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/pro/adp/index.html

金子邦彦

CC BY NC SA

C言語における整数データと浮動小数データの違いの理解、目的に応じた変数型の使い分け

【学習内容の構成】

1. **金種計算（例題1・2）**：整数の割り算と剰余演算を用いた紙幣・硬貨の枚数計算
2. **データ型の違い**：整数データ（int）と浮動小数データ（double）の精度・表現範囲・演算の差異
3. **入出力の書式**：scanf・printfにおける書式指定子（%d、%lf、%f）の正しい使い分け
4. **複利計算（例題3）**：pow関数による浮動小数演算と(int)キャストによる型変換
5. **演算の注意点**：1/2と1.0/2.0の違いに見る整数演算と浮動小数演算の結果の相違

- 前提：C言語の変数宣言、printf・scanfの基本的な使用方法

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，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

```
printf("kingaku=");
```

メッセージ「kingaku=」を表示
（出力文）

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

整数データを読み込み
（入力文）

```
sen = kingaku/1000;

en = kingaku%1000;
```

千円札と１円玉の数
を計算

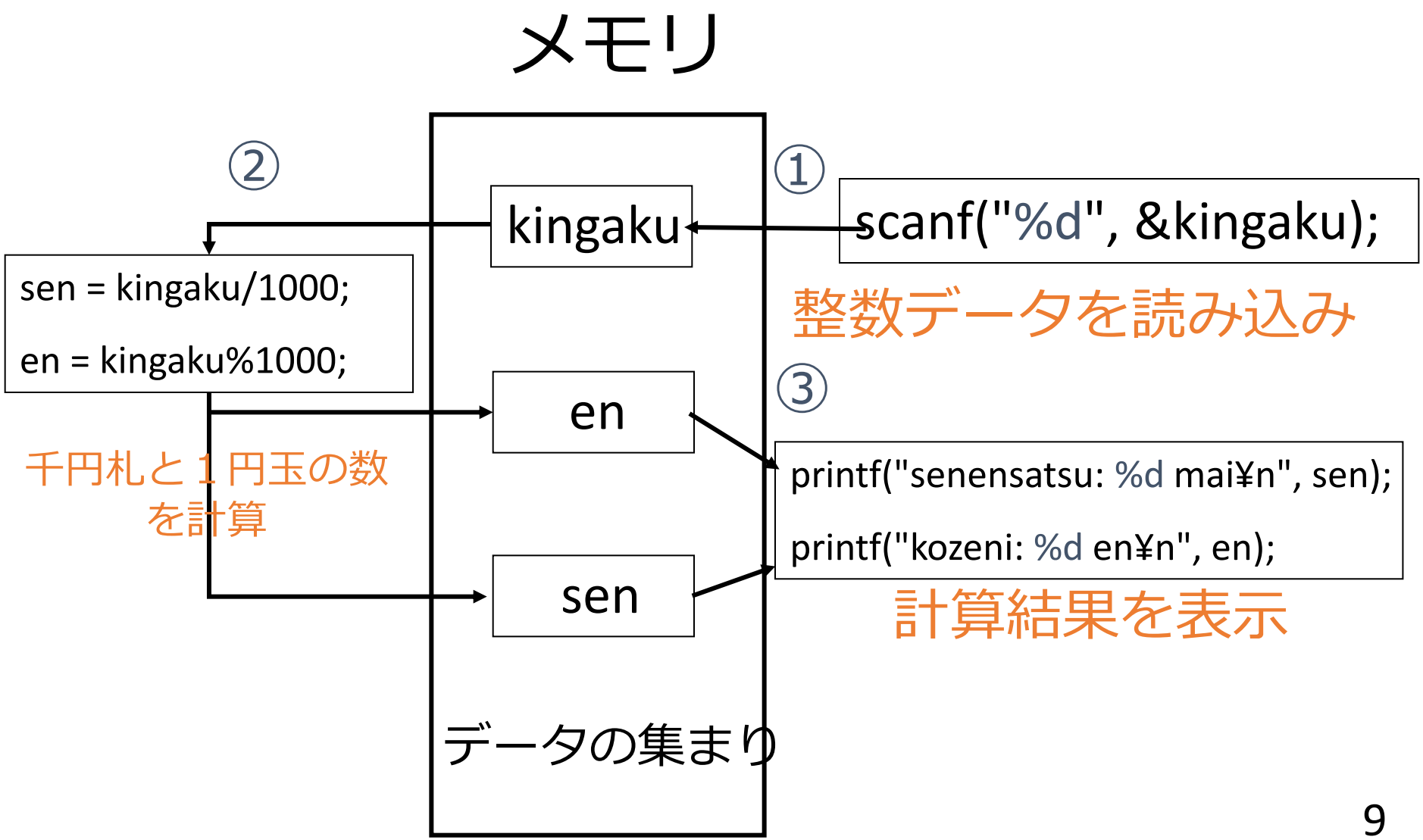
```
printf("senensatsu: %d mai¥n", sen);

printf("kozeni: %d en¥n", en);
```

