Ark Oak企画

# C言語入門

## 第3回

復習

# 構文の復習

## switch文

```
switch(式){
    case 定数式1:
        処理1;
        break;
    ・・・
    default:
        処理3;
        break;
}
```

## if文

```
if (条件①) {
        条件が成り立つときの処理
}
else if (条件②) {  //条件分岐が2つ以上
        条件が成り立つときの処理
}
else{
        条件が成り立たないときの処理
}
```

## for文

```
for (初期化; 条件; 次の一歩){
        繰り返す処理;
}
```

# 宿題

- 変数n を用意
  - n = 1　ならば月
  - n = 2　ならば火
  - ・・・
  - n = 7　ならば日

（例）n = 3;　とすると

水

と表示するようなプログラムを組んでくる

```c
#include <stdio.h>
int main(void){
    int n;
    scanf("%d",&n);
    switch(n){
        case 1:
            printf("月曜日です。");
            break;
        case 2:
            printf("火曜日です。");
            break;
        case 3:
            printf("水曜日です。");
            break;
        case 4:
            printf("木曜日です。");
            break;
        case 5:
            printf("金曜日です。");
            break;
        case 6:
            printf("土曜日です。");
            break;
        case 7:
            printf("日曜日です。");
            break;
        default:
            printf("エラー");
            break;
    }
}[EOF]
```

入力を変数へ
読み込む関数

```
yamamoto@yamamoto-PC ~/work
$ ./a
2
火曜日です。
yamamoto@yamamoto-PC ~/work
$ ./a
3
水曜日です。
yamamoto@yamamoto-PC ~/work
$ ./a
4
木曜日です。
yamamoto@yamamoto-PC ~/work
$ ./a
金曜日です。
yamamoto@yamamoto-PC ~/work
$ ./a
6
土曜日です。
yamamoto@yamamoto-PC ~/work
$ ./a
7
日曜日です。
yamamoto@yamamoto-PC ~/work
$ ./a
エラー
yamamoto@yamamoto-PC ~/work
$ ./a
h
エラー
```

# ～while文～

## 繰り返し処理(～まで)

# while文の構造

- 繰り返し処理

繰り返す回数が不明
(条件を満たすまで)

```
while (条件) {
        繰り返す処理;
}
```

繰り返す回数が既知

```
for (初期値; 条件; 次の一歩){
        繰り返す処理;
}
```

# while文とは

【例題】ある小学生が、親に頼みました。「今月は1円、来月は2円、再来月は4円と、前の月の倍額のおこづかいをちょうだい」さて、おこづかいが10000円を超えるのは何ヶ月目でしょうか？

```
int money = 1;  //現在の金額
int month = 1;  //現在の月

while(money < 10000){
        money = money * 2
        month = month + 1;
}
printf("%dヶ月¥n", month);
```

$a = a + 1 \cdots \ a++$ インクリメント
$a = a - 1 \cdots \ a--$ デクリメント

```
for(money; money<10000; money*=2){
    month++;
}
```

