第 4 章の代わりのテキスト

名古屋高校　情報科

2018 年度

》　問題の解決と処理手順の自動化

コンピュータはプログラムを実行することによって作業をしている。それはパソコンだけでなく，iPad のようなタブレットや携帯電話やゲーム機，あるいは自動販売機やデジタル時計でも同様である。我々が普段何気なく使っている機器の多くにコンピュータが搭載されている。それらを扱うということは，プログラムを実行する（あるいは実行中のプログラムを操作する）ことにほかならない。

この単元では，自分でプログラムを作ってみる。そのことを通じて，身の回りのコンピュータやプログラムがどのように動いているのか，考えを巡らせてみてほしい。

》第 1 節　アルゴリズム

アルゴリズムとは，問題を解決するための具体的な手順を記述したものである。たとえば数学の問題を解く手順，理科の実験の手順，駅で切符を買って電車に乗る手順なども一種のアルゴリズムといえる。

アルゴリズムを頭の中だけで考えるのは間違えやすいし，他人に伝えることも出来ない。そこでアルゴリズムを記述する方法として，この授業では**フローチャート**を用いる。

》第 2 節　PEN, PenFlowchart, WaPEN

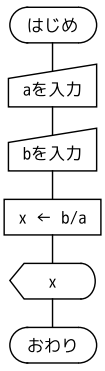
アルゴリズムはプログラムではないから，そのままではコンピュータで実行できない。そこで**プログラミング言語**を用いて，アルゴリズムを元にプログラムを作らなくてはいけない。この授業では xDNCL(*1) というプログラミング言語を用いる。

xDNCL の開発・実行環境として開発された PEN (*2) というプログラムがあるが，マウスでフローチャートを作ることによって xDNCL のプログラムを生成する PenFlowchart (*3) というプログラムを本校で開発したので，この授業ではこれを用いる。メニューから「開発」→「PenFlowchart」で起動しよう。

また，Web ブラウザで同様の作業ができる環境 WaPEN(Web-aided PEN)(*4) も開発したので，それを使ってもいい。ブラウザを起動して `http://www.nagoya-gakuin.ed.jp` から「WaPEN」を選べばいい。使うときには「フローチャート」のチェックボックスにチェックを入れておこう。

1　PenFlowchart，WaPEN の使い方

まず手始めに，一次方程式 $ax = b$ を解くプログラムを作成する。作業を進めることで，PenFlowchart や WaPEN の使い方を覚えることが目的だ。そのため，あえて少し問題のあるプログラムになっている。ツッコミどころはあるだろうが，とりあえず我慢してほしい。




```
整数 a,b,x
a を入力する
b を入力する
x ← b/a
x を表示する
```

出来上がりはこのようになる。これを今から作ってみよう。なお，今後もできるだけプログラムとフローチャートを併記するが，どちらを見てももう片方が思い浮かぶように，両方を理解するようにしてほしい。

1.1　変数

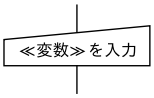
xDNCL のプログラムでは，どのような名前の変数を使うかを，プログラムの冒頭で宣言しなくてはいけない。具体的には，フローチャートの上の窓に変数の名前をコンマで区切って入力する。このプログラムでは `a`，`b`，`x` の 3 つ

(*1) 大学入試センター試験「情報関係基礎」で用いられている DNCL という言語が元になっている。
(*2) `http://www.media.osaka-cu.ac.jp/PEN/`で配布されている。Java の実行環境があれば Windows や Mac OS でも使用できる。
(*3) `https://watayan.net/prog/#penflowchart` で配布されている。Java の実行環境があれば Windows や Mac OS でも使用できる。
(*4) `https://watayan.net/prog/#wapen`，`https://github.com/watayan/WaPEN/`で配布されている。

が必要になる（話を簡単にするために，これらはすべて整数であるとする)。「整数」のところに「a,b,x」と入力しよう。これで，プログラムの中で a，b，x の 3 つの変数が使えるようになる。

1.2 入力

a，b の値は，ユーザがキーボードから**入力**することにしよう。入力はフローチャートでは右のような記号で表され，xDNCL では




```
《変数》 を入力する
```

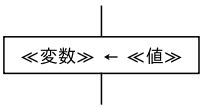
という構文になる。

PenFlowchart の上のパーツの中にある「入力」を，フローチャートの「はじめ」と「おわり」の間の線上にドラッグ&ドロップしよう（WaPEN では，線の上で右クリックして出てくるメニューで「入力」を選ぶ)。パーツが配置できたら，右クリックで表示されるメニューから「編集」を選択して，変数名を「a」にする。これができたら b の入力も同様に追加しよう。

フローチャートを編集すると，それに対応するプログラムが同時に表示される。両方を見ながら作業するようにしよう。

1.3 代入

変数に値を覚えさせることを**代入**という。$x = \frac{b}{a}$ だから，この計算結果を x に代入しよう。代入はフローチャートでは右のような記号で表され，xDNCL では




```
《変数》 ← 《値》
```

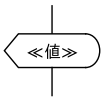
という構文になる。

これも入力と同様に，「代入」のパーツをフローチャートに追加して編集しよう。ただし，$\frac{b}{a}$ は b/a と記述する。

□ 入力と代入　入力と代入はどちらも変数に値を覚えさせることではあるのだが，まったく違う動作であることに注意してほしい。入力はユーザが入力した値を変数に覚えさせるものであり，代入はコンピュータが計算した値を変数に覚えさせるものである。

1.4 出力

計算した x の値を**出力**すれば目的は達成される。出力はフローチャートでは右のような記号で表され，xDNCL では




```
《値》 を表示する
```

という構文になる。もう説明しなくても何をしたらいいかわかるだろう。

1.5 実行

プログラムの画面の近くの「実行」ボタンを押すとプログラムが実行できる。さっそくやってみよう。$2x = 6$ や $3x = 96$ などが解けることを確認しよう。実行するとすぐ入力待ちの状態になるので，$ax = b$ の a の値を入力して Enter を押す。また入力待ちになるので，今度は b を入力して Enter を押す。そうすると x の値が表示されるはずだ。

