プログラミング

中西渉 *

情報処理学会　教員免許更新講習　2020 年 12 月 26 日

1　学校でのプログラミング

2020 年度から小学校でプログラミングが扱われること，2022 年度から高校で必履修となる「情報 I」ですべての高校生がプログラミングについて学習することなど，学校現場ではプログラミングが大きい話題になっている。筆者自身は「コンピュータを使う＝プログラムを作る」という少年時代を経験しているので何の違和感もないのだが，出来合いのアプリケーションを使うのが当たり前になってからコンピュータを使い始めた人にとっては，自分でプログラミングすることの必要性はさほど感じられないのかもしれない。

もちろん「プログラミング的思考」というキーワードで説明される能力を身につけることは有益であるが，ここではあえて筆者の私見を述べる。それは，プログラミングができる方が「自由」になるということだ。もちろん自由になる手段はいろいろあって，言葉で自由を手に入れる人，数学で自由を手に入れる人，音楽で自由を手に入れる人…そういう人の中に，プログラミングで自由を手に入れる人がいるのだと思っている。

2　プログラミング学習環境

プログラミング学習環境にはさまざまなものがあるが，担当者の一存では使用するソフトウェアを自由に選べないこともあると聞いているので，ここではインストールしなくても Web ブラウザを用いて使えるものを紹介する。検索すれば新しいものがいくらでも見つかってキリがないので，筆者が使用しているものに限定する。実際に学校の授業で使えるかどうかについては，利用条件などを考慮した上で判断していただきたい。

ただし，この中には Internet Explorer（以下 IE）では動作しないものが含まれていることにも注意されたい。Microsoft 社自身も使用を推奨していないとはいえ，学校の事情などにより IE しか使えないこともままあるわけで…。

* 名古屋高等学校

2.1 汎用的なプログラミング言語

まず，一般のアプリケーション開発でも用いられている「普通の」言語の学習環境を取り上げてみる。

2.1.1 Paiza.IO

https://paiza.io/ja/

オンラインで C，C++，Java，Perl，Python，Ruby など多くの言語が利用できるプログラミング環境。

2.1.2 Wandbox

https://wandbox.org

さまざまな言語が利用できるオンラインコンパイラサービス。各言語について多くの環境が用意されており，さらにコンパイラやそのバージョンによる違いなどを確認することもできる（たとえば次のコードが Python3.8 以降では動くが 3.7 では動かないとか）。

```
print(a:=3)
print(a)
```

https://github.com/melpon/wandbox でプログラムが公開されているので，自分のサーバなどにも設置できるはずなのだが，環境構築はとても難しい。冒頭にあげたサイトを利用するのが簡単であろう。

2.1.3 Monaca

https://edu.monaca.io

アプリ開発現場だけでなく，教育機関でも多く採用されているモバイルアプリ開発用ツール。HTML5 と JavaScript による開発ができ，デバッガをスマートフォンにインストールして設定しておけば，実機で動作を確認することもできる。プロジェクトはクラウドに保存される。アカウントを作成することが必要。書籍『Monaca で学ぶはじめてのプログラミング』も出版されている。

2.1.4 Colaboratory

https://colab.research.google.com

Google が機械学習の学習・研究のために開発したプラットフォームで，ブラウザから Python のコードを実行できる。Anaconda や Jupyter Notebook をパソコンにインストールしなくても，それらと同様の環境を利用することができる。Google アカウントが必要。

2.2 教育用プログラミング言語

続いて，教育用に作られた（あるいは筆者がそのように想像する）プログラミング言語について触れる。

2.2.1 アルゴロジック

https://home.jeita.or.jp/is/algo/ 問題解決型のプログラミング体験ゲーム。与えられた問題を解いていくものなのでプログラミング言語とは言いにくいが，くりかえしなどの構造の理解を助けるものだと考えている。

