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[1,a)]

概要：筆者は DNCL によるプログラミング学習環境を Web ブラウザ上で実行できる環境として WaPEN を開発し，勤務校での授業で用いてきた．新学習指導要領の「情報 I」に対応するべく，関数や手続きを自作できるよう改良を行った．さらにプログラムの実行結果が正しいかを判定する機能も追加した．また，WaPEN を自分のサーバに置いて使用する教員等がサンプルプログラムを簡単に差し替えられるようなプログラムも開発した．このことによって，授業がよりスムースに行われると考えるものである．

キーワード：プログラミング教育，DNCL，自動採点

Improvement of WaPEN Programming Learning Environment on Web Browser

Wataru Nakanishi[1,a)]

Abstract: I developed 'WaPEN' which is a programming learning environment with DNCL on a Web browser, and I used it in my classes. In order to respond to "Informatics I" of the new course of study, I made improvements to create own functions and procedures. And the auto scoring function has been added. Furthermore, I also developed other program so that teachers can easily replace sample programs. I think that this will make classes more smoothly.

Keywords: Education of programming, DNCL, Automatic scoring

1. はじめに

2018 年に告示された高等学校学習指導要領 [1] において，教科「情報」で必履修となった科目「情報 I」の内容に「コンピュータとプログラミング」が含まれることから，高校生のほとんどがプログラミングの学習をすることになった．これまでプログラミングを明示的に扱う「情報 B」や「情報の科学」を選択していた割合は小さかったため，高校現場では大きい変化として捉えられている．

特に，どのような言語・教材を使うかが話題になることが多い．先ごろ公開された「情報 I」教員研修資料 [2] では Python が用いられているが，JavaScript や VBA，Swift，ドリトル版も作られることが告知されていることから，特定の言語だけが推されることにはならないものと考えられる．だとすれば，後述する DNCL も選択肢にあげることができるだろう．

本稿では，筆者が開発した DNCL でのプログラミング学習環境である WaPEN について行った改良について述べる．まず自作関数・自作手続きを実装したことであるが，これは新学習指導要領「情報 I」に対応するためでもある．さらに課題に対する自動採点機能を追加し，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を開発した．

第 2 章では関連する先行研究について，第 3 章では WaPEN および本研究における改良点について，第 4 章では改良点および今後の開発方針についての考察をそれぞれ述べる．

[1] 名古屋高等学校
Nagoya Senior High School
[a)] watayan@meigaku.ac.jp

2. 先行研究

2.1 DNCL の実行環境

2.1.1 DNCL について

DNCL は大学入試センター試験「情報関係基礎」の出題に用いられる，日本語を元にした擬似言語である．言語に関する説明 [3] はあるがその中には「入力」はなく，関数が定義できることには言及しているがその記述方法は定められていない．またループは「WHILE〜WEND」「FOR〜NEXT」型の説明しかないのに，実際の試験問題には「DO〜UNTIL」型も用いられている．そこで以下に述べる実行環境では DNCL をそれぞれ適当に拡張している．

2.1.2 ローカルマシンで実行できる環境

PEN[4] は大阪学院大学情報学部西田研究室と大阪市立大学大学院創造都市研究科松浦研究室の共同プロジェクトとして開発された，初学者向けプログラミング学習環境である．DNCL は日本語に近いため，手入力すると微妙に間違った構文や命令を書いてしまいがちなので，それを防ぎ入力効率を良くするために入力支援ボタンが用意されている．

また，筆者は PEN を元にして，フローチャートでプログラムを生成できる PenFlowchart[5] を開発した．これはプログラムの実行部分は PEN をそのまま使い，フローチャートを編集すれば即座に DNCL のプログラムが生成される（プログラムをフローチャートにすることもできる）というものである．ただし，フローチャートは自作関数や自作手続きには対応していない．

2.1.3 Web ブラウザで実行できる環境

PEN や PenFlowchart は Java アプリケーションなので多種の OS で動作させることができるが，昨今の状況から Java の Runtime をインストールすることは敬遠される傾向にあるし，個人の判断で新しいソフトウェアをインストールすることが禁じられている学校も多々ある．そこで Web ブラウザ上でのプログラミング学習環境がいくつか提案され，使われている．

