#01
CPU・メモリ・機械語・変数・スタック

2022 年度 / プログラミング及び実習 III

角川裕次

龍谷大学　先端理工学部

もくじ

1 (独自) コンピュータアーキテクチャ

2 第 6-1 節　関数とは

今回 (#01) の内容: シラバスでの該当部分

小テーマ: CPU・メモリ・機械語・変数・スタック

第 1 回： メモリとアドレス
第 2 回： 機械語プログラムのメモリ配置

重要概念リスト

- CPU, メモリ, 機械語
- メモリの 3 区域
- `main` 関数
- ライブラリ関数
- 関数定義
- 実引数と仮引数
- 値渡し

今回の実習・課題 (manaba へ提出)

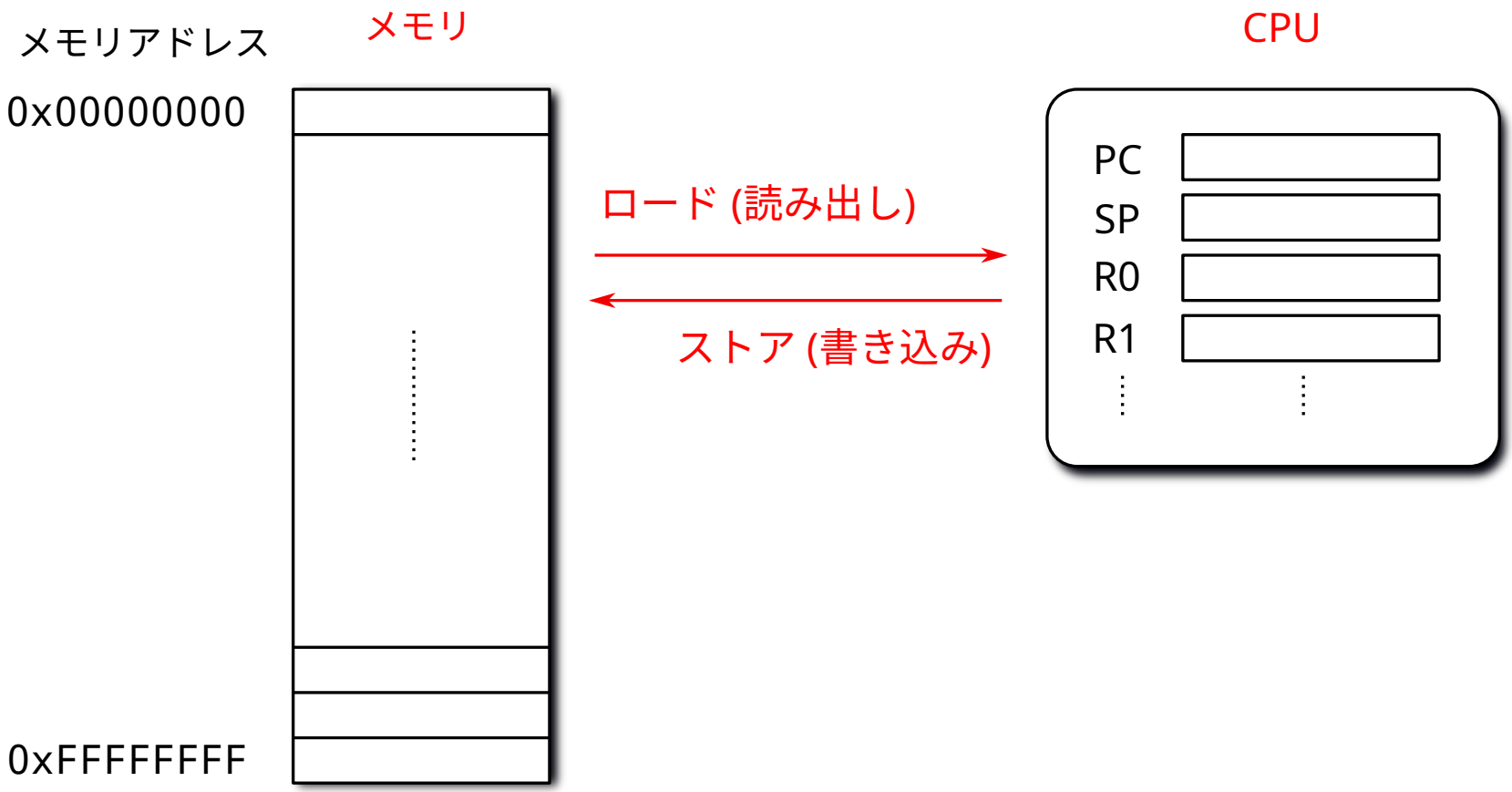
実習内容と課題内容は講義途中に提示します

(作成したファイル類は manaba に提出)

