1　プログラムとは

コンピュータが行う作業—ゲームやワープロなどすべて—は，プログラムというもので指示された手順に従って実行されます。すなわち，プログラムはコンピュータにやらせたい作業の手順を細かに記述したものです。

1.1　油分け算

塵劫記[†1)] に載っている**油分け算**というパズルを解くプログラムを例にとって説明しましょう。

1.1.1　問題

大きな甕に油が入っています。油を量るためのマスが 8 合マス[†2)] と 5 合マスの 2 つあります。これを使って油を 4 合量るにはどうしたらよいでしょうか。

1.1.2　方針

次の 3 つの動作をうまく組み合わせるとできます。

- 甕から 8 合マスに汲む。
- 8 合マスから 5 合マスに移す。
- 5 合マスから甕に戻す。

1.1.3　解答

		マス１	マス２
		0	0
1 回	かめ　→ マス１	8	0
2 回	マス１ → マス２	3	5
3 回	マス２ → かめ	3	0
4 回	マス１ → マス２	0	3
5 回	かめ　→ マス１	8	3
6 回	マス１ → マス２	6	5
7 回	マス２ → かめ	6	0
8 回	マス１ → マス２	1	5
9 回	マス２ → かめ	1	0
10 回	マス１ → マス２	0	1
11 回	かめ　→ マス１	8	1
12 回	マス１ → マス２	4	5
13 回	マス２ → かめ	4	0

[†1)] 江戸時代初期に吉田光由が著した日本最初の数学書。

[†2)] 1 合 = 1/10 升 = 180ml = 炊飯器についてくる米のカップ 1 杯分の量 です。

8 合マスと 5 合マスを使って目標の 4 合を量りとるという問題を解くだけならば，上の解答を示せば十分ですが，一般に「2 つのマスを使って目標の量を量りとる」という問題を解きたいですね。

それは，つぎのようにしてできます。

1.1.4　解答を求める手順

(1) マス１の容量を定める。

(2) マス２の容量を定める。

(3) 目標の量を定める。

(4) マス１を空にする。

(5) マス２を空にする。

(6) 次のことを繰り返す。

　(6.1) マス１が空ならば，甕からマス１に汲む。

　(6.2) マス１が空でないとき，

　　(6.2.1) マス２が満杯ならば，マス２から甕に戻す。

　　(6.2.2) マス２が満杯でないとき，マス１からマス２に移せるだけ移す。すなわち，

　　　(6.2.2.1) マス１に入っている量がマス２の空き量以下ならば，全部マス２に移す。

　　　(6.2.2.2) マス１に入っている量がマス２の空き量より多ければ，空き量分だけ移す。

(7) 2 つのマスに入っている合計が目標の量になったら終了する。

問 1.1 上の解答は，8 合マスで汲んで 5 合マスで戻す動作を繰り返しました。逆に，5 合マスで汲んで 8 合マスで戻したら何回の動作でできるでしょうか。マス１の容量を 5，マス２の容量を 8，目標量を 4 として，この手順に従って求めてください。

		マス１	マス２
		0	0
1 回　かめ	→ マス１	5	0

1.1.5　改良

たとえば 8 合マスと 6 合マスでは偶数合の量しか量れないので，目標量を 1 合にすると一連の動作を永久に繰り返してしまいます。2 つのマスが両方とも満杯になると，次にマス 2 から甕に戻して，マス 1 が満杯でマス 2 が空の状態になります。これは最初に甕からマス 1 に汲んだ状態と同じです。ですから，両方とも満杯になったらこの問題は不可能だということがわかります。

最後の終了条件を次のように変更すると，不可能な問題の場合も永久に繰り返すことがなくなります。

(7) 2 つのマスに入っている合計が目標の量または容量の和になったら終了する。

1.1.6　プログラム

この手順を，コンピュータが理解できる言語で書いたものがプログラムです。この授業では Pascal というプログラミング言語でプログラムを書きます。

1.2　実行

油分け算のプログラムを実行して，コンピュータがどのように振る舞うか見ることにします。

1.2.1　プログラムをコピー

プログラム `Aburawake.dpr` が `T:\ooya` [†3] に入っています。次のようにして `Z:\Programming1` にコピーして使います。

(1) マイコンピュータを起動する。

(2) `Z:` ドライブを表示する。

(3) ［ファイル｜新規作成］でフォルダを作成して，フォルダ名を `Programming1` にする。

(4) `T:` ドライブの `ooya` フォルダを表示する。

(5) `Aburawake.dpr` を右クリックして「コピー」をクリックする。

(6) `Z:\Programming1` を右クリックして「貼付け」をクリックする。

(7) マイコンピュータを終了する。

1.2.2　プログラムを読み込む

(1) Delphi（デルファイ） (Delphi5 または Delphi7) を起動する。

(2) ［ファイル｜プロジェクトを開く］で `Z:\Programming1\Aburawake.dpr` を開く。

(3) 1 行目の `//` の後に学生証番号と氏名を書く[†4]。

[†3] \ は，日本では ¥になります。
[†4] 今後，プログラムを作るたびに，同様に 1 行目の後に // に続けて学生証番号と氏名を書くようにしてください。