筆者は PenFlowchart を JavaScript で移植した WaPEN を開発した [6]．詳細は後に述べる．

オンラインプログラミング環境ビットアロー（Bit Arrow）[7] では様々な言語が使え，その中に DNCL が使える‘どんくり’[8] がある．これはオンラインで使うだけでなく，ダウンロードして使うこともできるので，インターネットへの接続が制限された状況でも使うことができる．私見であるが変数の宣言を必要としないことや「x を 2 増やす」などの構文を構えていることなど，DNCL の説明 [3] にはもっとも忠実な環境だと思われる．

大宮らによる wPen[9] は短冊を組み合わせてプログラムを作り，DNCL を内部的に JavaScript に変換して実行している．また，出題機能などを備えている．

2.2 自動採点

プログラミングの自動採点についてはソースコードを評価する方法（ホワイトボックステスト）と実行結果を評価する方法（ブラックボックステスト）が考えられる．

JavaDrill[10] や pgtracer[11] などはソースコードを評価する考え方がベースになっているが，いずれも穴埋め形式のものであって DNCL にはそぐわないと考える．また，模範解答との単純な文字列比較では完全な採点ができない（たとえば `n=n+1` と `n++`のどちらでも正しい場合があるなど）．そのため，結局は実行結果を評価する仕組みを作らざるを得ず，pgtracer は実際にそのような実装がなされている．

一方，実行結果を評価する場合はコンパイルエラーなどが取り除かれて実行できることが最低限必要となるが，これは初学者にとって決して簡単なことではない．逆に実行できて正しい結果が得られる場合でも，課題の条件を満たしているかの判断は別に行わなくてはならない [12]．

また，サーバ上にシステムを設置してサーバ上でコンパイルや実行を行う場合は，投稿されたプログラムが「悪さ」をしないよう配慮することが特に求められる．実際，田上らの自動採点システム [13] では `malloc` などのメモリ管理関数をフックして範囲外のメモリを破壊していないかを確認したり，実行を SandBox 上で行って悪意のあるプログラムを封じるなどの工夫がなされている．

3. WaPEN について

3.1 概要

WaPEN（以下, 本システム）は 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である．JavaScript で作られており，データベースなどを必要としないので，サーバにアップロードせずにローカルに置いたままでも使用することができる．動作確認は Google Chrome，Firefox，Microsoft Edge，Internet Explorer，Safari で行っている（ただし Safari では行数の表示がうまくいかない）．

実行イメージを**図 1** に示す．左側のテキストエリアがプログラムを入力する部分であり，画面下部の入力支援ボタンを押すと構文の雛形が入力されるので，ユーザは《変数》や《値》などの部分を書き換えるだけでよい．またプログラム入力のテキストエリアで右クリックすることによって，定義済み関数などが入力できる．実行結果は真ん中のテキストエリアに表示される．さらに下部にはサンプルプログラムを呼び出すボタンが用意されている．

右側のフローチャート画面では，縦棒上で右クリックすることによって部品が追加でき，部品は右クリックやダブルクリックで編集できる．フローチャートを編集するとただちにプログラムが修正されるが，プログラムを修正したときには「コード→フローチャート」ボタンを押さないとフローチャートに反映されない．

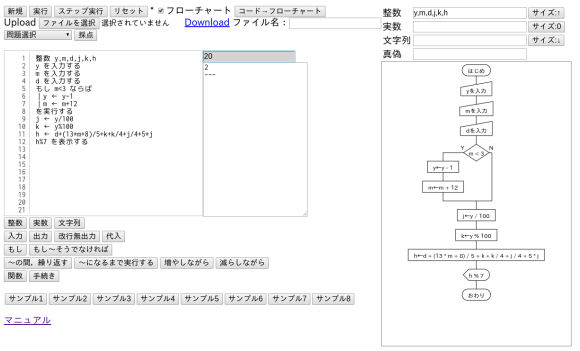

図 1　WaPEN の実行画面
Fig. 1　Screenshot of WaPEN

前述した Web ブラウザ上の DNCL 実行環境には，DNCL を JavaScript に「翻訳」して実行しているものがあるが，本システムでは JavaScript 用のパーサジェネレータ Jison[14] で生成したパーサで構文解析した結果を評価する形をとっている．これはフローチャート生成を簡単にするためであるが，一種の SandBox 的な効果も果たしている．

PEN にあるような変数表示画面は実装していないが，どんくりにあるような「すべての変数を表示する関数」を作ることでその代用になると考えている（未実装）．

2017 年度以降の勤務校での授業では本システムを用いている．また報処理学会の教員免許更新講習でも，センター試験「情報関係基礎」の第 3 問のプログラムを作るという実習でこれを用いている．

