

# Cプログラミング入門

## (基幹5クラス)

第11回 ポインタ, 演習

# 本日の講義・演習項目

---

## ▶ ポインタ（復習）

- ▶ アドレス演算子「&」 ... 変数のアドレスを知る
- ▶ ポインタ型変数 ... 変数のアドレスを記憶する
- ▶ 間接参照演算子「\*」 ... アドレスから変数を知る
- ▶ 関数とポインタ

## ▶ 配列とポインタ

# ポインタ（復習）

```
int a = 3, *pa;  
pa = &a;  
*pa = 5;
```

... **a**は**int型**変数, **pa**は**ポインタ型**変数

... **pa**に**&a** (**aのアドレス**) を代入

... **pa(=&a)**が指す**変数の値**を5に書き換え

変数

a



**\*pa**

変数のアドレス

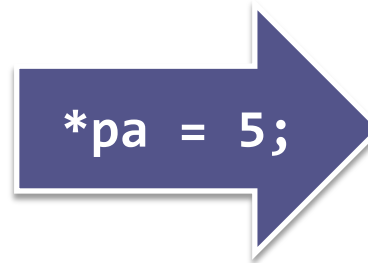
**&a**

||

**pa**

ポインタ型変数

間接参照



変数

a



**\*pa**

# 関数とポインタ（復習）

## ▶ 値渡し

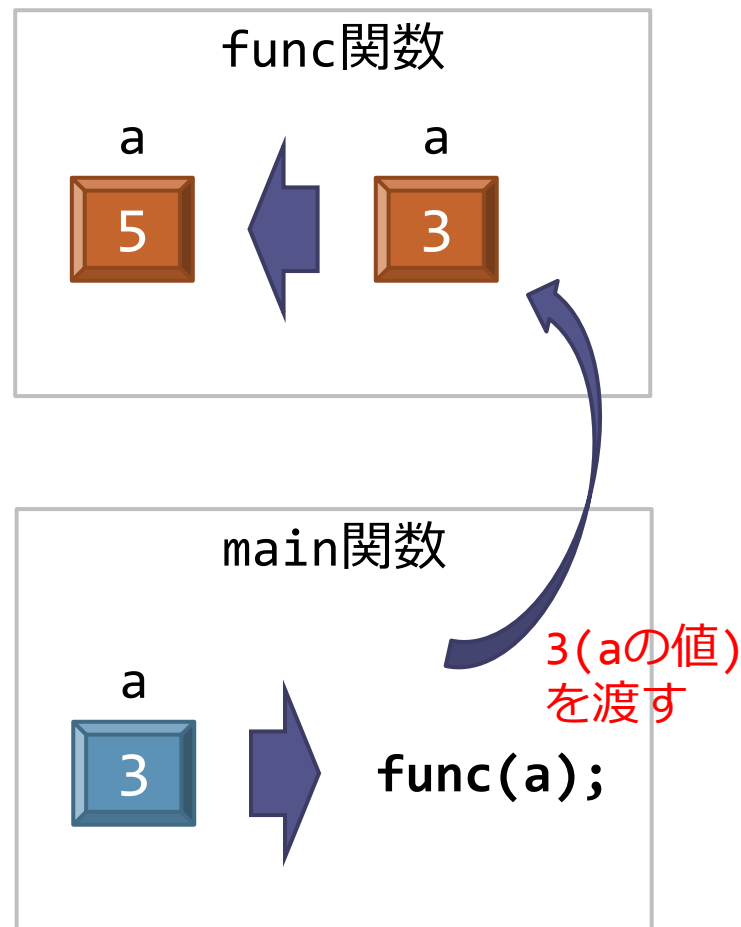
```
#include <stdio.h>

void func(int a){
    a = 5;
    return;
}

int main(void){
    int a = 3;

    func(a);

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



# 関数とポインタ（復習）

## ▶ 参照渡し

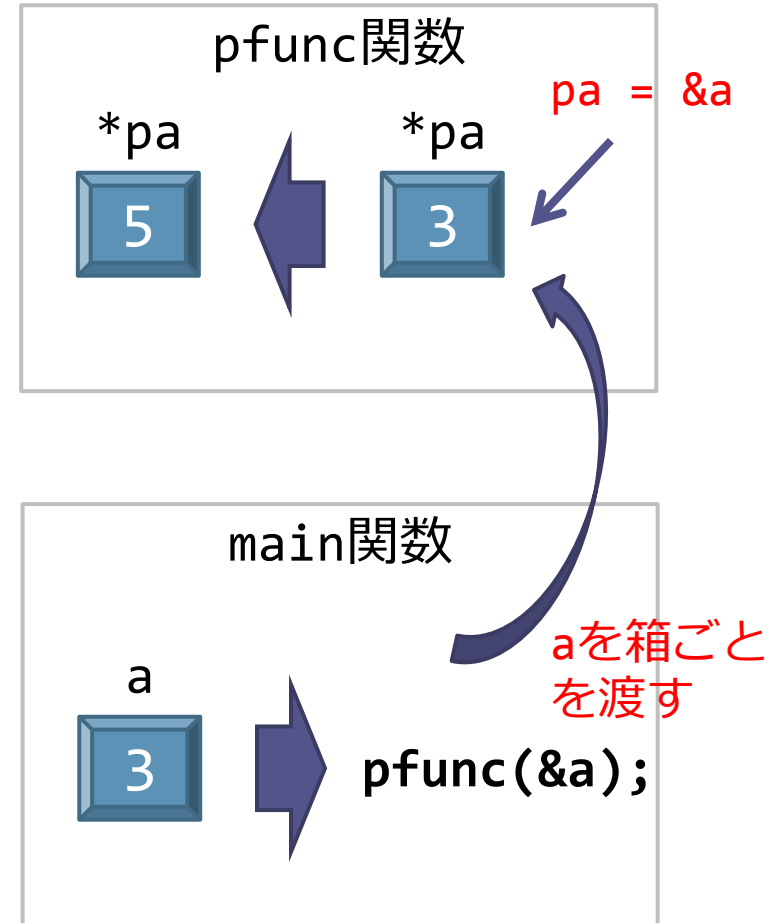
```
#include <stdio.h>

