

# cp-6. 整数データと浮動小数 データ

(C プログラミング入門)

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

金子邦彦



# 本日の内容



例題 1. 単純な金種計算

例題 2. 硬貨の金種計算

整数の変数

浮動小数と整数の違い

例題 3. 複利計算

整数の変数と、浮動小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「整数データ」と「浮動小数データ」の違いについて理解する
- 目的に応じて、整数の変数、浮動小数の変数を正しく使い分けることができるようになる

## 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 1 0 5 0 円 のとき,
    - 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、小銭の種別は考えない (1 円玉のみ) ということにする。
- 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

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int kingaku;
    int en;
    int sen;
    printf("kingaku=");
    scanf("%d", &kingaku);
    sen = kingaku/1000;
    en = kingaku%1000;
    printf("senensatsu: %d mai¥n", sen);
    printf("kozeni: %d en¥n", en);
    return 0;
}
```

入力部分

計算部分

出力部分

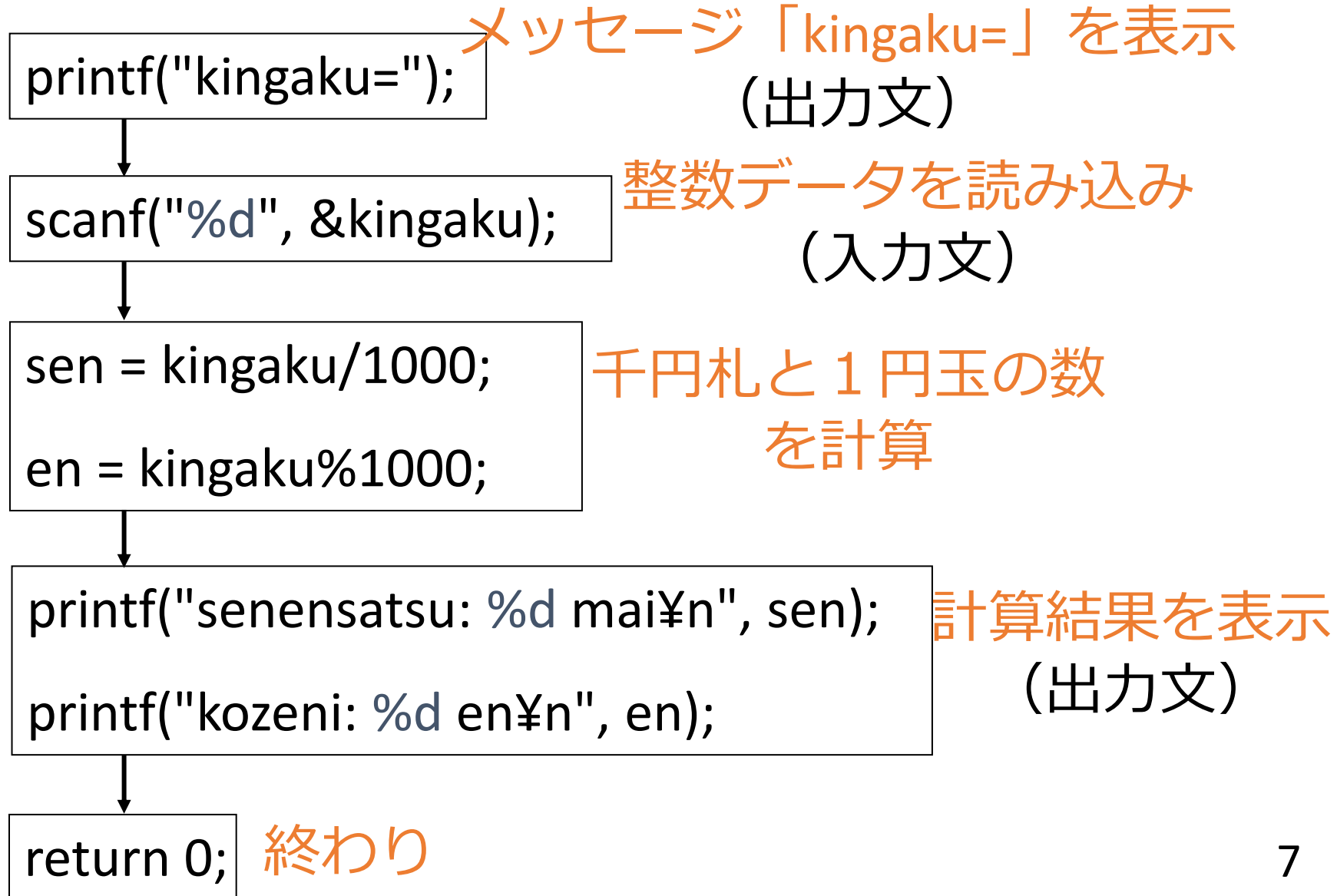
# 実行結果例

kingaku=13500

senensatsu: 13 mai

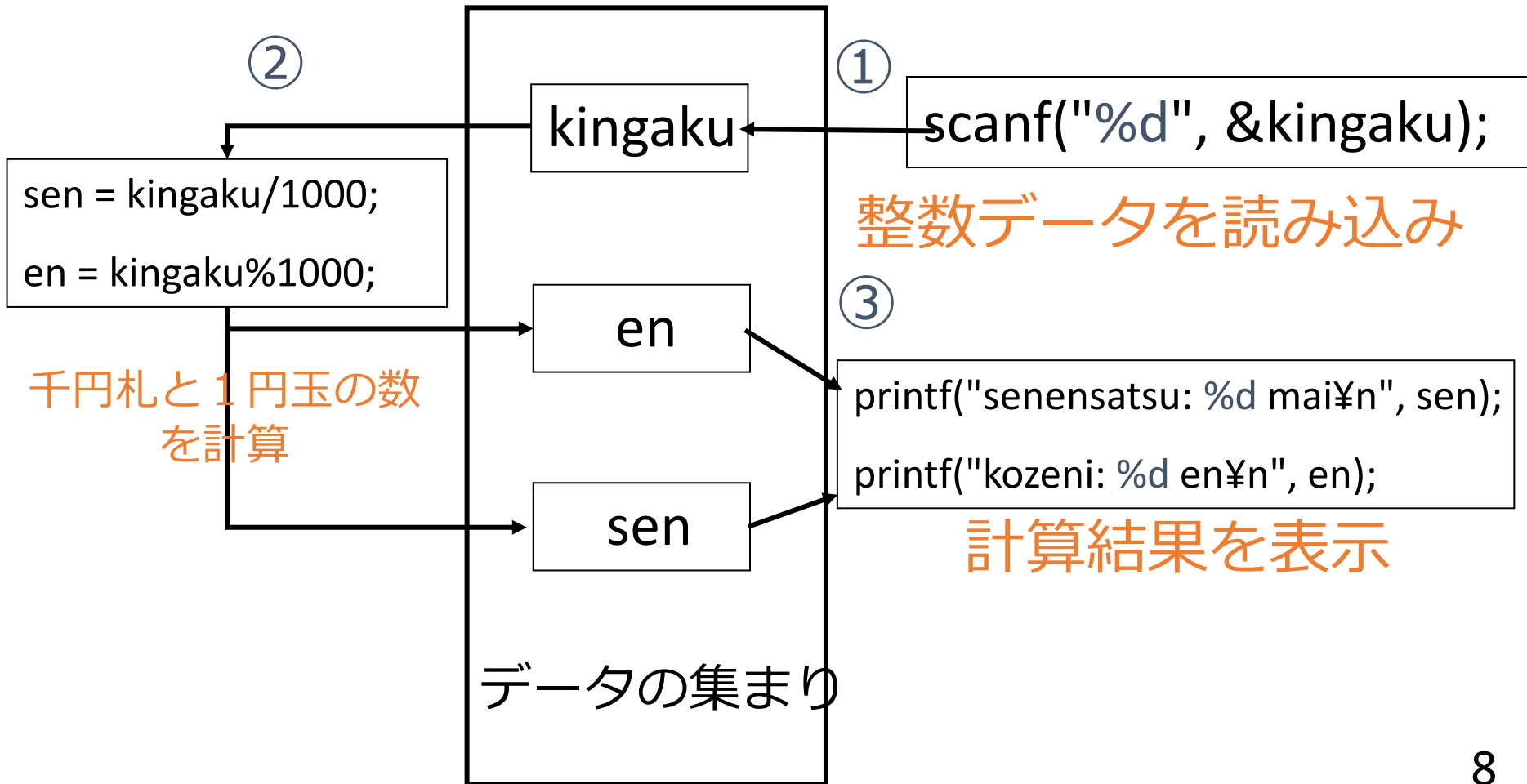
kozeni: 500 en

# プログラム実行順



# プログラムとデータ

## メモリ





# 整数データと浮動小数データ

## • 整数データ

- 整数
- およそ4桁まで
- 割り算では小数点以下切り捨て

例)

0  
3  
2 8  
4 7 7 8  
