

Cプログラミング入門

(基幹5クラス)

第9回 関数・数学ライブラリ

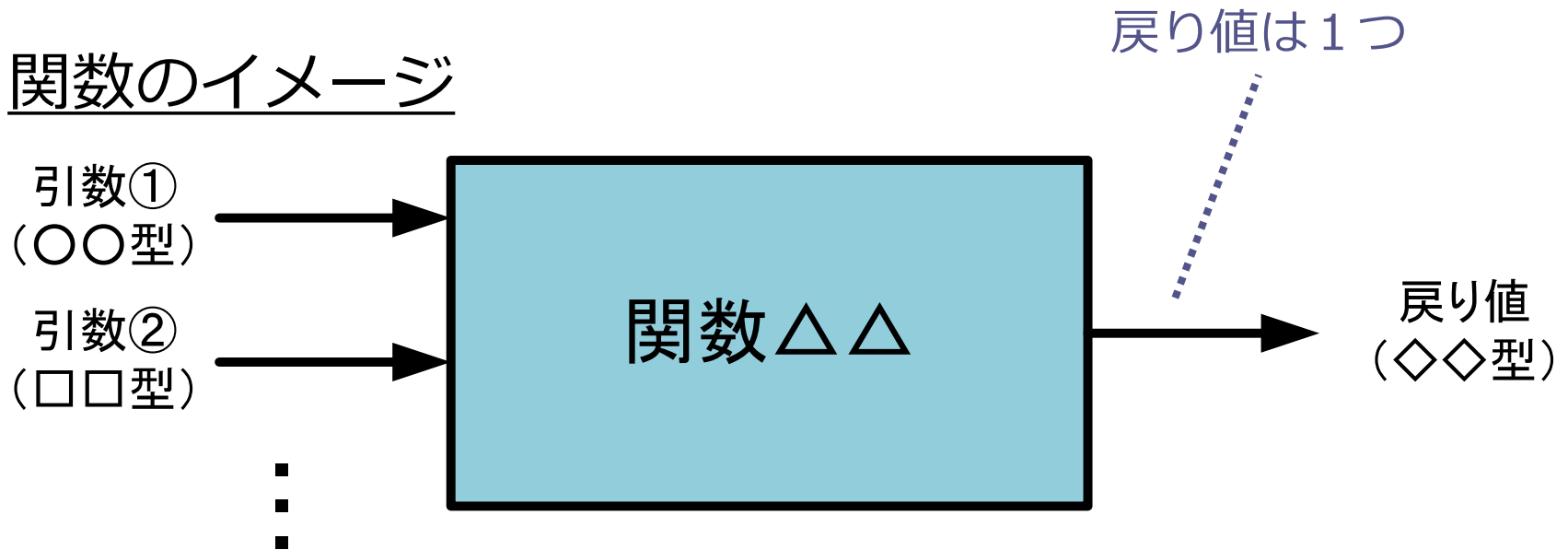
本日の講義・演習項目

- ▶ 授業内演習の解説 ... 第8回講義の解説スライドに記載
- ▶ 関数
- ▶ 数学ライブラリ

関数とは？（１）

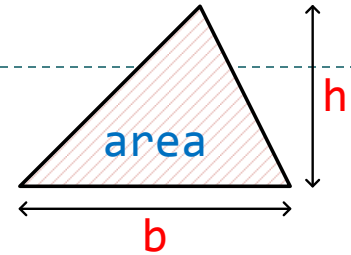
- ▶ 関数：処理のまとめ
- ▶ **関数名**が付く
- ▶ **引数**（入力）と**戻り値**（出力）を持つ
 - ▶ 引数と戻り値には**型**を指定

関数のイメージ



関数とは？（２）

- ▶ Cプログラムにおける関数の記述例
 - ▶ 三角形の面積を計算する関数：triArea



```
double triArea(double b, double h){  
    double area;  
    area = b * h / 2.0;  
    return area;  
}
```

