

# Cプログラミング入門

## (基幹5クラス)

第10回 ポインタ

# 本日の講義・演習項目

---

- ▶ ポインタ
  - ▶ アドレス演算子「&」
  - ▶ ポインタ型変数
  - ▶ 間接参照演算子「\*」
- ▶ 関数とポインタ

# アドレス演算子「&」

- ▶ 変数：数値や文字を記憶しておく箱
- ▶ 変数の**アドレス**：PCメモリ上での変数の**位置**
  - ▶ 変数の宣言時に自動で割り当てられる

```
int a; ... 変数aの宣言
```

変数

a

変数のアドレス



← **&a**

- ▶ アドレスの表示

```
printf("%p", &a); ... 変数aのアドレス表示
```

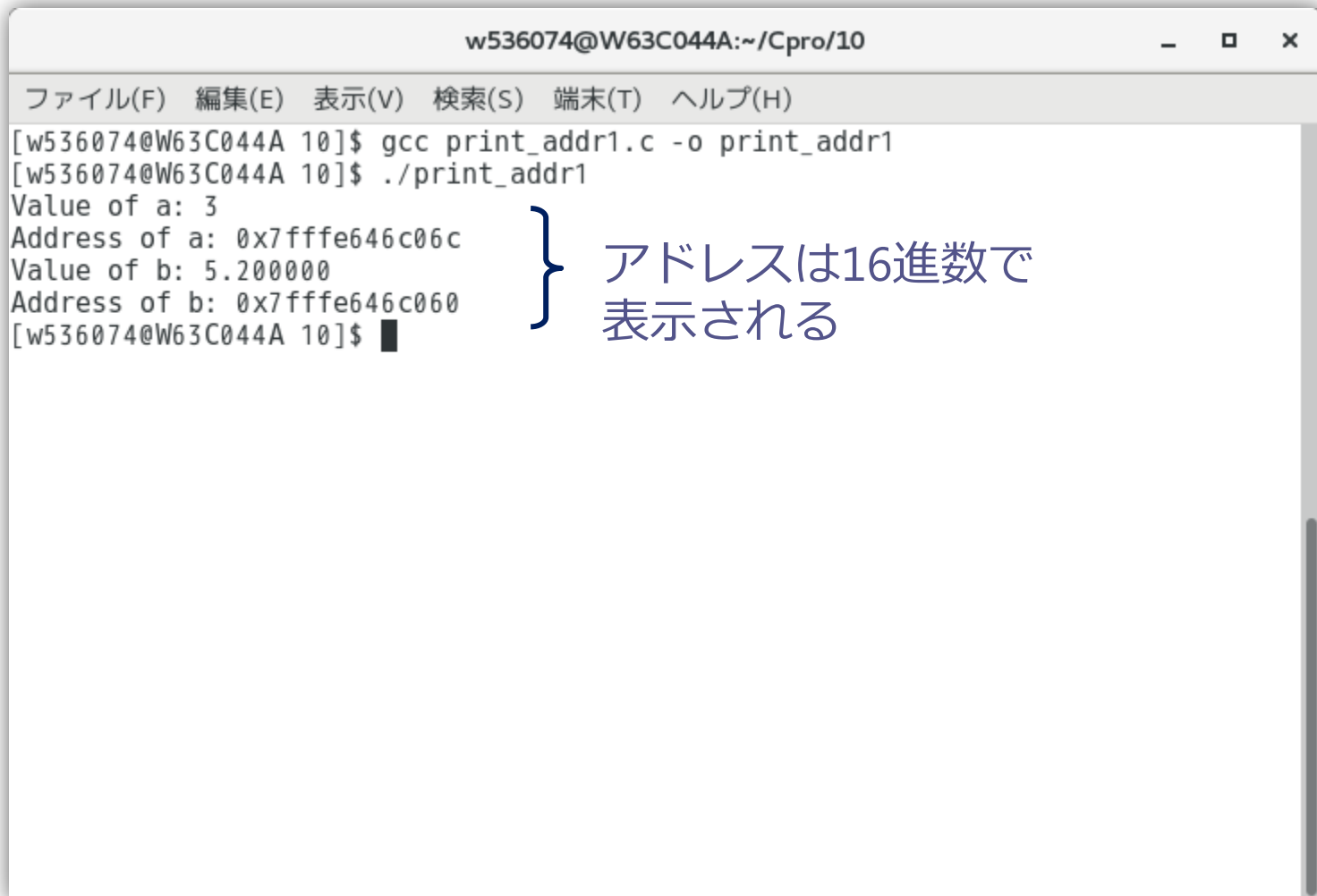
## 演習①

- ▶ int型変数a, double型変数bを宣言し, それぞれの値とアドレスを表示するプログラム print\_addr1.c を作成せよ。

```
#include <stdio.h>
int main(void){
    int a = 3;
    double b = 5.2;
    printf("Value of a: [?] %n", [?]);
    printf("Address of a: [?] %n", [?]);
    printf("Value of b: [?] %n", [?]);
    printf("Address of b: [?] %n", [?]);
    return 0;
}
```