※for文を使うとわかりにくい

15か月

# その他の制御文

**break文**

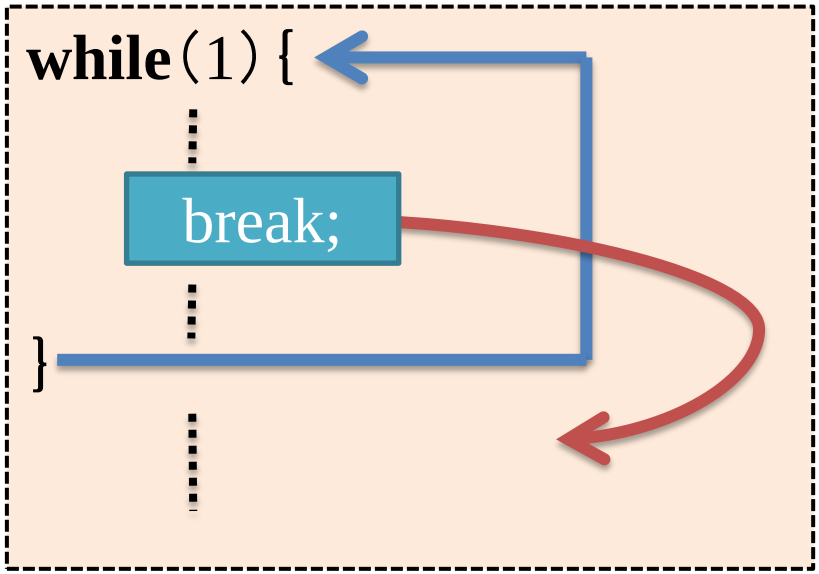
```
break;
```

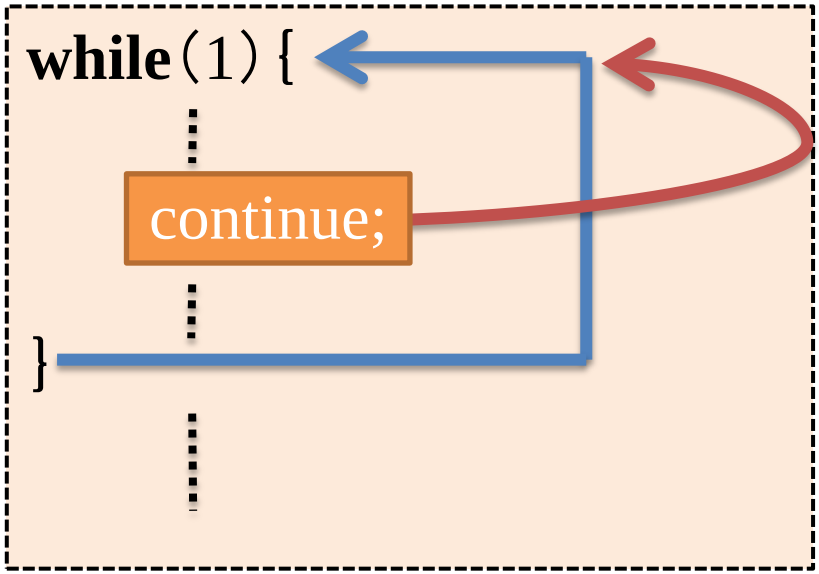
繰り返しの処理を途中で中断。→ループの外へジャンプ

**continue文**

```
continue;
```

繰り返しの処理のその回を中断。→次の回からスタート



# 【練習】while文の練習

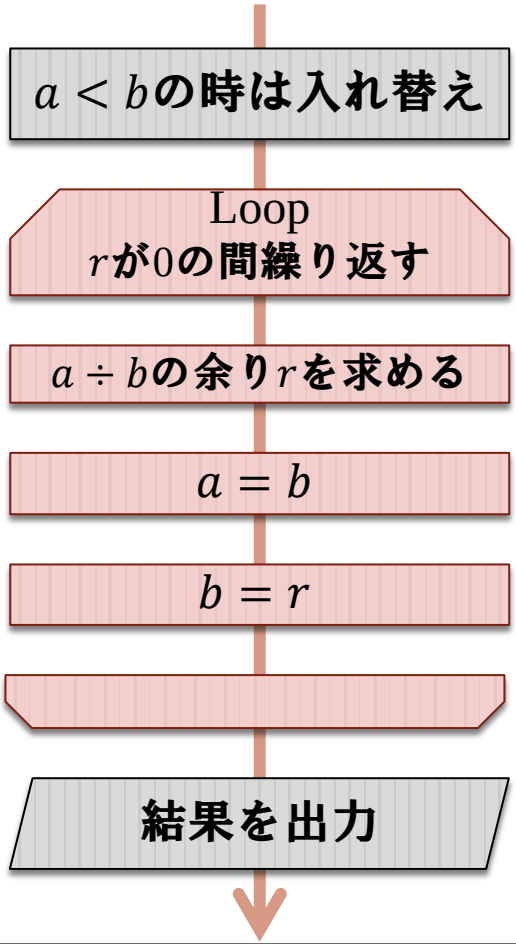
**現在大島さんの列には10000人の人が並んでいて、毎分100人ずつ増えています。一方渡辺さんの列は現在5000人の人が並んでいて毎分350人の人が並んでいます。渡辺さんの列が大島さんの列よりも長くなるのは何分後でしょう？**

```
3  int main(void){↓
1
1
1
1
1
16 }↓
```

# 【練習】ユークリッドの互除法 ～280と220の最大公約数を求める～

**フローチャート**

$a = 220$ b= 280 r…余り




例

280 ÷ 220 = 1 … 60
220 ÷ 60 = 3 … 40
60 ÷ 40 = 1 … 20
40 ÷ 20 = 2 … 0

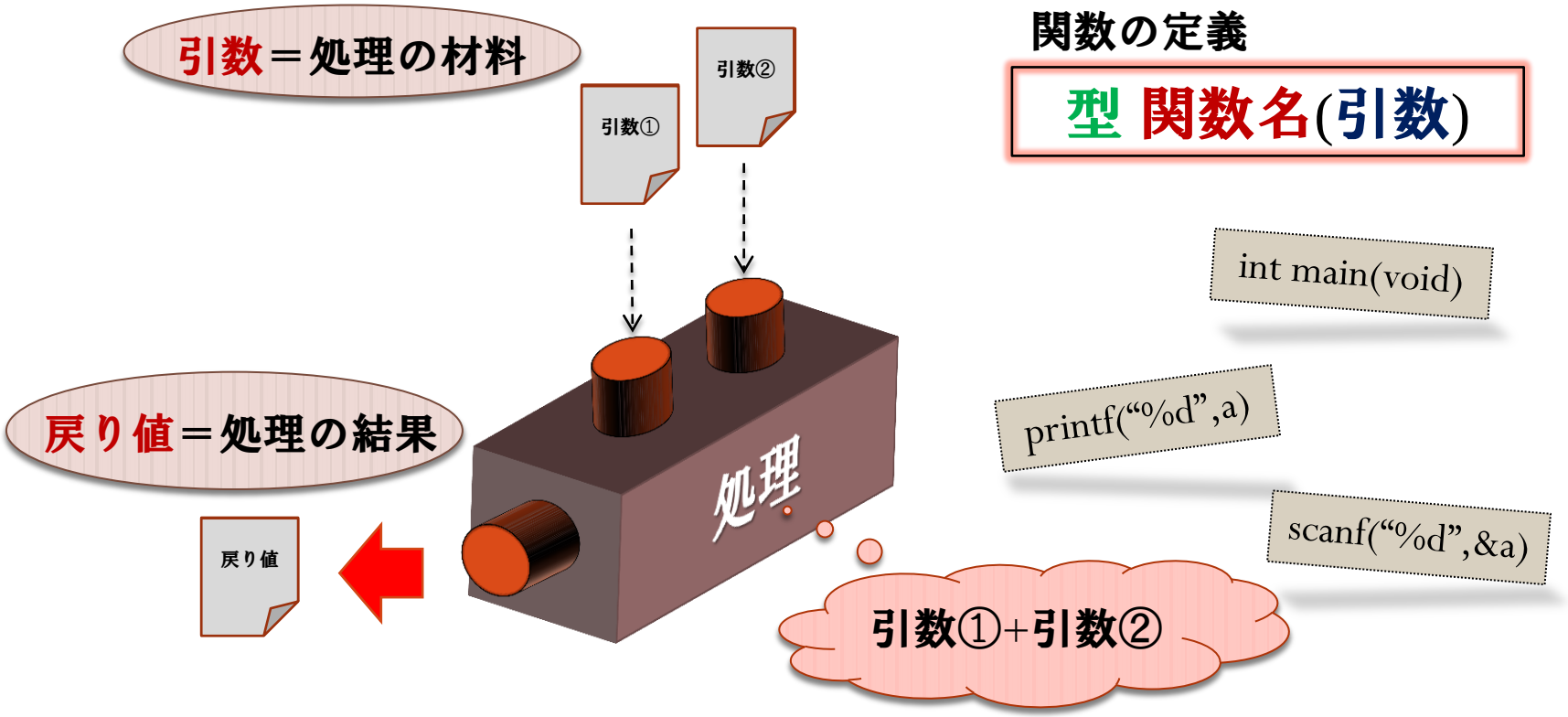
最大公約数

```
3  main(){↓
4      int a=220,b=280,c,r;↓
5      ↓
6      if(a < b){ //入れ替え↓
7          c = a;↓
8          a = b;↓
9          b = c;↓
10     }    ↓          //余りが0になるまで繰り返す
11     while(r != 0){↓
12         r = a % b;↓
13         a = b;↓
14         b = r;↓
15     }    ↓
16     printf("最大公約数は%d¥n",a);
17 }↓
```