void pfunc(int *pa){
    *pa = 5;
    return;
}

int main(void){
    int a = 3;

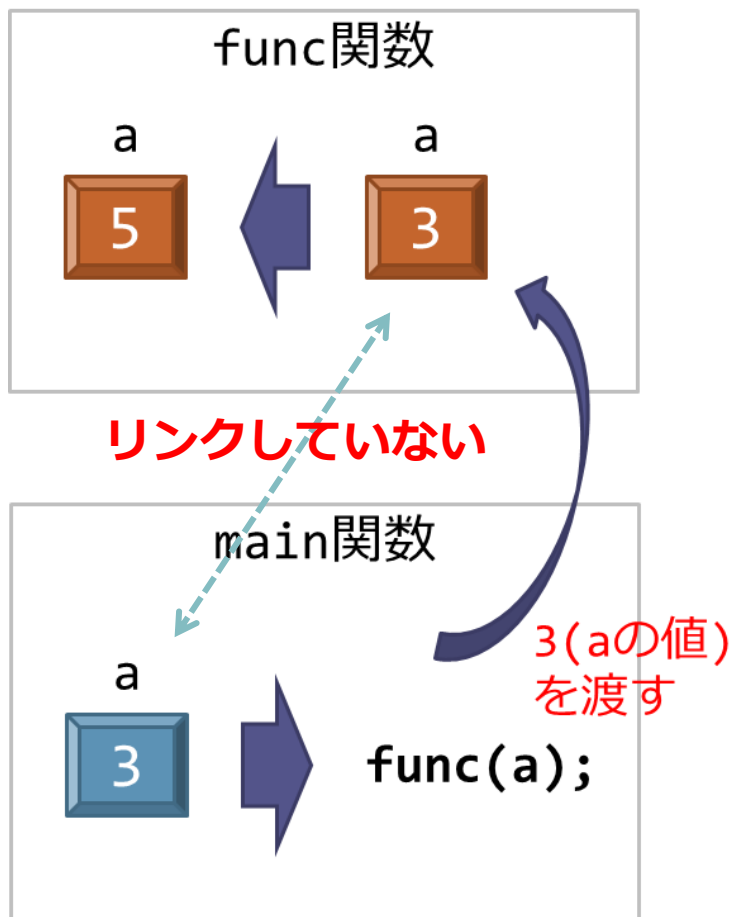
    pfunc(&a);