3.2　改良点

3.2.1　関数・手続きの実装

次期学習指導要領 [1] では，「情報 I」の内容（3）「コンピュータとプログラミング」の取扱いについて，「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」と記述されている．そこで，情報 I の実習を行うためには関数や手続きを作って使うことができる環境を用意しなくてはならない．

PEN やどんくりではこれらが既にできているが，Pen-Flowchart のフローチャート部分はこれに対応していない（プログラム実行部分は PEN をそのまま使っているので実行はできる）．WaPEN はこれまでまったく対応していなかったので，実行できるように実装した．ただし，フローチャートは対応していないので，自作関数や自作手続きが使われているプログラムでは「コード→フローチャート」ボタンを押しても「対応していません」と表示して無視するようにしている．

定義は PEN と同じ構文で図 **2** のように記述し，使うときは（引数がないときでも）() をつけて呼び出す．関数名や手続き名は変数名と同様に英文字で始まる英数字列に限

```
関数 《関数名》(《引数》)        手続き 《手続き名》(《引数》)
｜...                            ｜...
｜《値》を返す                   手続き終了
関数終了
```

図 2　関数・手続きの定義
Fig. 2　Definition of functions and procedures

```
"use strict";
var Quizzes=[
new Quiz('敬称',
'文字列を 1 つ受け取って，その後ろに「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',
'2 つの整数を受け取って，大きい方を表示しなさい．',
[[23,21],[100,200],[10,10],[-13,12]],
[23,200,10,12])
,
...];
```

図 3　answer.js の例
Fig. 3　Example of answer.js

定している[*1]．関数と手続きの違いは，関数は必ず値を返すこと（値を返さずに終了するとエラーになる），手続きは命令として呼び出せることである．

3.2.2　自動採点機能

いくつかのテストケースを用意し，入力に応じた出力が得られるかどうかを自動的に判定する機能を追加した．

テストケースは answer.js に図 **3** のように記述する．Quiz の引数は順に次の通りである：

(1) タイトル
(2) 問題文
(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) 出力終了までの制限時間（ミリ秒，省略すれば 10000 ミリ秒）

採点を行った画面イメージを図 **4** に示す．競技プログラミングのような場面では失敗した場合には理由を示すべきではないのかもしれないが，授業では入力や出力を忘れているケースが多々見受けられたので，入力の回数間違いなどは結果の間違いと区別がつくようにメッセージを出すようにした．場合によっては入力値と正しい出力結果と誤った出力結果を並べて表示してもいいのかもしれない．

正解であるかどうかの判定は，プログラム終了までの出力をすべて結合し，期待される出力と文字列比較して行っている（最後の改行の有無には拘泥したくないので，末尾の空白文字は除去している）．そのため，たとえば‘「ABC」を出力しなさい’という問題に対して，図 **5** のように解答しても正解になるが，これを不正解とする理由はない．

この機能の使用を想定しているのは出題者が採点の手間

[*1] 日本語文字を含められるようにすることは容易であるが，そのように実装するかどうか迷っている．

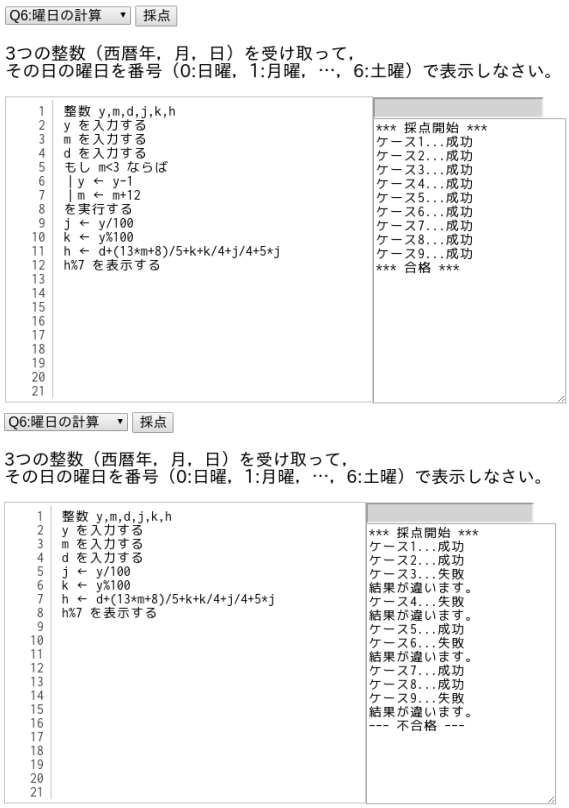

図 4 自動採点機能
Fig. 4 Auto scoring

```
「AB」を改行なしで表示する
「C」を表示する
```

図 5 ‘「ABC」を出力しなさい’ の解答例
Fig. 5 An answer example of ‘output “ABC”’