# 演習① ~実行例~

---



```
w536074@W63C044A:~/Cpro/10
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C044A 10]$ gcc print_addr1.c -o print_addr1
[w536074@W63C044A 10]$ ./print_addr1
Value of a: 3
Address of a: 0x7fffe646c06c
Value of b: 5.200000
Address of b: 0x7fffe646c060
[w536074@W63C044A 10]$
```

} アドレスは16進数で表示される

# ポインタ型変数

- ▶ ポインタ型変数：変数のアドレスを記憶するための箱

```
int *pa;      ... int型変数のアドレスを記憶する箱を用意  
double *pb;  ... double型変数のアドレスを記憶する箱を用意
```

- ▶ ポインタ型変数への代入

```
pa = &a;      ... paはint型変数のアドレスを記憶できる  
pb = &b;      ... pbはdouble型変数のアドレスを記憶できる
```

## 演習②

---

- ▶ int型変数a, double型変数bを宣言し, それぞれのアドレスを表示するプログラム print\_addr2.c を作成せよ。
- ▶ 次スライドを参考に, 二通りの方法でアドレスを表示させてみる。
  - ▶ 変数a, bを用いた表示 (演習①と同じ方法)
  - ▶ ポインタ型変数pa, pbを用いた表示

## 演習②

```
#include <stdio.h>
int main(void){
    int a = 3, *pa;
    double b = 5.2, *pb;
    pa = &a;
    pb = &b;
    printf("Address of a: [?] ¥n", [?] );
    printf("Address of a: [?] ¥n", [?] );
    printf("Address of b: [?] ¥n", [?] );
    printf("Address of b: [?] ¥n", [?] );
    return 0;
}
```