2 デバッグ

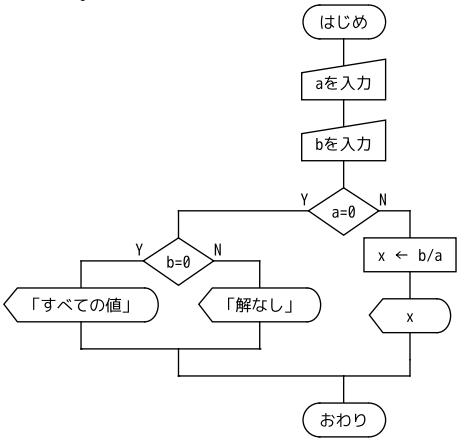
さて，これでプログラムが完成したわけだが，完璧だと言っていいだろうか。試しに次の方程式を解いてみてほしい。

$$0x = 2 \qquad\qquad 2x = 7$$

どちらも困ったことになってしまう。最初の例ではエラーになってしまうし，2 つめの例も $x = 3$ はこの方程式の解ではない。このようなことが起きてしまうようでは完璧なプログラムとはいえない。このようなプログラムの誤りを**バグ**といい，バグを取り除いて正しいプログラムにすることを**デバッグ**という。デバッグした結果，できあがったプログラム，フローチャートは以下のようになる。実際に作って実行してみよう。現時点では意味は詳しくわからなく

てもいいが...それでも何となくわかるのではないかな。

```
実数 a,b,x
a を入力する
b を入力する
もし a=0 ならば
  | もし b=0 ならば
  |   | 「すべての値」を表示する
  | を実行し，そうでなければ
  |   | 「解なし」を表示する
  | を実行する
を実行し，そうでなければ
  | x ← b/a
  | x を表示する
を実行する
```



3　プログラムの直接編集

プログラム自体を直接編集することもできる。その際には画面下部にある「入力支援ボタン」を使うことで，非常に少ない手数でプログラムを入力することができる。細かい間違いを修正するときはこの画面でやった方が手っ取り早いこともある。

ただし，PenFlowchart でも WaPEN でも，フローチャートを変更するとプログラムは強制的にフローチャートの通りに変更されてしまう。そのため，プログラムを編集したら「フローチャート」ボタン（WaPEN では「コード→フローチャート」ボタン）を押して，変更をフローチャートに反映させなくてはいけない。

4　課題提出の方法

課題を提出するときは，PEN のプログラムをコピーして貼り付ける。プログラムの画面をクリックしてから Ctrl+A を押すと全文選択，これをコピーして（Ctrl+C）貼り付ければいい（Ctrl+V）。

》第 3 節　変数・値

xDNCL で使える変数は次の 4 種類の型がある。変数の名前は，英字で始まる英数文字列でなくてはいけない（半角文字）。また，大文字と小文字は区別される。

整数　整数が保存できる。
実数　実数が保存できる...ということだが，小数が扱えると考えればいい。
文字列　文字列が保存できる。プログラム中では文字列は「」で囲む。
真偽　条件の真偽を保存する。TRUE（真）または FALSE（偽）のどちらかの値をとる（この授業では使用しない）。

値の計算でも整数と実数は区別され，整数どうしの割り算では余りが切り捨てられる。次のプログラムで確認してみよう（「*」は ×，「/」は ÷ の意味）。

```
7/2 を表示する
7.0/2.0 を表示する
1.0/3.0*3.0 を表示する
1/3*3 を表示する
1*3/3 を表示する
```

上述したように整数どうしの割り算は余りを切り捨てるので，数学の式と考えたときには同じ式だとしても，プログラムでは別の式になることがある（上の 1/3*3 と 1*3/3 のように）ので，注意してほしい。

□ 課題：偶数奇数の判定　ここでは課題の作成・提出に必要な手順を述べる。実際にやって提出しなさい（以後の課題も，基本的にすべてこの手順で作成・提出を行う）。作るプログラムは次のものとする：

整数を一つ入力し，それが偶数なら「偶数」，奇数なら「奇数」と表示する。

プログラムの作成は次の手順で行なう。

変数　まずどのような変数が必要かを考えよう。
アルゴリズム　どのような順番で処理をしたらいいかを考えよう。
プログラム作成　考えたアルゴリズムの通りにプログラムを作ろう。
動作確認　実行して，正しい結果が得られるかどうか確認しよう。うまくいかなかったら前の手順に戻って考えて直してみる。

さて，偶数奇数の判定のために，新しい記号を一つ覚えよう。整数の割り算の「余り」を計算する記号として % が使える。次のプログラムで使い方を確認してみよう。

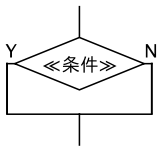
```
13%5 を表示する
7%2 を表示する
8%2 を表示する
```

また，このプログラムでは偶数のときと奇数のときで別の動作を行なうことになるので，その方法について説明する。このような処理は**選択**と呼ばれ，フローチャートでは右下のような記号で表される。xDNCL では下のような構文になる。

《条件》は a=0 や a>0 のように，等式や不等式で表す（詳細は後の授業で説明する）。

```
もし 《条件》 ならば
  |
を実行する
```

```
もし 《条件》 ならば
  |
を実行し，そうでなければ
  |
を実行する
```



完成したらプログラムをコピーして提出する。プログラムの画面で Ctrl+A を押して全文選択して「編集」→「コピー」(*5)，返信メール（または moodle）の画面で「編集」→「ペースト」（または「貼り付け」）すれば，本文にプログラムが挿入されるので，これを送信すればいい。

コピーは Ctrl+C，貼り付けは Ctrl+V でできるので，このキー操作も覚えてほしい。

□ 曜日の計算　2019 年の 1 月は火曜日から始まる。日付の数字を入力したら，1 月のその日が何曜日であるかを答えるプログラムを作ろう。ただし，結果を文字列で表示するのは大変なので，日曜日を 0，月曜日を 1，...，土曜日を 6 で表すことにする。

曜日は 7 日周期なので，7 で割った余りを考えればいい。しかし日付をそのまま 7 で割ったのでは，日曜日が 0 にならない。たとえば 1 日が火曜日だから 1 を入力したら火曜日を表す 2 が出力されるように，6 日が日曜日だから 6 を入力したら 0 が表示されるようにしたいわけだが，だったら日付の数字に何を足してから7 で割ったらいい？

□ 課題：Zeller の公式　特定の月の曜日だけがわかるというだけでは大して面白くもない。どうせなら自分の生まれた日の曜日が分かるくらいのものを作ろう。そこで，曜日を計算する方法としてもっともよく知られている Zeller（ツェラー）の公式を紹介する。これで曜日の計算をするプログラムを作って提出するのが今回の課題だ (*6)

西暦 y 年 m 月 d 日の曜日は次のように計算できる。ただし 1 月，2 月は前の年の 13 月，14 月として計算する（たとえば 2019 年 1 月は 2018 年 13 月として計算する—つまり 1 月と 2 月は y を 1 減らして m を 12 増やす）。y の上 2 桁を j，下 2 桁を k とし，

$$h = d + \left\lfloor \frac{13m+8}{5} \right\rfloor + k + \left\lfloor \frac{k}{4} \right\rfloor + \left\lfloor \frac{j}{4} \right\rfloor + 5j$$