(独自) コンピュータアーキテクチャ

CPU とメモリ：ロードとストアでデータ転送

- CPU：プログラム実行装置 (作業の一時記憶用にレジスタを持つ)
- メモリ：記憶装置 (メモリアドレスでアクセス場所を指定)



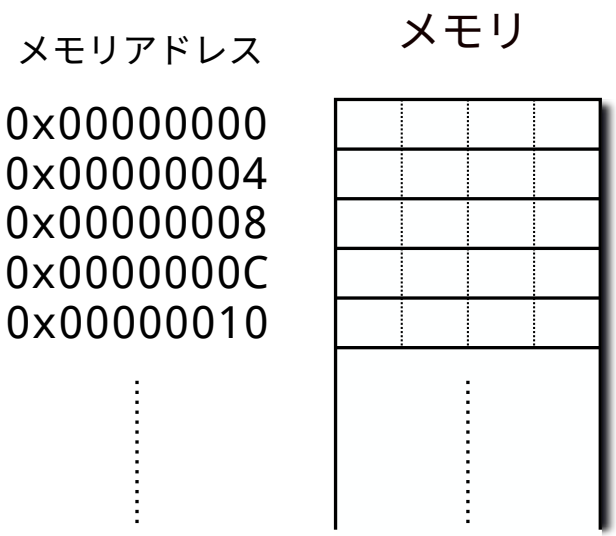
メモリ

1 語：データ処理の基本となるメモリ量

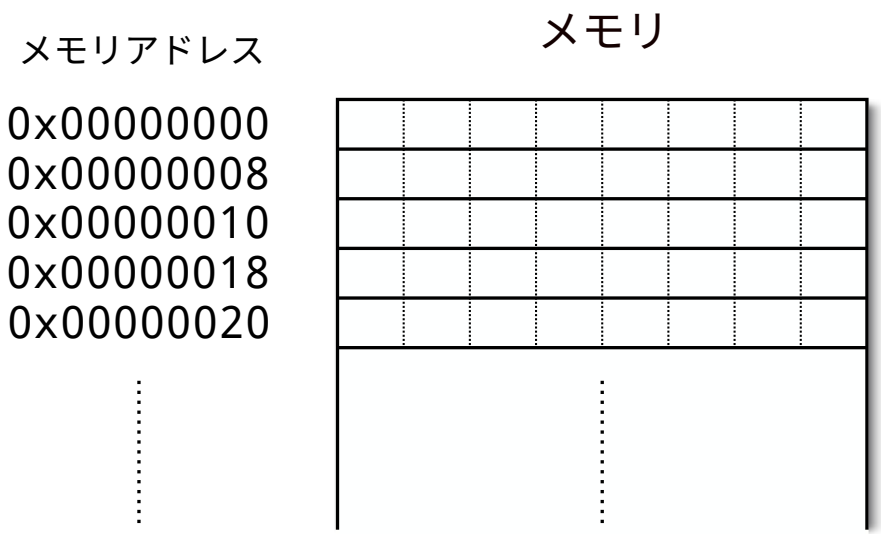
- 整数値の表現に一般的に使用されるメモリ量
- C 言語の int 型のデータ記憶の単位 (の場合が多い)

CPU の構造に依存 (CPU ごとに異なる)

- 32 ビット CPU は 4 バイト, 64 ビット CPU は 8 バイト (が多い)



