



# pe-2. 計算

(Pascal プログラミング入門)

URL: <https://www.kkaneko.jp/cc/pascal/index.html>

金子邦彦





```
program sum;
var start_x, step_x, x, y:real;
var i:integer;
begin
  write('Please Enter start_x:');
  readln(start_x);
  write('Please Enter step_x:');
  readln(step_x);
  for i:=1 to 20 do
  begin
    x := start_x + ( i * step_x );
    y := sin(x);
    writeln('sin(', x:8:3, ') =', y:8:3);
  end;
  readln
end.
```

キーボードからの  
データの読み込みを  
行っている部分

計算の繰り返しを  
行っている部分

画面へのデータの  
書き出しを行って  
いる部分

# 例題 1 のプログラム実行結果 (例)



```
main.pas
1 program sum;
2 var start_x, step_x, x, y:real;
3 var i:integer;
4 begin
5   write('Please Enter start_x:');
6   readln(start_x);
7   write('Please Enter step_x:');
8   readln(step_x);
9   for i:=1 to 20 do
10    begin
11      x := start_x + ( i * step_x );
12      y := sin(x);
13      writeln('sin( ', x:8:3, ' ) = ', y:8:3);
14    end;
15   readln
16 end.
```

キーボードから、データ「0.4」と「0.1」を読み込んでいる

```
Copyright (c) 1993-2017 by Florian Klaempfl and others
Target OS: Linux for x86-64
Compiling main.pas
Linking a.out
/usr/bin/ld.bfd: warning: link.res contains output sect
17 lines compiled, 0.1 sec
Please Enter start_x:0.4
Please Enter step_x:0.1
sin( 0.500) = 0.479
sin( 0.600) = 0.565
sin( 0.700) = 0.644
sin( 0.800) = 0.717
sin( 0.900) = 0.783
sin( 1.000) = 0.841
sin( 1.100) = 0.891
sin( 1.200) = 0.932
sin( 1.300) = 0.964
sin( 1.400) = 0.985
sin( 1.500) = 0.997
sin( 1.600) = 1.000
sin( 1.700) = 0.992
sin( 1.800) = 0.974
sin( 1.900) = 0.946
sin( 2.000) = 0.909
sin( 2.100) = 0.863
sin( 2.200) = 0.808
sin( 2.300) = 0.746
sin( 2.400) = 0.675
```

計算を 20 回繰り返して、計算結果を表示している

# 例題 1 のプログラムの機能



1. キーボードからのデータの読み込み  
次の2つの値を読み込む

start\_x, step\_x

2. 計算の繰り返し

sin( x ) の計算を 20 回繰り返す

x = start\_x + step\_x,

start\_x + 2 × step\_x,

...

start\_x + 20 × step\_x

} 20回分

3. 画面へのデータの書き出し

計算した sin( x ) の値を書き出す

# これ以降の内容



## 例題 1. 自由落下距離

- 四則演算

## 例題 2. 三角形の面積

- 浮動小数点数の変数, 入力文, 出力文, 代入文
- write 文と writeln 文の違い
- 「プログラムの終わりのreadln文」の機能

## 例題 3. sin 関数による三角形の面積

## 例題 4. ライブラリ関数

- 種々のライブラリ関数

# 目標



- 自分の思い通りの計算ができるようになる
  - 四則演算
  - ライブラリ関数 (三角関数, 対数・指数関数など)
- 見やすいプログラムを書くために, 字下げを行う



- **プログラミングを行えるオンラインのサービス**

<https://www.onlinegdb.com>

- **ウェブブラウザを使う**

- **たくさんの言語を扱うことができる**

Pascal, Python3, Java, C/C++, C#, JavaScript,  
R, アセンブリ言語, SQL など

- **オンラインなので、「秘密にしたいプログラム」  
を扱うには十分な注意が必要**

# Online GDB で Pascal を動かす手順



① ウェブブラウザを起動する

② 次の URL を開く

<https://www.onlinegdb.com>

A screenshot of a web browser's address bar. The address bar is a light gray rectangle with a thin border. On the left side, there is a magnifying glass icon. To the right of the icon, the text "https://www.onlinegdb.com" is displayed in a light gray font. Below the address bar, there is a thin horizontal line, and below that, a larger light gray rectangular area representing the browser's content area.





