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

オンライン開発環境 Online GDB

- **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

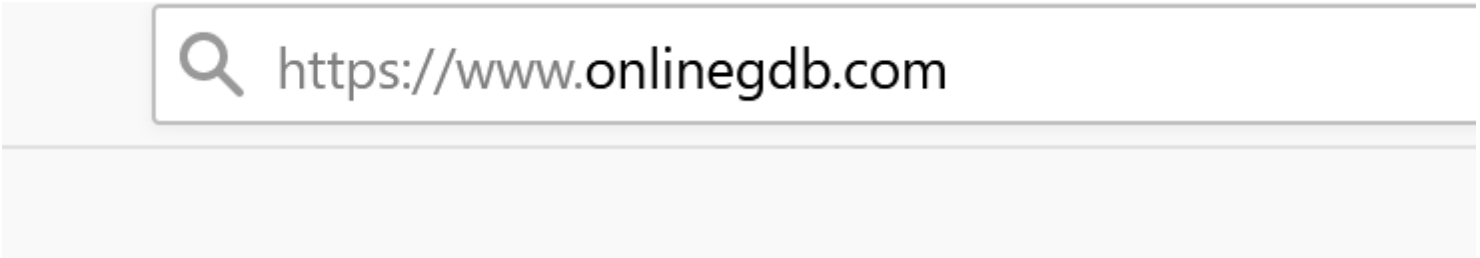
- オンラインなので、「秘密にしたいプログラム」を扱うには十分な注意が必要

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

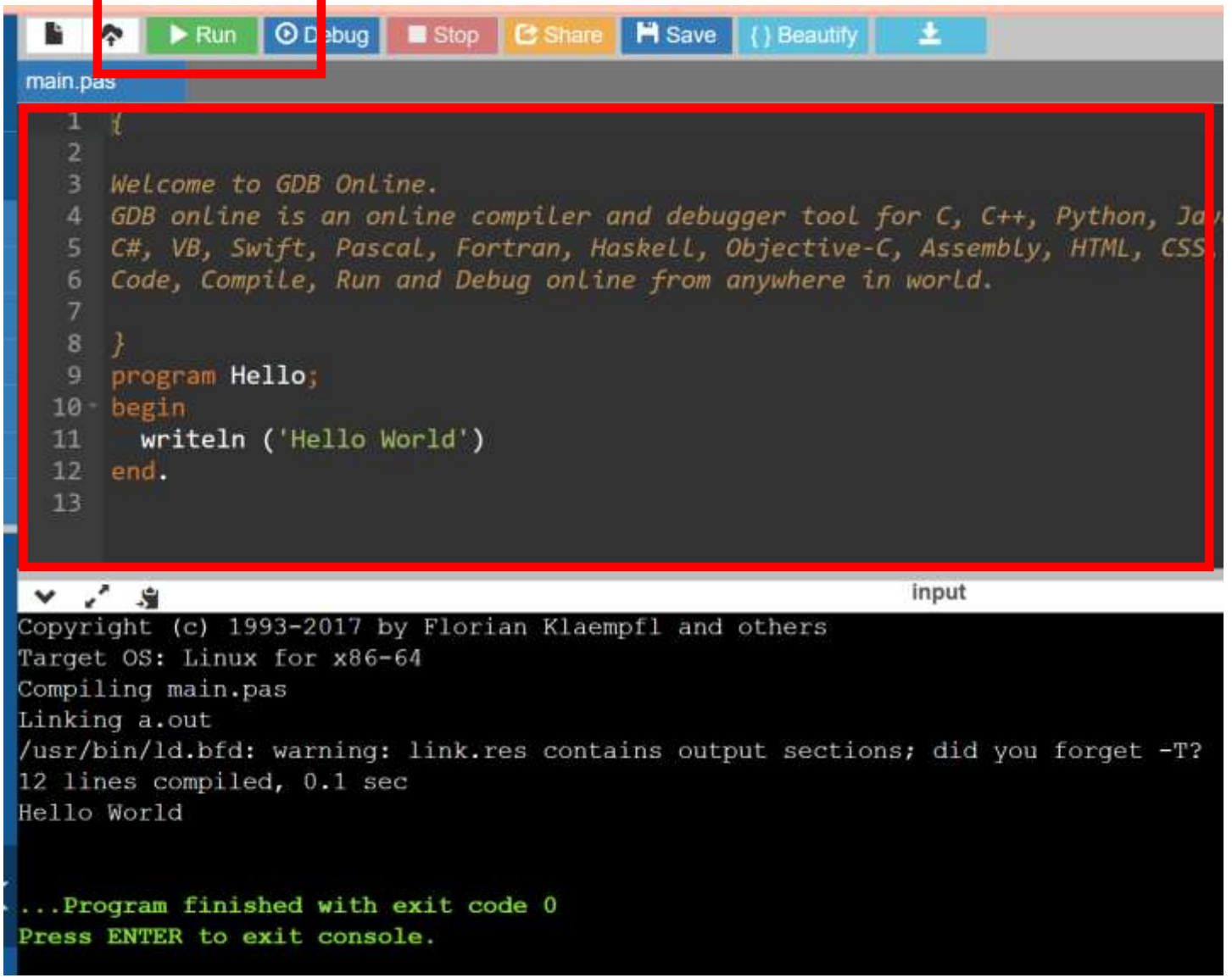


③「**Language**」のところで，「**Pascal**」を選ぶ

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



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１０５０円のとき，

千円札： １枚

１円玉： ５０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
  write('kingaku ?: ');
  readln(kingaku);
  sen := kingaku div 1000;
  en := kingaku mod 1000;
  writeln('1000 en =', sen:8, ' mai');
  writeln('1 en =   ', en:8, ' 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