- 1  
- 1 0  
- 1 2 5 0

## • 浮動小数データ

- 小数付きの数も可
- およそ10桁まで

例)

0  
3  
2 8  
4 7 7 8  
- 1  
- 1 0  
- 1 2 5 0  
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

# 整数データと浮動小数データの精度

- 整数データ(int)
  - $-32767$ から $+32767$ までの範囲の数
  - 数学での「整数」では無い
- 浮動小数データ(double)
  - 精度が10桁で,  $10^{-37}$ 乗から $10^{+37}$ 乗までの正と負の範囲数
  - 数学での「実数」とは違う

# 整数データと浮動小数データの違い

	整数データ	浮動小数データ
変数宣言	<code>int kingaku;</code> <code>int en;</code>	<code>double teihen;</code> <code>double takasa;</code>
入力	<code>scanf("%d", &amp;kingaku);</code>	<code>scanf("%lf", &amp;takasa);</code>
出力	<code>printf("kozeni: %d en¥n", en);</code>	<code>printf("takasa: %f ¥n", takasa);</code>
四則演算	四則演算には, +, -, *, / を使う	四則演算には, +, -, *, / を使う
剰余	<code>en = kingaku%1000;</code>	<code>z = fmod(x,y);</code>

# 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
- 変数宣言では、名前と型を書く

## 「型」の例

- 整数データ           int
- 浮動小数データ   double

# 整数データの算術演算子



+ 和

- 差

\* 積

/ 商

% 剰余 (整数データの場合)

→ 浮動小数データの剰余は  
fmod(x,y)を使うこと

# 入力文と出力文の機能

- printf の機能

- メッセージの表示
- 整数データの表示
- 浮動小数データの表示

- scanf の機能

- 整数データの読み込み
- 浮動小数データの読み込み

# 入力文