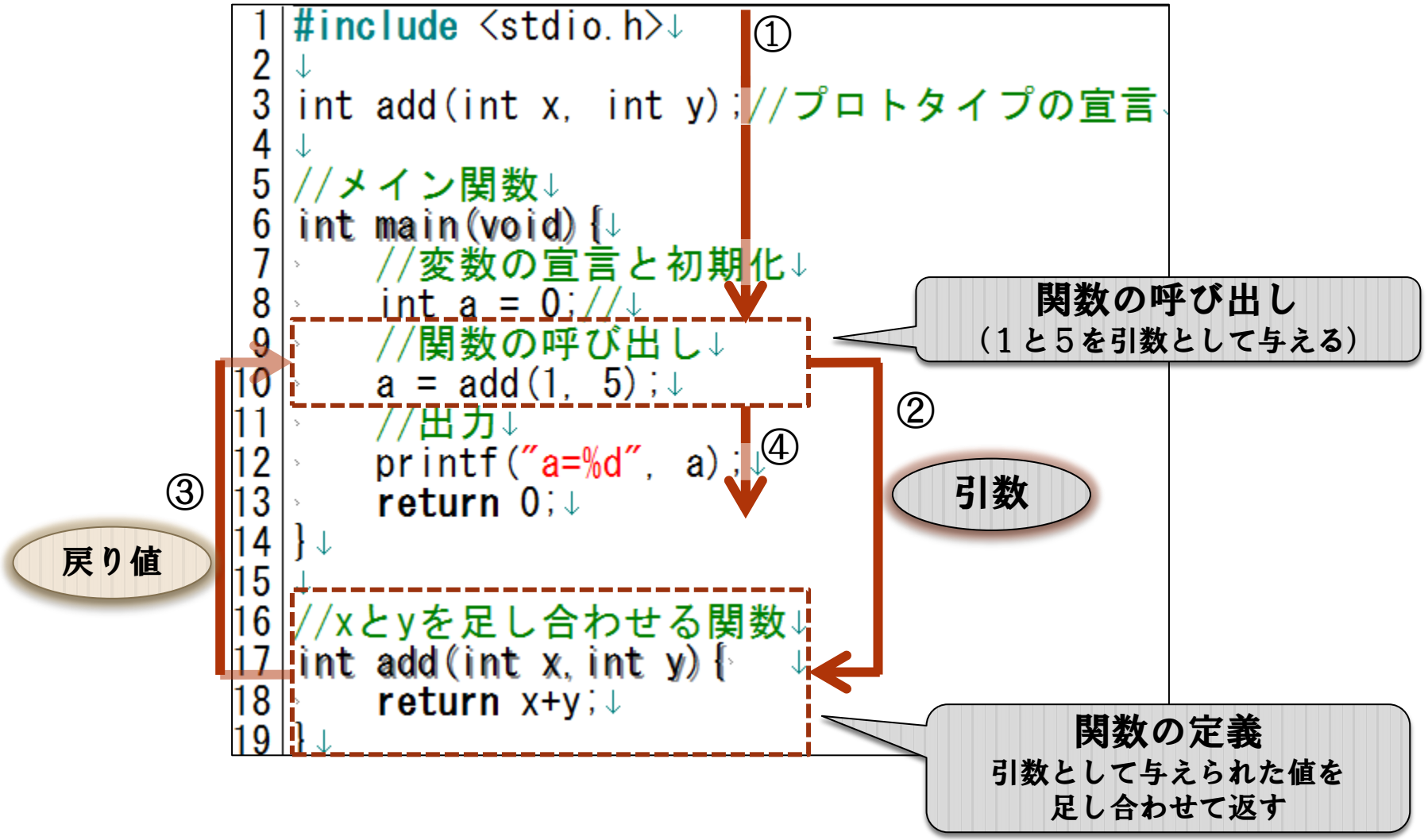
# ～関数～

# 関数とは？？

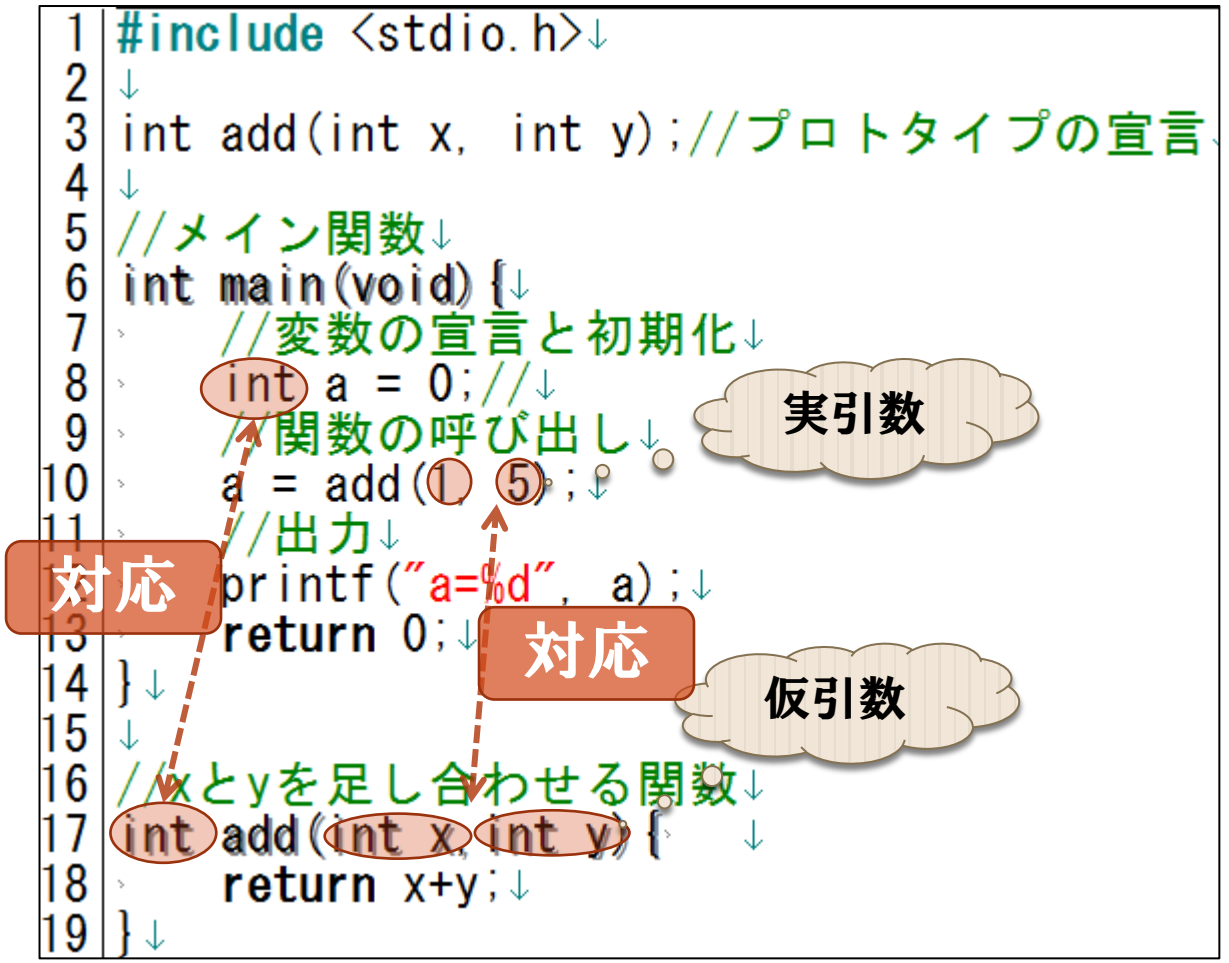
**プログラマが与えた値(引数)を**
**指示通りに処理し、結果を吐き出す箱**



# 関数の定義と利用


1 #include <stdio.h>
2
3 int add(int x, int y);//プロトタイプの宣言
4
5 //メイン関数
6 int main(void){
7 //変数の宣言と初期化
8 int a = 0;//
9 //関数の呼び出し
10 a = add(1, 5);
11 //出力
12 printf("a=%d", a);
13 return 0;
14 }
15
16 //xとyを足し合わせる関数
17 int add(int x,int y){
18 return x+y;
19 }
①
②
③
④
関数の呼び出し
（1と5を引数として与える）
引数
戻り値
関数の定義
引数として与えられた値を
足し合わせて返す

# 関数の定義と利用


#include <stdio.h>
int add(int x, int y);//プロトタイプの宣言
//メイン関数
int main(void){
//変数の宣言と初期化
int a = 0;//
//関数の呼び出し
a = add(1, 5);
//出力
printf("a=%d", a);
return 0;
}
//xとyを足し合わせる関数
int add(int x, int y){
return x+y;
}
実引数
仮引数
対応
対応

# 【練習】関数の活用
## ～関数に慣れよう～

**以下の二つの関数を作って呼び出してみよう！！**
**①hello()関数：「こんにちは」と表示する**
**②bye()関数：「さようなら」と表示する**

```
 3 void hello();//関数のプロトタイプの宣言↓