を省くためではなく，学習者が提出前に自分のプログラムが正しいことを確認するという場面である．というのも，勤務校での授業で手が止まっている生徒に対して，次のような問答をすることがよくあるからである：

教師「できたの？」
生徒「...」
教師「実行してみた？」
生徒「先生，このプログラム合ってますか？」
教師「そんなの実行してみたらわかるでしょ」
生徒「...（ようやく実行ボタンを押す）」
教師「...入力しなきゃ進まないよ」
生徒「何を入力したらいいんでしょう」

なぜかわからないのだが，実行ボタンを押すのに躊躇している様子なのだ．個人的には「実行」ボタンも「採点」ボタンも違わないと思うのだが，実は「実行したらわかる」ためには自分で入力を考えた上に実行結果が正しいかどうかも自分で考えることが要求されている．問題文を読めばそれくらい自分で考えられるはずだと言いたいのはやまやまであるが，完全に指定されたこと以外の行動をためらう者がいるのも事実である．後ろ向きな対応ではあるが，その判断が自動化されるということによって，ハードルを下げる効果はあるかもしれない．

```
"use strict";
var sample=[
"7/2 を表示する\n"+
"1/3*3 を表示する\n"+
"1*3/3 を表示する"
,
"13%5 を表示する\n"+
"7%2 を表示する\n"+
"8%2 を表示する"
,
...];
```

図 6 `sample.js` の例
Fig. 6 Example of `sample.js`

筆者は本システムについてはサーバ上に設置しなくても動作することを求めているので，現行では入力値や期待される出力を JavaScript のプログラム中のデータとして与えている．そのため解答者はそれをのぞき見して，与えられたテストケースだけをクリアするようなプログラムで正解を得ることもできてしまう．しかし，筆者が想定している学習者のレベルではそのようなプログラムを作るほうがむしろ難しいし，提出されたものを見れば簡単に見分けが付くので，特に問題であるとは考えていない．自動採点の結果をそのまま評価に利用しなければいいだけのことである．この機能がブラックボックステストに徹しているのはそのためでもある．

3.2.3 MakeSample の作成

JavaScript では Web サーバ上に置いたファイルを取り込むことはできるが，ローカルマシンにあるファイルを取り込むことは難しく，主要な Web ブラウザに限っても共通して使える方法は見つけられなかった．そのため，サンプルプログラムは `sample.js` というファイルに JavaScript の配列の要素としての文字列として書く方法で実現している．したがってこのファイルを書き換えればサンプルプログラムを差し替えることができるのだが，その場合には**図 6** のように各行を"でくくり，行末には`\n` や文字列結合のための+を追加しなくてはならない．書き方を少し間違えるだけでシステム全体が使えなくなるため，かなり面倒である．

これを手作業でしなくても済むように，複数の`*.pen` ファイルをまとめて `sample.js` に書き込む Perl スクリプトを同梱してはいるが，取り込むファイル名の順序によって順番が固定されるし，Windows ユーザの多くは Perl をインストールしていないものと思われるので，あまり有効に使われるような気がしない．

そこで，使うファイルを指定すればそれらをまとめた `sample.js` を生成する GUI プログラム ‘MakeSample’ を作成した．筆者の開発環境は Linux であるが，このプロ

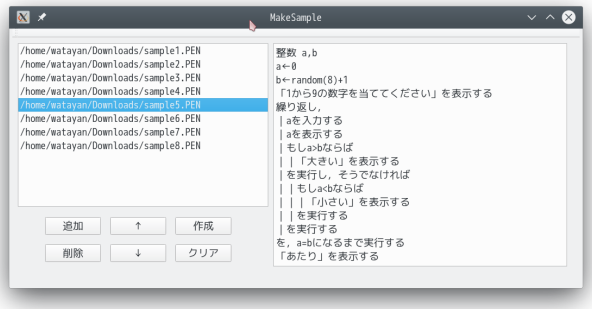

図 7 MakeSample の実行画面
Fig. 7 Screenshot of MakeSample

グラムは Windows など他の OS でも動く必要があるので Qt[15] を用いて開発した．Qt は多くの OS に対応したアプリケーションフレームワークである．実行画面を**図 7** に示す．取り込むファイルとその順序を決定して「作成」ボタンを押すと，指定したフォルダに `sample.js` を作成する．これによってサンプルプログラムの差し替えはずいぶん楽になっている．