計算結果を表示
（出力文）

```
return 0;
```

終わり

プログラムとデータ


メモリ
②
①
kingaku
scanf("%d", &kingaku);
整数データを読み込み
sen = kingaku/1000;
en = kingaku%1000;
③
en
千円札と１円玉の数
を計算
printf("senensatsu: %d mai¥n", sen);
printf("kozeni: %d en¥n", en);
sen
計算結果を表示
データの集まり

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
２８
４７７８
－１
－１０
－１２５０
```

- 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
２８
４７７８
－１
－１０
－１２５０
１２７８７４８６２３
－４５６３７５９３９８
２．１９０８７２
０．０００１７８
```

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の－３７乗から１０の＋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	int kingaku; int en;	double teihen; double takasa;
入力	scanf("%d", &kingaku);	scanf("%lf", &takasa);
出力	printf("kozeni: %d en¥n", en);	printf("takasa: %f ¥n", takasa);
四則演算	四則演算には，+, -, *, / を使う	四則演算には，+, -, *, / を使う
剰余	en = kingaku%1000;	z = fmod(x,y);

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

  %lf : 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

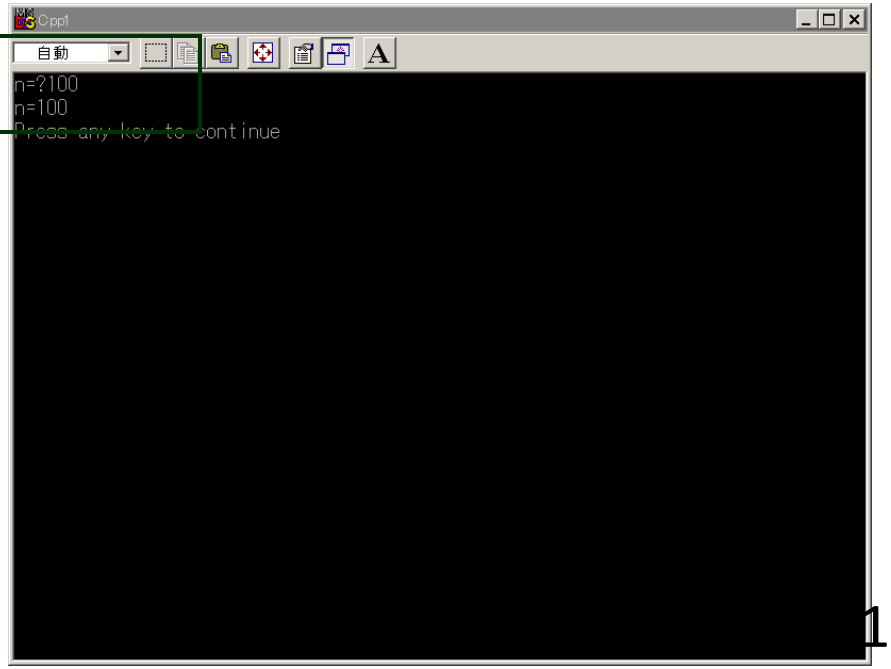
整数データの読み込みと表示 (1/3)