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

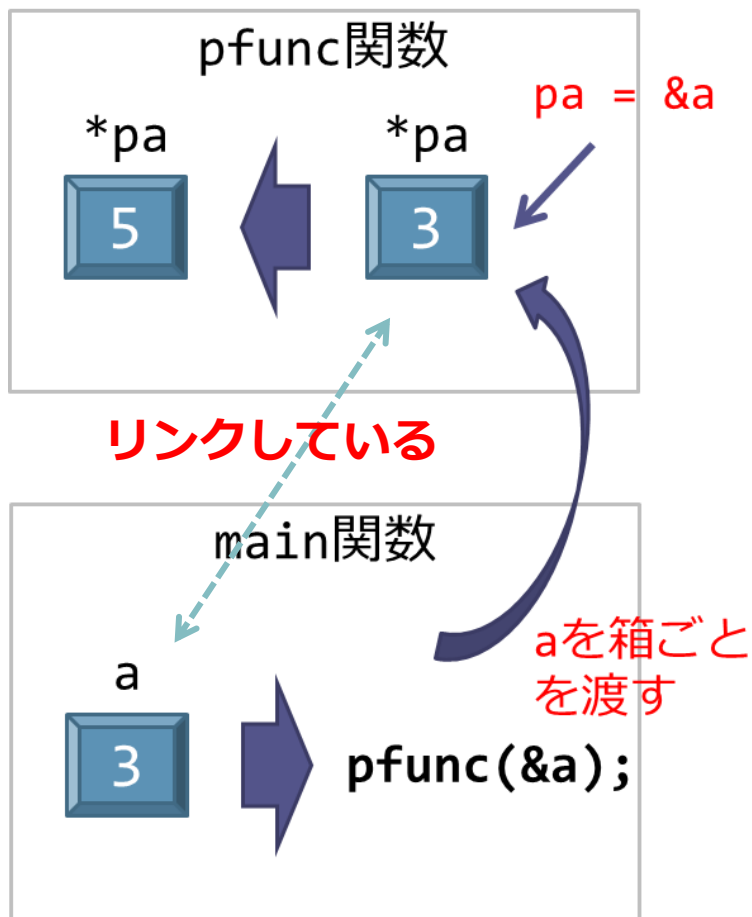


# 関数とポインタ（復習）

## 値渡し



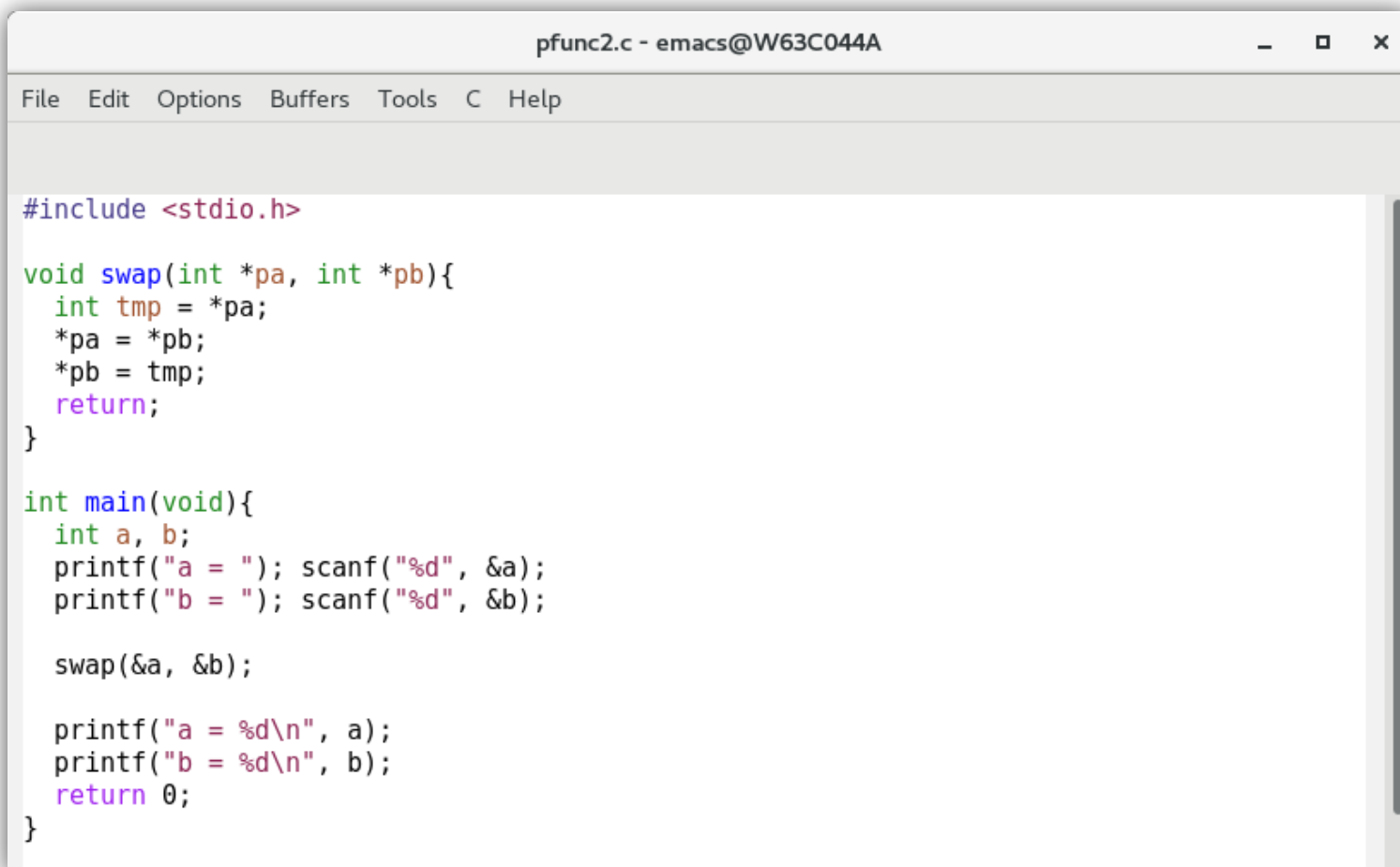
## 参照渡し



# 関数とポインタ（復習）

---

## ▶ プログラム例：swap関数



The screenshot shows an Emacs editor window titled "pfunc2.c - emacs@W63C044A". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "C", and "Help". The code is as follows:

```
#include <stdio.h>

void swap(int *pa, int *pb){
    int tmp = *pa;
    *pa = *pb;
    *pb = tmp;
    return;
}

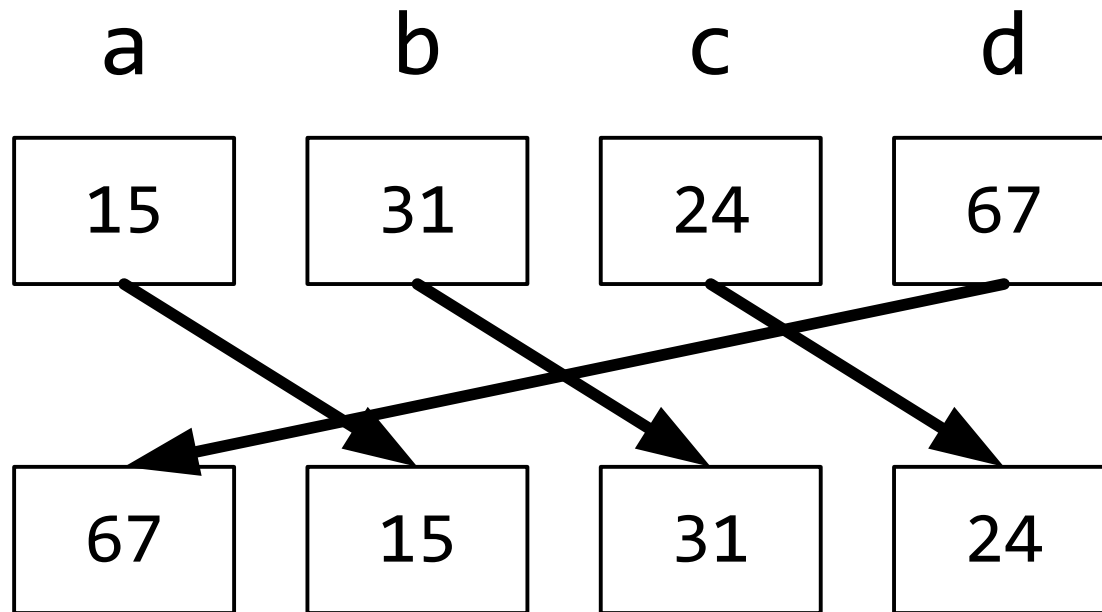
int main(void){