とするとき，h を 7 で割ったときの余りが 0,1,⋯,6 であれば，その日は日曜，月曜，...，土曜である。ただし $\lfloor x \rfloor$ は x の小数点以下を切り捨てた値である (*7)。もっとも xDNCL では，整数で割り算すれば余りが切り捨てられるので，単に割り算をすればいい (*8)。

なお，長い計算式は左から順番に計算されるが，*，/，% は +，- よりも優先して計算されるし，カッコがあればそ

(*5) PEN の画面では，右クリックでメニューが出てこないので画面上部のメニューから選択する。
(*6) あとの授業で文字列での表示ができるようにバージョンアップするので，このプログラムは完成したら保存して，それまで残しておくこと。
(*7) 床関数という。数学では $[x]$ という表記で習い，「ガウスの記号」と呼ぶ。
(*8) プログラミング言語によっては整数どうしの割り算の結果を小数にするものもあり，そういう言語では切り捨ての処理を行わなくてはいけない。

の中を先に計算する。また，xDNCL では掛け算記号は省略できない。たとえば $13m+8$ は `13m+8` でなく `13*m+8` と表すし，ab は `ab` でなく `a*b` と表す。

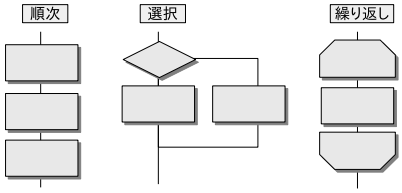
完成したら，いくつかの日付を入力して，正しい曜日が出力できることを確認してから提出しなさい。たとえば

2019 年 1 月 1 日	火曜 (2)	2020 年 1 月 1 日	水曜 (3)
2019 年 3 月 1 日	金曜 (5)	1964 年 10 月 10 日	土曜 (6)
2019 年 12 月 31 日	火曜 (2)	2020 年 7 月 24 日	金曜 (5)

といった日付で答え合わせをしてみるといい。これらが正しく計算できるようになっていれば，自分の生まれた日が何曜日であるかもわかるだろう。

》第 4 節　構造化プログラミング

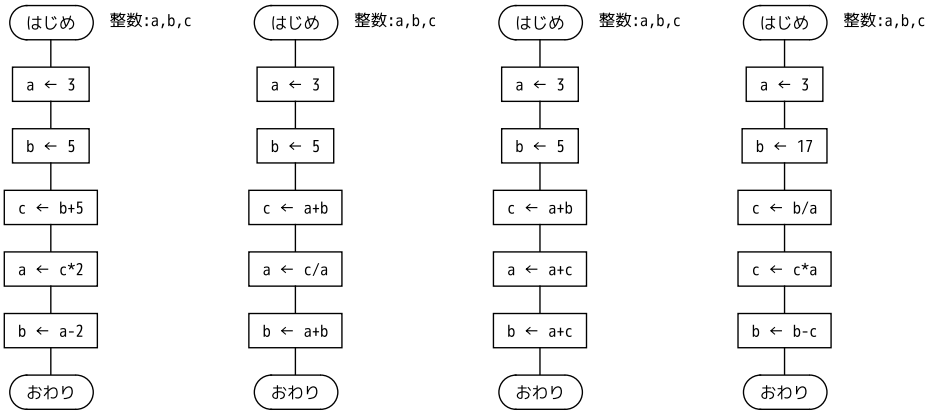
世の中にはとても複雑なプログラムがあり，そのアルゴリズムは当然複雑なものであるが，実は「順次」「選択」「繰り返し」の組み合わせで作られているものが多い。実際 PenFlowchart や WaPEN では，この 3 つの構造で構成されたプログラムしか作ることはできないが，アルゴリズムを表現するにはそれで十分なのである。では，この 3 つの構造を順に見ていこう。



1　順次

順次というのは，上から順番に処理を実行していくだけである。それは一見簡単なことのように思われる。実際，機械が実行するならこれほど簡単なものはない。しかし人間が考えるときには注意が必要だ—人間の頭には工夫するという能力があるので，つい先回りしたり，順番を入れ替えたり，おかしそうな部分を勝手に直したりしてしまいがちだ。たとえば変数に値を代入するという単純な処理でも，順番を入れ替えてしまうと望み通りの結果は得られない。

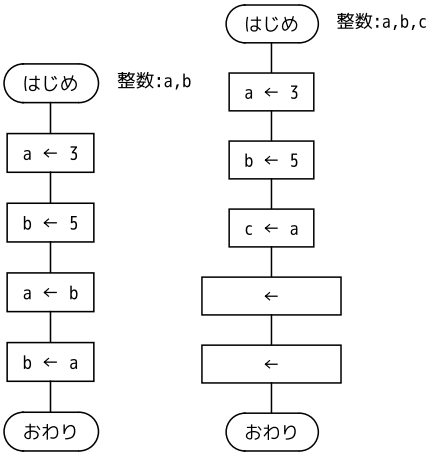
□ **問題 1**　次の各アルゴリズムが終了したときの，各変数の値を答えよ。



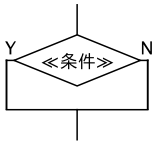
問 1　　問 2　　問 3　　問 4

□ **問題 2**　右図の左側は a と b の値を入れ替えようとしたものだが，失敗している（うまくいかないことを確認せよ）。実は変数の値の入れ替えをするには，もう一つ変数を用意しなくてはいけない。a と b の値の入れ替えが成功するように，右側の空欄を埋めなさい。

なお，この問題の右側は変数の値を入れ替えるときの常套手段である。そのまま丸暗記するようなものではないが「もう一つ変数を用意する」ということだけは覚えておいてほしい。



2　選択



選択は，ある条件が満たされるときとそうでないときで別の処理を行なうというものである。条件は，2 つの値を比較して，等しいとか，等しくないとか，どちらが大きいとかいうように Yes/No で判断できるものである。

条件を表すために用いられる演算子は次のとおりである：

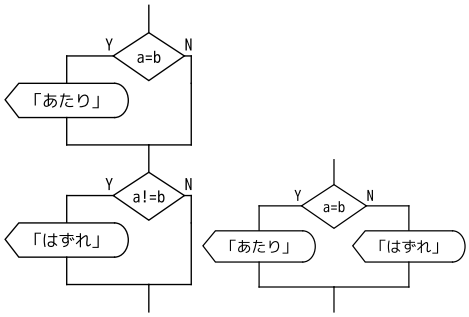
`=`	等しい
`!=`	等しくない（≠ の意味）
`>`	大なり
`<`	小なり
`>=`	大なりイコール（≧ の意味）
`<=`	小なりイコール（≦ の意味）