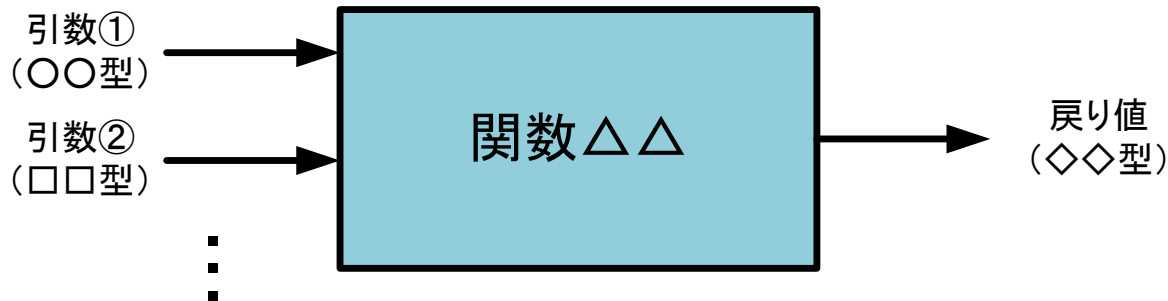
- ▶ Cプログラムにおける関数の使用例

```
double bottom = 2.5, height = 1.5, area;  
area = triArea(bottom, height);
```

関数とは？（３）

▶ 関数の構成

```
戻り値の型 関数名(引数の型 引数①, ...){  
  
    関数内の処理  
  
    return 戻り値;  
}
```



関数とは？（４）

- ▶ Cプログラムは関数の寄せ集めで構成される
- ▶ 関数の**種類**
 - ▶ main関数
 - ▶ 標準で用意されている関数
 - ▶ 自作の関数

関数とは？（５）

▶ main関数

- ▶ プログラムを実行したときに**最初に実行される**関数
- ▶ すべてのCプログラムに必要

```
int main(void){  
    ... } 処理  
    ...  
    return 0;  
}
```

main関数において
戻り値が0であることは
正常終了を意味する

引数なし
(void)

main関数

戻り値 = 0
(int型)

関数とは？（6）

- ▶ 標準で用意されている関数
 - ▶ 使うためにはヘッダファイルのインクルードが必要
 - ▶ `#include <stdio.h>` → printf関数, scanf関数 など
 - ▶ `#include <stdlib.h>` → rand関数 など
 - ▶ `#include <time.h>` → time関数 など
 - ▶ `#include <math.h>` → 様々な数学関数
- ▶ 処理内容・引数・戻り値が予め決まっている

関数とは？（7）

▶ 標準で用意されている関数の例

1. sqrt関数：平方根を計算する関数

```
double x = 5.0;  
double y = sqrt(x);
```



2. rand関数：乱数を生成する関数

```
int num;  
num = rand()%6 + 1;
```



3. printf関数：文字列を標準出力する関数

```
printf("Hello!");
```



関数とは？（８）

▶ 自作の関数

- ▶ 関数名・引数・戻り値・処理内容を自由に決めることができる
- ▶ 通常、main関数より前に記述

▶ 関数を作ると便利になること

- ▶ 共通する処理を一箇所にまとめることができる
- ▶ 機能的に分かりやすい単位で書くことができる

▶ **可読性の高いプログラムを書けるようになるう！**

演習①

- ▶ int型変数 x を端末から入力し, $y = x^3 + 1$ の結果 y を標準出力するプログラム func1.c を作成せよ。
- ▶ ただし, main関数を以下の通り記述した上で, 自作の関数 **func** を #includeとmain関数の間に記述すること

```
#include <stdio.h>

int main(void){
    int x, y;
    printf("Input x: ");
    scanf("%d", &x);

    y = func(x);

    printf("y = x^3 + 1 = %d¥n", y);
    return 0;
}
```

演習①

▶ 自作の関数**func**のヒント

```
#include <stdio.h>
```

```
int func(int x){
```

```
    int y;
```

```
    ?
```

```
    return y;
```

```
}
```