    int a, b;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);

    swap(&a, &b);

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

## 演習①

- ▶ 端末から入力した4つの整数を**シフト**して標準出力するプログラム pfunc4.c を作成せよ。



(シフトの様子)



# 演習①

---

```
#include <stdio.h>

void rot1(int *pa, int *pb, int *pc, int *pd){
    

?


    return;
}

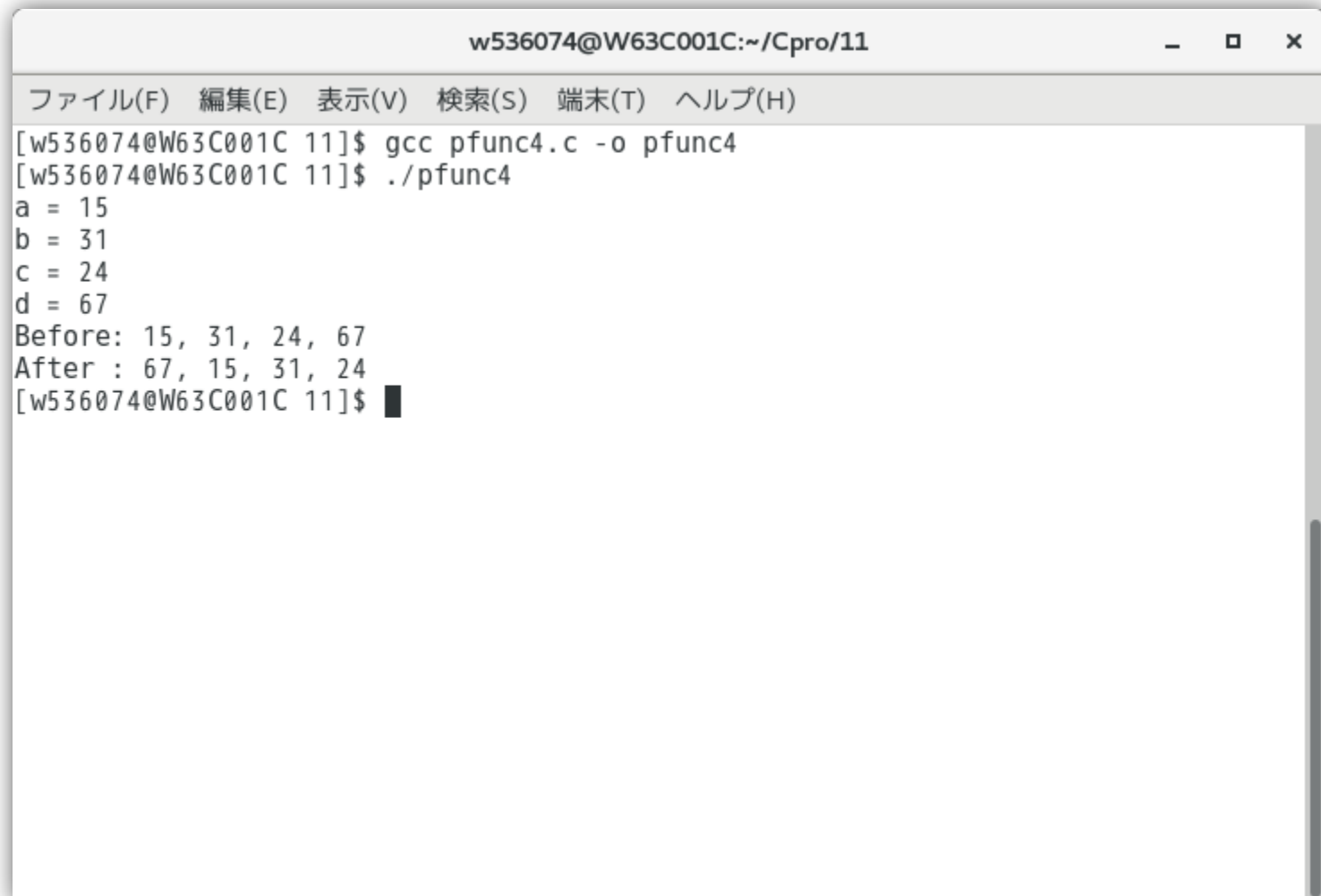
int main(void){
    int a, b, c, d;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);
    printf("c = "); scanf("%d", &c);
    printf("d = "); scanf("%d", &d);
    printf("Before: %d, %d, %d, %d\n", a, b, c, d);