□ 例題:数あてゲーム　1 桁の乱数（1〜9）を発生させ，ユーザが入力した数値がそれと一致したら「あたり」，違っていたら「はずれ」と表示するプログラムを作りなさい。

乱数とはランダムに数を発生させる仕組みであり，ゲームや占いには欠かせない（もちろん遊びでない目的にも使われる）。xDNCL では `random(n)`（`n` は自然数）で 0 以上 `n` 以下の（整数の）乱数が得られる。たとえばこのプログラムを何度も実行して，0〜10 の値が表示されることを確認してほしい。

```
random(10) を表示する
```

では，1 から 9 の乱数を作るにはどうしたらいいだろう。`random(9)` では 0〜9 になってしまうから...。



注意

このプログラムを作る場合，左側のアルゴリズムでも正しく動作はするが，こういう作り方は良くない。判定が 2 回になってしまって効率が悪いし，もし当たりの条件を変えることになったら 2 箇所を修正しなくてはいけないからだ。うっかり片方だけ直してしまうと，見つけにくいバグを作ってしまうことになる（直すのは人間だから，こういうミスはよく起きる）。右側のようにするのが良い。

□ 課題:数あてゲーム（発展）　上の数あてゲームを，違っていたときにユーザが入力した数値が乱数の値よりも大きければ「大きい」，小さければ「小さい」と表示するように改造しなさい。

2.1　複雑な条件

条件を複数組み合わせることもできる。その場合には，たとえば「`a>3 または b>3`」のように「または」や「かつ」を用いる [*9]。

数学では $1 < x < 3$ というような表し方ができるが，xDNCL では `1<x<3` という表現はできない。これは `1<x` と `x<3` がともに成り立つということだから「`1<x` かつ `x<3`」と表す。同様に，$x < 1, 3 < x$ も「`x<1` または `3<x`」と表す。

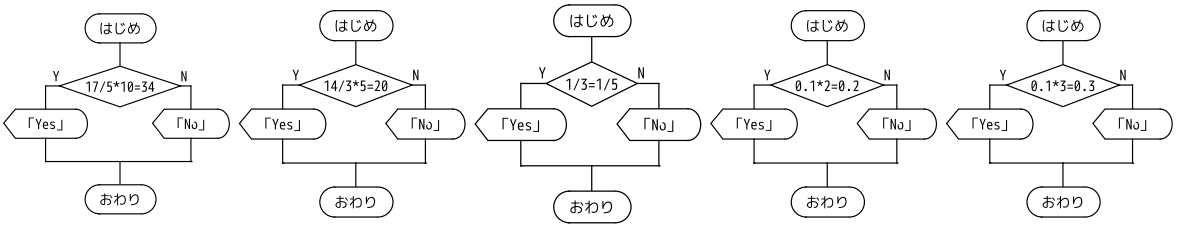
よくある間違いなので注意しておくが，例えば $x = 1, 3$ を「`x=1` または `3`」と書いてはいけない。「`3`」では条件にならないからだ。この場合は面倒でも「`x=1` または `x=3`」と書く。

□ 例題:点数入力　テストの点数は 0 点以上 100 点以下である。この条件に合わない（0 より小さいか，100 より大きい）数値が入力されたときには「間違いです」と表示するプログラムを作る。どのような条件にすればよいか。

□ 例題:点数入力（続）　今度は正しい数値が入力されたときに「よろしい」と表示するようにしたい。どのような条件にすればよいか。

[*9] 否定を表す「でない」もあるのだが，この授業では扱わない。たとえば「`a=0` でない」なら「`a!=0`」，「`a>3` でない」なら「`a<=3`」のように書けるからだ。

□ **問題**　次のアルゴリズムで「Yes」「No」どちらが表示されるか考えなさい。その後で，実際にプログラムを作って確かめなさい。



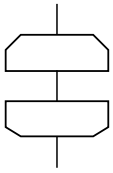
問 1　問 2　問 3　問 4　問 5

問 4，問 5 について：十進法の 0.1 は有限小数だが，二進法だと無限小数になり，コンピュータ内部の計算では有限の適当な桁数で打ち切ってしまう。そのため，最後の桁には誤差が含まれるので，このような結果になってしまった。問 4 と問 5 のどちらが「Yes」かなどということはどうでもいいのであって，次のことを覚えておいてほしい。

- 実数（小数）の計算には誤差が含まれる。
- 数値が一致するかどうかの判断を実数（小数）でしてはいけない。
  差の絶対値が十分小さいかどうかで判断するのがよい。
- 整数なら，一致するかどうかの判断は正確にできる。

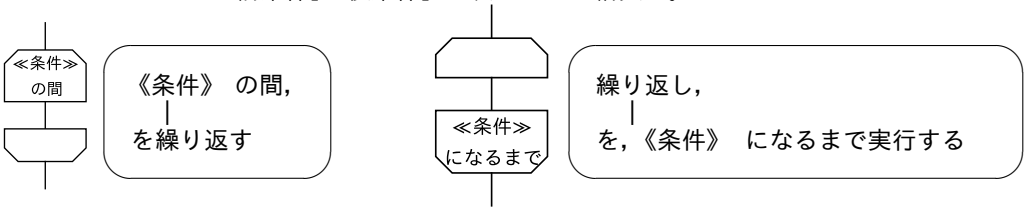
3　繰り返し

「繰り返し」はある条件が満たされるまで（あるいは満たしている間）同じ処理を繰り返すというものである。フローチャートでは図のような記号で表し，上端か下端に繰り返しの終了または継続に関する条件を書く。本当は JIS で決まっている正式な書き方があるのだが，この授業では（規格の上では不正確ではあるが）直観的にわかりやすいと思われる表記を用いる。



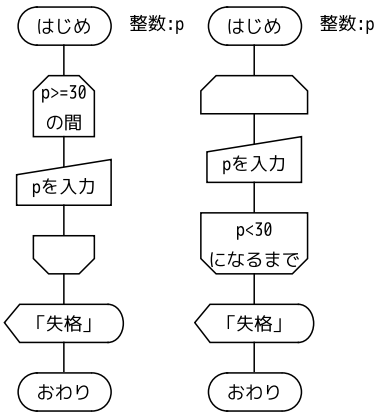
3.1　前条件，後条件

PenFlowchart の「前条件」「後条件」は次の 2 つの構文だ。



見ての通り，繰り返しの上端，あるいは下端に来たときに条件判断をして，繰り返しを続けるか終了するかを決めるというものだ。どちらを使っても構わない場合も多いが，場合によってはどちらかでなくてはいけない場合もある。