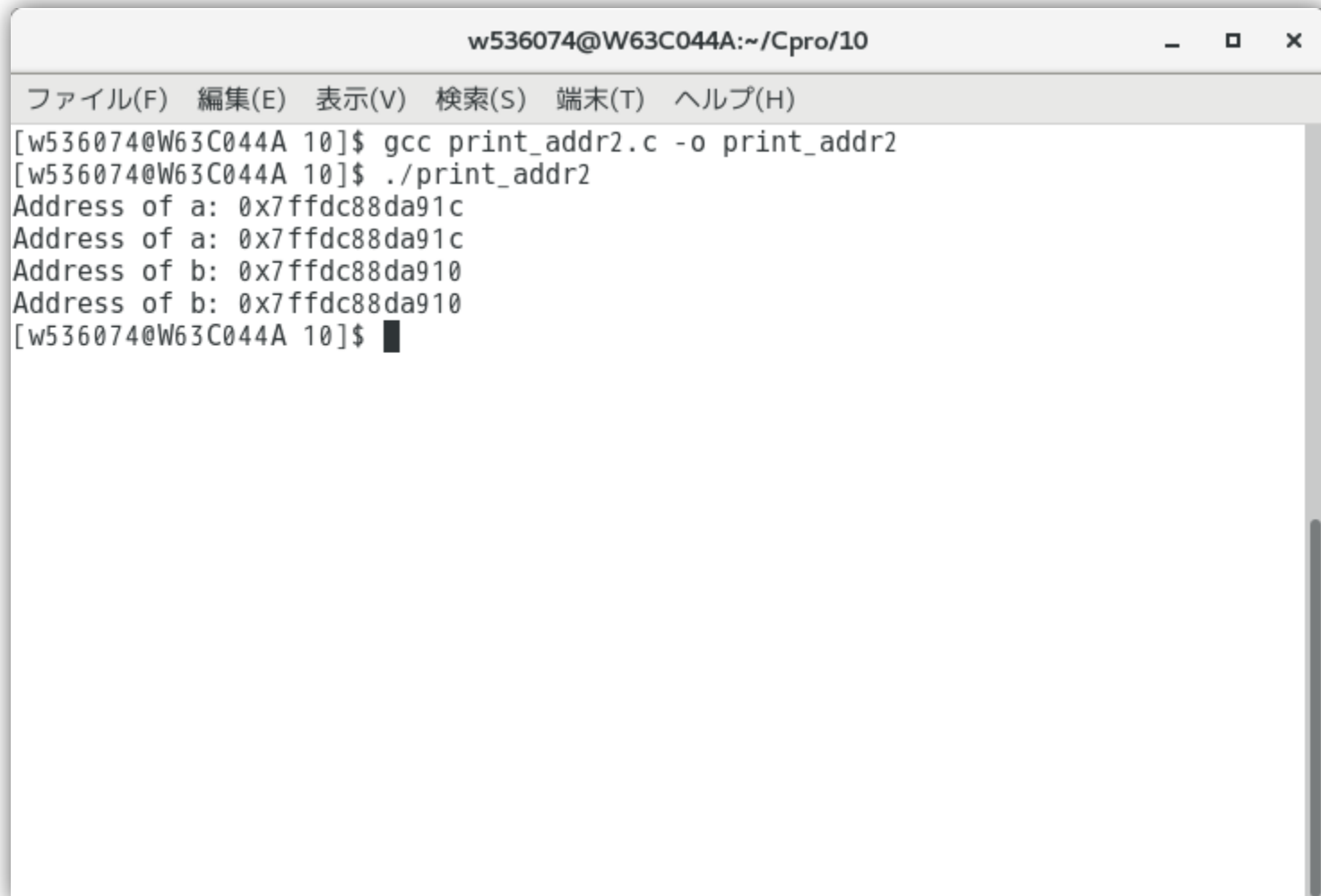
アドレス  
演算子

ポインタ  
型変数



## 演習② ~実行例~

---



A terminal window titled "w536074@W63C044A:~/Cpro/10" with standard window controls. The menu bar contains "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 "print\_addr2".

```
w536074@W63C044A:~/Cpro/10
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C044A 10]$ gcc print_addr2.c -o print_addr2
[w536074@W63C044A 10]$ ./print_addr2
Address of a: 0x7ffdc88da91c
Address of a: 0x7ffdc88da91c
Address of b: 0x7ffdc88da910
Address of b: 0x7ffdc88da910
[w536074@W63C044A 10]$
```

# 間接参照演算子「\*」

- ▶ ポインタ型変数が指し示す変数の**値**にアクセス

```
int a = 3, *pa;
```

```
pa = &a;
```

```
*pa = 5;    ... この「*」が間接参照演算子
```

変数

a



\*pa

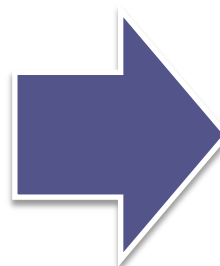
変数のアドレス

&a

||

pa

間接参照



a



\*pa

\*pa = 5;

## 演習③

---

- ▶ int型変数a, double型変数bを宣言し, それぞれの値を表示するプログラム print\_value.c を作成せよ。
- ▶ 次スライドを参考に, 二通りの方法で値を表示させてみる。
  - ▶ 変数a, bを用いた表示 (演習①と同じ方法)
  - ▶ ポインタ型変数pa, pbと間接参照演算子を用いた表示

## 演習③

```
#include <stdio.h>
int main(void){
    int a = 3, *pa;
    double b = 5.2, *pb;
    pa = &a;
    pb = &b;
    printf("Value of a: [?] ¥n", [?]);
    printf("Value of a: [?] ¥n", [?]);
    printf("Value of b: [?] ¥n", [?]);
    printf("Value of b: [?] ¥n", [?]);
    return 0;
}
```