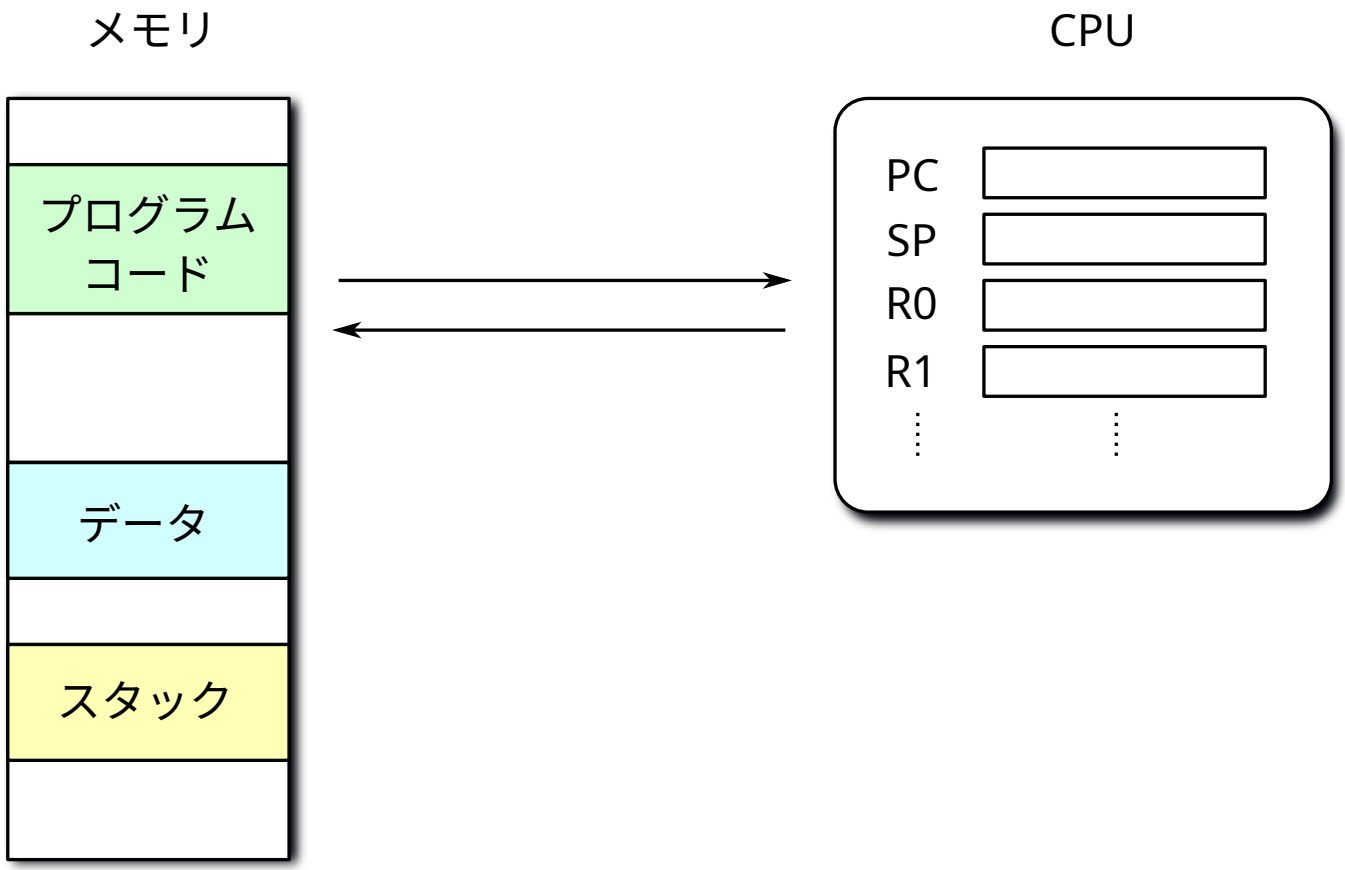
1 語 4 バイトの場合



1 語 8 バイトの場合

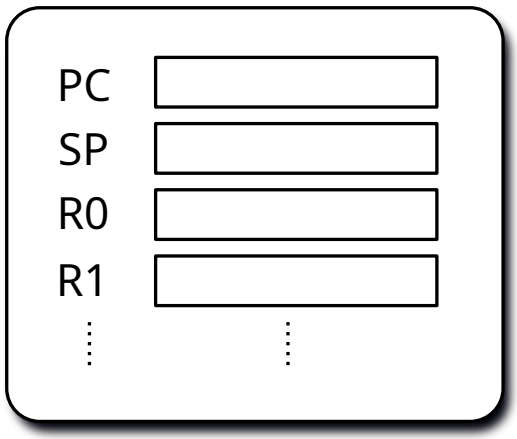
メモリの 3 区域

プログラムコード：実行する命令列を記憶
データ：変数 (静的; static) の値や動的を記憶
スタック：変数 (自動; auto) の値や関数呼び出し・復帰情報を記憶



CPU (Cetral Processing Unit)