□ **例題**　30 点未満が入力されるまで点数入力を繰り返し，30 点未満が入力されたら「失格」を表示して終わるプログラムのフローチャートは右図のどちらが正しいか。



注意

次の 2 つの例題は，入力する整数を記憶する変数のほかに，繰り返しのための変数が 1 つ必要になる。こういったことは課題の文にはあらわれないので，自分で判断して追加しなくてはいけない。

□ **例題：カウントダウン**　整数を入力し，その整数から 0 までカウントダウンするプログラムを作りなさい。

□ **例題：約数**　整数を入力し，その整数の約数を小さい順にすべて表示するプログラムを作りなさい。

□ **課題：ユークリッドの互除法**　2 つの整数の最大公約数を求めるときに，ユークリッドの互除法という方法が用いられる。その手順は次の通りだ。

1. a，b をその 2 数とする。
2. c に a % b を代入する。
3. a に b を代入する。
4. b に c を代入する。
5. c が 0 でなければ 2. に戻る。
6. a が最大公約数である。

これを用いて，839835391 と 527215277 の最大公約数を求めるプログラムを作って提出しなさい。5. で「戻る」となっていることから，これは繰り返しを使う。なお，最初は簡単な値（35 と 21 とか）で正しい答えが出ることを確認すること。

大事な余談

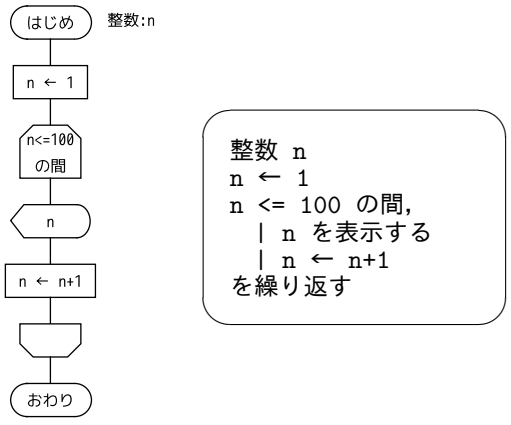
前に例題で作った「約数をすべて求めるプログラム」を使えば公約数を求めることもできるが，このような大きい数だとすごく時間がかかる。実際，素因数分解は「時間がかかる」処理であり，2 学期に習った RSA 暗号は「時間がかかる」から現実には解けないと考えられている。だからもし画期的に速い素因数分解のアルゴリズムが開発されたら，RSA 暗号は役に立たなくなる。

□ **例題：$\sqrt{2}$ の近似値**　$\sqrt{2} = 1.414\ldots$ の値をもっと詳しく求めるプログラムを作ろう。ここでは「二分法」というアルゴリズムを使う。これをプログラムにしてみよう。なお，このプログラムで使う変数はすべて実数型である。

1. a $< \sqrt{2} <$ b となる a，b を適当に定める [*10]。
2. (a+b)/2 を x に代入する。
3. x*x が 2 より大きければ（x は $\sqrt{2}$ より大きいのだから）b に x を代入し，そうでなければ a に x を代入する。その結果やはり a $< \sqrt{2} <$ b であり，b と a の差は半分になるから，a，b ともに $\sqrt{2}$ に近づいていく。
4. 十分な精度（あらかじめ決めておく。たとえば a と b の差が 0.00001 より小さくなるまで，とか）が得られるまで 2. に戻って繰り返す。

3.2　カウントアップ，カウントダウン

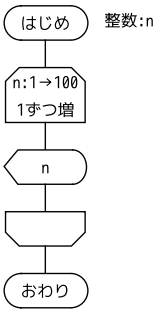
1 から 100 までの整数を小さい順に表示するプログラムのアルゴリズムは次のようになる。



しかし，ある範囲の数を順に扱うケースはよくあるので，xDNCL ではこれを簡単に記述する方法が与えられている。それが次のフローチャート [*11] とプログラムだ。PenFlowchart の繰り返し記号の「増やし」「減らし」を使えばいい（WaPEN では「増やしながら」「減らしながら」になる）。

[*10] だからといって a を $-\sqrt{2}$ より小さい値にしてしまうと困ったことになる。a と b で $\sqrt{2}$ を挟む，というイメージ。

[*11] この表記も正式なものではないが，直感的にわかりやすいのでこの授業ではこれを用いる。




```
整数 n
n を 1 から 100 まで 1 ずつ増やしながら，
  | n を表示する
を繰り返す
```

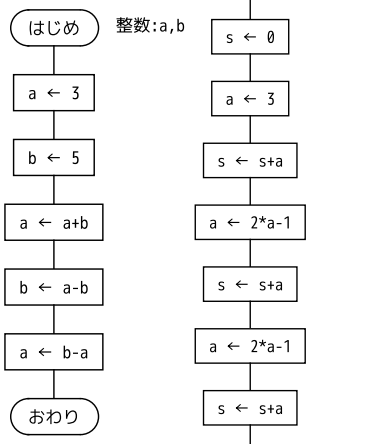
□ **例題:2 桁の奇数**　2 桁の奇数を小さい順に表示するプログラムを作りなさい。

□ **例題:カウントダウン**　入力した整数から 0 までカウントダウンするプログラムを「減らしながら」を使って作りなさい。なお，ここでは変数を 2 つ用意する。入力した整数（つまりカウントダウンのスタートになる値）を表すものと，カウントダウンしていく数を表すものを別に扱うためだ（次の課題でも同様）。

□ **課題:約数**　入力した整数の約数を小さい順にすべて表示するプログラムを作って提出しなさい。前は「前条件」「後条件」の繰り返しを使ってこれを作ったが，今回は「増やし」を使って作ること。

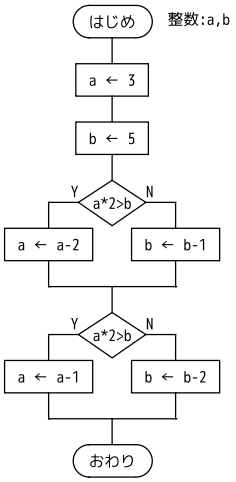
4　練習問題

次の各アルゴリズムが完了した後，各変数の値はいくつになっているか。

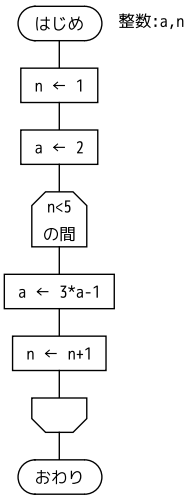


(a) a，b の値

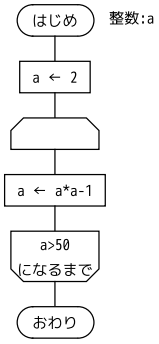
(b) a，s の値



(c) a，b の値



(d) a，n の値



(e) a の値

正解は (a) $a = -5$，$b = 3$ (b) $a = 9$，$s = 17$ (c) $a = 1$，$b = 3$ (d) $a = 122$，$n = 5$ (e) $a = 63$