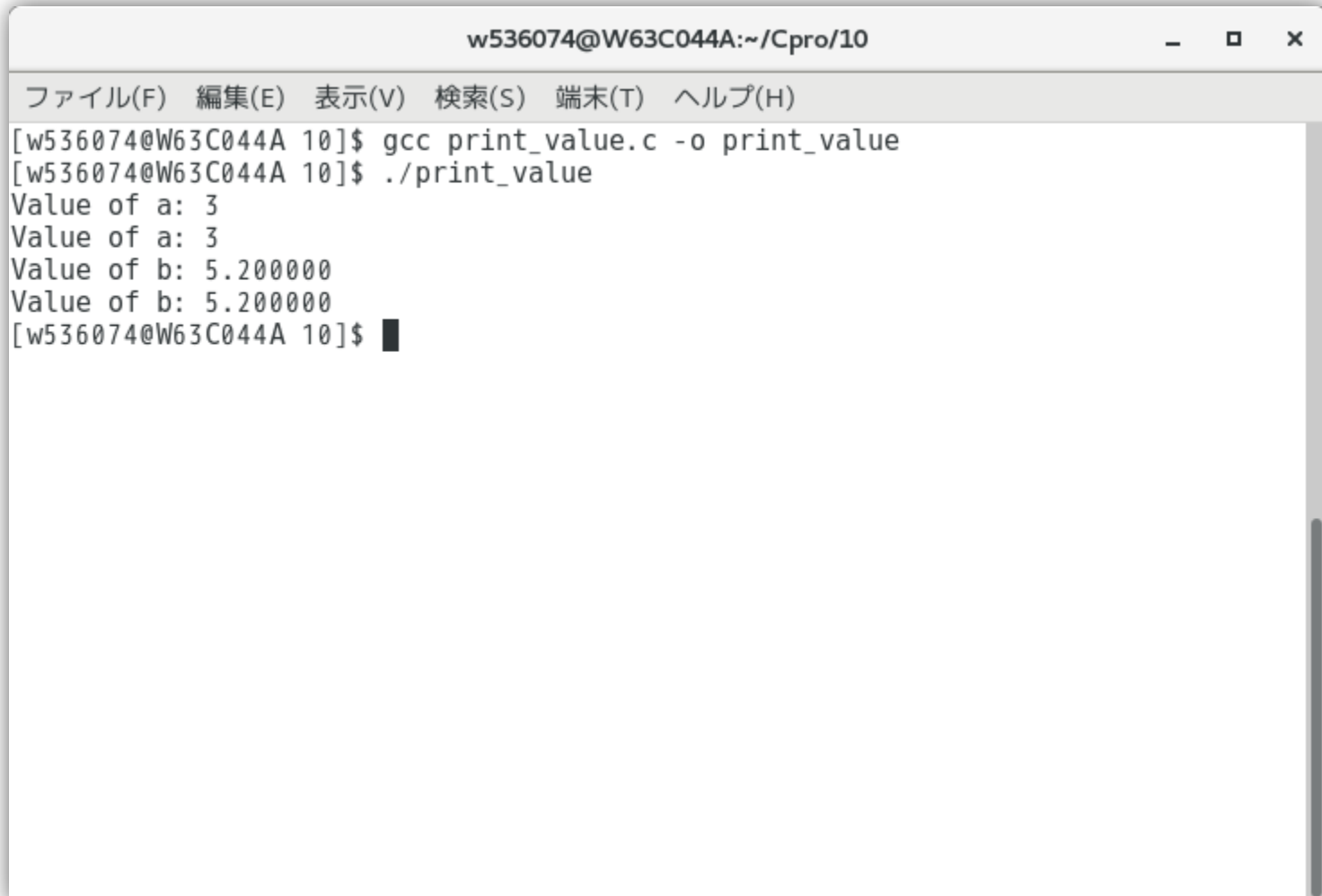
変数

ポインタ型変数と  
間接参照演算子



## 演習③ ~実行例~

---



```
w536074@W63C044A:~/Cpro/10
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C044A 10]$ gcc print_value.c -o print_value
[w536074@W63C044A 10]$ ./print_value
Value of a: 3
Value of a: 3
Value of b: 5.200000
Value of b: 5.200000
[w536074@W63C044A 10]$
```

# 本日の講義・演習項目

---

## ▶ ポインタ

- ▶ アドレス演算子「&」
- ▶ ポインタ型変数
- ▶ 間接参照演算子「\*」

## ▶ 関数とポインタ

スライド15とスライド16のプログラムの違いを理解することが**超重要**です！！

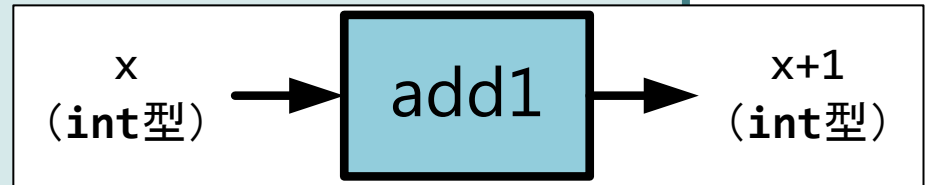
# 関数への「値渡し」

## ▶ プログラム例

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

```
int add1(int x){  
    int y;  
    y = x + 1;  
    return y;  
}
```

```
int main(void){  
    int x;  
    printf("x = "); scanf("%d", &x);  
  
    x = add1(x);  
  
    printf("x = %d\n", x);  
    return 0;  
}
```