```
scanf("%d", &kingaku);
```

書式

& 読み込むべき変数名

- 入力文とは、データを読み込むための文
- 「書式」と読み込むべき変数名を書く

## 書式の書き方

### %d: 整数データ

キーボードから読み込まれる数字を10進の整数データとして解釈

### %f: 浮動小数データ

キーボードから読み込まれる数字、小数点などを10進の浮動小数データとして解釈

- 変数名の前には「&」を付けること

# 出力文



```
printf("kozeni: %d en¥n", en);
```

書式

表示すべき変数名

- データとメッセージを表示するための文
- 「書式」と読み込むべき変数名を書く

## 書式の書き方

**%d: 整数データ**

10進数の数字に直してこの位置（表示文字列の一部）に置け、という指示

**%f: 浮動小数データ**

10進数の数字に直してこの位置（表示文字列の一部）に置け、という指示

- 変数名の前には「&」は付けない



# 整数データの読み込みと表示 (1/3)



プログラム例

```
#include <stdio.h>
int main()
{
    int n;
    printf( "n=?" );
    scanf( "%d", &n );
    printf( "n=%d¥n", n );
    return 0;
}
```

実行結果例

```
n=?100
n=100
```

A screenshot of a terminal window titled "Oppl" with a menu bar containing "自動" and various icons. The terminal output shows the program's execution: "n=?100", "n=100", and "Press any key to continue". A green arrow points from the "n=?100" line in the execution result box to the corresponding input in the terminal. Another green arrow points from the "n=100" line in the execution result box to the output in the terminal. The terminal background is black with white text.

# 整数データの読み込みと表示 (2/3)



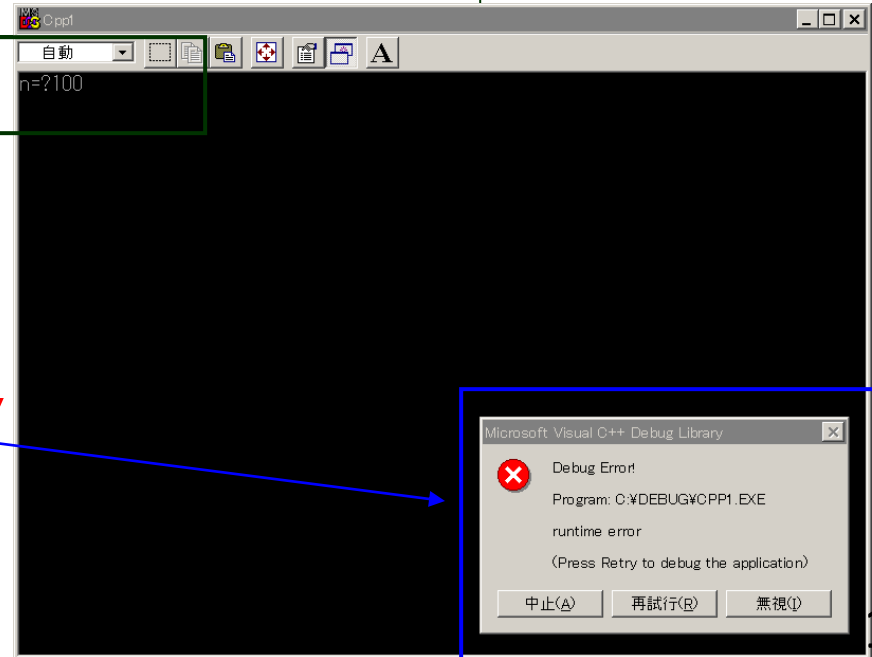
プログラム例

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int n;
    printf( "n=?" );
    scanf( "%lf", &n );
    printf( "n=%d¥n", n );
    return 0;
}
```