    rot1(&a, &b, &c, &d);

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

# 演習① ~実行例~

---

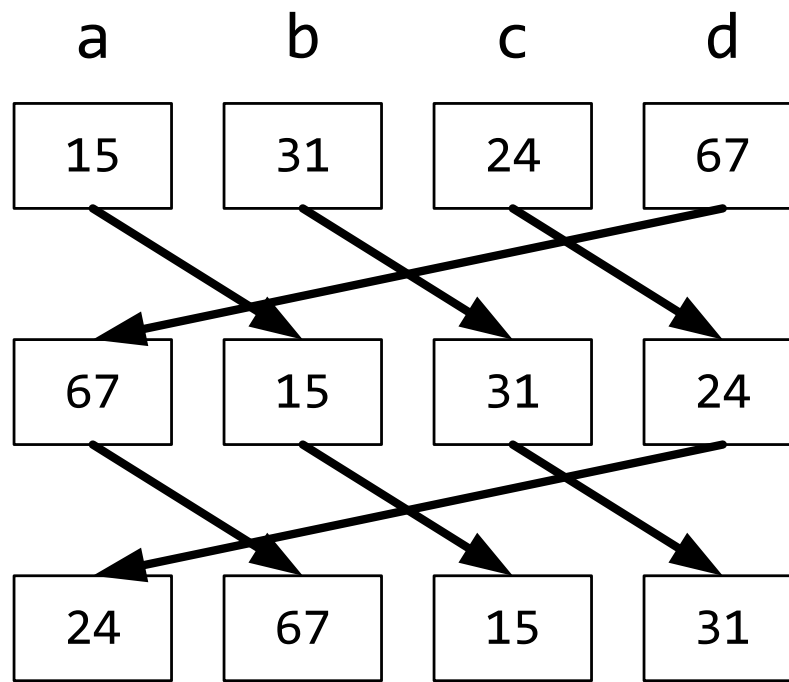


A terminal window titled "w536074@W63C001C:~/Cpro/11" with standard window controls. The menu bar includes "ファイル(F)", "編集(E)", "表示(V)", "検索(S)", "端末(T)", and "ヘルプ(H)". The terminal output shows the compilation and execution of a program named "pfunc4".

```
w536074@W63C001C:~/Cpro/11
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 11]$ gcc pfunc4.c -o pfunc4
[w536074@W63C001C 11]$ ./pfunc4
a = 15
b = 31
c = 24
d = 67
Before: 15, 31, 24, 67
After : 67, 15, 31, 24
[w536074@W63C001C 11]$
```

## 演習②

- ▶ 端末から入力した4つの整数を**N回シフトして**標準出力するプログラム pfunc5.c を作成せよ。
- ▶ ただし, 回数Nも端末から入力するものとする。



(例) **N=2**の場合.

## 演習②

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

```
void rot1(int *pa, int *pb, int *pc, int *pd){  
    ...  
    ...  
}
```

} 演習①を流用

```
void rotN(int *pa, int *pb, int *pc, int *pd, int N){
```

?

} 複数回rot1を呼び出すように記述

```
    return;  
}
```

```
int main(void){  
    int a, b, c, d, N;  
    printf("a = "); scanf("%d", &a);  
    printf("b = "); scanf("%d", &b);  
    printf("c = "); scanf("%d", &c);  
    printf("d = "); scanf("%d", &d);  
    printf("Before: %d, %d, %d, %d\n", a, b, c, d);  
    printf("N = "); scanf("%d", &N);  
  
    rotN(&a, &b, &c, &d, N);  
  
    printf("After : %d, %d, %d, %d\n", a, b, c, d);  
    return 0;  
}
```

## 演習② ~実行例~

---

```
w536074@W63C001C:~/Cpro/11
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 11]$ gcc pfunc5.c -o pfunc5
[w536074@W63C001C 11]$ ./pfunc5
a = 15
b = 31
c = 24
d = 67
Before: 15, 31, 24, 67
N = 2
After : 24, 67, 15, 31
[w536074@W63C001C 11]$ ./pfunc5
a = 15
b = 31
c = 24
d = 67
Before: 15, 31, 24, 67
N = 3
After : 31, 24, 67, 15
[w536074@W63C001C 11]$
```