### ③ 「Language」 のところで、「Pascal」 を選ぶ

The screenshot shows the GDB Online web interface. At the top, there is a navigation bar with buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. A 'Language' dropdown menu is open, showing a list of programming languages. The 'Pascal' option is highlighted with a red box. The main area displays source code for a C program that prints 'Hello World'.

```
1 - /*****  
2  
3 Welcome to GDB OnLine.  
4 GDB online is an online compiler and debugger tool for C, C++, Python  
5 C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17
```



## 実行ボタン

The screenshot shows the GDB Online IDE interface. At the top, there is a toolbar with several buttons: Run (green), Debug (blue), Stop (red), Share (orange), Save (blue), Beautify (blue), and a download icon. The 'Run' button is highlighted with a red box. Below the toolbar is the editor area, which contains the following Pascal code:

```
1 {  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python, Java,  
5 C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS,  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 }  
9 program Hello;  
10 begin  
11     writeln ('Hello World')  
12 end.  
13
```

Below the editor is the console output, which shows the compilation and execution process:

```
input  
Copyright (c) 1993-2017 by Florian Klaempfl and others  
Target OS: Linux for x86-64  
Compiling main.pas  
Linking a.out  
/usr/bin/ld.bfd: warning: link.res contains output sections; did you forget -T?  
12 lines compiled, 0.1 sec  
Hello World  
...Program finished with exit code 0  
Press ENTER to exit console.
```

## エディタ画面

プログラムを  
書き換えること  
ができる

# 例題 1 . 自由落下距離



- **自由落下距離を求めるプログラムを作る**
  - 地上で物を落とし始めた後の自由落下距離を求める
  - 重力加速度  $g$  は  $9.8$  とする
  - 自由落下距離を求めるために、プログラム中に、計算式  $y := (9.8 / 2.0) * x * x$  を書く



```
program sum;
var start_x, step_x, x, y:real;
var i:integer;
begin
  write('Please Enter start_x:');
  readln(start_x);
  write('Please Enter step_x:');
  readln(step_x);
  for i:=1 to 20 do
  begin
    x := start_x + ( i * step_x );
    y := ( 9.8 / 2.0 ) * x * x;
    writeln('x=', x:8:3, ', y=', y:8:3);
  end;
  readln
end.
```

キーボードからのデータの読み込みを行っている部分

計算の繰り返しを行っている部分

自由落下運動の式

画面へのデータの書き出しを行っている部分

# 実行結果例



```
10 lines compiled, 0.1 s
Please Enter start_x:0
Please Enter step_x:0.1
x= 0.100, y= 0.049
x= 0.200, y= 0.196
x= 0.300, y= 0.441
x= 0.400, y= 0.784
x= 0.500, y= 1.225
x= 0.600, y= 1.764
x= 0.700, y= 2.401
x= 0.800, y= 3.136
x= 0.900, y= 3.969
x= 1.000, y= 4.900
x= 1.100, y= 5.929
x= 1.200, y= 7.056
x= 1.300, y= 8.281
x= 1.400, y= 9.604
x= 1.500, y= 11.025
x= 1.600, y= 12.544
x= 1.700, y= 14.161
x= 1.800, y= 15.876
x= 1.900, y= 17.689
x= 2.000, y= 19.600
```