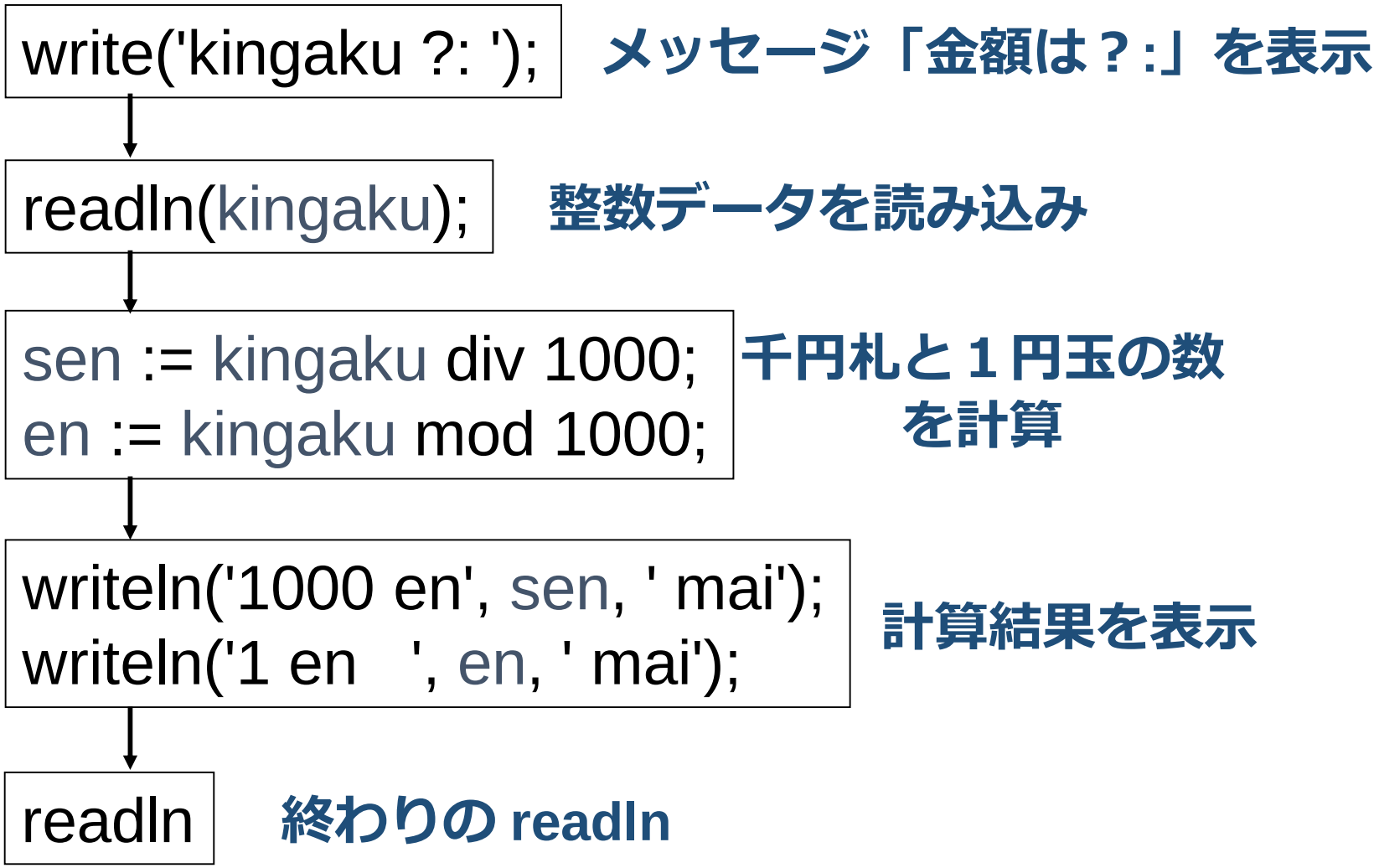
単純な金種計算

実行結果の例

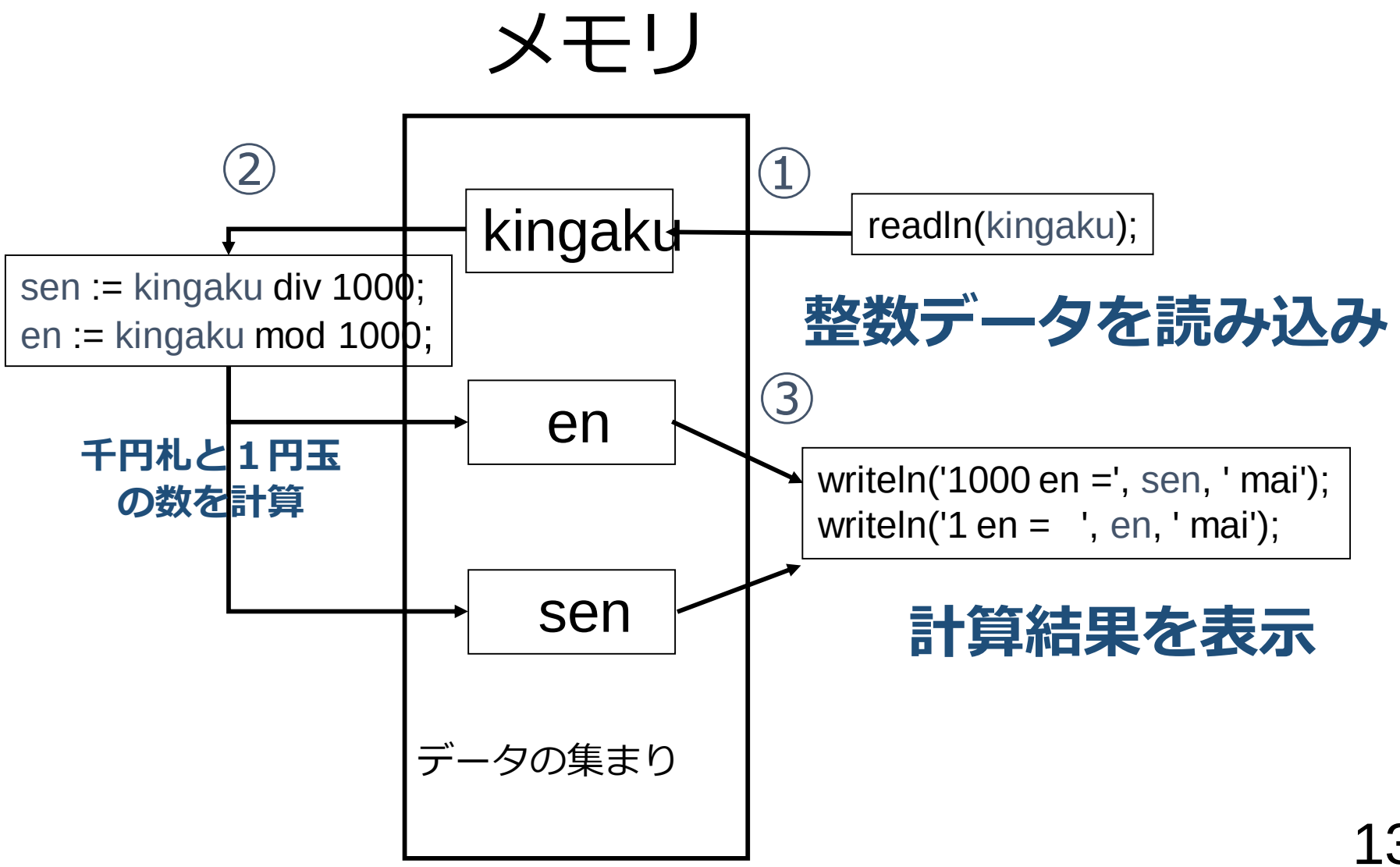
```
kingaku ?: 3020
1000 en =          3 mai
1 en =            20 mai
```