## 演習③

- ▶ 端末から入力した整数a, bをもとに**a÷bの商と余り**を標準出力するプログラム pfunc6.c を作成せよ。

```
#include <stdio.h>

void div(int a, int b, int *pq, int *pr){
    

?


    return;
}

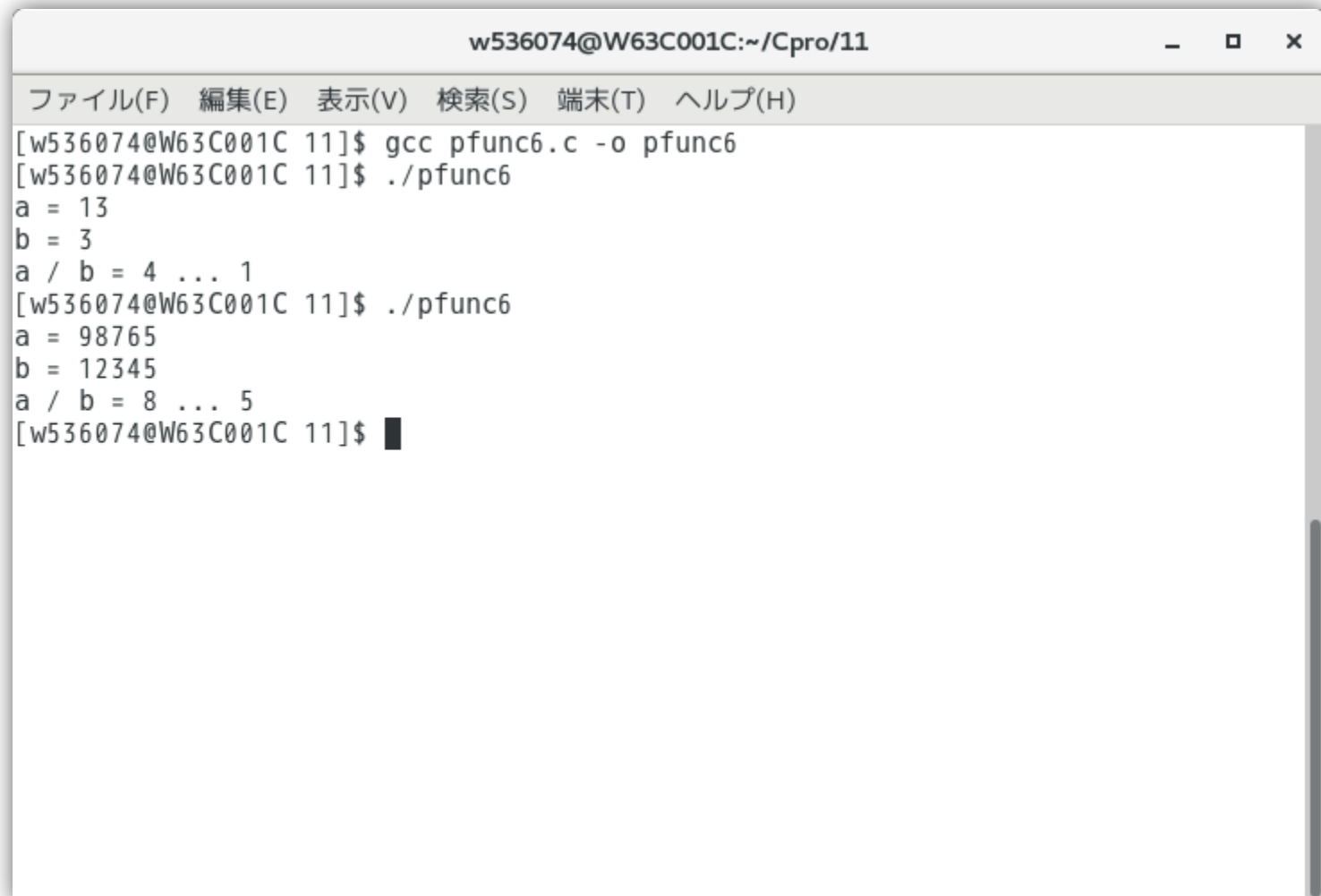
int main(void){
    int a, b, q, r;
    printf("a = "); scanf("%d", &a);
    printf("b = "); scanf("%d", &b);

    div(a, b, &q, &r);

    printf("a / b = %d ... %d¥n", q, r);
    return 0;
}
```

## 演習③ ~実行例~

---



The screenshot shows a terminal window with the title bar "w536074@W63C001C:~/Cpro/11". 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 content is as follows:

```
[w536074@W63C001C 11]$ gcc pfunc6.c -o pfunc6
[w536074@W63C001C 11]$ ./pfunc6
a = 13
b = 3
a / b = 4 ... 1
[w536074@W63C001C 11]$ ./pfunc6
a = 98765
b = 12345
a / b = 8 ... 5
[w536074@W63C001C 11]$
```

