

# pe-4. 整数データと浮動小数点 数データ

(Pascal プログラミング入門)

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

金子邦彦



# 内容



- 例題 1. 単純な金種計算
- 例題 2. 硬貨の金種計算
- 例題 3. うるう年の判定

整数の変数

浮動小数点数と整数の違い

- 例題 4. 複利計算

整数の変数と、浮動小数点数の変数を混在させるときに気を付けねばならないこと

# 目標



- プログラムでの**整数と浮動小数点数**の違いについて理解する
- 目的に応じて、**整数の変数**、**浮動小数点数の変数**を**正しく使い分ける**ことができるようになる

- 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 search bar with a magnifying glass icon on the left and the text "https://www.onlinegdb.com" entered inside. The search bar is set against a light gray background.

🔍 <https://www.onlinegdb.com>

### ③ 「Language」 のところで、「Pascal」 を選ぶ

SPONSOR Slack — Bring your team together with Slack, the collaboration hub for work.

Run Debug Stop Share Save { } Beautify Language -- select --

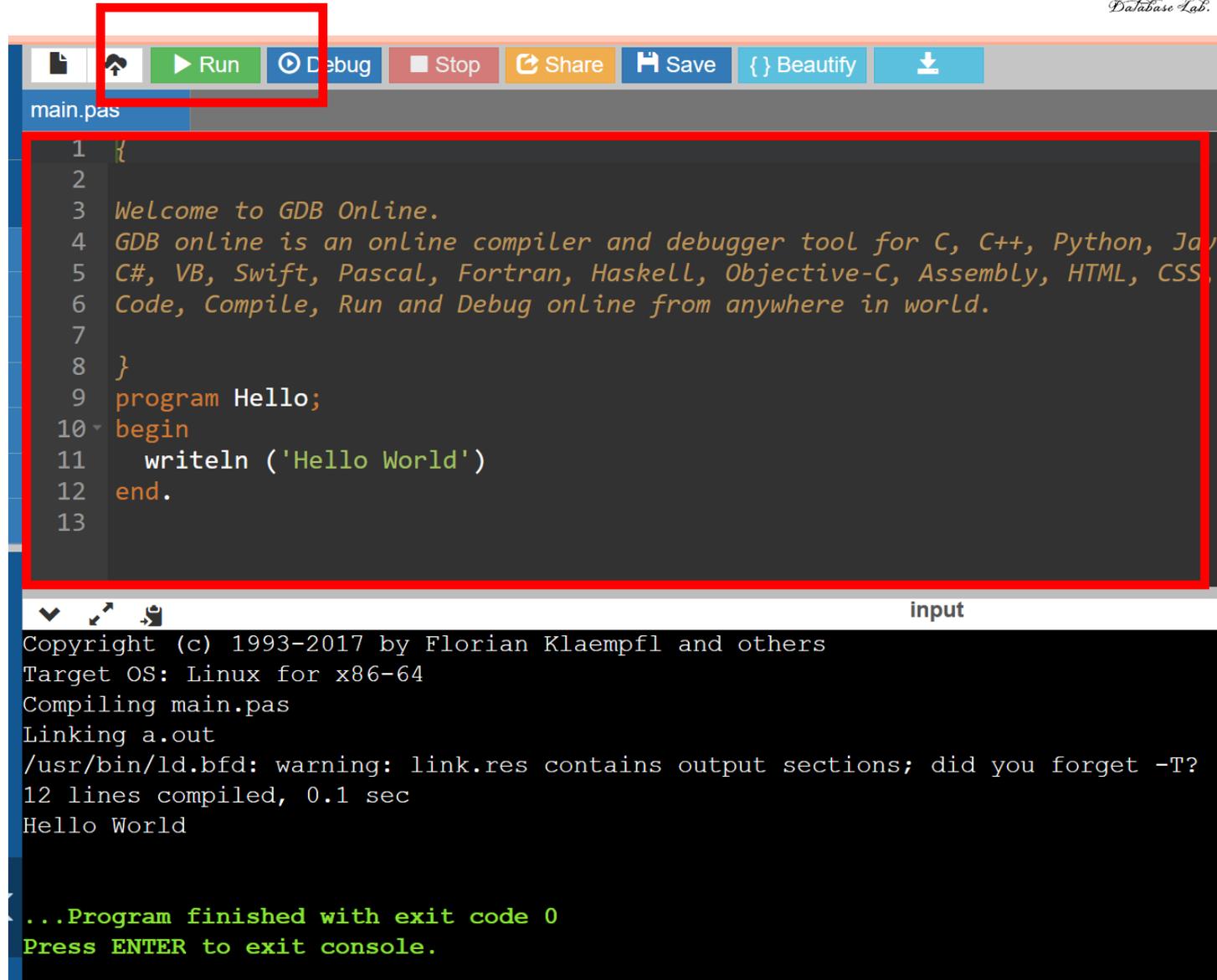
```
1 /*****  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python  
5 C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int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

-- select --  
C  
C++  
C++ 14  
C++ 17  
Java  
Python  
PHP  
C#  
VB  
HTML,JS,CSS  
Ruby  
Perl  
Pascal  
R  
Fortran  
Haskell  
Assembly(GCC)  
Objective C  
SQLite

# 実行ボタン

エディタ画面

プログラムを  
書き換えること  
ができる