》第 5 節　文法まとめ

xDNCL の文法のうち，この授業で必要となるものをまとめておく。

1　変数の宣言

変数はプログラムの先頭で宣言しないと使えない。変数が複数あるときはコンマで区切って記述する。配列は [ ] の中に最後の番号を書いて宣言する（配列の番号は 0 から始まる）。

```
整数　《変数名》
実数　《変数名》
文字列　《変数名》
```

2　計算

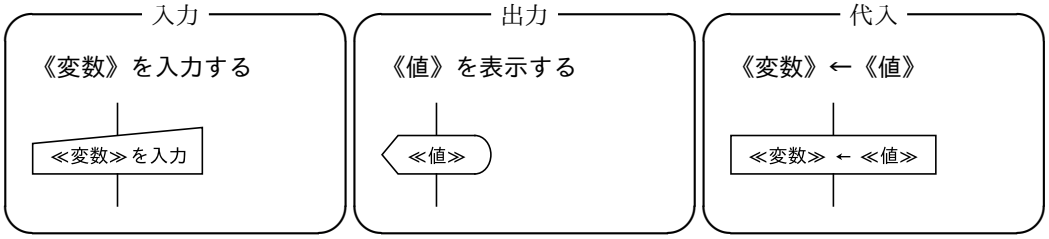
+，−，*，/，% が使える。*は掛け算，/は割り算を表し，% は割り算をしたときの余りを表す。なお，整数どうしを/で割ったときは，余りを切り捨てて計算が行なわれる。

3　条件

=，!=，>，<，>=，<=によって二つの値を比較する（!=は $\neq$，>=は $\geqq$，<=は $\leqq$ をそれぞれ表す）。条件と条件を ‘かつ’ や ‘または’ で組み合わせることもできる。

4　入力・出力・代入

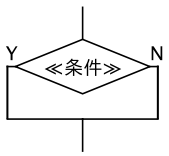
文字列は「」で囲む。複数の値や文字列を ‘と’ でつなぐことができる（例：「x は」 と x を表示する）。



5　選択

```
もし《条件式》ならば
  | 《処理》
を実行する
```

```
もし《条件式》ならば
  | 《処理》
を実行し，そうでなければ
  | 《処理》
を実行する
```



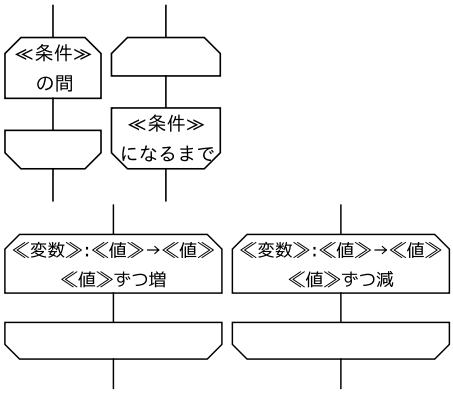
6　繰り返し

```
《条件式》の間，
  | 《処理》
を繰り返す
```

```
繰り返し，
  | 《処理》
を，《条件式》になるまで実行する
```

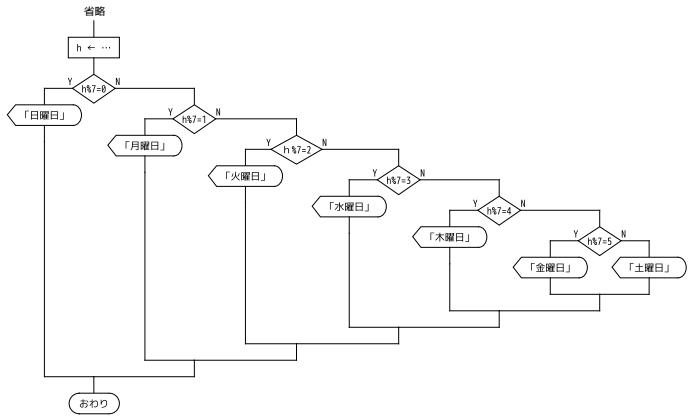
```
《変数》を《値》から《値》まで《値》ずつ増やしながら，
  | 《処理》
を繰り返す
```

```
《変数》を《値》から《値》まで《値》ずつ減らしながら，
  | 《処理》
を繰り返す
```



》第 6 節　配列

前に，Zeller の公式を使って，日付を入力すると曜日を 0〜6 の数字で表示するプログラムを作った。これを「日曜日」「月曜日」...「土曜日」と表示したい。次のようなアルゴリズムでもできるが，決してスマートな方法とはいえない。




```
h ← …
もし h%7=0 ならば
  | 「日曜日」を表示する
を実行し，そうでなければ
  | もし h%7=1 ならば
  |   | 「月曜日」を表示する
  | を実行し，そうでなければ
  |   | もし h%7=2 ならば
  |   |   | 「火曜日」を表示する
  |   | を実行し，そうでなければ
  |   |   | もし h%7=3 ならば
  |   |   |   | 「水曜日」を表示する
  |   |   | を実行し，そうでなければ
  |   |   |   | もし h%7=4 ならば
  |   |   |   |   | 「木曜日」を表示する
  |   |   |   | を実行し，そうでなければ
  |   |   |   |   | もし h%7=5 ならば
  |   |   |   |   |   | 「金曜日」を表示する
  |   |   |   |   | を実行し，そうでなければ
  |   |   |   |   |   | 「土曜日」を表示する
  |   |   |   |   | を実行する
  |   |   |   | を実行する
  |   |   | を実行する
  |   | を実行する
  | を実行する
を実行する
```

xDNCL では番号付きの変数である**配列**を使うことができる。これは変数の宣言時に変数名の後ろに [n] をつけることで（n は負でない整数），変数名の後ろに [0]，[1]，...，[n] をつけた $n+1$ 個の変数が使えるというものだ。これを使うことによって，プログラムがすっきりすることがよくある。たとえばこの曜日のプログラムであれば，

```
文字列 a[6]
a[0] ← 「日曜日」
a[1] ← 「月曜日」
a[2] ← 「火曜日」
a[3] ← 「水曜日」
a[4] ← 「木曜日」
a[5] ← 「金曜日」
a[6] ← 「土曜日」
(中略)
a[h%7] を表示する
```