# 関数への「参照渡し」

## ▶ プログラム例

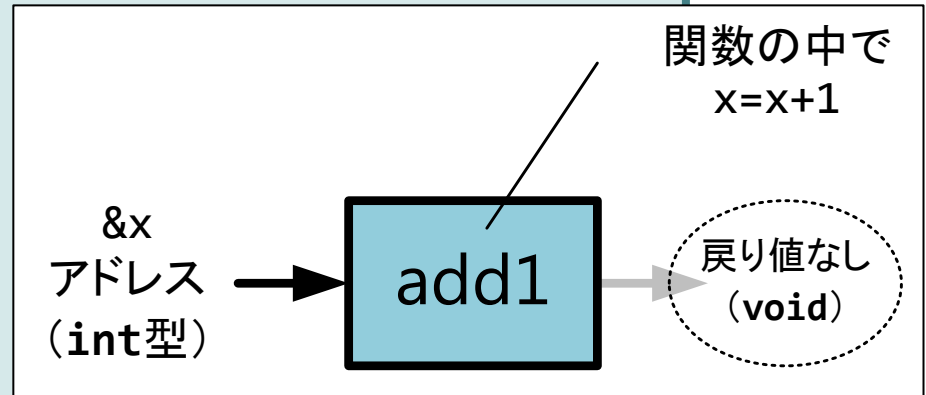
```
#include <stdio.h>

void add1(int *px){
    *px = *px + 1;
    return;
}
```

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

    add1(&x);

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





# 関数への「参照渡し」

## ▶ プログラム例

px=変数xのアドレス

```
#include <stdio.h>

void add1(int *px){
    *px = *px + 1;
    return;
}
```

アドレスを受け取る  
(ポインタ型変数)

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

    add1(&x);

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

アドレスを渡す

&x  
アドレス  
(int型)

add1

関数の中で  
x=x+1

戻り値なし  
(void)

# 関数への「参照渡し」

## ▶ プログラム例

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

```
void add1(int *px){  
    *px = *px + 1;  
    return;  
}
```

間接参照  
→ pxが指す変数  
に対する処理

```
int main(void){  
    int x;  
    printf("x = "); scanf("%d", &x);  
  
    add1(&x);  
  
    printf("x = %d\n", x);  
    return 0;  
}
```

&x  
アドレス  
(int型)

add1

関数の中で  
x=x+1

戻り値なし  
(void)

# 関数への「参照渡し」

## ▶ プログラム例

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

```
void add1(int *px){  
    *px = *px + 1;  
    return;  
}
```

何も返す必要はない

```
int main(void){  
    int x;  
    printf("x = "); scanf("%d", &x);  
  
    add1(&x);  
  
    printf("x = %d\n", x);  
    return 0;  
}
```

&x  
アドレス  
(int型)

add1

関数の中で  
x=x+1

戻り値なし  
(void)

# 参照渡しをすると何が嬉しいか？

- ▶ 関数の引数に指定した変数を関数内で変更することができる
  - ▶ スライド16のプログラム例の通り
- ▶ 関数の戻り値を実質的に増やすことができる
  - ▶ 値渡し

```
max = max3(a, b, c);  
min = min3(a, b, c);
```