関数**func**本体の記述

```
int main(void){
```

```
    int x, y;
```

```
    printf("Input x: ");
```

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

```
    y = func(x);
```

```
    printf("y = x^3 + 1 = %d\n", y);
```

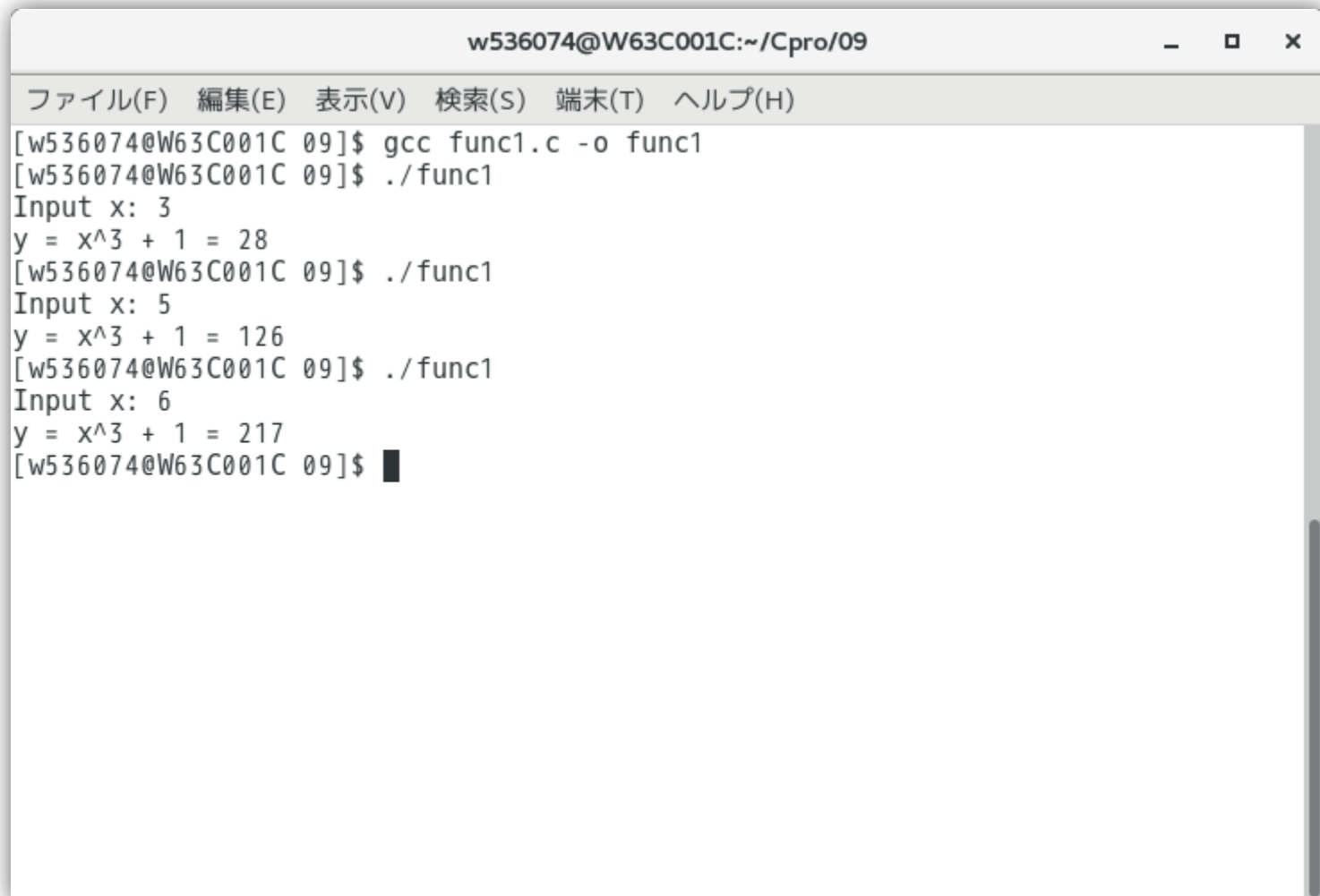
```
    return 0;
```

```
}
```



関数**func**の呼び出し

演習① ~実行結果~



A terminal window titled "w536074@W63C001C:~/Cpro/09" with standard window controls. The menu bar includes "ファイル(F)", "編集(E)", "表示(V)", "検索(S)", "端末(T)", and "ヘルプ(H)". The terminal output shows the compilation and execution of a program named "func1". The program takes an input 'x' and calculates $y = x^3 + 1$. It is executed three times with inputs 3, 5, and 6, resulting in outputs 28, 126, and 217 respectively. The prompt character is a thick black bar.

```
w536074@W63C001C:~/Cpro/09
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 09]$ gcc func1.c -o func1
[w536074@W63C001C 09]$ ./func1
Input x: 3
y = x^3 + 1 = 28
[w536074@W63C001C 09]$ ./func1
Input x: 5
y = x^3 + 1 = 126
[w536074@W63C001C 09]$ ./func1
Input x: 6
y = x^3 + 1 = 217
[w536074@W63C001C 09]$ █
```

変数のスコープ

- ▶ スコープ：有効範囲
- ▶ 関数の引数に指定した変数・関数内で宣言した変数はローカル変数と呼ばれ、関数内のみで有効
- ▶ 別の関数で宣言された変数どうしは（たとえ名前が同じでも）別物として扱われる
 - ▶ main関数内のxとfunc関数内のxは別物
 - ▶ main関数内のyとfunc関数内のyは別物

演習②

- ▶ 以下を参考に，端末から入力した3つの整数の**最大値**を標準出力するプログラム func2.c を作成せよ。