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

プログラム実行順


write('kingaku ?: ');
メッセージ「金額は？:」を表示
readln(kingaku);
整数データを読み込み
sen := kingaku div 1000;
en := kingaku mod 1000;
千円札と１円玉の数
を計算
writeln('1000 en', sen, ' mai');
writeln('1 en ', en, ' mai');
計算結果を表示
readln
終わりの readln

プログラムとデータ


メモリ
②
①
kingaku
readln(kingaku);
sen := kingaku div 1000;
en := kingaku mod 1000;
整数データを読み込み
③
en
千円札と１円玉
の数を計算
writeln('1000 en =', sen, ' mai');
writeln('1 en = ', en, ' mai');
sen
計算結果を表示
データの集まり

整数と浮動小数点数

- **整数**

  整数（正か負か０）

  例）

  ０

  ３

  ２８

  ４７７８

  －１

  －１０

  －１２５０

- **浮動小数点数**

  小数付きの数も**可**

  例）

  ０

  ３

  １２７８７４８６２３

  －４５６３７５９３９

  ８

  ２．１９０８７２

  ０．０００１７８

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円のとき，

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

```
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

硬貨の金種計算

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　５００

例えば，12687 円のとき

５００円玉：　12687 div 500 → 25

１００円玉：　(12687 mod 500) div 100 → 1

12687 を 500 で割った余り（値は 187）

５０円玉：　(12687 mod 100) div 50 → 1

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 mod 400) = 0) or (((y mod 100) <> 0) and ((y mod 4) = 0))

400の倍数である　**100の倍数でない**　**4の倍数である**

かつ

または

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);
  r := 1 + ( nenri * 0.01 );
  ganri := trunc( gankin * Power( r, nensu ) );
  writeln('ganri = ', ganri:8, ' en');
  writeln('risoku =', ( ganri - gankin ):8, ' en');
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

硬貨の金種計算

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：

$$元利 = 元金 \times （1 + 年利）^{年数}$$

- 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 → 3，3.6 → 4，　-1.6 → 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 → 3，3.6 → 3，　-1.6 → -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える**こと
  - 例題２のプログラムを参考にすること

例） 金額が１３，４８６円のとき，

１万円札： １	５００円： ０
５千円札： ０	１００円： ４
千円札： ３	５０円： １
	１０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