```
main.pas
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
```

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.
```

# 例題 1 . 単純な金種計算

- **金額を読み込んで、適切な紙幣と小銭の枚数を求め、表示するプログラムを作る。**

**例) 金額が 1 0 5 0 円するとき、**

**千円札 : 1 枚**

**1 円玉 : 5 0 枚**

- 例題では、簡単のため、紙幣は**千円札のみ**、硬貨は**1 円玉のみ**（種別は考えない）ことにする
- 金額、千円札の枚数、1 円玉の枚数を扱うために、**整数の変数**を使う

- 1000円札の枚数  
金額を1000で割った商（小数点以下切り捨て）
  
- 1円玉の枚数  
金額を1000で割った余り

```
program sum;  
var kingaku, sen, en: integer;  
begin
```

キーボードからの  
データの読み込みを  
行っている部分

```
write('kingaku ?: ');  
readln(kingaku);
```

```
sen := kingaku div 1000;  
en := kingaku mod 1000;
```

計算を  
行っている部分

```
writeln('1000 en =', sen:8, ' mai');  
writeln('1 en = ', en:8, ' mai');
```

画面へのデータの  
書き出しを行ってい  
る部分

```
readln
```

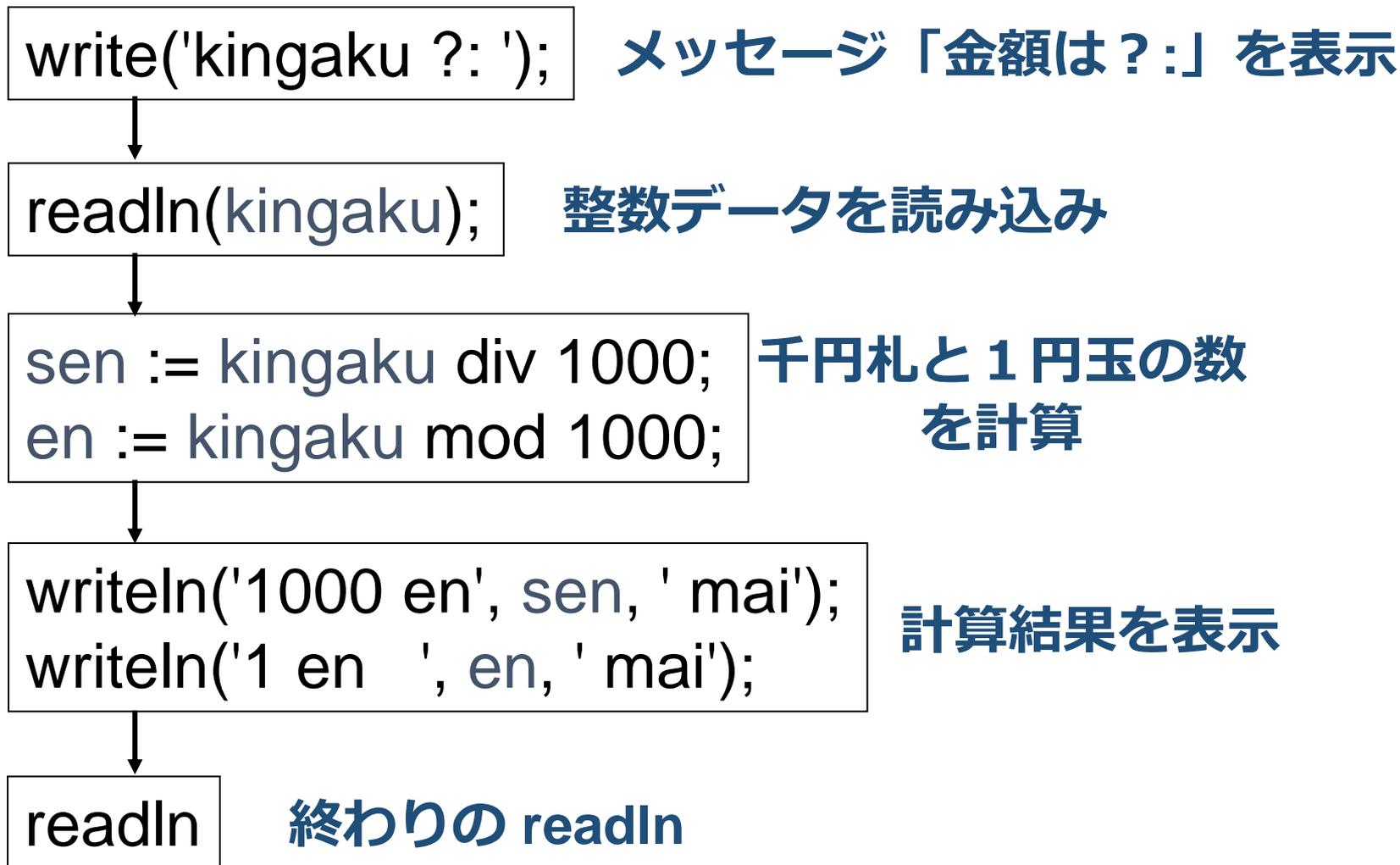
```
end.
```

