.com/watayan/WaPEN.git
cd WaPEN
copy sample.js-dist sample.js
copy answer.js-dist answer.js
```

更新方法

```
git pull
```

おまけ

こんなの作り中...https://github.com/watayan/PyPEN.git

```
1  整数 a,b
2  a←0
3  b←random(8)+1
4  「1から9の数字を当ててください」を表示する
5  a!=bの間繰り返す：
6      aを入力する
7      aを表示する
8      もしa>bならば：
9          「大きい」を表示する
10     そうでなければ：
11         もしa<bならば：
12             「小さい」を表示する
13 「あたり」を表示する
14
15
16
17
18
19
20
21
```

```
3
1から9の数字を当ててください
4
大きい
1
小さい
2
小さい
3
あたり
---
```