 4 void bye();↓
 5 ↓
 6 int main(void){↓
 7 ›   hello(); //hello関数を呼び出す↓
 8 ›   bye(); //bye関数を呼び出す↓
 9 ›   ↓
10 ›   return 0;↓
11 }↓
12 ↓
13 void hello (void){
14 ›   [redacted]
15 }↓
16 ↓
17 void bye (void){
18 ›   [redacted]
19 }↓
```

呼び出し先で処理

# main関数・標準ライブラリ関数

**main()関数：プログラムの開始地点となる関数**

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

**引き数を省略、戻り値int**

```
int main(int argc, char *argv[])
{
    return 0;
}
```

**引き数と戻り値を指定(基本パターン)**

**標準ライブラリ関数：コンパイラが最初から用意**

stdio.h

**標準入出力**
printf()
scanf()
fopen() **など**

math.h

**数学関係**
sin()
pow()
sqrt() **など**

# ローカル変数とグローバル変数

関数の外で宣言：
グローバル変数

関数の中で宣言：
ローカル変数

```
int z;
void func()                    変数 z のスコープ
{
    int y;                     変数 y のスコープ
    z = 2;
    …
}
int main(void)
{
    int x = 1;                 変数 x のスコープ
    z = x + 1;
    y = 3;
    return 0;
}
```