# 本日の講義・演習項目

---

## ▶ ポインタ（復習）

- ▶ アドレス演算子「&」 ... 変数のアドレスを知る
- ▶ ポインタ型変数 ... 変数のアドレスを記憶する
- ▶ 間接参照演算子「\*」 ... アドレスから変数を知る
- ▶ 関数とポインタ

## ▶ 配列とポインタ



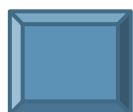
# 配列（復習） ... 第8回講義

## ▶ 配列の宣言

```
int i;           ... 変数の宣言
int a[5];        ... 配列の宣言
```

変数

i



配列

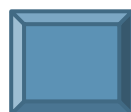
a[0]

a[1]

a[2]

a[3]

a[4]



添え字は0~4  
(1~5ではない)

## ▶ 配列の要素への代入

```
a[0] = 23;
a[1] = 34;
...
```

配列の何番目に代入するのかを  
記述する必要がある

# 配列とポインタ

---

- ▶ 配列を使う場合には必ず添え字が必要
  - ▶ 配列の*i*番目の要素は`a[i]`と表される
- ▶ `&a[0]` = 配列の先頭要素`a[0]`のアドレス
- ▶ プログラム中で配列名`a`は`&a[0]`を表す

# 配列とポインタ

## ▶ アドレスを表示するプログラム

```
#include <stdio.h>
int main(void){
    int a[5] = {10, 11, 12, 13, 14};
    printf("Address of a[0] = %p\n", &a[0]);
    printf("Address of a[0] = %p\n", a);
    return 0;
}
```

## ▶ 実行結果



```
w536074@W63C001C:~/Cpro/11
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 11]$ gcc print_addr.c -o print_addr
[w536074@W63C001C 11]$ ./print_addr
Address of a[0]: 0x7fff59bad2b0
Address of a[0]: 0x7fff59bad2b0
[w536074@W63C001C 11]$
```

← 同じアドレスを示す

# 配列とポインタ

## ▶ プログラム例：zero関数


```
#include <stdio.h>
#define SIZE 5

void zero(int *a){
    int i;
    for(i = 0; i < SIZE ; i++){
        a[i] = 0;
    }
    return;
}

int main(void){
    int i, a[SIZE];

    zero(a);

    for(i = 0 ; i < SIZE ; i++){
        printf("a[%d] = %d\n", i, a[i]);
    }
    return 0;
}
```



配列aの先頭要素の  
アドレスを渡す

# 配列とポインタ

## ▶ プログラム例：zero関数

```
#include <stdio.h>
#define SIZE 5

void zero(int *a){
    int i;
    for(i = 0; i < SIZE ; i++){
        a[i] = 0;
    }
    return;
}

int main(void){
    int i, a[SIZE];

    zero(a);

    for(i = 0 ; i < SIZE ; i++){
        printf("a[%d] = %d\n", i, a[i]);
    }
    return 0;
}
```

配列aの先頭要素の  
アドレスを受け取る  
→ ポインタ型変数

添え字を付ければ  
main関数内と同様に  
配列を扱える

## 演習④

- ▶ 配列の各要素を2倍に変更する関数 twice を作成せよ。
- ▶ プログラム名：parray1.c

```
#include <stdio.h>

void twice(int *a){
    

?



    return;
}

int main(void){
    int i, a[5] = {11, 22, 33, 44, 55};
    for(i = 0 ; i < 5 ; i++){ printf("%d ", a[i]); }
    printf("\n");

    twice(a);

    for(i = 0 ; i < 5 ; i++){ printf("%d ", a[i]); }
    printf("\n");
    return 0;
}
```

要素数は5で固定



## 演習④ ~実行例~

---



```
w536074@W63C001C:~/Cpro/11
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 11]$ gcc parray1.c -o parray1
[w536074@W63C001C 11]$ ./parray1
11 22 33 44 55
22 44 66 88 110
[w536074@W63C001C 11]$
```

## 演習⑤

- ▶ 配列の要素を逆順に変更する関数 reverse を作成せよ。
- ▶ プログラム名：parray2.c

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

```
void reverse(int *a){
```

?

```
    return;
```

```
}
```

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

```
    int i, a[5] = {11, 22, 33, 44, 55};
```

```
    for(i = 0 ; i < 5 ; i++){ printf("%d ", a[i]); }
```

```
    printf("\n");
```

```
    reverse(a);
```

```
    for(i = 0 ; i < 5 ; i++){ printf("%d ", a[i]); }
```

```
    printf("\n");
```

```
    return 0;
```

```
}
```