# 四則演算のための演算子



+ 和

- 差

\* 積

/ 商

## 例題 2. 三角形の面積



- **底辺と高さを読み込んで、面積を計算するプログラムを作る**

例) 底辺が 2.5, 高さが 5 のとき,  
面積 : 6.25

- **底辺, 高さ, 面積を扱うために、浮動小数点数の変数を 3 つ使う**



```
program sum;  
var teihen: real;  
var takasa: real;  
var menseki: real;  
begin
```

```
write('Please Enter teihen: ');  
readln(teihen);  
write('Please Enter takasa: ');  
readln(takasa);  
menseki := teihen*takasa*0.5;  
writeln('menseki =', menseki:8:3);
```

```
readln
```

```
end.
```

キーボードからの  
データの読み込みを  
行っている部分

## 三角形の面積の式

画面へのデータの  
書き出しを行ってい  
る部分



# 実行結果画面 (例)



```
Please Enter teihen: 2.5  
Please Enter takasa: 5  
menseki = 6.250
```

# プログラム実行順



```
write('Please Enter teihen: ');
```

メッセージ

「 'Please Enter teihen: 」を表示

```
readln(teihen);
```

浮動小数点数データを読み込み

```
write('Please Enter takasa: ');
```

メッセージ

「 'Please Enter takasa: 」を表示

```
readln(takasa);
```

浮動小数点数データを読み込み

```
menseki := teihen * takasa * 0.5;
```

計算

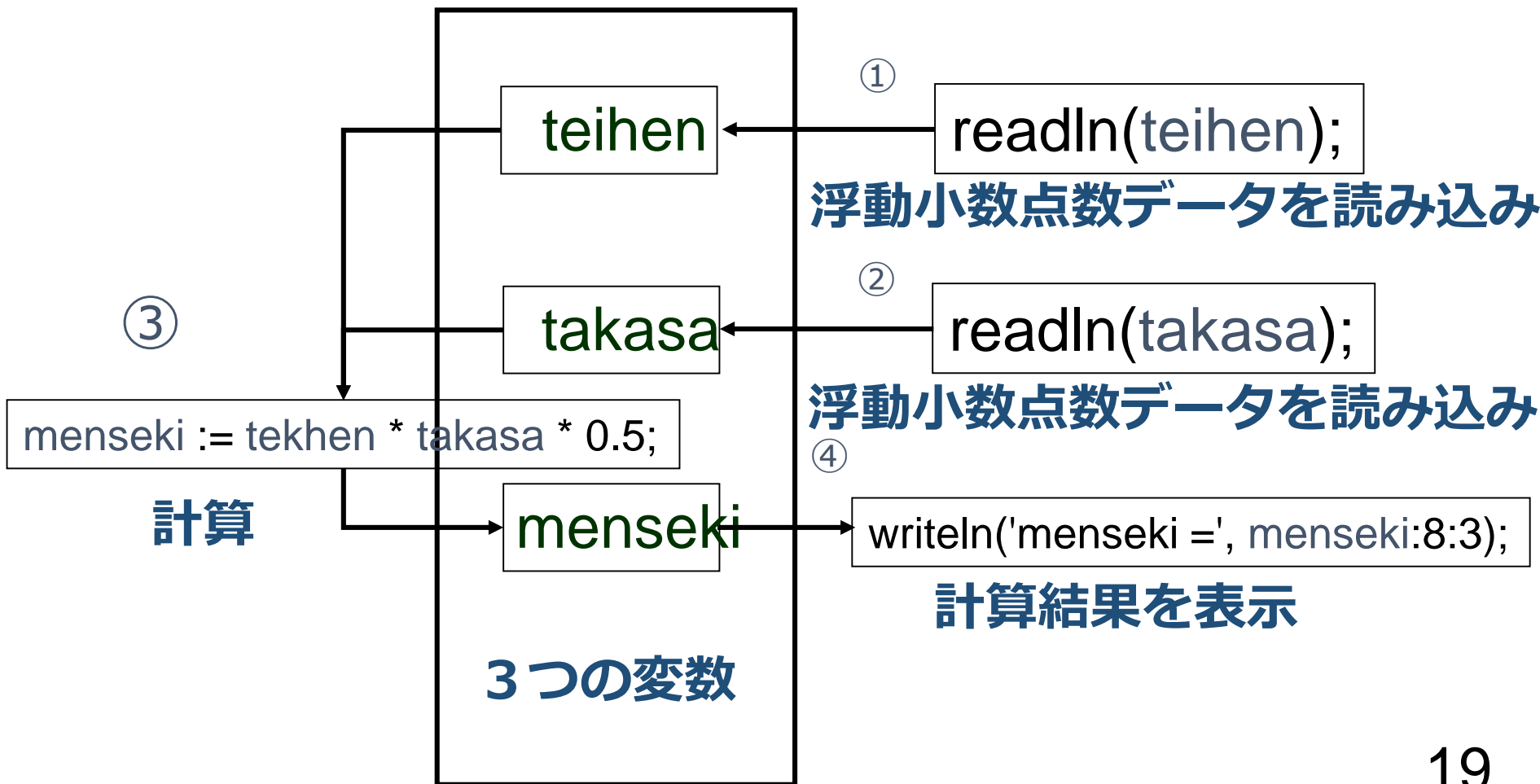
```
writeln('menseki =', menseki:8:3);
```