main()関数から
参照できない

※グローバル変数は便利だが、多用するべきではない！！

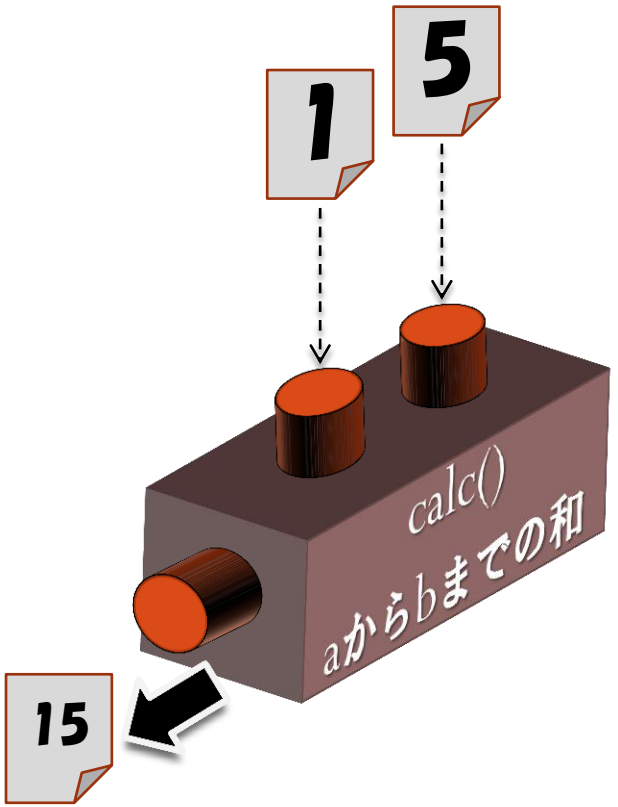
# 【練習】関数の活用
## ～１から５までの和を求める～

```c
#include <stdio.h>
int calc(int, int);//プロトタイプの宣言
//メイン関数
int main(void){
    //変数の宣言と初期化
    int ans = 0;//
    //関数の呼び出し
    ans = calc(1, 5);
    //出力
    printf("ans=%d", ans);
    return 0;
}
//a～bまでを足し合わせる関数を定義
int calc(int a,int b){

}
```

引数

戻り値

数字を足し合わせる処理



# まとめ①
# ～関数のメリット・デメリット～

- 関数は引数(与えた値)を指示通りに処理し、戻り値(結果)を吐き出す箱のような物
- なるべく機能ごとにモジュール分けするべき

**☆可読性が良くなる**

・入力と出力が明確

・同様の処理を何度も行いたい場合

☆デバッグが関数単位で行える

・開発効率が上がる。

**☆呼び出し元と処理を行う記述が別々**

・大きなプログラムだとわかりにくい

☆関数定義→先定義か後定義かの順序認識が必要

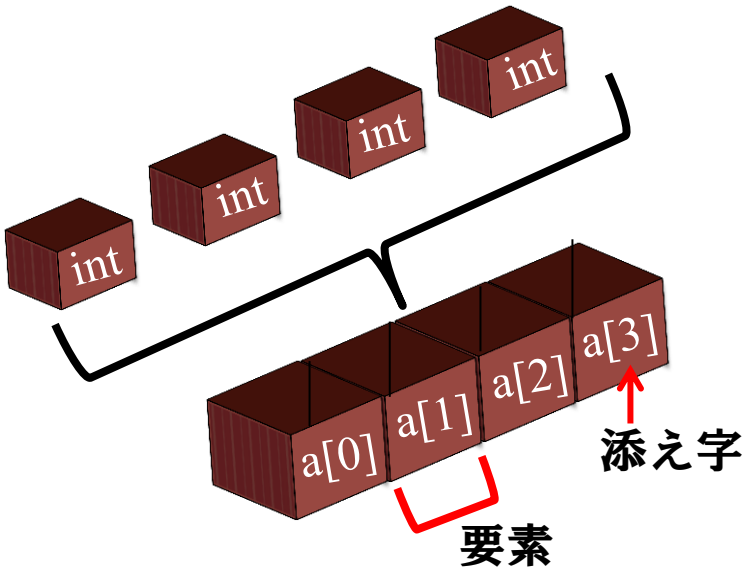
# ～配列～

# 配列とは？？

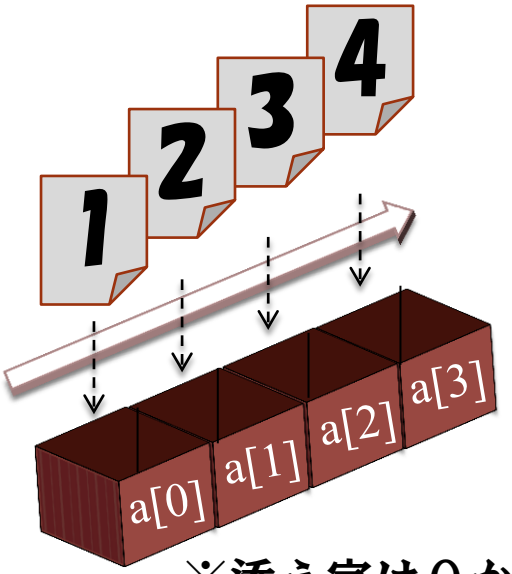
**複数の同じ型の変数を一つにまとめたもの。**

```
int a[4];
```

**型 配列名[配列の大きさ];**




```
int a[4] = {1, 2, 3, 4};
```




※添え字は０から始まる

# 配列要素の参照と代入

```
 1 #include <stdio.h>↓
 2 ↓
 3 int main(void){↓
 4 ↓
 5     int a[4]; //配列の宣言↓
 6 ↓
 7     a[0] = 1; //値を代入↓
 8     a[1] = 2;↓
 9     a[2] = 3;↓
10     a[3] = 4;↓
11 ↓
12     //a[0]の値を表示↓
13     plintf("a[0]=%d¥n",a[0]);↓
14 ↓
15     return 0;↓
16 }↓
```

int型の箱を４つ用意

**初期化は必ず行う！！**

普通の変数と同様に値を代入

普通の変数と同様に値を参照

※a[4]は配列の範囲外！！

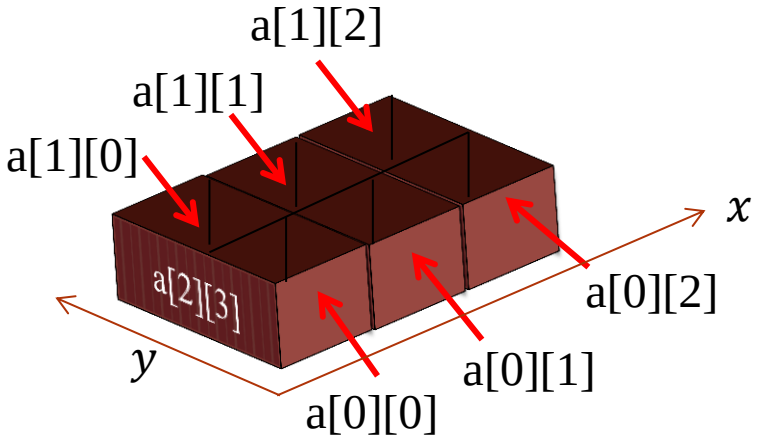
# 多次元配列について

**２次元配列**

int a[2][3]

