

# プログラミング基礎

<http://bit.ly/prog2d>

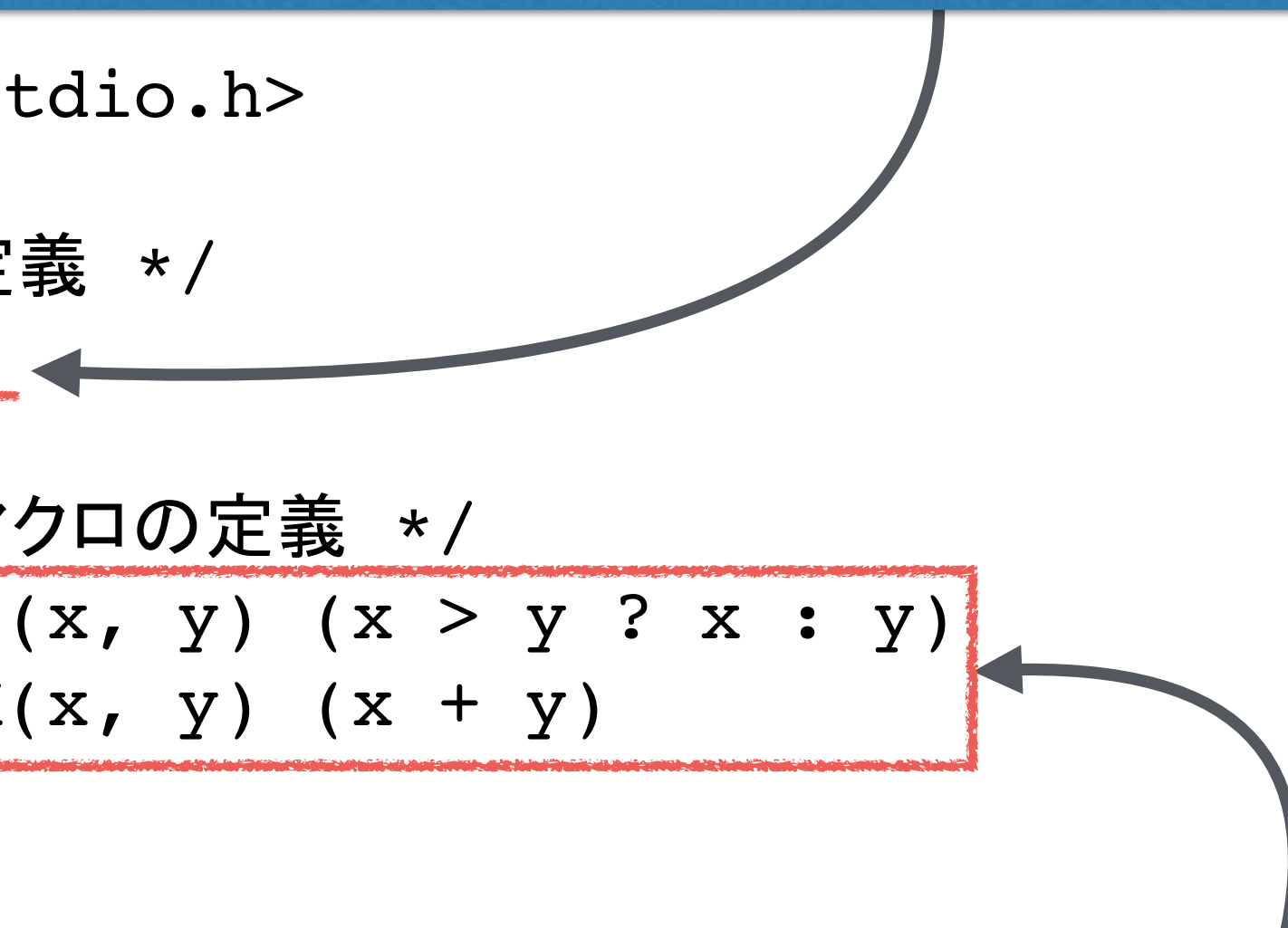
## 配列

前期 第4週

2017/5/8

【Point 1】 Nを5に置き換えるように定義している  
(p.194~195)

```
1: #include <stdio.h>
2:
3: /* マクロの定義 */
4: #define N 5
5:
6: /* 関数形式マクロの定義 */
7: #define MAX(x, y) (x > y ? x : y)
8: #define SUM(x, y) (x + y)
9:
```