## 実行結果の例

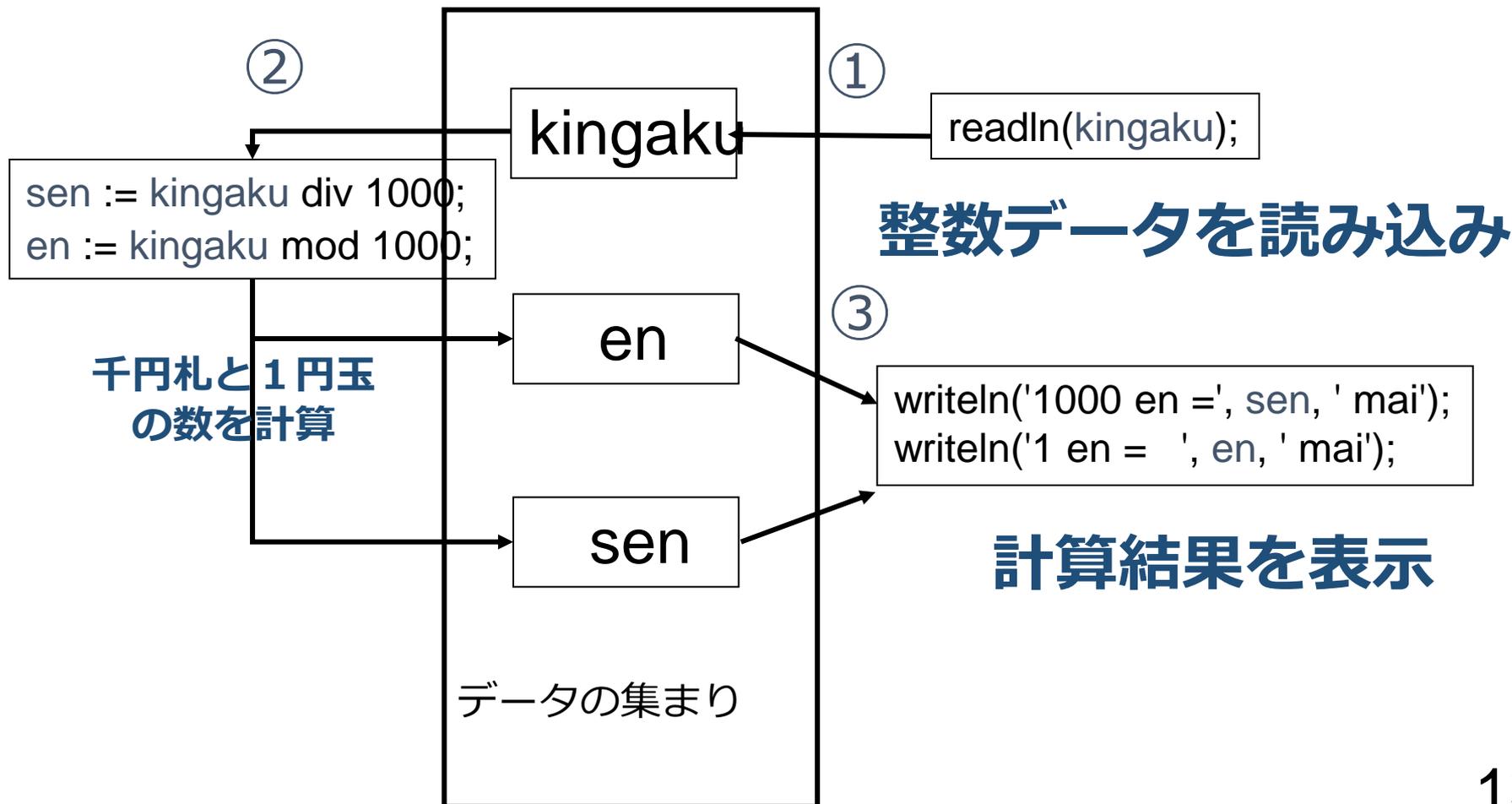
```
kingaku ? : 3020
1000 en =      3 mai
1 en =        20 mai
```

```
kingaku ? : 12345
1000 en =     12 mai
1 en =       345 mai
```

# プログラム実行順



## メモリ



## • 整数

整数（正か負か0）

例)

0

3

2 8

4 7 7 8

- 1

- 1 0

- 1 2 5 0

## • 浮動小数点数

小数付きの数も可

例)

0

3

1 2 7 8 7 4 8 6 2 3

- 4 5 6 3 7 5 9 3 9

8

2. 1 9 0 8 7 2

0. 0 0 0 1 7 8

# 整数と浮動小数点数



	整数	浮動小数点数
変数宣言	kingaku :integer;	teihen :real;
四則演算	+ , - , * , div •div は割り算 (余りは切り捨て)	+ , - , * , /
剰余	mod	

# 変数



- **変数**には, **名前**と**型** (型とはデータの種類のこと)がある
  - 整数 integer
  - 浮動小数点数 real
- **変数宣言**では, **名前**と**型**を指定する

# 整数データの算術演算子



+	和
-	差
*	積
div	割り算の商（余りは切り捨て）
mod	割り算の剰余

## 例題 2. 硬貨の金種計算



- **金額を読み込んで，適切な小銭の枚数を求め，表示するプログラムを作る。**

**例) 金額が768円するとき，**

**500円玉：1枚**

**100円玉：2枚**

**50円玉：1枚**

**10円玉：1枚**

**5円玉：1枚**

**1円玉：3枚**

- 例題では，簡単のため，**紙幣は考えない（硬貨のみ）**ということにする
- 各硬貨の枚数を扱うために，**整数の変数を使う** 18

```
program sum;  
var kingaku, n500, n100, n50, n10, n5, n1: integer;  
begin
```

```
  write('kingaku ?: ');
```

```
  readln(kingaku);
```

```
  n500 := kingaku div 500;
```

```
  n100 := ( kingaku mod 500 ) div 100;
```

```
  n50 := ( kingaku mod 100 ) div 50;
```

```
  n10 := ( kingaku mod 50 ) div 10;
```

```
  n5 := ( kingaku mod 10 ) div 5;
```

```
  n1 := kingaku mod 5;
```

```
  writeln('500 en', n500:4, ' mai');
```

```
  writeln('100 en', n100:4, ' mai');
```

```
  writeln('50 en ', n50:4, ' mai');
```

```
  writeln('10 en ', n10:4, ' mai');
```

```
  writeln('5 en ', n5:4, ' mai');
```

```
  writeln('1 en ', n1:4, ' mai');
```

```
  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

## 実行結果の例