内部にいくつかのレジスタ (register) を持つ



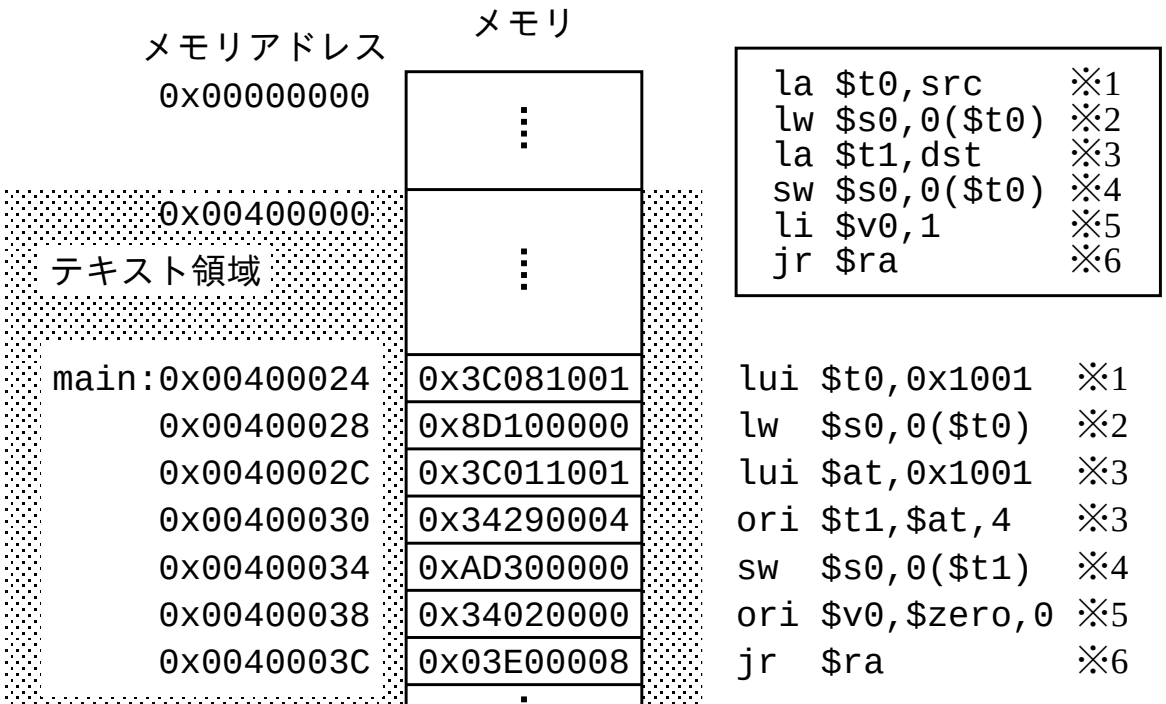
レジスタの分類
- プログラムカウンタ (PC) : 次に実行する命令のメモリアドレス
- スタックポンタ (SP) : 関数呼び出しの制御用
- 汎用レジスタ (R0, R1, ...) : 数値の一時的な記憶

プログラム：CPU の命令の列で構成

命令の種類

・ロード命令：メモリの内容を読み出して CPU のレジスタへ
・ストア命令：CPU のレジスタの値をメモリに書き込む
・演算命令：レジスタの値を演算 (加減乗除)
・比較命令：レジスタの値を比較
・ジャンプ命令：次に実行する命令のアドレスを変更

プログラムの例 (MIPS32 プロセッサ)



第 6-1 節　関数とは

★重要★

□ main 関数とライブラリ関数 p.142

main 関数：プログラム実行時に最初に呼び出される関数

- プログラム中で 1 つだけ必要

Fig.6-1 main 関数

```
#include <stdio.h>
int main(void)
{
  /* ... */
  return 0;
}
```

ライブラリ (library) 関数：C 言語が予め用意している関数

標準 C ライブラリ (libc)：C 言語の標準規格で規定

- printf, scanf, puts など

□ 関数とは　p.142

引数から結果を求める一連の手続き

- 関数名と引数で呼び出し

関数は自分でプログラム内で自由に定義・使用可能

関数使用の 2 ステップ

1. 関数の定義 — 作る
2. 関数の呼出し — 利用する

ライブラリ関数

- 上記ステップ 1 が予め済んでいる
- 呼び出すだけで利用できる

□ 関数定義 p.143

Fig.6-3：2 つの整数値の小さくない方の値を返す関数の例

```
int max2(int a, int b)
{
  if (a > b)
    return a;
  else
    return b;
}
```