- ▶ **cp /share/func2.c ./** ... 講義中のみ実行できます

```
#include <stdio.h>

int max3(int a, int b, int c){
    

?


}

int main(void){
    int a, b, c, max;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);
    printf("c = "); scanf("%d", &c);

    max = max3(a, b, c);

    printf("Max is %d\n", max);
    return 0;
}
```

演習② ~実行結果~

```
w536074@W63C001C:~/Cpro/09
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 09]$ gcc func2.c -o func2
[w536074@W63C001C 09]$ ./func2
a = 2
b = 5
c = 8
Max is 8
[w536074@W63C001C 09]$ ./func2
a = 15
b = 69
c = 39
Max is 69
[w536074@W63C001C 09]$ ./func2
a = -8
b = -9
c = -100
Max is -8
[w536074@W63C001C 09]$ █
```


演習③

- ▶ 以下を参考に，端末から入力した2つの整数の**差の絶対値**を標準出力するプログラム func3.c を作成せよ。
- ▶ **cp /share/func3.c ./** ... 講義中のみ実行できます

```
#include <stdio.h>

int diff2(int a, int b){
    

?


}

int main(void){
    int a, b, diff;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);

    diff = diff2(a, b);

    printf("|a - b| = %d\n", diff);
    return 0;
}
```

数学ライブラリ (math.h) を
使用しないで書くこと

演習③ ~実行結果~



```
w536074@W63C001C:~/Cpro/09
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 09]$ gcc func3.c -o func3
[w536074@W63C001C 09]$ ./func3
a = 5
b = 3
|a - b| = 2
[w536074@W63C001C 09]$ ./func3
a = 19
b = 26
|a - b| = 7
[w536074@W63C001C 09]$ ./func3
a = -8
b = -4
|a - b| = 4
[w536074@W63C001C 09]$
```