現時点ではインストーラの準備などができていないのでバイナリの配布はしていないが，ソースコードは GitHub で公開している[*2]．

また些細なことではあるが，サンプルプログラムの数に合わせてサンプルボタンを自動的に増やすようにした．

3.2.4 その他

生徒たちは全角半角の違いに無頓着であるため，それが原因でのエラーをよく起こしてしまう．そこで数値や変数名などが全角文字で入力されたことによるエラーを回避できるオプションを用意した．設定ファイルである `setting.js` の `zenkaku_mode` で強制的に半角に変換するか，エラーにするかを選択できる．

これについては Facebook 上で賛否両方の意見をいただいている[*3]．

- アルゴリズムに専念するためにはエラーを回避できるのがいい
- 間違いをはっきりエラーにすることで，それを直せる能力をつけさせるべきである
- エラー扱いはしないにしても警告はすべきである
- 簡単に切り替えられるようにするとよい
- 全角半角の違いが意味の違いにならないようにしてもらいたい（ドリトルがうまくやっているように）

など複数の助言をいただき，どちらにも決めかねるということで，設定ファイルで選択できるようにした次第である．

[*2] https://github.com/watayan/MakeSample.git
[*3] https://www.facebook.com/photo.php?fbid=2267703269959823

4. 考察

4.1 フローチャート

PenFlowchart や WaPEN ではフローチャートでプログラムを作成する機能がある．初学者にとってそれは有効であるが [16]，ある程度上達した者にとってはこれがかえって足かせになっている面もあるように思われるし，開発する上でもそのように感じることがある．

たとえば「そうでなくもし」(else-if) や，複数の代入をコンマ区切りで 1 行に書くこと（どちらも DNCL の説明 [3] に明記されている）を筆者のこれらのプログラムで採用していないのは，フローチャートでそれを表現する方法をうまく決められていないからである．もちろん JIS のフローチャートの規格 [17] ではそれらを表現することができるようになっているのだが，それをどのように実装するかが思いついていない．また，関数や手続きのフローチャートは画面を切り替えることなどで可能だとは考えているが，それを実装する意味があるのかは疑問である．

県内の他校で，情報の教科書の例題プログラムを参考にしつつ数学の大学入試問題を WaPEN のプログラミングで解くという実践が行われたのだが，実施者は短い時間（実質 1 時間半）でこれを教えるには，フローチャートよりも入力支援ボタンが有効であるという感想を述べていた．「入力支援ボタンを押す」→「《〜》の部分を穴埋め問題のように適当なものに置き換えていく」という手順がわかってしまえば，フローチャートでの編集より速いのだという．筆者が勤務校で行っている実習でも，少し大きいプログラムになるとやたら縦に長いフローチャートになってしまって見通しが悪くなる．

これらのことからフローチャートでの操作は導入や初期段階にとどめておいて，早々に入力支援ボタンを使ったテキストでのプログラミングに移行するのが良いのではないかと考えている．勤務校では 3 学期にプログラミングの実習を行うので，そのような扱いに切り替える予定である．

4.2 自動採点用データに関して

サンプルプログラムを収める `sample.js` の編集については MakeSample を作ることで作業が簡単になった．一方，自動採点用データの `answer.js` も同様に素の JavaScript のコードであるから，これも同様のプログラムで生成できるようにしたい．この実装は比較的容易だと考えられるので，近いうちに行おうと考えている．

4.3 バックエンドとの連携

本システムは JavaScript だけでできており，ファイルをサーバもしくはローカルマシンに置くだけで使うことができる．しかしデータベースなどと連携すれば，課題の提

出や提出された課題の自動採点結果を集計するなどの機能を追加することも可能であるし（ただし 3.2.2 の末尾に述べたように，行っているのはブラックボックステストなので，プログラム自体を見ずに実行結果だけを問うてはいけない），自動採点のテストケースを隠蔽することもできるだろう．そうすれば採点の手間は大きく軽減できるのだが，その分サーバやその上の設定が必要になるので設置のハードルが高くなる．