正しくは「%d」

実行結果例

n=?100



読み込もうとすると、  
エラーが現れて、  
結局読み込めない

# 整数データの読み込みと表示 (3/3)



プログラム例

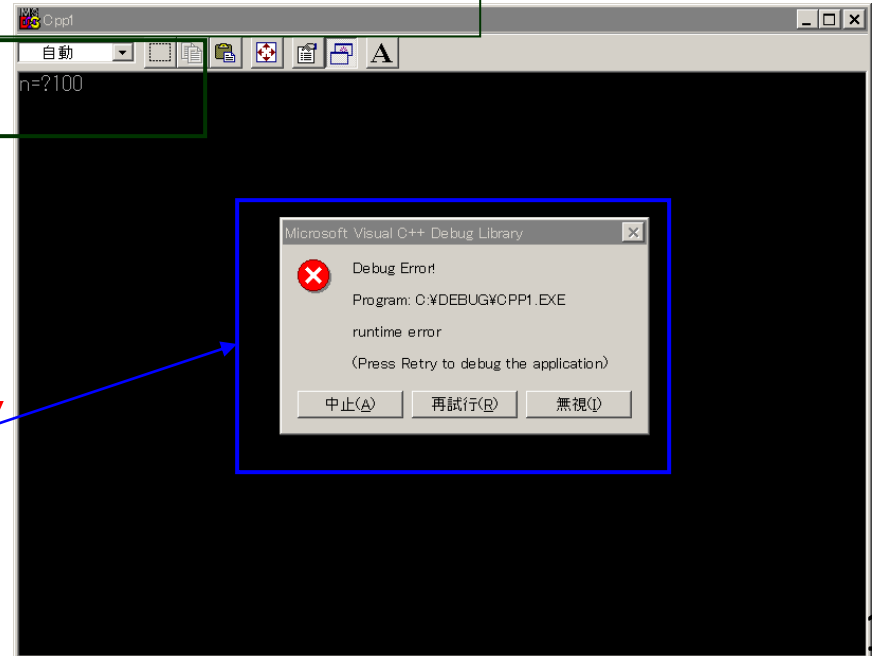
```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int n;
    printf( "n=?" );
    scanf( "%d", &n );
    printf( "n=%f¥n", n );
    return 0;
}
```

正しくは「%d」

実行結果例

n=?100

表示しようとする  
と、エラーが現れて、  
結局表示されない



## 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 7 6 8 円するとき、

5 0 0 円玉 : 1 枚

1 0 0 円玉 : 2 枚

5 0 円玉 : 1 枚

1 0 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、整数データの変数を使う

```
#include <stdio.h>
#pragma warning(disable:4996)
int main()
{
    int kingaku;
    int gohyaku, hyaku, gojyu, jyu, go, ichi;
    printf("kingaku=");
    scanf("%d", &kingaku);
    gohyaku = ( kingaku % 1000 ) / 500;
    hyaku = ( kingaku % 500 ) / 100;
    gojyu = ( kingaku % 100 ) / 50;
    jyu = ( kingaku % 50 ) / 10;
    go = ( kingaku % 10 ) / 5;
    ichi = kingaku % 5;
    printf( "五百円 = %d¥n", gohyaku );
    printf( "百円 = %d¥n", hyaku );
    printf( "五十円 = %d¥n", gojyu );
    printf( "十円 = %d¥n", jyu );
    printf( "五円 = %d¥n", go );
    printf( "一円 = %d¥n", ichi );
    return 0;
}
```