とすればよい。a[0]〜a[6] に「日曜日」〜「土曜日」を代入するのが多少面倒ではあるが，それでも最初に書いた「もし」の連続に比べれば，行数は半分以下だ。

実は次のようにまとめて代入することもできる。{ }とコンマを忘れないように（カッコは{ }でも [ ] でもよい）。

```
a ← {「日曜日」,「月曜日」,「火曜日」,「水曜日」,「木曜日」,「金曜日」,「土曜日」}
```

たとえば，40 人のクラスで点数入力をするなら次のようにすればいい。合計点などの計算も，配列を使うととても楽だ（これは次の項で扱う）。

```
整数 a[39], b
b を 0 から 39 まで 1 ずつ増やしながら,
  | a[b] を入力する
を繰り返す
```

□ **例題：おみくじ**　乱数を用いて「大吉」「吉」「小吉」「凶」「大凶」のどれかがでるようなおみくじのプログラムを作りなさい。余裕があれば，出るメッセージの種類を増やしてみなさい。

モデル化とシミュレーション

サイコロはそれぞれの目が $\frac{1}{6}$ の確率で出る。だったら 60 回サイコロを振れば，それぞれの目が 10 回ずつ出るだろうか。60 回くらいなら実際に試してみてもいいのだが，後で 600 回，6000 回ならどうなるかということについても実験してみたい。そうなると手作業で数えるのは大変だ。そこで，サイコロの実物を使うのでなく，プログラムでその状況を作り出して実験してみるとしよう。このように，実物を使うことが困難であるようなときに，代用品で同じような状況を作って試してみることを**シミュレーション**(*12)という。

シミュレーションは必ずしもコンピュータを使うものばかりではない。部屋の模様替えをするときに，家具の配置を適当な縮尺の方眼紙の上で考えるのもシミュレーションだし，車や飛行機の開発現場で部品の模型を風洞に入れて空気の流れを計測するのもそうだ。また，物を使うものばかりではなく，計算で行なうものもある（これについてはコンピュータが活躍する場面も多い）。その意味では数学の文章題や理科の計算問題は，実はシミュレーションの一種でもある。

シミュレーションを行なうためには，問題となる状況から必要な要素を抽出して**モデル化**を行なう。この場合だと「サイコロは 6 つの目が等確率で出る」「前に出た目は次に出る目に影響を及ぼさない」「回数を重ねても各目の確率は変化しない」などといったことになるだろう。

□ 課題：サイコロのシミュレーション　サイコロを 60 回振ったら各目が 10 回ずつ出るかどうかを調べるプログラムを作りなさい。プログラムを簡単にするために，サイコロは 0〜5 の目が出るものとして(*13)，60 回サイコロを振ったときに各目の出た回数を数えて表示するプログラムを作ることにする。アルゴリズムは次のようになるだろう。繰り返し用の変数が 1 つ必要になることに注意しよう。

1. a[0]〜a[5] の値をすべて 0 にする。
2. 次の (a)(b) を 60 回繰り返す。
   （a）0〜5 の乱数を b に代入する。
   （b）a[b] の値を 1 増やす。
3. a[0]〜a[5] の値を順に表示する(*14)。

》第 7 節　集計

データの集計をするプログラムをいろいろ作ってみよう。集計というからにはたくさんの数値を扱いたいのだが，それを毎回入力するのも，代入をたくさん繰り返すのも面倒だ。そこで，ここでは決まった値を先に代入しておくことにしよう。

```
整数 a[9]
a ← {59, 34, 24, 99, 38, 1, 29, 89, 81, 6}
```

この節では集計をするプログラムをこれの続きとして作っていく。

1　合計の計算

a[0]〜a[9] の合計を求めるプログラムを考える。もちろん

```
a[0]+a[1]+a[2]+a[3]+a[4]+a[5]+a[6]+a[7]+a[8]+a[9] を表示する
```

とすれば可能ではあるが，これだとデータが 10 個でないときにはプログラムを書きなおさなくてはいけないから，こんなやり方では話にならない。繰り返しを使って合計を求めるのが適切なやり方だ。ここでは合計を s で表すことにする。まず s の値を 0 にし，順次 a[0]，a[1]，...，a[9] を加算していく。これを繰り返しを使わずに書くと

(*12)「シュミレーション」という間違いは非常に恥ずかしいのでしないように。
(*13) 普通のサイコロは 1〜6 であるが，配列の番号が 0 から始まるから 0〜5 にした方が面倒がない。
(*14) 実は「a を表示する」でまとめて表示できる。

```
s ← 0
s ← s + a[0]
s ← s + a[1]
(中略)
s ← s + a[9]
```

となるが，これをそのまま入力するのではなく，繰り返しを使おう。`s ← s + a[b]` を，`b` を 0〜9 の値にして行なえばいい。つまり「`b` を 0 から 9 まで 1 ずつ増やしながら『`s ← s + a[b]`』を繰り返す」わけだ[(*15)]。また最後に結果の表示を追加しよう（この節の後の処理でも同じ）。

2　最大値，最小値

`a[0]`〜`a[9]` の最大値を求める方法を考えよう。ここでは最大値を `m` としよう。アルゴリズムは次のようになる。

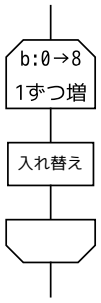
1. `m` の初期値を適切に定める。
2. 次の (a) を，`b` を 0〜9 の値にして行なう。
   （a）`m<a[b]` なら `m ← a[b]`。

やはり「`b` を 0〜9 の値にして…」というのが繰り返しだ。最小値は (a) の不等号を逆向きにすればいい。ところで，1. で定める初期値はどのようにするといいだろうか。

3　ソート

値を順番に並べることを**ソート**という。`a[0]`〜`a[9]` を小さい順に並べるプログラムを作ってみよう。なお，小さい順を「**昇順**」，大きい順を「**降順**」という。アルゴリズムの基本的な考えは次のようになる。

1. `a[0]`〜`a[9]` の最小値が `a[0]` になるように値を入れ替える。
2. `a[1]`〜`a[9]` の最小値が `a[1]` になるように値を入れ替える。
3. `a[2]`〜`a[9]` の最小値が `a[2]` になるように値を入れ替える。
   （以下，最後まで繰り返す）