関数頭部

- 関数の返却値型：int 型
- 関数名：max2
- 仮引数型並び：int 型の a, int 型の b

関数本体

- 関数の動作を記述
- return 文：関数を終了し関数値を返す (呼び出し元に戻る)

□ 関数呼出し p.143

List 6-1 (1/2) : 関数 max2 を定義・呼び出すプログラム例

```
#include  <stdio.h>

int max2(int a, int b)
{
  if (a > b)
    return a;
  else
    return b;
}
```

関数呼出し (つづき)

List 6-1 (2/2)：関数 max2 を定義・呼び出すプログラム例

```
int  main(void)
{
   int   n1, n2;

   puts("二つの整数を入力せよ。");
   printf("整数１：");    scanf("%d", &n1);
   printf("整数２：");    scanf("%d", &n2);

   printf("大きい方の値は%dです。\n", max2(n1, n2));

   return (0);
}
```

- プログラム実行開始時に main 関数が呼び出される
- ソースコード中で max2 が先に出現するが出現順序は関係ない
- main 関数の中で max2 を呼び出す

関数呼び出しの実行詳細

関数呼び出し部分

```
printf("大きい方の値は%dです。\n", max2(n1, n2));
```

実引数の評価と関数呼出し

1. 第 1 引数 n1 の値を評価 — 変数 n1 の値
2. 第 2 引数 n2 の値を評価 — 変数 n2 の値
3. それらの値を引数 (実引数) として関数 max2 を呼び出す

関数 max2 での仮引数の値の設定 `int max2(int a, int b)`

1. 仮引数 a : n1 の値を設定
2. 仮引数 b : n2 の値を設定

関数本体の実行

1. return 文の値を関数値として返す
   (後述: 値を返さない関数も定義可能...return 文の値が無指定)

関数呼び出しの形いろいろ

引数に定数値 （定数値を実引数に)

```
    max2(n1, 5);
```

引数に式 (式の計算結果を実引数に)

```
    max2(n1+5, n2);
```

引数の変数名が仮引数の変数名と同じ (式の計算結果を実引数に)

```
    max2(a, b);
```

関数 max2 の実現いろいろ (1/3)

仕様：引数 a と b のうち小さくない方を返す

- 教科書での「大きい方」という表現は不正確
- 理由：a = b の場合「大きい方」は存在しない
- 「小さくない方」と表現するのが正確

実現 1：最大値を記憶する変数を使用

```
int max2(int a, int b)
{
  int max;
  if (a > b)
      max = a;
  else
      max = b;
  return max;
}
```

関数 max2 の実現いろいろ (2/3)

実現 2：最大値を記憶する変数の値を初期化

```
int max2(int a, int b)
{
  int max = a;
  if (b > max)
      max = b;
  return max;
}
```

実現 3：条件演算子の使用

```
int max2(int a, int b)
{
  return (a > b) ? a: b;
}
```

関数 max2 の実現いろいろ (3/3)

実現 4：仮引数の変数名は重要ではない (1)

```
int max2(int b, int a)
{
  return (b > a) ? b : a;
}
```

実現 5：仮引数の変数名は重要ではない (2)

```
int max2(int fjjjizoo, int abc)
{
  return (fjjjizoo > abc) ? fjjjizoo : abc;
}
```

- そうなんだけども, 分かりやすい変数名を付けましょうね
- バグで苦しむのは自分自身ですよ
- そのバグを人に押し付けないでね

□ 3 値の最大値を求める関数 p.147

List 6-2 (部分)

```
int max3(int a, int b, int c)
{
  int max = a;
  if (b > max)
    max = b;
  if (c > max)
    max = c;
  return max;
}
```

3 値の最大値：main 関数

List 6-2 (部分)

```
int main(void)
{
  int a, b, c;
  puts("三つの整数を入力せよ。");
  printf("整数１：");  scanf("%d", &a);
  printf("整数２：");  scanf("%d", &b);
  printf("整数３：");  scanf("%d", &c);
  printf("最大値は%dです。\n", max3(a, b, c));
  return 0;
}
```

実引数と仮引数は別物：値のみが渡される

main 関数の変数 a, b, c と関数 max3 の仮引数 a, b, c

- 名前は同じでも別物
- 理由：異なるメモリを使用して記憶するため

以下のように関数 max3 の定義を変更しても動作結果は同一

```
int max3(int b, int c, int a)
{
  int max = b;
  if (c > max)
      max = c;
  if (a > max)
      max = a;
  return max;
}
```