2.2.2 Viscuit

https://www.viscuit.com「メガネ」という仕組みだけで構成されるプログラミング言語。

2.2.3 Scratch

https://scratch.mit.edu あまりにも有名なプログラミング環境であるため説明に困る。これを「教育用」と言い切ってしまうのは筆者自身違和感もある。

2.2.4 Bit Arrow

https://bitarrow.eplang.jp JavaScript，ドリトル，簡易 C，DNCL，Python，Tonyu に対応したプログラミング学習環境。

2.2.5 Pictogramming

https://pictogramming.org/ ピクトグラムとプログラミングを組み合わせたプログラミング学習環境。IE でも動作する。たとえば下のプログラムを PC 版で実行すると…。

```
回転待ち 左肩 -150 1
繰返し 3
回転待ち 左肘 -60 0.5
回転待ち 左肘 60 0.5
終わり
移動待ち 500 0 2
```

2.2.6 WaPEN

https://watayan.net/prog/wapen.html 大学入試センター試験「情報関係基礎」の問 3 で使われていたプログラミング言語 DNCL を用いたプログラミング学習環境。

2.2.7 PyPEN

https://watayan.net/prog/pypen.html WaPEN の構文表記を Python のインデントによる方法に近づけたもの。大学入試共通テストに「情報」が導入される場合を想定して試作された問題[1]の第 5 問で使われていたプログラミング言語の文法がこれに似ていたことに驚いた。

3 プログラミング実習

前章の最後に取り上げた PyPEN を用いて簡単なプログラムを作る実習を行う。筆者のサイトにある https://watayan.net/prog/PyPEN/ で実際に入力してもいいし，この PDF 中の青い文字はプログラムが入力された PyPEN の画面へのリンクになっているので，それを利用してもいい。

3.1 変数・代入・演算

変数とは，数値や文字列などの値を覚えておくための入れ物のようなものであり，PyPEN では英数字の文字列（ただし 1 文字目は英字）で表す。変数に値を覚えさせることを「代入」といい，「《変数》←《値》」のように表す[2]。値には「整数」「実数」「文字列」「真偽」の型がある。

演算で使う記号（演算子）は次の通りである。

+	和	/	余りを出さない割り算の商（結果は実数になる）
-	差	//	余りを出す割り算の商（商の整数部分）
*	積	%	余りを出す割り算の余り

基本的に左から順に計算すること，+や-よりも*，/，//，%が優先されること，カッコがあればそれが最優先であることなどは算数や数学と同じである[3]。なお，/と//の使い分けは Python と同じになっている。

では次の各プログラムが終了したときに変数はどのような値になっているか，たどってみてください。答え合わせは「正解」で表示されるプログラムを実行すればいい。

正解

```
a ← 3
b ← 5
c ← b+5
a ← c*2
b ← a-2
```

正解

```
a ← 3
b ← 25
c ← a+b
a ← c//a
b ← c%a
```

正解

```
a ← 3
b ← 5
もし a*2>b ならば：
    a ← a+2
そうでなければ：
    b ← b-1
a ← a+1
```

正解

```
n ← 1
a ← 2
n<5 の間繰り返す：
    n ← n+1
    a ← 3*a-1
```

1） 情報処理学会「大学入学共通テストへの『情報』の出題について」https://www.ipsj.or.jp/education/edu202012.html 参考資料参照。

2） 実は多くの言語でそうなっているように「《変数》=《値》」でもよい。

3） 数学ではカッコの外のカッコは中カッコを使ったりするが，（PyPEN も含めて）ほとんどのプログラミング言語ではカッコはすべて ( ) である。

3.2 配列

番号つきの変数があるといろいろ便利なので，配列というものがある。たとえば 10 人の点数を集計するプログラムを作るとき，点数を保存する変数を a,b,c,…,j とするのでは人数が増えたときに困るし，合計するにしても a+b+c+⋯+j という式を書かなくてはいけない。a0,a1,a2,…,a9 にしたとしても，変数名についている数字はただの文字なので結局同じことになる。