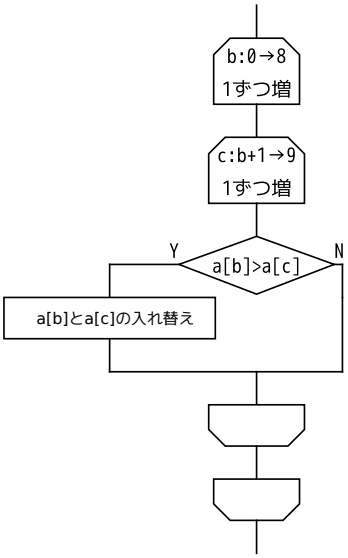


これをもう少し詳しくいうと次のようになるだろう。

1. `a[1]`〜`a[9]` の中で `a[0]` よりも小さいものがあれば，それと `a[0]` を入れ替える。
2. `a[2]`〜`a[9]` の中で `a[1]` よりも小さいものがあれば，それと `a[1]` を入れ替える。
3. `a[3]`〜`a[9]` の中で `a[2]` よりも小さいものがあれば，それと `a[2]` を入れ替える。
   （以下，最後まで繰り返す）

1.,2.,... をまとめて言うと次のようになる（右上図の「入れ替え」と書かれている部分がこれ）。`b` を 0〜8 にしてこれを行えばいい：

> **`a[b+1]`〜`a[9]` の中で `a[b]` よりも小さいものがあれば，それと `a[b]` を入れ替える。**



この「入れ替え」をさらに言い換えると次のようになる。

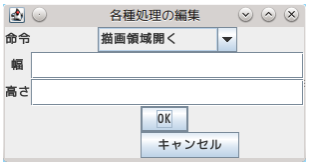
1. `c` を `b+1`〜9 にして，(a) を行なう。
   （a）もし `a[b]>a[c]` なら，`a[b]` と `a[c]` を入れ替える。

では，実際にこれをプログラムにしてみよう。なお，変数の入れ替えにはもう一つ変数を用意する必要があることを，ずっと前に学習した（5 ページの問題）。

(*15) 数学だと $\sum_{b=0}^{9} a_b$ のような式に相当する。

》第 8 節　グラフィック

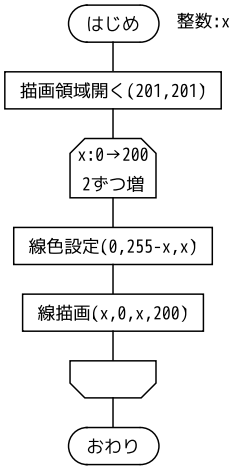
1　グラフィックの基本



PenFlowchart や WaPEN にはグラフィックの命令があるので，それを使ってみよう。パーツの「各種処理」をフローチャートに追加して，編集で「描画領域開く」を選択すると図のようなダイアログが開く。

これを実行すると，ここで指定したサイズの描画用ウィンドウが開かれる。その他の描画用命令もすべて「各種処理」にあるので自由に探してみてほしい。座標は左上が $(0,0)$ で，x 軸が右，y 軸が下に伸びていると考えればいい。

次のプログラムでは縦の直線が 101 本引かれ，緑から青へのグラデーションになる。隙間をあけて線を引いているが，もちろん隙間なしになるように変えてみてもいい。




```
整数 x
描画領域開く (200,200)
x を 0 から 200 まで 2 ずつ増やしながら,
  ┃線色設定 (0,255-x,x)
  ┃線描画 (x,0,x,200)
を繰り返す
```

次のプログラムでは，自由な位置に自由な大きさの円を 100 個描く。「線色設定」を使えば色を変えるなどの発展も考えられるだろう。その場合は，色の赤，青，緑の数字はそれぞれ 0〜255 の範囲にすること。

```
整数 x,y,r,i
描画領域開く (200,200)
i を 1 から 100 まで 1 ずつ増やしながら,
  ┃ x ← random(200)
  ┃ y ← random(200)
  ┃ r ← random(50)
  ┃円描画 (x,y,r)
を繰り返す
```

これを応用して，いろいろな模様を描いてみよう。なお，PenFlowchart では

描画領域保存 (「CG.PNG」,「PNG」)

を最後に入れておけば，CG.PNG というファイル名で画像を保存できる。WaPEN の場合は画像を右クリックして，名前をつけてファイルに保存すればよい。

2　サンプルプログラム

次のプログラムは同心円の色を徐々に変えてグラデーションを表現してみたものである。

```
整数 i
描画領域開く (200,200)
i を 100 から 1 まで 1 ずつ減らしながら,
  ┃線色設定 (2*i,0,200-2*i)
  ┃円描画 (100,100,i)
を繰り返す
```

次のプログラムは直線だけを描画しているのだが，曲線が現れる。これがなんという曲線であるかは...。

```
整数 i
描画領域開く (200,200)
i を 1 から 200 まで 1 ずつ増やしながら,
  |線色設定 (0,i,200-i)
  |線描画 (i,0,0,200-i)
を繰り返す
```

1=1 は常に成り立つので，「1=1 の間」というのは無限ループになる。止めるには，PenFlowchart なら「初めに戻る」，WaPEN なら「リセット」のボタンを押す。

```
整数 x,y,w,h,i
描画領域開く (200, 200)
1=1 の間,
  |描画領域全消去 ()
  |i を 1 から 100 まで 1 ずつ増やしながら,
  |  |x ← random(200)
  |  |y ← random(200)
  |  |w ← random(50)+1
  |  |h ← random(30)+1
  |  |矩形描画 (x, y, w, h)
  |を繰り返す
を繰り返す
```

次のプログラムは長いが，配列 a に代入した値を棒グラフで表す。max に 0 でなく 1 を代入しているのは，a の値が全部 0 であってもエラーにならないようにするためである。

```
整数 a[4],i,h,w,x,y,max,wb,hb
h ← 400
w ← 400
a ← {100,240,220,150,180}
max ← 1
i を 0 から 4 まで 1 ずつ増やしながら,
  |もし a[i]>max ならば
  |  |max ← a[i]
  |を実行する
を繰り返す
描画領域開く (w,h)
x ← w*0.1
y ← h*0.9
w ← w*0.8
h ← h*0.8
wb ← w/5
線描画 (x,y,x+w,y)
線描画 (x,y,x,y-h)
塗色設定 (0,255,0)
文字サイズ設定 (16)
i を 0 から 4 まで 1 ずつ増やしながら,
  |hb ← a[i]*h/max
  |矩形塗描画 (x+wb*0.1,y-hb,wb*0.8,hb)
  |文字描画 (a[i],x+wb*0.1+16,y-hb-2)
  |文字描画 (i+1+「番目」,x+wb*0.1,y+16)
  |x ← x+wb
を繰り返す
```