**３次元配列**

int a[2][2][3]

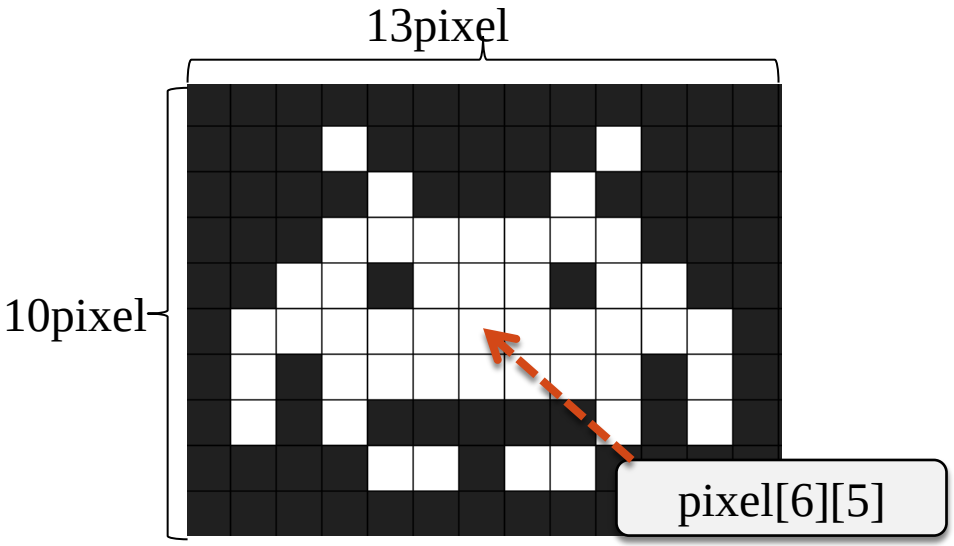



**例１：表の管理**

| | | | | |
|---|---|---|---|---|
| Oさん | 2 | 1 | 2 | 1 |
| Wさん | 4 | 5 | 5 | 2 |
| Kさん | 9 | 8 | 3 | 3 |

```
number[3][4];   //配列の宣言
number[0][0] = 2; //配列に値を代入
printf("%d", number[2][3]) ;//出力
```

**例２：画像データを格納**

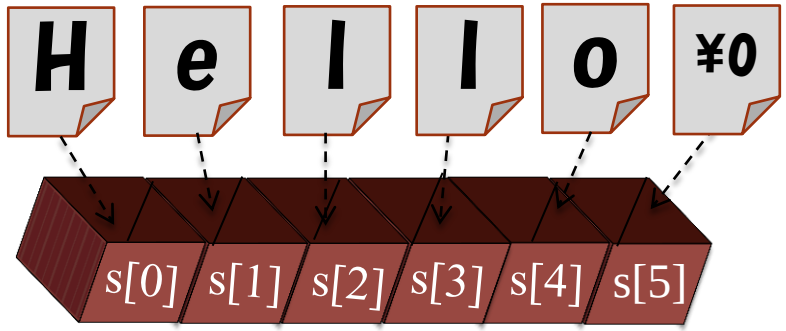


# 配列で文字列

**文字列**　C言語･･･文字列は配列を使った文字の集まり

```
char s[6] = “Hello”;
```

文字数よりも1文字多く宣言
(書かなくてもよい)




Null文字
文字列がここで最後

```
printf(“%c”,s[1]);
```

→ e

```
printf(“%s”,s);
```

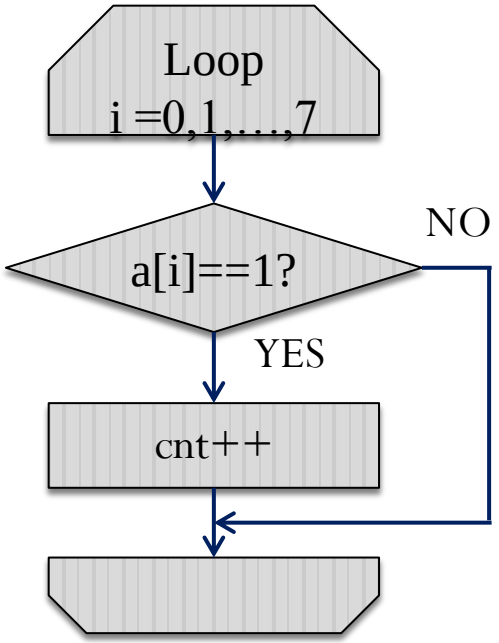
→ Hello

# 【練習】配列の活用
# ～数字を探索しよう～

①{1,1,9,2,1,1,1,5}を配列に格納

②for文を使って「１」が何個あるか求める！

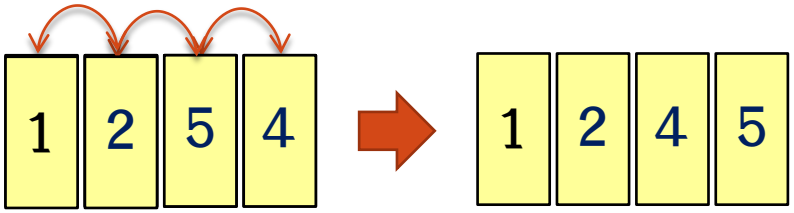
```
1  #include <stdio.h>
2
3  int main(void){
4      int a[] = {1,1,9,2,1,1,1,5};
5      int i;
6      int cnt=0;
7
8
9
10
11
12
13
14     printf("1の数は%dです",cnt);
15
16     return 0;
17 }
18
```