計算結果を表示

```
readln
```

終わり

## メモリ



# 変数宣言



- **変数**は、データを入れるためのメモリ
- **変数宣言**とは、変数を使うために、名前と型を書いて、変数の使用をコンピュータに伝えること

```
var teihen: real;
```

} 浮動小数点数データで、変数名は「teihen」

```
var takasa: real;
```

} 浮動小数点数データで、変数名は「takasa」

```
var menseki:
```

} 浮動小数点数データで、変数名は「menseki」

```
real;
```

「real」とは、浮動小数点数データ  
という意味。

# 代入文



- 計算結果 ( $\text{teihen} * \text{takasa} * 0.5$ ) を, 変数 `menseki` に格納する (このことを, 代入という)

「`:=`」は, 代入の意味

```
menseki := teihen*takasa*0.5;
```

# 入力, 出力とは



- 入力
  - データの読み込み
  - (読み込まれたデータは変数に格納される)
  - readln 文を使う
- 出力
  - メッセージの表示
  - データの表示
  - (変数に格納されたデータが表示される)
  - 式の計算結果の表示 (例題 4 参照)
  - write文あるいは writeln文を使う

# readln



```
program sum;
var teihen: real;
var takasa: real;
var menseki: real;
begin
  write('Please Enter teihen: ');
  readln(teihen);
  write('Please Enter takasa: ');
  readln(takasa);
  menseki := teihen*takasa*0.5;
  writeln('menseki =', menseki:8:3);
  readln
end.
```

キーボードの「Enter」キー  
を待つ

# 入力文



- **入力文**は、**データを読み込む**ための文
- **読み込むべき変数名を指定**

```
readln(teihen);
```

読み込むべき変数名



# 出力文



- **出力文**は、**データやメッセージを表示**するための文
- メッセージ部分と、変数部分を「, 」で区切る
- 変数に、書式を付ける  
書式「: 8 : 3」の意味： 8桁で、小数点以下3桁表示

```
writeln('menseki =', menseki:8:3);
```

メッセージ      表示すべき変数名      書式

# いろいろな出力



```
write('Please Enter teihen: ');
```

メッセージ 「Please Enter  
teihen:」 の表示 (改行無し)

```
writeln('sin(', x:8:3, ') =', y:8:3);
```

「sin( 0.800) = 0.717」  
のように、メッセージと変数の  
中身を並べて表示 (改行あり)