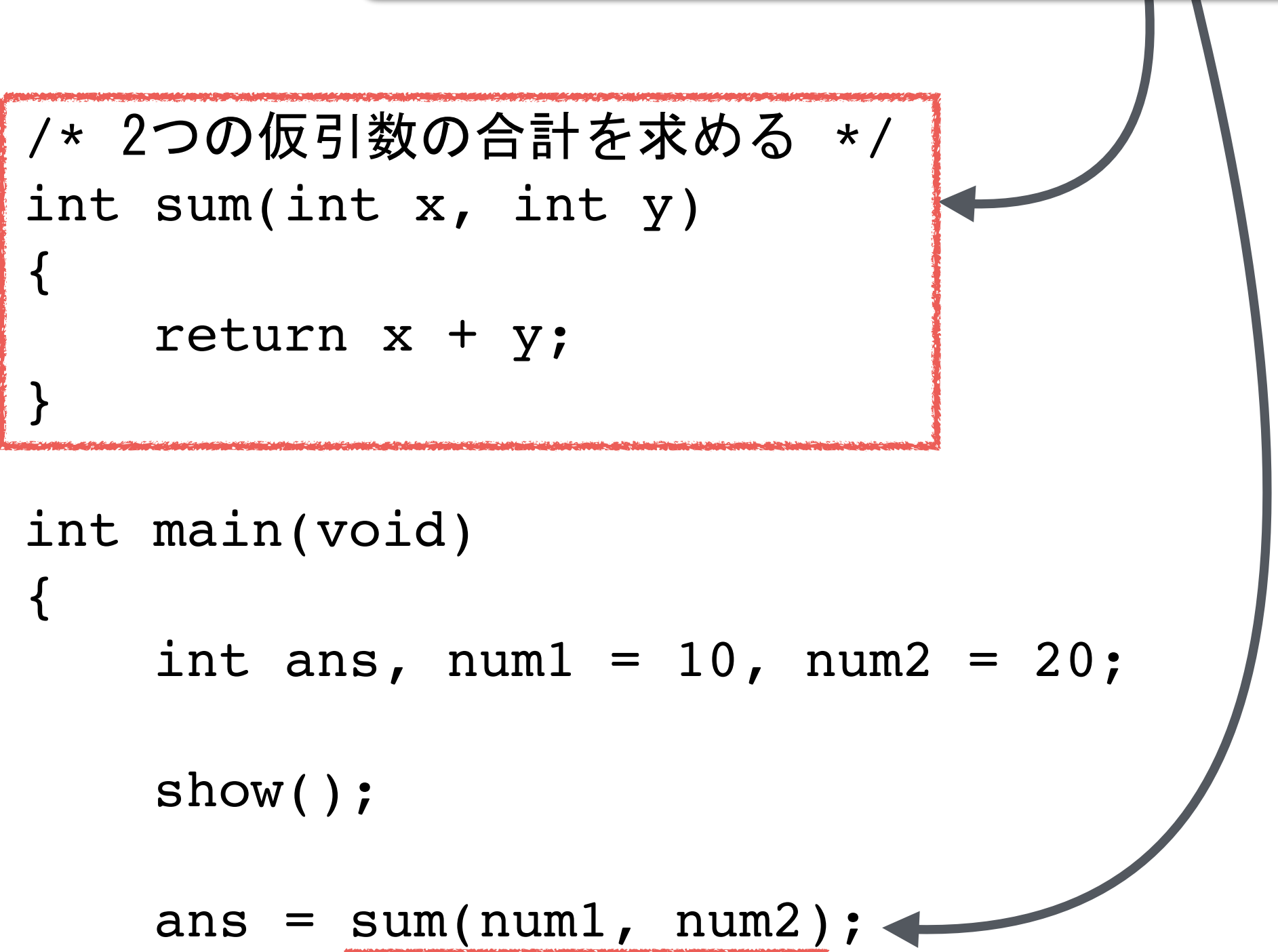
【Point 3】 マクロに引数を付けると、関数のような書き方ができる  
(p. 245~246)

```
10: int test[N] = {80, 60, 22, 50, 75};
11:
12: /* 関数のプロトタイプ宣言 */
13: void show();
14: int sum(int x, int y);
15:
16: /* 配列testの要素を出力する */
17: void show()
18: {
19:     int i;
20:     for(i=0; i<N; i++) {
21:         printf("test[%d]: %d\n", i, test[i]);
22:     }
23: }
```

【Point 2】コンパイラ時にNが5に置き換わる  
(p. 194~195)

## 引数付きマクロと同じ処理を関数でも作ってみる

```
24:  /* 2つの仮引数の合計を求める */
25:  int sum(int x, int y)
26:  {
27:      return x + y;
28:  }
29:
30:  int main(void)
31:  {
32:      int ans, num1 = 10, num2 = 20;
33:
34:      show( );
35:
36:      ans = sum(num1, num2);
37:      printf("sum: %d\n", ans);
```



【Point 4】コンパイル時に、この部分が「num1 + num2」に置き換わる (p. 246)

```
38:
39:     ans = SUM(num1, num2);
40:     printf("SUM: %d\n", ans);
41:
42:     ans = MAX(num1, num2);
43:     printf("MAX: %d\n", ans);
44:
45:     return 0;
46: }
```