演習④

- ▶ 以下を参考に，端末から入力した2つの整数の**平均値**を標準出力するプログラム func4.c を作成せよ。
- ▶ **cp /share/func4.c ./** ... 講義中のみ実行できます

```
#include <stdio.h>

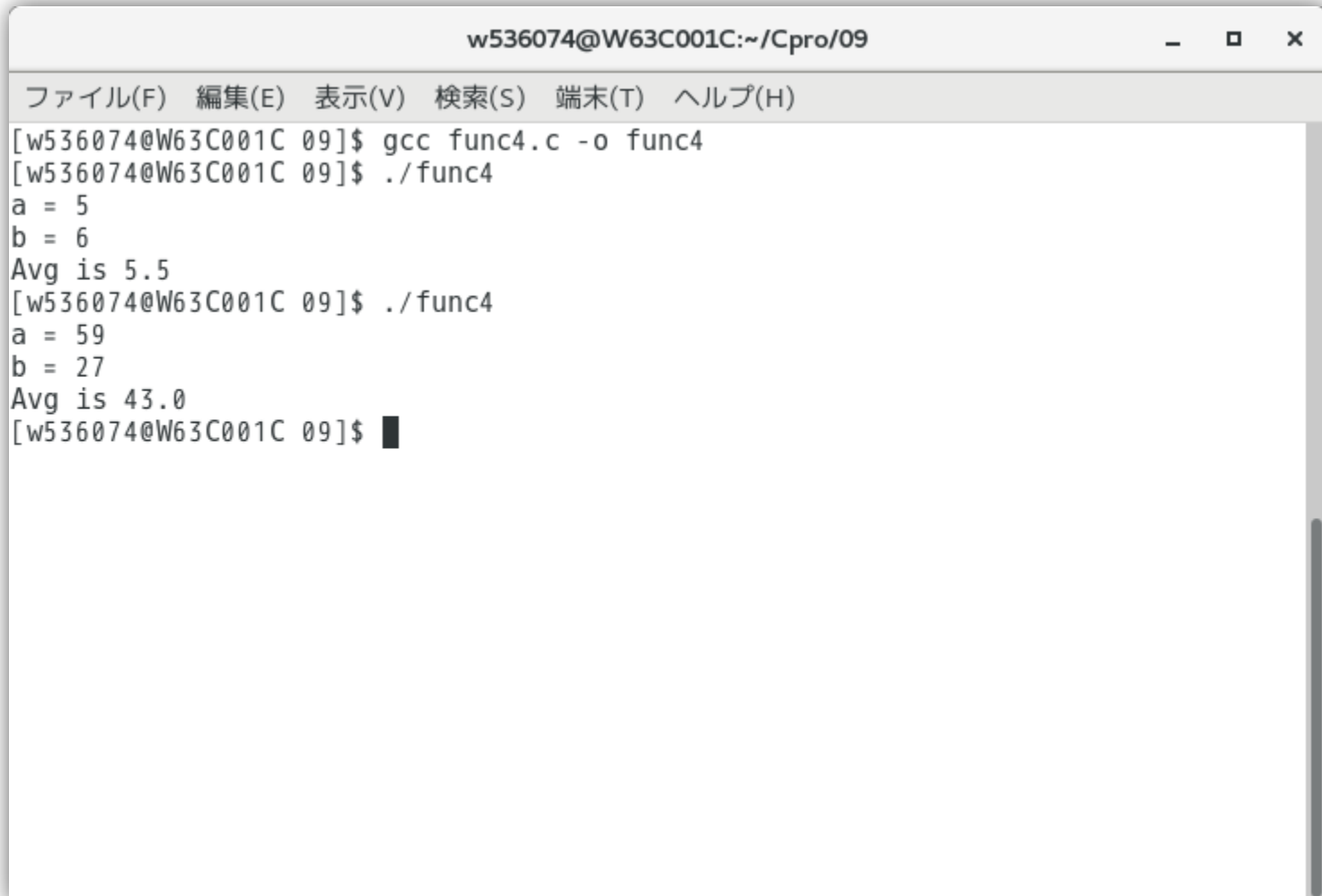
? avg2(int a, int b){
    ?
}

int main(void){
    int a, b;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);

    double avg = avg2(a, b);

    printf("Avg is %.1f¥n", avg);
    return 0;
}
```