入力部分

計算部分

出力部分

# 実行結果例

kingaku=768

五百円 = 1

百円 = 2

五十円 = 1

十円 = 1

五円 = 1

一円 = 3

# 課題 1 . 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（金種計算）を作りなさい。
  - 但し、すべての種類の紙幣と硬貨を考えること。

紙幣： **1万円札, 5千円札, 千円札**

硬貨： **500円, 100円, 50円, 10円, 5円, 1円**

例) 金額が13,486円するとき,

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

# 課題 1 の実行結果例

kingaku=13486

一万円 = 1

五千円 = 0

千円 = 3

五百円 = 0

百円 = 4

五十円 = 1

十円 = 3

五円 = 1

一円 = 1



# 課題 1 のヒント



1 0 0 0 円札の枚数は

「(金額) を 5 0 0 0 円で割った余り」 / 1 0 0 0

例えば, 1 7, 6 0 0 円するとき

$$1 \text{ 万円札} : 17,600 / 10,000 = 1$$

$$5 \text{ 千円札} : \underbrace{(17,600 \% 10,000)} / 5000 = 1$$

17,600 を 10,000 で割った余りのこと (値は 7,600)

$$1 \text{ 千円札} : \underbrace{(17,600 \% 5,000)} / 1000 = 2$$

17,600 を 5,000 で割った余りのこと (値は 2,600)

## 課題 2 . 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

## 例題 3 . 複利計算

- 元金 `gankin` 円を年利 `nenri` (パーセント) で `nensu` 年運用したときの利息を求めるプログラムを作る
  - 単利計算では, 年数に比例.
  - 複利計算では, 利息が利息を生む.
  - 複利計算を行うために, `pow`関数を使う
  - 年数は整数データ, 利息は浮動小数データ

```
#include <stdio.h>
#include <math.h>
#pragma warning(disable:4996)
int main()
{
    int gankin, nensu, ganri;
    double nenri, r;
    printf("gankin=");
    scanf("%d", &gankin);
    printf("nenri=");
    scanf("%lf", &nenri);
    printf("nensu=");
    scanf("%d", &nensu);
    r = 1 + nenri * 0.01;
    ganri = (int)(gankin * pow(r, nensu));
    printf("risoku = %d¥n", ganri - gankin);
    return 0;
}
```

計算部分



# 複利の計算

- 複利の公式：

$$ganri = gankin \times (1 + nenri)^{nensu}$$

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

```
ganri = (int)(gankin * pow(r, nensu));
```

## (int) の意味

```
ganri = (int)(gankin * pow(r, nensu));
```

整数                      整数                      浮動小数   整数

- 右辺は浮動小数の精度で計算される
- (int) の意味：
  - 結果の小数点以下を切り捨てて、整数部だけを ganri に代入.
  - (int) は、変数 ganri への代入時に、「データの精度が落ちておかまわらない」ことをコンピュータに教える

## (int) が必要な場合

```
ganri = (int)(gankin * pow(r, nensu));
```

整数                      整数                      浮動小数   整数

- 左辺が整数の変数，右辺が浮動小数を含む式
  - このとき、右辺は、浮動小数の精度で計算され、最後に、左辺の変数の代入される

# 1/2 の値は 0

```
#include <stdio.h>
int main()
{
    double r;
    r = 1 / 2;
    printf("r = %f¥n ", r );
}
```

このプログラムの実行結果は、直感とは一致しないかも知れない

```
r = 0.000000
```

右辺に整数の変数しか登場しないので、右辺は整数の精度で計算される



# 1.0/2.0 の値は 0.5

```
#include <stdio.h>
int main()
{
    double r;
    r = 1.0 / 2.0;
    printf("r = %f\n ", r );
}
```

「1/2」と「1.0/2.0」は、意味が違う

```
r = 0.500000
```

右辺に浮動小数の変数が登場するので、右辺は浮動小数の精度で計算される

# 1/2 と 1.0/2.0 の違い



- **1/2** は, 整数と整数の割り算
  - 文法的には「**2000/30 (値は66)** 」と書くのと同じ
  - **1/2** の値は **0** (やはり整数)
- **1.0/2.0** は, 浮動小数と浮動小数の割り算
  - **1.0/2.0** の値は **0.5** (浮動小数)