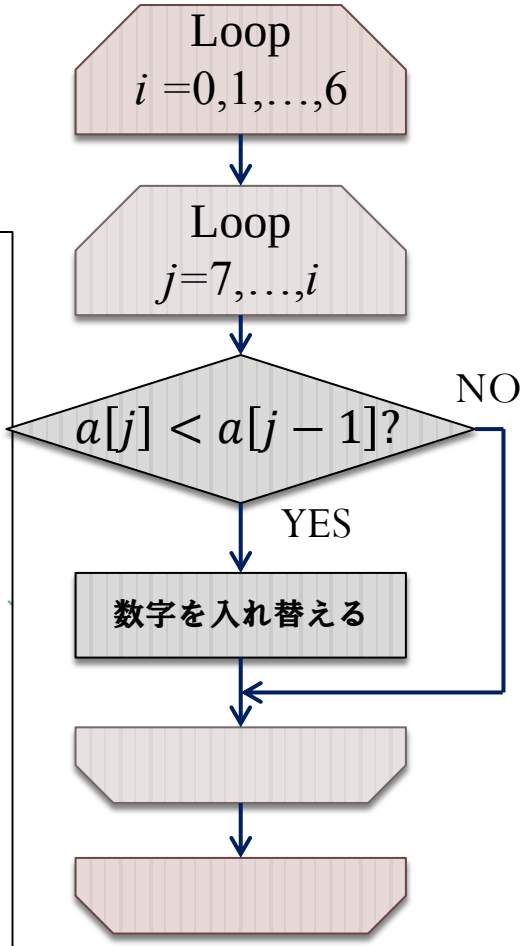

線形探索法（リニアサーチ）

１の数は５です

# 【練習】配列の活用
## ～数字の並び替え(バブルソート)～




```
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```



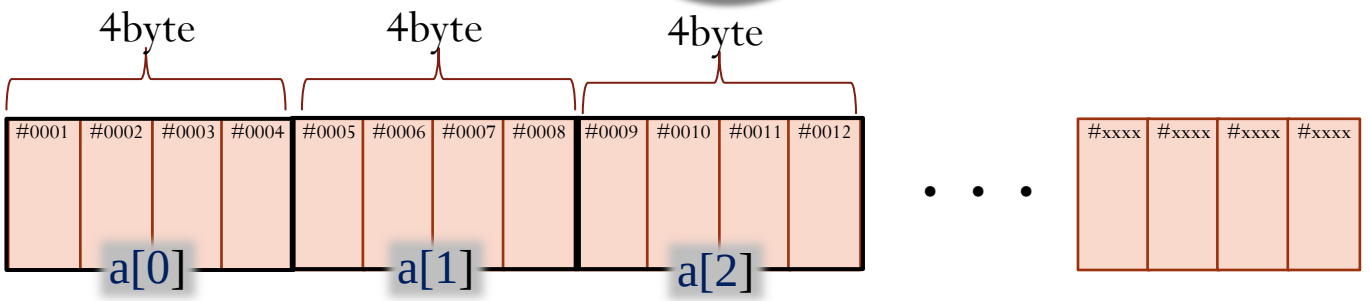
# まとめ②
# ～配列とメモリについて～

**配列を宣言**

int a[3];

メモリ上に領域確保(自動)

**◎静的なメモリ確保**




**☆処理をまとめて行うことができる**
**☆変数の数を減らすことができ可読性が良くなる**

**☆余分に宣言するとメモリがあまってしまったとき無駄**
**☆もっと必要になったときに拡張しにくい**

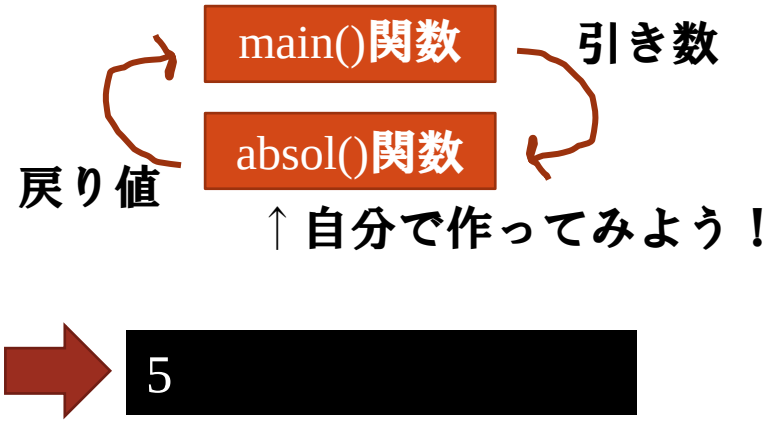
# 【宿題】

**標準ライブラリ関数の中身を考えて作ってみよう！！**

**①abs()　絶対値を求める関数**

**②pow()　累乗を求める関数(今回は引数はint型でOK)**

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

int main(void){
  int a = -5;
  printf("%d,",abs(a)); //aの絶対値
}
```