- 関数 max3 の仮引数 b：main 関数の変数 a の値が設定される

□ 関数の返却値を引数として関数に渡す p.148

List 6-3 (1/2) : 二つの整数の 2 乗値の差を求める

```
#include <stdio.h>

/* nの2乗値を返す */
int sqr(int n)
{
  return n * n;
}

/* aとbの差を返す (=a-bの絶対値) */
int diff(int a, int b)
{
  return (a > b) ? a - b : b - a;
}
```

関数の返却値を引数として関数に渡す (つづき)

List 6-3 (2/2) : 二つの整数の 2 乗値の差を求める

```
int main(void)
{
  int x, y;
    ... 略 ...
  printf("xの2乗とyの2乗の差は%dです。\n",
         diff(sqr(x), sqr(y)));
  return 0;
}
```

例: x=4, y = 5 の場合の実行の流れ

1. sqr(4) を呼び出す... 関数値 16 を得る
2. sqr(5) を呼び出す... 関数値 25 を得る
3. diff(16, 25) を呼び出す... 関数値 9 を得る
4. printf(....) を呼び出す... 9 を表示

□ 自作の関数を呼び出す関数 p.149

4 つの整数の最大値を求める

List 6-4 (1/2)：自作の関数で他の関数を呼び出せる

```
#include <stdio.h>

int max2(int a, int b)
{
  return (a > b) ? a : b;
}

int max4(int a, int b, int c, int d)
{
  return max2(max2(a, b), max2(c, d));
}
```

関数内で任意の関数を呼び出せる

- 自作/ライブラリ関数どちらでも OK

自作の関数を呼び出す関数 (つづき)

List 6-4 (2/2) : 自作の関数で他の関数を呼び出せる

```
int main(void)
{
  int n1, n2, n3, n4;
    ... 略 ...
  printf("最も大きい値は%dです。\n",
         max4(n1, n2, n3, n4));
  return 0;
}
```

関数内で任意の関数を呼び出せる

- 自作/ライブラリ関数どちらでも OK

★重要★

□ 値渡し　p.150

x の n 乗 $(= x^n)$ を返す関数 power

- x を n 回掛け合わせた値を返せば良い

List 6-5 (1/2) : べき乗を求める

```
#include <stdio.h>

double power(double x, int n)
{
  double tmp = 1.0;
  for (int i = 1; i <= n; i++) {
    tmp *= x;
  }
  return tmp;
}
```

power 関数を呼び出す main 関数

★重要★

List 6-5 (2/2) : べき乗を求める

```
int main(void)
{
  double a;
  int b;
  printf("aのb乗を求めます。\n");
  printf("実数a:");  scanf("%lf", &a);
  printf("整数b:");  scanf("%d",  &b);
  printf("%.2fの%d乗は%.2fです。\n",
         a, b, power(a, b));
  return 0;
}
```

関数 power：別実装 (1/2)

★重要★

べき乗を求める (別実装)

List 6-6 (部分)

```
double power(double x, int n)
{
  double tmp = 1.0;
  while (n-- > 0)
    tmp *= x;
  return tmp;
}
```

注目点：仮引数 n の値を関数 power 内で書き換えている

- n の値が 0 になれば while 文を終了

関数 power：別実装 (2/2)

★重要★

List 6-5 (部分)：関数 power を呼び出す main 関数

```
int main(void)
{
  double a;
  int b;
      ... 略 ...
  printf("%.2fの%d乗は%.2fです。\n",
         a, b, power(a, b));
  return 0;
}
```

Q：変数 b は関数 power の呼出で値が変わる? (b は 0 になる?)
　関数 power 内で n の値を書き換えてるよ!!!

A: b の値は変わりません.「値渡し (call by value)」だからです.

- 仕組み：実引数の値の計算結果が関数の仮引数に渡される
- 変数そのものが渡されるのではない

値呼び出しなのでこれでも大丈夫

着目点：n が main 関数で power 関数の両方で使用

```
double power(double x, int n)
{
  double tmp = 1.0;
  while (n-- > 0)
    tmp *= x;
  return tmp;
}

int main(void)
{
  double e = 2.718;
  int n = 5;
  printf("%.2fの%d乗は%.2fです。\n",
         e, n, power(e, n));
  return 0;
}
```

- power 関数内で n を書き換え
- その影響は main 関数での n には及ばない

おわり