... 戻り値は1個のみ

... 戻り値は1個のみ

- ▶ 参照渡し

```
comp3(a, b, c, &max, &min);
```

... 戻り値は実質2個

## 演習④

- ▶ 端末から入力した 3 つの整数の**最大値と最小値**を標準出力するプログラム pfunc1.c を作成せよ。

```
#include <stdio.h>

void comp3(int a, int b, int c, int *pmax, int *pmin){
    

?


    return;
}

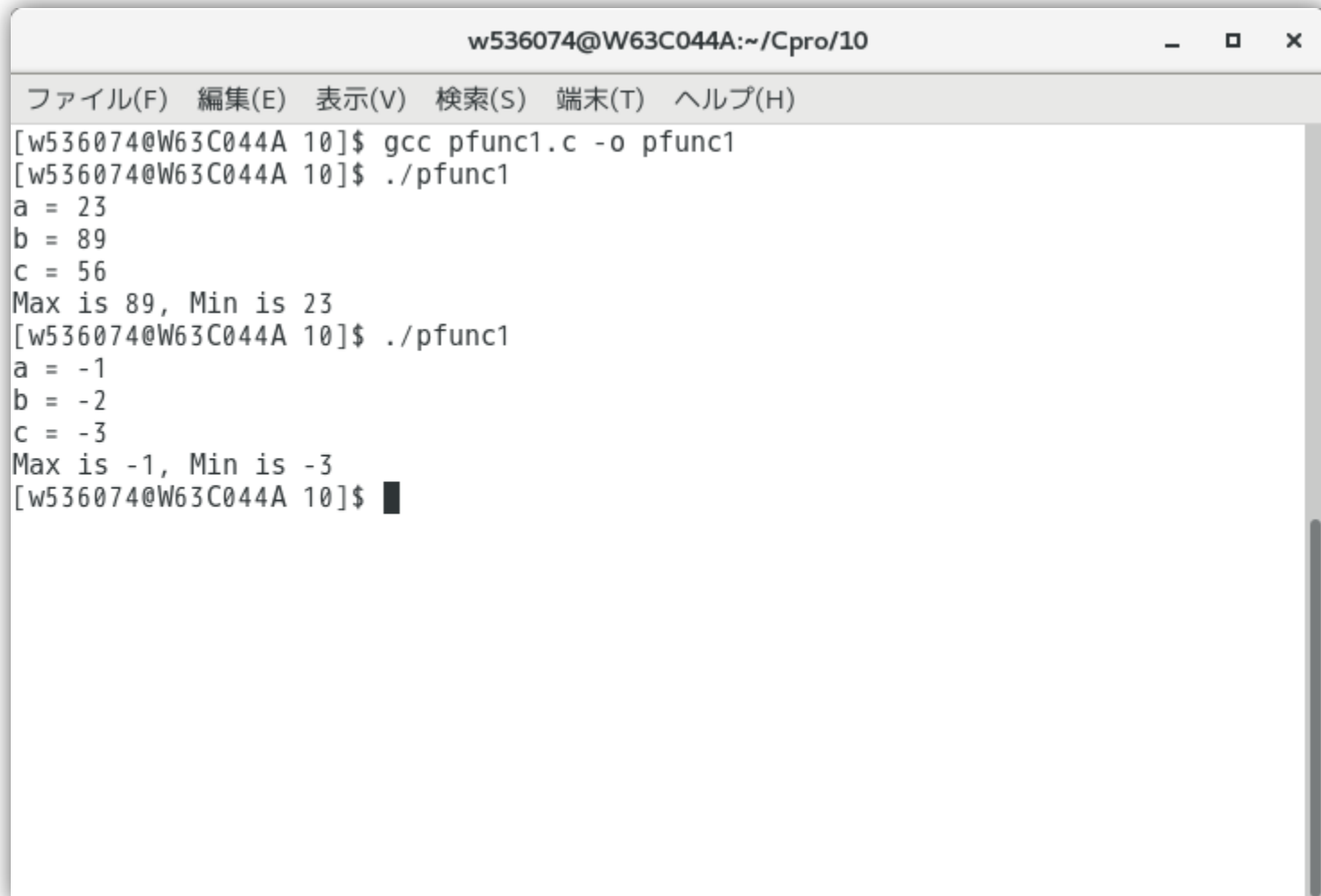
int main(void){
    int a, b, c, max, min;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);
    printf("c = "); scanf("%d", &c);

    comp3(a, b, c, &max, &min);

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

## 演習④ ~実行例~

---



A terminal window titled "w536074@W63C044A:~/Cpro/10" with standard window controls. The menu bar contains "ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)". The terminal shows the compilation and execution of a C program named "pfunc1.c".

```
w536074@W63C044A:~/Cpro/10
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C044A 10]$ gcc pfunc1.c -o pfunc1
[w536074@W63C044A 10]$ ./pfunc1
a = 23
b = 89
c = 56
Max is 89, Min is 23
[w536074@W63C044A 10]$ ./pfunc1
a = -1
b = -2
c = -3
Max is -1, Min is -3
[w536074@W63C044A 10]$
```

## 演習⑤

- ▶ 端末から入力した 2 つの整数を**入れ替えて**標準出力するプログラム pfunc2.c を作成せよ。

```
#include <stdio.h>

void swap(int *pa, int *pb){
    

?