```
kingaku ? : 12687  
500 en 25 mai  
100 en 1 mai  
50 en 1 mai  
10 en 3 mai  
5 en 1 mai  
1 en 2 mai
```

# 硬貨の金種計算



## • 500円玉の枚数は

「(金額) を1000円で割った余り」 割る 5  
00

例えば, 12687 円の時

$$500 \text{ 円玉} : 12687 \text{ div } 500 \rightarrow 25$$

$$100 \text{ 円玉} : (12687 \text{ mod } 500) \text{ div } 100 \rightarrow 1$$

$\underbrace{\hspace{10em}}$   
12687 を 500 で割った余り (値は 187)

$$50 \text{ 円玉} : (12687 \text{ mod } 100) \text{ div } 50 \rightarrow 1$$

$\underbrace{\hspace{10em}}$   
12687 を 100 で割った余り (値は 87)

10円, 5円, 1円も同様に考える

# 例題 3. うるう年の判定



- 「西暦年」を読み込んで、うるう年かどうか表示するプログラムを作る。
  - うるう年の判定のために、**比較演算**と**論理演算**を組み合わせる
  - 西暦年が 4, 100, 400 の倍数であるかを調べるために `mod` を使う

**2021** → うるう年でない

**2024** → うるう年である

# グレゴリオ暦でのうるう年



- **うるう年**とは： 2月が29日までである年
  - うるう年は400年に97回で， 1年の平均日数は365.2422日
  - **うるう年の判定法**
    - **年数が4の倍数の年** → うるう年
    - **但し， 100の倍数の年で400の倍数でない年**
      - **うるう年ではない**（4の倍数だが**例外**とする）
- (例) 2024年： うるう年（4の倍数）  
2020年： うるう年（4の倍数）  
2000年： うるう年（4の倍数）  
1900年： **うるう年ではない**  
（100の倍数だが400の倍数でない）  
1800年： **うるう年ではない**  
（100の倍数だが400の倍数でない）

```
program sum;
var y: integer;
begin
  write('year ?: ');
  readln(y);
  if ((y mod 400) = 0) or (((y mod 100) <> 0) and ((y mod 4) = 0)) then begin
    writeln(y, ' is leap year');
  end
  else begin
    writeln(y, ' is NOT leap year');
  end;
  readln
end.
```

条件式



条件が成り立つ場合に  
実行される部分

条件が成り立たない場合に  
実行される部分

# うるう年の判定



## 実行結果の例

```
year ? : 2024  
2024 is leap year
```

```
year ? : 2000  
2000 is leap year
```

```
year ? : 1900  
1900 is NOT leap year
```

# うるう年の判定式



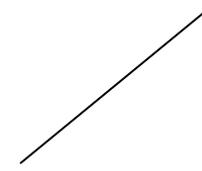
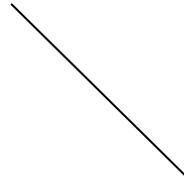
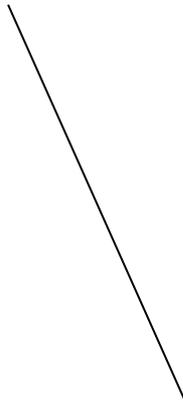
$((y \bmod 400) = 0) \text{ or } (((y \bmod 100) \neq 0) \text{ and } ((y \bmod 4) = 0))$



400の倍数である

100の倍数でない

4の倍数である



または

かつ

# 例題 4 . 複利計算



- **元金**をある**年利**で，ある**年数**だけ運用したときの**利息**を表示する
  - 複利計算では，**利息が利息**を生む。
  - 複利計算を行うために，**Power 関数** (**Power(x,y)**でxのy乗) を使う
  - 整数データと浮動小数点数データが混在する
    - 元金 gankin: 整数データ (単位は 円)
    - 年数 nensu: 整数データ (単位は 年)
    - 年利 nenri: 浮動小数点数データ (単位は %)