要素数は5で固定





## 演習⑤ ~実行例~

---

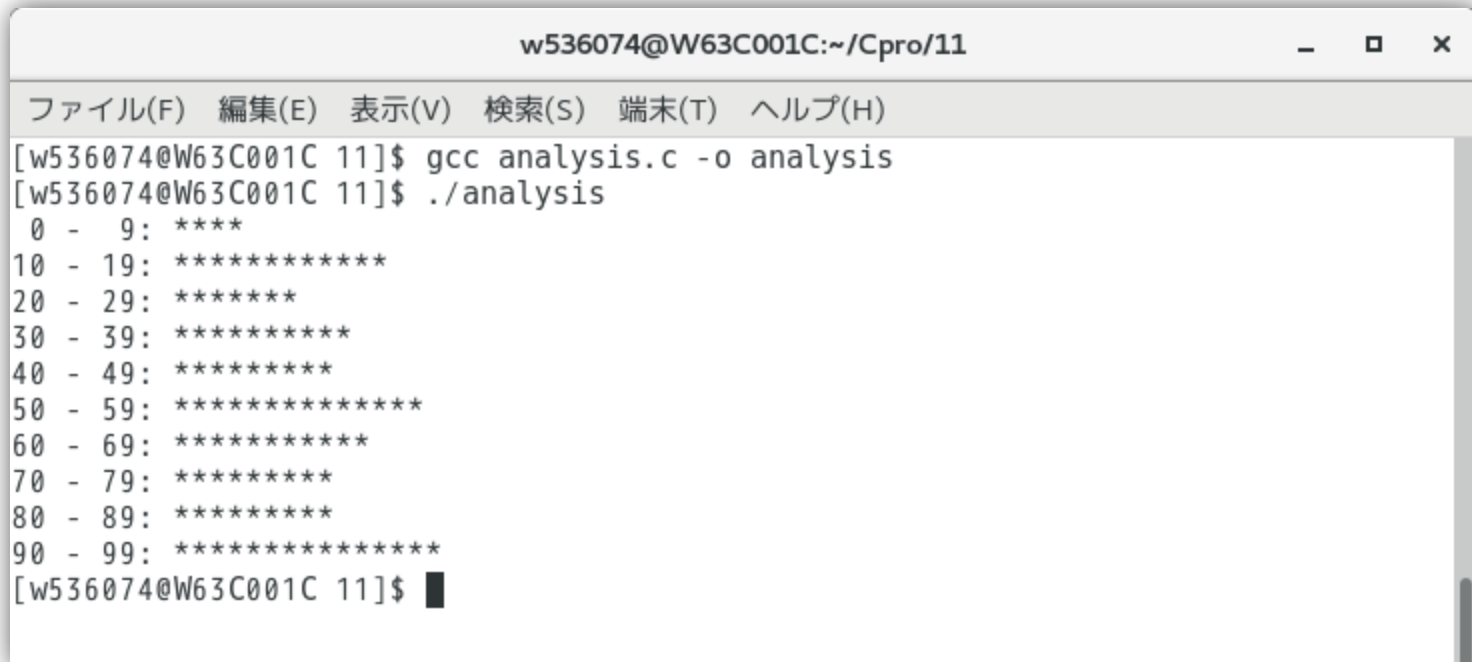


A terminal window titled "w536074@W63C001C:~/Cpro/11" with standard window controls. The menu bar includes "ファイル(F)", "編集(E)", "表示(V)", "検索(S)", "端末(T)", and "ヘルプ(H)". The terminal content shows the compilation and execution of a program named "parray2.c". The output of the program is displayed on two lines.

```
w536074@W63C001C:~/Cpro/11
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 11]$ gcc parray2.c -o parray2
[w536074@W63C001C 11]$ ./parray2
11 22 33 44 55
55 44 33 22 11
[w536074@W63C001C 11]$
```

## 演習⑥（余力がある人向け）

- ▶ 0~99の乱数を100個生成し，10段階のヒストグラムを表示するプログラム analysis.c を作成せよ。
- ▶ 実行例




```
w536074@W63C001C:~/Cpro/11
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[w536074@W63C001C 11]$ gcc analysis.c -o analysis
[w536074@W63C001C 11]$ ./analysis
 0 -  9: ****
10 - 19: *****
20 - 29: *****
30 - 39: *****
40 - 49: *****
50 - 59: *****
60 - 69: *****
70 - 79: *****
80 - 89: *****
90 - 99: *****
[w536074@W63C001C 11]$
```

## 演習⑥（余力がある人向け）

### ▶ プログラムの一部

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 100
```

二つの関数を  
自作する



```
void analyze_data(int *data, int *score, int width){ ... }
void print_score(int *score, int num){ ... }
```

```
int main(void){
    srand((unsigned)time(NULL));
    int i, data[SIZE];
    for(i = 0 ; i < SIZE ; i++){
        data[i] = rand() % 100;
    }
    int score[10];
    for(i = 0 ; i < 10 ; i++){ score[i] = 0; }

    analyze_data(data, score, 10);
    print_score(score, 10);
    return 0;
}
```

## 演習⑥（余力がある人向け）

---

- ▶ `analyze_data(int *data, int *score, int width){ ... }`
  - ▶ データを解析する関数
  - ▶ `score[0]`, `score[1]`, ..., `score[9]`を計算する
  - ▶ `score[0]` → 0~9の範囲のデータ個数（階級0）
  - ▶ `score[1]` → 10~19の範囲のデータ個数（階級1）
  - ▶ `width` → 階級の幅（10）
- ▶ `print_score(int *score, int num){ ... }`
  - ▶ 解析結果を表示する関数
  - ▶ `num` → 階級数（階級0～階級9）

# キーワード，次回の講義

---

- ▶ 本日のキーワード：
- ▶ 次回は7/5
- ▶ 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