1.2.3　実行する

(1) F9 キーを押して実行する。

(2) 実行結果を確認したら，Enter キーを押して終了する。

問 1.1 の解答が出力されます。正しくできていましたか。

1.2.4　監視式を追加する

変数の値を監視するために，変数を監視式に追加します。

(1) Ctrl+F5 を押す（Ctrl キーをおさえながら F5 キーを押す）。

(2) 「監視式のプロパティ」というダイアログが現れるので，変数名 `Youryou1` を入れる。

(3) (1),(2) を繰り返して，すべての変数を監視式に追加する。

1.2.5　ステップ実行する

1 ステップずつ実行して，変数やコンソール画面の変化を調べます。

(1) F8 キーを押す。
コンソール画面（黒い画面）が現れ，プログラムの実行部の先頭（`begin`）の行の色が変わる。
コンソール画面，プログラムの画面，監視式一覧の画面が同時に見えるように，位置，サイズを調節する。オブジェクトインスペクタの画面は必要ないので，右上の×をクリックして消す。

(2) F8 キーを押して，1 ステップずつ実行する。

(3) (2) を繰り返す。

(4) 最後の `ReadLn;` を実行すると，コンソール画面で入力待ちになるので，Enter キーを押す。

問 1.2 1 ステップ実行するたびに，つぎのことを調べなさい。
実行する前に結果を推理して，実行後それが正しいことを確認しましょう。正しくなかったら，なぜ間違えたか反省して次は正しく推理できるようにしましょう。

(1) `:=` を含む行を実行するとき，監視式一覧の変数の値がどう変わるか。

(2) `if` で始まる行を実行するとき，次にどの行に進むか。

(3) `until` で始まる行を実行するとき，次にどの行に進むか。

(4) `Write` で始まる行を実行するとき，コンソール画面に何が出力されるか。

問 1.3 マス 1 の容量，マス 2 の容量，目標量をいろいろ変えて実行しなさい。

1.3　解説

1.3.1　代入文

変数 := **式**	右辺の**式**を計算して，その値を左辺の**変数**に代入する。

四則演算子

演算	演算子	例	値
足し算	+	75 + 8	83
引き算	-	75 - 8	67
掛け算	*	75 * 8	600
割り算	/	75 / 8	9.375
整数の割り算（商）	div	75 div 8	9
（余り）	mod	75 mod 8	3

1.3.2　if 文

if **条件** then begin **文**$_{t,1}$; ⋮ **文**$_{t,k}$; end else begin **文**$_{e,1}$; ⋮ **文**$_{e,m}$; end	**条件**の真偽を計算して，真（True）のとき then の後の **文**$_{t,1}$;…;**文**$_{t,k}$; を実行し，偽（False）のとき else の後の **文**$_{e,1}$;…;**文**$_{e,m}$; を実行する。

比較演算子

数学の書き方	$=$	$\neq$	$<$	$>$	$\leqq$	$\geqq$
Pascal の書き方	=	<>	<	>	<=	>=

論理演算子

演算	演算子	例	意味
論理積	and	(10 <= N) and (N < 100)	$10 \leqq N < 100$
論理和	or	(N = 2) or (N = 7)	N は 2 または 7 である
否定	not	not (N = 5)	N は 5 でない （N <> 5 と同じ）

1.3.3　repeat 文

repeat 　**文**$_1$; 　　⋮ 　**文**$_k$; until **終了条件**	**文**$_1$;...;**文**$_k$; を実行した後で，**終了条件**の真偽を計算して，真ならば終了する。偽ならば，repeat まで戻って繰り返す。

1.3.4　出力文

(1) Write(**データ**$_1$, ... , **データ**$_k$)	(1) **データ**$_1$ ... **データ**$_k$ を書く。書いた後改行しない。
(2) WriteLn	(2) 改行する。
(3) WriteLn(**データ**$_1$, ... , **データ**$_k$)	(3) **データ**$_1$ ... **データ**$_k$ を書く。書いた後改行する。

データ

’ ’でくくってある場合，くくられている文字列を書きます。

```
Write('32+65*2');                32+65*2
```

’ ’でくくってない場合は，式[†5)] を計算してその値を書きます。

```
Write(32+65*2);                  162
```

データの後に :**桁数** をつけると，全体の桁数が**桁数**になるように左に空白を足して書きます。

```
WriteLn('123456789');            123456789
WriteLn(32+65*2:7);                  162
WriteLn(3*5>3+5:7);                 true
WriteLn(92 mod 3:7);                   2
```

`Write` と `WriteLn`

`Write(データ)` は，データを書いた後に改行しません。

```
Write('32+65*2=');               32+65*2=162
Write(32+65*2);
```

`WriteLn(データ)` は，データを書いた後に改行します。

```
WriteLn('32+65*2=');             32+65*2=
WriteLn(32+65*2);                162
```

[†5)] 変数 1 個だけの単項式もあります。