【Point 5】コンパイル時に、この部分が「num1 > num2 ? num1 : num2」に置き換わる

「A ? B : C」という書き方は条件演算子と呼び、  
「Aの条件を満たせばB、そうでなければC」という  
演算をする (p. 142)

# 【練習4-1】

サンプルプログラムをコンパイルして、  
実行結果を確認しましょう。

# 【練習4-2】

サンプルプログラムにおいて、Nを10の定義に変更すると、34行目のshow()の処理がどのような結果になるのか確認してみましょう。（テキストp.192の説明にあるような結果になります。）

# 【課題4-1】

マクロSUMを参考に、2つの変数x, yの積を求めるマクロPROを定義し、main()で動作を確認するプログラムを作成してください。

[main()での処理]

```
int ans, num1 = 10, num2 = 20;  
ans = PRO(num1, num2);  
printf("PRO: %d\n", ans);
```

[実行結果]

PRO: 200



# 【課題4-2】

マクロMAXを参考に、2つの変数x, yの小さい方の値を決めるマクロMINを定義し、main()で動作を確認するプログラムを作成してください。

[main()での処理]

```
int ans, num1 = 10, num2 = 20;  
ans = MIN(num1, num2);  
printf("MIN: %d\n", ans);
```

[実行結果]

```
MIN: 10
```

# 【課題4-3】

整数値を格納する配列testの中から最大値を見つけ、その値を戻り値とする関数max\_array()を作成してください。ただし、マクロMAXを繰り返し使って最大値を見つけてください。

[この関数のプロトタイプ宣言]

```
int max_array();
```

```
/* 配列testに対する繰り返し処理の中でMAXを使うように作ってみる */
```

```
/* 見つけた最大値を戻り値として返すようにする */
```

# 【課題4-3】

[ 配列をグローバル変数として宣言する ]

```
int test[N] = {80, 60, 22, 50, 75};
```

[ main() での処理 ]

```
int ans;  
ans = max_array();  
printf("最大値: %d\n", ans);
```

[ 実行結果 ]

最大値: 80

# 【課題4-4】

関数形式マクロの引数は、型が指定されていないため、  
どのような型の変数にも使うことができます。  
(p.248参照)

課題4-3と同じように最大値を見つける処理を、  
float型の配列test2に対して処理する関数  
max\_array2()を作成してください。  
ただし、マクロMAXを繰り返し使って最大値を  
見つけてください。

# 【課題4-4】

[この関数のプロトタイプ宣言]

```
float max_array2();
```

```
/* 課題4-3の処理をfloat型の配列に対して行うように変更する */
```

```
/* マクロMAXは変更することなくfloat型にそのまま使うことができる */
```

```
/* 見つけた最大値を戻り値として返すようにする */
```

# 【課題4-4】

[ 配列をグローバル変数として宣言する ]

```
float test2[N] = {2.3, 5.9, 9.1, 1.8, 7.6};
```

[ main() での処理 ]

```
float ans2;  
ans2 = max_array2();  
printf("最大値: %f\n", ans2);
```

[ 実行結果 ]

```
最大値: 9.100000
```

# まだ余裕のある人は…【課題4-5】

(1) 「引数xに対して偶数かどうかを調べて、偶数でなかったら0とする（偶数だったらxのまま）」

マクロEVENを定義してください。「引数xが偶数かどうか」の条件は「**xを2で割った余りが0かどうか**」で表すことができます。

(2) **マクロEVENを繰り返し使って**、配列testの偶数の要素のみを残す関数even\_array()を作成してください。

# 【課題4-5】

[この関数のプロトタイプ宣言]

```
void even_array();
```

```
/* testの全ての要素に対してマクロEVENを処理するように作る */
```

```
/* 奇数の要素は0のまま残してよい */
```



# 【課題4-5】

[ 配列をグローバル変数として宣言する ]

```
int test[N] = {80, 60, 22, 50, 75};
```

[ main( ) での処理 ]

```
even_array();
```

```
show();
```

```
/* 結果が確認できたら、testの要素の値を変えて試してみましょう */
```

[ 実行結果 ]

```
test[0]: 80
```

```
test[1]: 60
```

```
test[2]: 22
```

```
test[3]: 50
```

```
test[4]: 0
```

# 小テストについて

## 小テストの注意点

- 他人の力は借りずに、自分だけでプログラムを作成する。（つまり定期試験と同様）
- 小テスト中は、演習室外へのネットワークアクセスは遮断される。

# 小テストについて

## 小テスト中に参照できるもの

- 教科書, 配付資料
- 自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル
- 小テストでは紙媒体のものは参照可能
- 上記以外の情報を参照することはカンニング行為とする  
例：USBで接続された機器に保存されているファイルの参照  
ネットワークを介した情報の参照、など