演習④ ~実行結果~



```
w536074@W63C001C:~/Cpro/09
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 09]$ gcc func4.c -o func4
[w536074@W63C001C 09]$ ./func4
a = 5
b = 6
Avg is 5.5
[w536074@W63C001C 09]$ ./func4
a = 59
b = 27
Avg is 43.0
[w536074@W63C001C 09]$
```

演習⑤

- ▶ 端末から入力した整数 x, y ($y > 0$ とする) にもとづいて x^y を計算するプログラム func5.c を作成せよ。
- ▶ **cp /share/func5.c ./** ... 講義中のみ実行できます

```
#include <stdio.h>

int pow2(int x, int y){
    

?


}

int main(void){
    int x, y, p;
    printf("x = "); scanf("%d", &x);
    printf("y = "); scanf("%d", &y);

    p = pow2(x, y);

    printf("x^y = %d\n", p);
    return 0;
}
```

数学ライブラリ (math.h) を
使用しないで書くこと

演習⑤ ~実行結果~



```
w536074@W63C001C:~/Cpro/09
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 09]$ gcc func5.c -o func5
[w536074@W63C001C 09]$ ./func5
x = 2
y = 10
x^y = 1024
[w536074@W63C001C 09]$ ./func5
x = 5
y = 3
x^y = 125
[w536074@W63C001C 09]$
```

本日の講義・演習項目

- ▶ 授業内演習の解説 ... 第8回講義の解説スライドに記載
- ▶ 関数
- ▶ **数学ライブラリ**

数学ライブラリ

数学関数の例

<code>fabs(x)</code>	絶対値	<code>exp(x)</code>	指数
<code>log(x)</code>	自然対数	<code>log10(x)</code>	常用対数
<code>pow(x,y)</code>	x^y	<code>sqrt(x)</code>	平方根
<code>floor(x)</code>	切り捨て	<code>ceil(x)</code>	切り上げ
<code>sin(x)</code>	<code>sin</code>	<code>cos(x)</code>	<code>cos</code>
<code>tan(x)</code>	<code>tan</code>		

数学定数の例

<code>M_PI</code>
π
<code>M_E</code>
e

- ▶ `#include <math.h>` をプログラム先頭に記述
- ▶ コンパイル時に `-lm` (エル・エム) オプションを付加
 - ▶ 例) `gcc test.c -lm -o test`

演習⑥

- ▶ 端末から入力した2つの整数の**差の絶対値**を標準出力するプログラム math1.c を数学ライブラリ内の関数 fabs を用いて作成せよ。(演習③と同じ動作)

```
#include <stdio.h>
#include <math.h>
int main(void){
    int a, b, diff;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);

    diff = 

    printf("|a - b| = %d\n", diff);
    return 0;
}
```



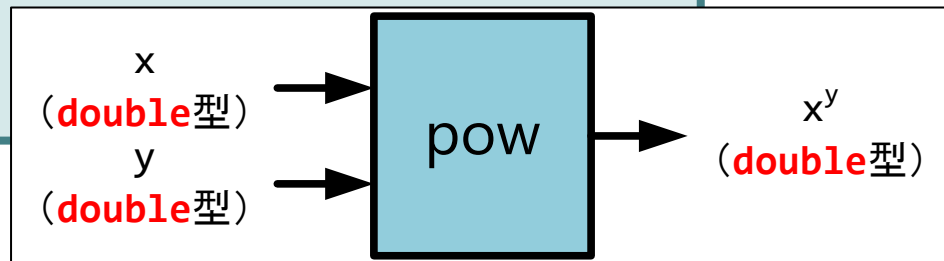
演習⑦

- ▶ 端末から入力した整数 x, y ($y > 0$ とする) にもとづいて x^y を計算するプログラム math2.c を数学ライブラリ内の関数 pow を用いて作成せよ。(演習⑤と同じ動作)

```
#include <stdio.h>
#include <math.h>
int main(void){
    int x, y, p;
    printf("x = "); scanf("%d", &x);
    printf("y = "); scanf("%d", &y);

    p = 

    printf("x^y = %d\n", p);
    return 0;
}
```