そこで配列を用いる。具体的には a[0],a[1],a[2],…,a[9] という変数を用意して，[ ] の中の番号を変化させて処理を行うのである。たとえば合計を求めるのであれば，合計を表す変数 s を用意して

```
s ← 0
n を 0 から 9 まで 1 ずつ増やしながら繰り返す：
    s ← s+a[n]
```

とすればいい。これなら人数が 100 人になっても繰り返しの回数を変えるだけだ。

PyPEN では配列の値は [ ] にコンマ区切りで値を並べて表す。たとえば

a ← [100,90,80,70]

とすると，a[0] が 100，a[1] が 90，a[2] が 80，a[3] が 70 ということになる。

3.3 ガチャのシミュレーション

ガチャはスマホのゲームなどによくあるシステムで，何種類かのアイテムなどのどれかがランダムに当たるものである。ガチャを何回引けばアイテムがコンプリートできるかについてシミュレーションを行う。ここでは話を簡単にするために，次のように設定する：

- 1 回のガチャで，10 種類のアイテムのうち 1 つがもらえる
- どのアイテムがあたる確率も $\frac{1}{10}$ で毎回同じ

プログラムは次のようになる。まず a に [0,0,0,0,0,0,0,0,0,0] を代入することで，a[0]〜a[9] をすべて 0 にする。そしてアイテムの番号を 0〜9 とし（random(9) で 0〜9 の乱数が得られる），b 番目のアイテムを取得したら a[b] に 1 を代入する。この操作を，全アイテムが出揃って a が [1,1,1,1,1,1,1,1,1,1] になるまで繰り返せばいい。なお，n はガチャを引いた回数である。

余裕のある人はこれを入力してもいいし（0 や 1 の個数を間違えないように注意），リンクでプログラムが入力された PyPEN の画面を呼び出してもいい。何度か試してみて，コンプリートに必要な回数を調べてみよう。

プログラムへのリンク

```
a ← [0,0,0,0,0,0,0,0,0,0]
n ← 0
a!=[1,1,1,1,1,1,1,1,1,1] の間繰り返す：
    n ← n+1
    b ← random(9)
    a[b] ← 1
    n，a を表示する
```

何度もやればだいたいの回数はわかってくるが，あくまでも「だいたい」でしかない。せっかくコンピュータを使っているのだから，これを 100 回繰り返して平均を取ってみるのはどうだろうか。その計算を簡単にするために，全アイテムが揃うまでの回数を数える部分を関数にして，その回数の合計を 100 でわって平均を求めるプログラムを作ると次のようになる。

プログラムへのリンク

```
関数 gacha()：
    a ← [0,0,0,0,0,0,0,0,0,0]
    n ← 0
    a!=[1,1,1,1,1,1,1,1,1,1] の間繰り返す：
        n ← n+1
        b ← random(9)
        a[b] ← 1
    n を返す

s ← 0
k ← 100
i を 1 から k まで 1 ずつ増やしながら繰り返す：
    s ← s+gacha()
s/k を表示する
```

実は n 個のアイテムが揃うまでの回数の期待値は $n(\frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{n})$ なので[4]，$n=10$ の場合はおよそ 29.3 回である。

他にどんな拡張が考えられるだろうか。たとえば…

- 「レアアイテム」を作ってみる。たとえば元のプログラムの `b ← random(9)` の部分を `b ← (random(99)+10)//11` にすれば，0 が 1%，1〜9 がそれぞれ 11%の確率になる。揃うまでの回数もずっと多くなるだろう。
- 回数をすべて記録して，ヒストグラムで分布を図示する。グラフ描画の説明をすると長くなるので，ここでは具体的なプログラムへのリンクを張るだけにしておく[5]。

4） Coupon collector problem というらしい。証明は https://mathtrain.jp/completegacha など。

5） 実行前にフローチャートを消しておくと見やすい。また，100 回だと分布の形がはっきりわからないので 1000 回とかにするのがいいかもしれない。