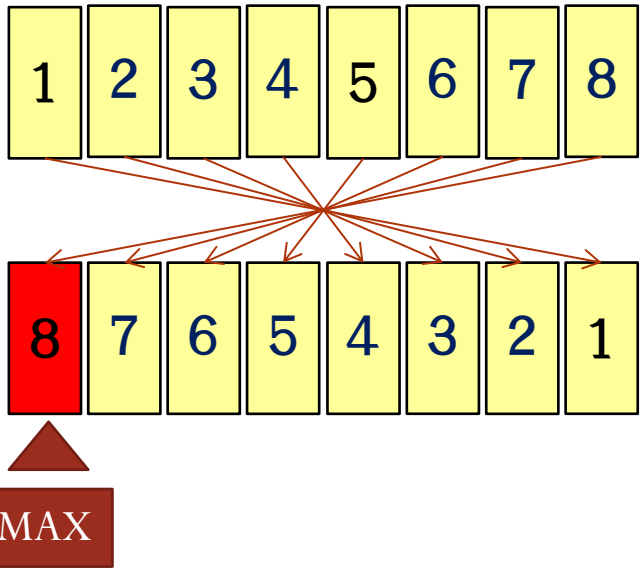



中身がどうなっているか考えてみよう！

# 【宿題】

①数字の並び替えをやってみよう！
　配列に8個の整数(適当)を格納して逆に並び替える
②同じ配列の中から最大値を求めてみよう！

①配列の並び替え




②最大値を調べる

探索などで調べてみよう！

# 【練習】while文の練習

**現在大島さんの列には10000人の人が並んでいて、毎分100人ずつ増えています。一方渡辺さんの列は現在5000人の人が並んでいて毎分350人の人が並んでいます。渡辺さんの列が大島さんの列よりも長くなるのは何分後でしょう？**

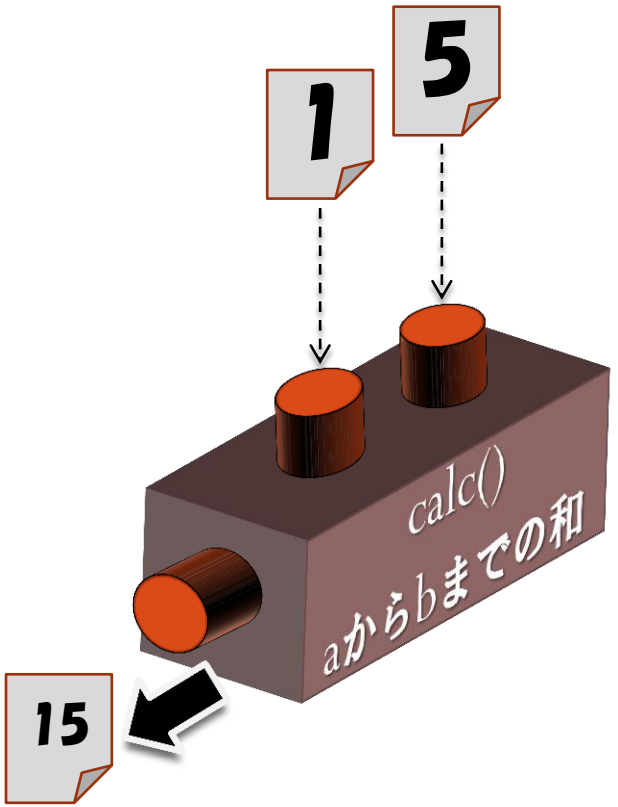
```
 3 int main(void){↓
 4 ›   int yuko = 10000; //大島さんの列↓
 5 ›   int mayu = 5000; //渡辺さんの列↓
 6 ›   int min = 0; //時間↓
 7 ›   ↓
 8 ›   while(yuko > mayu){↓
 9 ›   ›   yuko += 100; //100人ずつ増える↓
10 ›   ›   mayu += 350; //350人ずつ増える↓
11 ›   ›   min++; ↓
12 ›   }↓
13 ›   printf("%d¥n",min); ↓
14 ›   ↓
15 ›   return 0;↓
16 }↓
```

# 【練習】関数の活用
## ～１から５までの和を求める～

```c
#include <stdio.h>
int calc(int, int);//プロトタイプの宣言
//メイン関数
int main(void){
    //変数の宣言と初期化
    int ans = 0;//
    //関数の呼び出し
    ans = calc(1, 5);
    //出力
    printf("ans=%d", ans);
    return 0;
}
//a～bまでを足し合わせる関数を定義
int calc(int a,int b){
    int i, n = 0;
    for(i = a; i <= b; i++){
        n+= i;
    }
    return n;
}
```

引数

戻り値

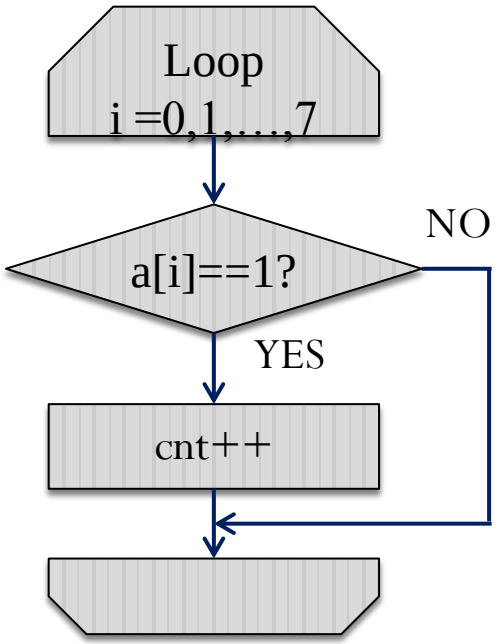


# 【練習】配列の活用
# ～数字を探索しよう～

**①{1,1,9,2,1,1,1,5}を配列に格納**

**②for文を使って「１」が何個あるか求める！**

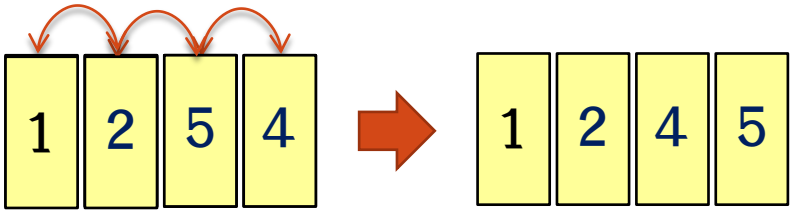
```
#include <stdio.h>
int main(void){
    int a[] = {1,1,9,2,1,1,1,5};
    int i;
    int cnt=0;

    for(i = 0; i < 8; i++){
        if(a[i] == 1){
            cnt++;
        }
    }

    printf("1の数は%dです",cnt);

    return 0;
}
```




**線形探索法（リニアサーチ）**

**１の数は５です**

# 【練習】配列の活用
## ～数字の並び替え(バブルソート)～




```
5   int i, j;
6
7   //最後の要素を除いてすべての要素を並び替える
8   for(i = 0; i < 7; i++){
9       //下から上に順番に比較する
10      for(j = 7; j > i; j--){
11          //上の方が大きいときは互いに入れ替える
12          if(a[j] < a[j-1]){
13              int t = a[j];
14              a[j] = a[j-1];
15              a[j-1] = t;
16          }
17      }
18  }
19
```