    return;
}

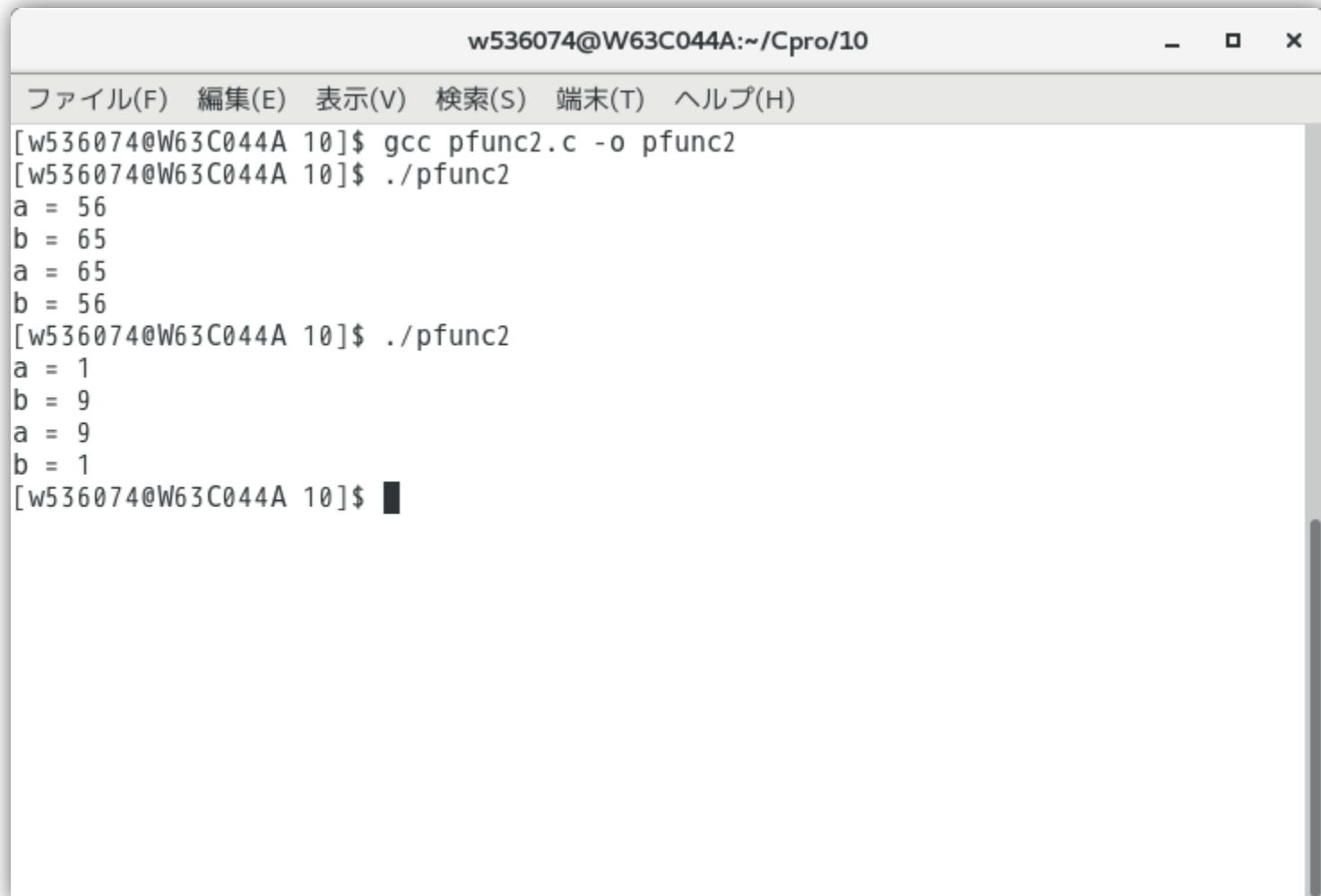
int main(void){
    int a, b;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);

    swap(&a, &b);

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

## 演習⑤ ~実行例~

---



```
w536074@W63C044A:~/Cpro/10
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C044A 10]$ gcc pfunc2.c -o pfunc2
[w536074@W63C044A 10]$ ./pfunc2
a = 56
b = 65
a = 65
b = 56
[w536074@W63C044A 10]$ ./pfunc2
a = 1
b = 9
a = 9
b = 1
[w536074@W63C044A 10]$
```