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
```



整数データの読み込みと表示 (2/3)

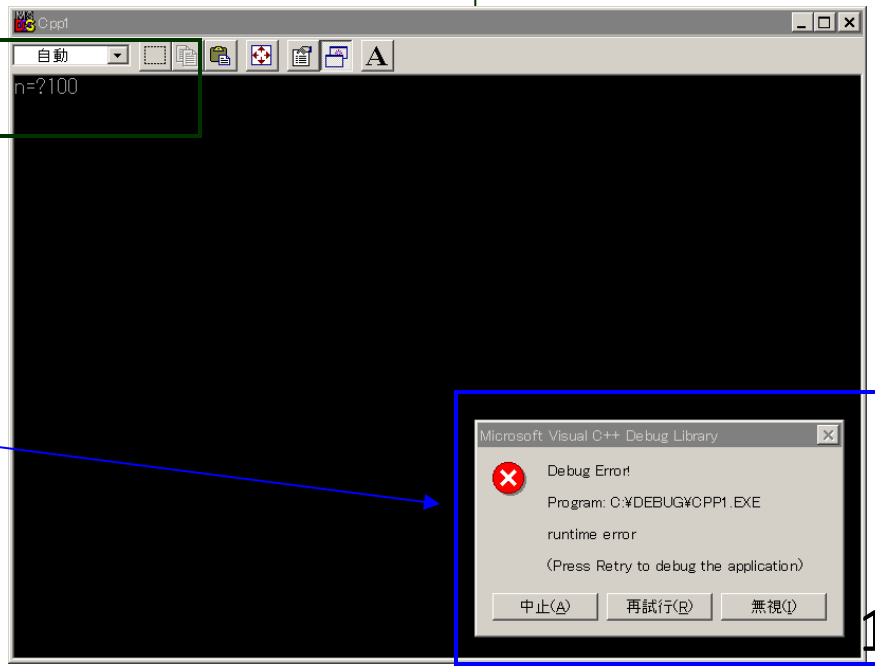
プログラム例

```
#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)

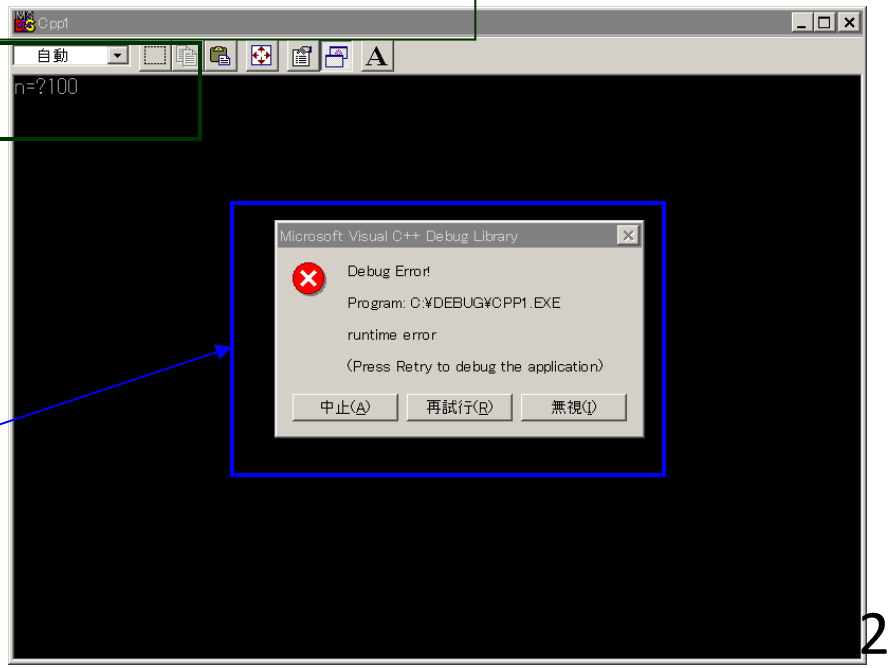
プログラム例

```
#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円のとき，

１万円札： 1
５千円札： 0
千円札： 3

５００円： 0
１００円： 4
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

１０００円札の枚数は

「（金額）　を５０００円で割った余り」　／１０００

例えば，１７，６００円のとき

１万円札：　１７，６００／１０，０００　＝　１

５千円札：　（１７，６００　％　１０，０００）／５０００　＝　１

１７，６００を１０，０００で割った余りのこと（値は７，６００）

１千円札：　（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もかまわない」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）