演習⑧（発展問題）

- ▶ 円周率の値をプログラムで求めるための方法のひとつとして、**ライプニッツの公式**を用いる方法がある。

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \lim_{N \rightarrow \infty} \sum_{i=0}^N \frac{(-1)^i}{2i+1} = \frac{\pi}{4}$$

- ▶ 本公式を用いて近似的に円周率の値を求め、それを数学ライブラリ内の定数M_PIと比較することで誤差が 10^{-5} 未満となるような最小の*N*を求めたい。
- ▶ *N*を求めるプログラム Leibniz.c を作成せよ。

演習⑧－ 1

$$4 \times \sum_{i=0}^N \frac{(-1)^i}{2i+1}$$

- ▶ 端末から入力した N をもとに、円周率の近似値を求め、誤差を計算するプログラムを作成せよ。
- ▶ ここでは、近似値と定数M_PIの差の絶対値を誤差と定義する

```
#include <stdio.h>
#include <math.h>
int main(void){
    int N;
    printf("N = "); scanf("%d", &N);

    int i;
    double sum = 0.0; error;

    printf("Approximation of pi = %.12f¥n", 4*sum);
    printf("Error = %.12f¥n", error);
    return 0;
}
```

演習⑧ – 1 ~実行結果~

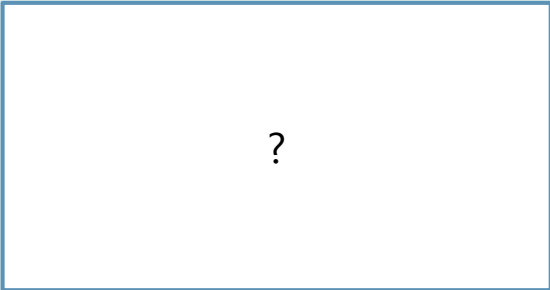
```
w536074@W63C001C:~/Cpro/09
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 09]$ gcc Leibniz1.c -o Leibniz1
[w536074@W63C001C 09]$ ./Leibniz1
N = 10
Approximation of pi = 3.232315809406
Error = 0.090723155816
[w536074@W63C001C 09]$ ./Leibniz1
N = 100
Approximation of pi = 3.151493401071
Error = 0.009900747481
[w536074@W63C001C 09]$ ./Leibniz1
N = 1000
Approximation of pi = 3.142591654340
Error = 0.000999000750
[w536074@W63C001C 09]$ ./Leibniz1
N = 10000
Approximation of pi = 3.141692643591
Error = 0.000099990001
[w536074@W63C001C 09]$
```

演習⑧ – 2

- ▶ 演習⑧ – 1 のプログラムを書き換え，誤差が 10^{-5} 未満となるような最小の **N** を求めるプログラムを作成せよ。

```
#include <stdio.h>
#include <math.h>
int main(void){
    int i;
    double sum = 0.0; error;


    





    printf("N = %d\n", ? );
    printf("Approximation of pi = %.12f\n", 4*sum);
    printf("Error = %.12f\n", error);
    return 0;
}
```

演習⑧ – 2 ~実行結果~



```
w536074@W63C001C:~/Cpro/09
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 09]$ gcc Leibniz2.c -o Leibniz2
[w536074@W63C001C 09]$ ./Leibniz2
N = 100000
Approximation of pi = 3.141602653490
Error = 0.000009999900
[w536074@W63C001C 09]$
```

キーワード，次回の講義

- ▶ 本日のキーワード：
- ▶ 次回は6/21
- ▶ 次回講義までに予習ビデオ「第11回 ポインタ」を視聴し，各自プログラミング実習