# write と writeln の違い



- 機能は同じ
- 使い方も同じ
- 表示の後に改行するかしないかだけが違う

## 例題 3. sin 関数による三角形の面積



- 三角形の2辺の長さ  $a$ ,  $b$  とその挟角  $\theta$  を読み込んで、面積  $S$  を計算するプログラムを作る
  - 面積を求めるために、sin関数を使う
  - 円周率 $\pi=3.14159$  とする

$$S = \frac{1}{2} ab \sin \theta$$



```
program sum;  
var a: real;  
var b: real;  
var theta: real;  
var S: real;  
begin  
  write('Please Enter a: ');  
  readln(a);  
  write('Please Enter b: ');  
  readln(b);  
  write('Please Enter theta: ');  
  readln(theta);  
  S := 0.5 * a * b * sin( theta * 3.14159 / 180.0 );  
  writeln('S =', S:8:3);  
  readln  
end.
```

キーボードからの  
データの読み込みを  
行っている部分

三角形の面積の式

画面へのデータの  
書き出しを行って  
いる部分

# 実行結果例



```
Please Enter a: 10  
Please Enter b: 6  
Please Enter theta: 30  
S = 15.000
```

## 例題 4. ライブラリ関数



- 浮動小数点数データ  $x$  (但し  $0 < x < 1$ ) を読み込んで、次の計算を行うプログラムを作る
  - 指数, 対数, 平方根
  - 三角関数
  - 絶対値



```
program sum;
uses Math;
var x: real;
begin
  write('Please Enter x: ');
  readln(x);
  writeln('exp(x) =', exp(x):8:3);
  writeln('Ln(x) =', Ln(x):8:3);
  writeln('sqrt(x) =', sqrt(x):8:3);
  writeln('ArcCos(x) =', ArcCos(x):8:3);
  writeln('ArcSin(x) =', ArcSin(x):8:3);
  writeln('ArcTan(x) =', ArcTan(x):8:3);
  writeln('cos(x) =', cos(x):8:3);
  writeln('sin(x) =', sin(x):8:3);
  writeln('tan(x) =', tan(x):8:3);
  writeln('abs(x) =', abs(x):8:3);
  readln
end.
```



# 実行結果例



```
Please Enter x: 0.5  
exp(x) = 1.649  
Ln(x) = -0.693  
sqrt(x) = 0.707  
ArcCos(x) = 1.047  
ArcSin(x) = 0.524  
ArcTan(x) = 0.464  
cos(x) = 0.878  
sin(x) = 0.479  
tan(x) = 0.546  
abs(x) = 0.500
```

# 出力文と計算式の組み合わせ



```
writeln('sqrt(x) =', sqrt(x):8:3);
```

```
sqrt(x) = 0.707
```

のように、メッセージと計算結果  
を並べて表示

→ 「:8:3」は、計算結果を8桁で、  
小数点以下3桁表示という意味



# ライブラリ関数

- 指数, 対数, 平方根
  - exp 指数関数 (eのz乗)
  - Ln 対数関数 (底をeとする自然対数)
  - sqrt 平方根
  - Power(x,y) べき乗 (xのy乗)
- 三角関数
  - ArcCos 逆コサイン
  - ArcSin 逆サイン
  - ArcTan 逆タンジェント
  - cos コサイン
  - sin サイン
  - tan タンジェント
- 絶対値
  - abs 絶対値

# いろいろな計算



```
y := sin( x );
```

$\sin x$  を計算し,  $y$  に格納

```
y := sqrt( x );
```

$\sqrt{x}$  を計算し,  $y$  に格納

```
d := sqrt( ( x * x ) + ( y * y ) );
```

$\sqrt{x^2 + y^2}$  を計算し,  $d$  に格納

```
x := sqrt( a * ( a - b ) * ( a - c ) );
```

$\sqrt{a(a-b)(a-c)}$  を計算し,  $x$  に格納

# 演習 1 . Heron の公式



- **三角形の3辺の長さ a, b, c を読み込んで, 面積 A を計算するプログラムを作りなさい.**
  - Heronの公式を用いること

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = (a+b+c)/2$$