それでも，Web サーバ上で若干の手作業をしさえすれば動かせる程度のものなら作れそうだとも思われるので，そのような発展形をオプションとして今後考えてみたい．

5. おわりに

Web ブラウザ上で動作する，DNCL でのプログラミング学習環境 WaPEN に改良を施した．これによって，より使いやすくなったものと考えている．

WaPEN は筆者のサイト[*4]と GitHub [*5]で配布している．前者には ZIP ファイルが用意してあるので取扱いがわかりやすいが，後者は git さえインストールしてあれば最新版への更新が簡単にできるのでこちらを薦めたい．

参考文献

[1] 文部科学省：高等学校学習指導要領（平成 30 年告示）(2018).

[2] 文部科学省：高等学校情報科「情報 I」教員研修用教材，文部科学省（オンライン），入手先 ⟨http://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416756.htm⟩（参照 2019-05-22）.

[3] 独立行政法人大学入試センター：センター試験用手順記述標準言語（DNCL）の説明，独立行政法人大学入試センター（オンライン），入手先 ⟨https://www.dnc.ac.jp/albums/abm.php?f=abm00004841.pdf&n=H23_dncl.pdf⟩（参照 2019-05-30）.

[4] 中村亮太，西田知博，松浦敏雄：プログラミング入門教育用学習環境 PEN，情報処理学会研究報告　コンピュータと教育（CE），Vol. 2005-CE-081, No. 104 (2005).

[5] 中西　渉：PenFlowchart の開発，研究報告コンピュータと教育（CE），Vol. 2012-CE-013, No. 13 (2012).

[6] 中西　渉：WaPEN…DNCL の Web ブラウザ上の実行環境におけるフローチャートなどの実装，情報教育シンポジウム論文集，Vol. 2018, No. 31, pp. 210–214 (2018).

[7] 兼宗　進，並木美太郎，長　慎也：オンラインプログラミング環境ビットアロー（Bit Arrow），大阪電気通信大学，東京農工大学，明星大学（オンライン），入手先 ⟨https://bitarrow.eplang.jp/⟩（参照 2019-05-30）.

[8] 本多佑希，兼宗　進：ブラウザ上で動作する DNCL 学習環境「どんくり」の開発，研究報告コンピュータと教育（CE），Vol. 2018-CE-147, No. 10 (2018).

[9] 大宮大地，松本嵩大，松浦敏雄，中西通雄：試験問題作成機能と学習及び受験用機能を持つ DNCL プログラミング環境，研究報告コンピュータと教育（CE），Vol. 2018-CE-148, No. 7 (2019).

[10] 内田保雄：初級プログラミング学習のための自動作問システム，研究報告コンピュータと教育（CE），Vol. 2007-CE-092 (2007).

[11] 掛下哲郎，柳田　崚，太田康介：穴埋め問題を用いたプログラミング教育支援ツール pgtracer の開発と評価，情報処理学会論文誌教育とコンピュータ（TCE），Vol. 2, No. 2, pp. 20–36 (2016).

[12] 内藤広志：ヒューリステックを用いた，プログラミング演習用の効率的な自動採点システム，情報処理学会第 66 回全国大会講演論文集，Vol. 2004, No. 1 (2004).

[13] 田上恒大，阿部公輝：比較的大きなプログラミング課題のための自動採点システム，研究報告コンピュータと教育（CE），Vol. 2006-CE-083 (2006).

[14] Zachary Carter: Jison, (online), available from ⟨https://zaa.ch/jison/⟩ (accessed 2019-05-31).

[15] The Qt Company: Qt Cross–platform software development for embedded & desktop, (online), available from ⟨https://www.qt.io/⟩ (accessed 2019-05-29).

[16] 中西　渉，辰己丈夫，西田知博：PenFlowchart を用いた，フローチャートによるプログラミング学習の効果に対する評価，情報処理学会論文誌教育とコンピュータ（TCE），Vol. 1, No. 4, pp. 75–82 (2015).

[17] 日本規格協会：JIS X 0121:1986　情報処理用流れ図・プログラム網図・システム資源図記号 (1986).

[*4] https://watayan.net/prog/wapen.html

[*5] https://github.com/watayan/WaPEN.git