## 演習⑥

- ▶ 端末から入力した 3 つの整数を**降順に並び替えて**標準出力するプログラム pfunc3.c を作成せよ。

```
#include <stdio.h>

void sort3(){
    
}

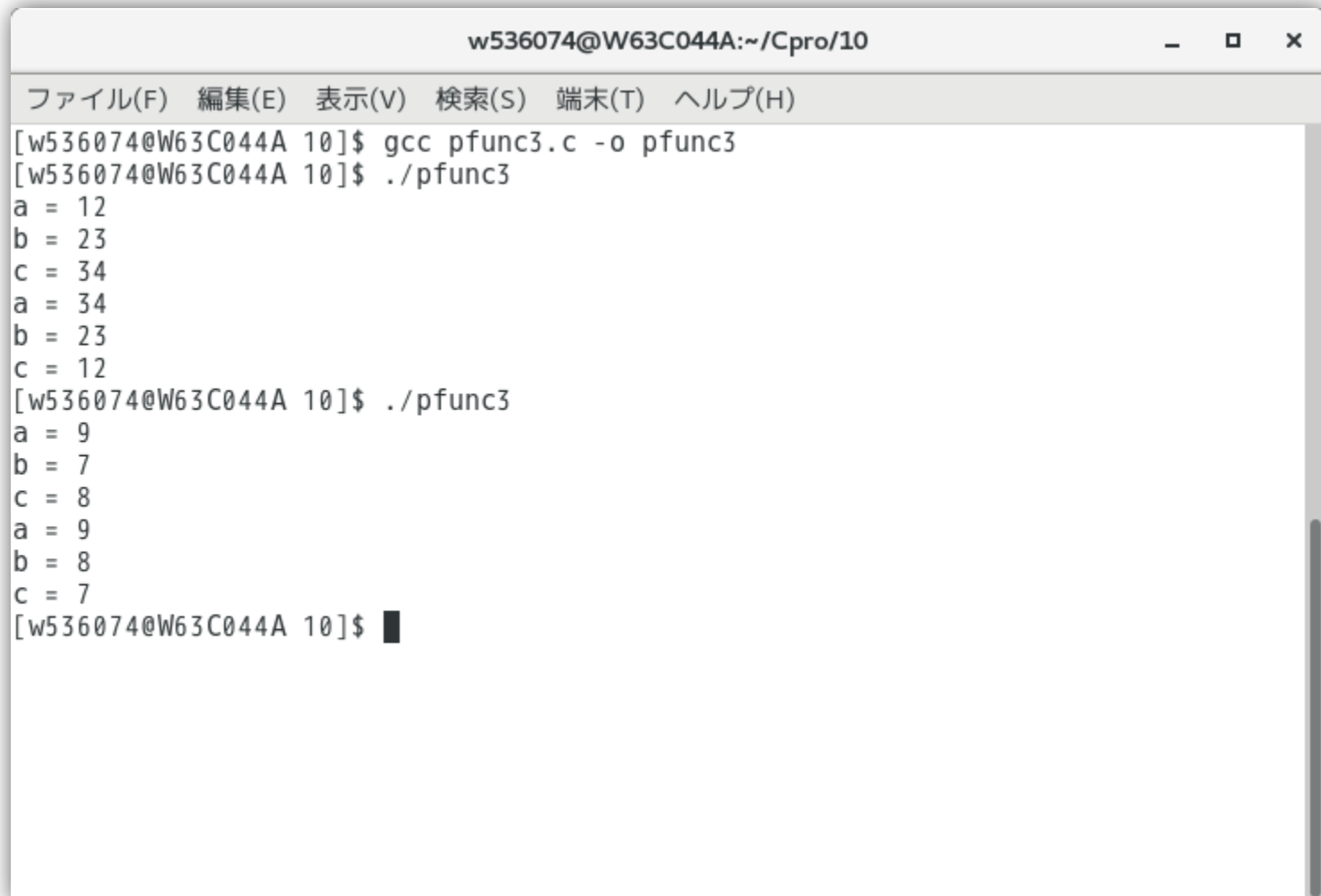
int main(void){
    int a, b, c;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);
    printf("c = "); scanf("%d", &c);

    sort3(&a, &b, &c);

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

## 演習⑥ ~実行例~

---



```
w536074@W63C044A:~/Cpro/10
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C044A 10]$ gcc pfunc3.c -o pfunc3
[w536074@W63C044A 10]$ ./pfunc3
a = 12
b = 23
c = 34
a = 34
b = 23
c = 12
[w536074@W63C044A 10]$ ./pfunc3
a = 9
b = 7
c = 8
a = 9
b = 8
c = 7
[w536074@W63C044A 10]$
```

# キーワード，次回の講義

---

- ▶ 本日のキーワード：
- ▶ 次回は6/28
- ▶ 次回講義までに予習ビデオ「第12回 文字と文字列」を視聴し，各自プログラミング実習
- ▶ 次回講義の最後に課題②を出題（×切：7/8）

## 演習⑦－ 1 （余力がある人向け）

- ▶ 端末から入力したdouble型変数  $a \geq 0$  の平方根  $\sqrt{a}$  を求めるプログラムを数学ライブラリを用いて作成せよ。

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

int main(void){
    double a, ra;
    printf("a = "); scanf("%lf", &a);

    printf("square root of a = %f\n", ra);
    return 0;
}
```

## 演習⑦－ 2 （余力がある人向け）

- ▶ 端末から入力したdouble型変数  $a \geq 0$  の平方根  $\sqrt{a}$  を求める関数 **double my\_sqrt(double a)** を自作せよ。
  - ▶ 作成したら演習⑦－ 1 と実行結果を比較してみる
- ▶ 以下の漸化式（**ニュートン法**）を用いて  $\sqrt{a}$  の近似値を求めることができる。

$$x_{n+1} = \frac{1}{2} \left( x_n + \frac{2}{x_n} \right), \quad x_1 = 1.0$$

- ▶ 補足
  - ▶  $x_1 = 1.0$  としているが,  $x_1 > 0$  の任意の実数を指定可能
  - ▶ 収束条件は  $|x_{n+1} - x_n| < 0.000001$  とする

## 演習⑦－ 3 （余力がある人向け）

---

- ▶ 端末から入力したdouble型変数  $a \geq 0$  の三乗根  $\sqrt[3]{a}$  を求める関数 `double my_cube_root(double a)` を自作し、`pow(a, 1/3.0)` の結果と比較せよ。
- ▶ ニュートン法について各自で調査し、演習⑦－ 2 と同じような漸化式を導出する。

??