```
program sum;  
uses Math;  
var gankin, nensu, ganri: integer;  
var nenri, r: real;  
begin
```

```
write('gankin ? (en): ');  
readln(gankin);  
write('nensu ? (nen): ');  
readln(nensu);  
write('nenri ? (%): ');  
readln(nenri);
```

キーボードからの  
データの読み込みを  
行っている部分

```
r := 1 + ( nenri * 0.01 );  
ganri := trunc( gankin * Power( r, nensu ) );
```

計算を  
行っている部分

```
writeln('ganri = ', ganri:8, ' en');  
writeln('risoku =', ( ganri - gankin ):8, ' en');
```

画面へのデータの  
書き出しを行ってい  
る部分

```
readln  
end.
```

## 実行結果の例

```
gankin ? (en) : 10000
nensu ? (nen) : 10
nenri ? (%) : 2
ganri =      12189 en
risoku =     2189 en
```

# 複利の計算



- 複利の公式：

$$\text{元利} = \text{元金} \times (1 + \text{年利})^{\text{年数}}$$

- べき乗  $x^y$  の計算のために、ライブラリ関数 `Power(x,y)` を使用する

```
ganri := trunc( gankin * Power( r, nensu ) );
```

**r は年利**

# 整数に関する関数



- round      浮動小数点数を  
最も近い整数へ変換  
例)  $3.4 \rightarrow 3$ ,  $3.6 \rightarrow 4$ ,  $-1.6 \rightarrow 2$
- trunc      浮動小数点数を  
ゼロ方向に向かった最も近い整数へ変換  
例)  $3.4 \rightarrow 3$ ,  $3.6 \rightarrow 3$ ,  $-1.6 \rightarrow -1$

# trunc の意味



- 浮動小数点数で計算される
- **計算結果が浮動小数点数であるとき，整数の変数に代入するには，小数点以下切り捨てか，四捨五入が行われねばならない**
- trunc の意味：
  - 右辺の計算結果の小数点以下を切り捨てて，左辺の変数に代入する。

`ganri` := trunc(`gankin` \* Power(`r`, `nensu`));

整数                      整数                      浮動小数点数 整数

# 演習 1 . trunc



- 例題 4 のプログラムを次のように変更し，元金は 10000 円，年数は 10 年，年利は 2% として実行せよ．実行結果を確認せよ．

## 変更前

```
var gankin, nensu, ganri: integer;  
(途中省略)  
ganri := trunc( gankin * Power( r, nensu ) );  
writeln('ganri = ', ganri:8, ' en');  
writeln('risoku = ', ( ganri - gankin ):8, ' en');
```

## 変更後

```
var gankin, nensu: integer;  
var ganri: real;  
(途中省略)  
ganri := gankin * Power( r, nensu );  
writeln('ganri = ', ganri:8:3, ' en');  
writeln('risoku = ', ( ganri - gankin ):8:3, ' en');
```

## 演習 2. 金種計算



- **金額**を読み込んで、適切な**紙幣と小銭の枚数**を求め、表示するプログラムを作成しなさい。
  - 但し、**すべての種類の紙幣と硬貨を考慮**すること
  - 例題 2 のプログラムを参考にする

例) 金額が 1 3, 4 8 6 円するとき,

1 万円札 : 1                      5 0 0 円 : 0

5 千円札 : 0                      1 0 0 円 : 4

千円札 : 3                      5 0 円 : 1

1 0 円 : 3

5 円 : 1

1 円 : 1

## 演習 3. 時間の換算

- **秒数  $x$**  を読み込んで,  **$h$  時,  $m$  分,  $s$  秒**を計算するプログラムを作りなさい.

例)  $x=3723$  のとき,

1 h, 2 m, 3 s