いわーく勉強会

第1回

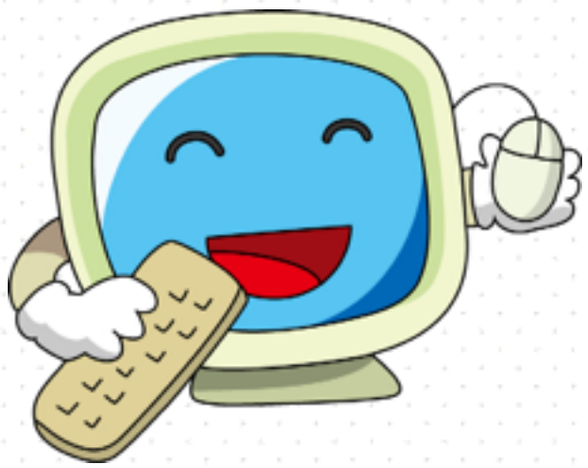
僕らはコンピュータを
動かしたい
コンピュータって何だろう

コンピュータの３大原則

1. コンピュータは、**入力**、**演算**、**出力**を行う装置である。

2. プログラムは、**データ**と**命令**の集合体である。

3. コンピュータの都合は、人間の感覚と異なる場合がある。



数字なら任せてよ！

コンピュータの３大原則

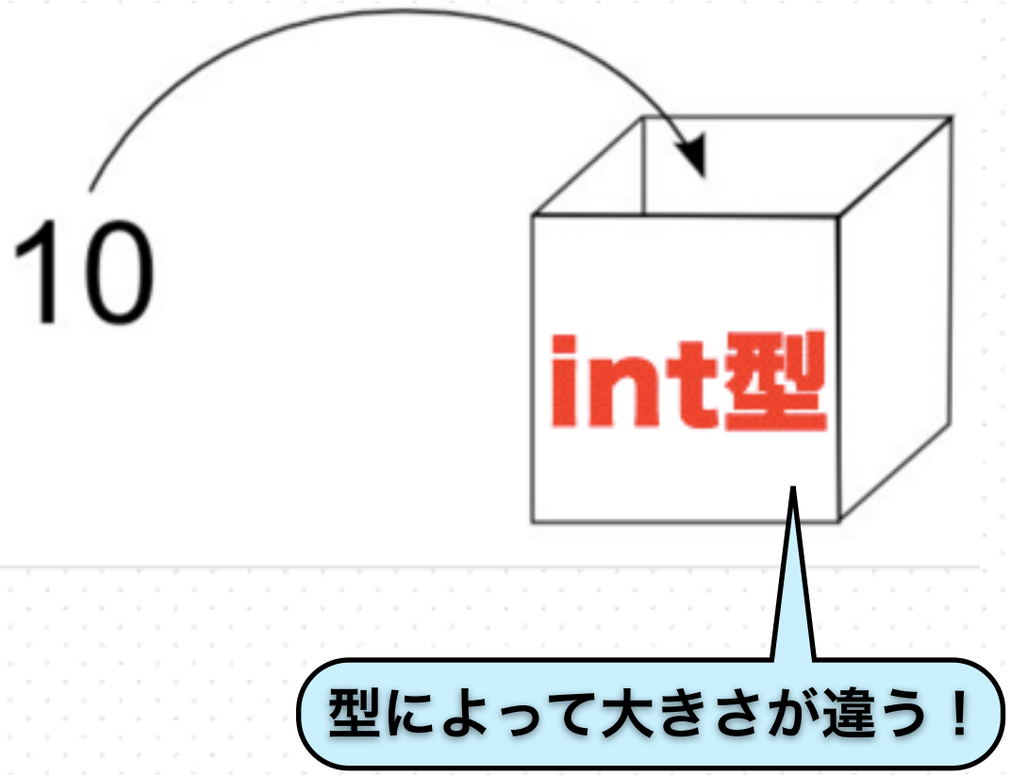
2．プログラムは、**データ**と**命令**の集合体である。

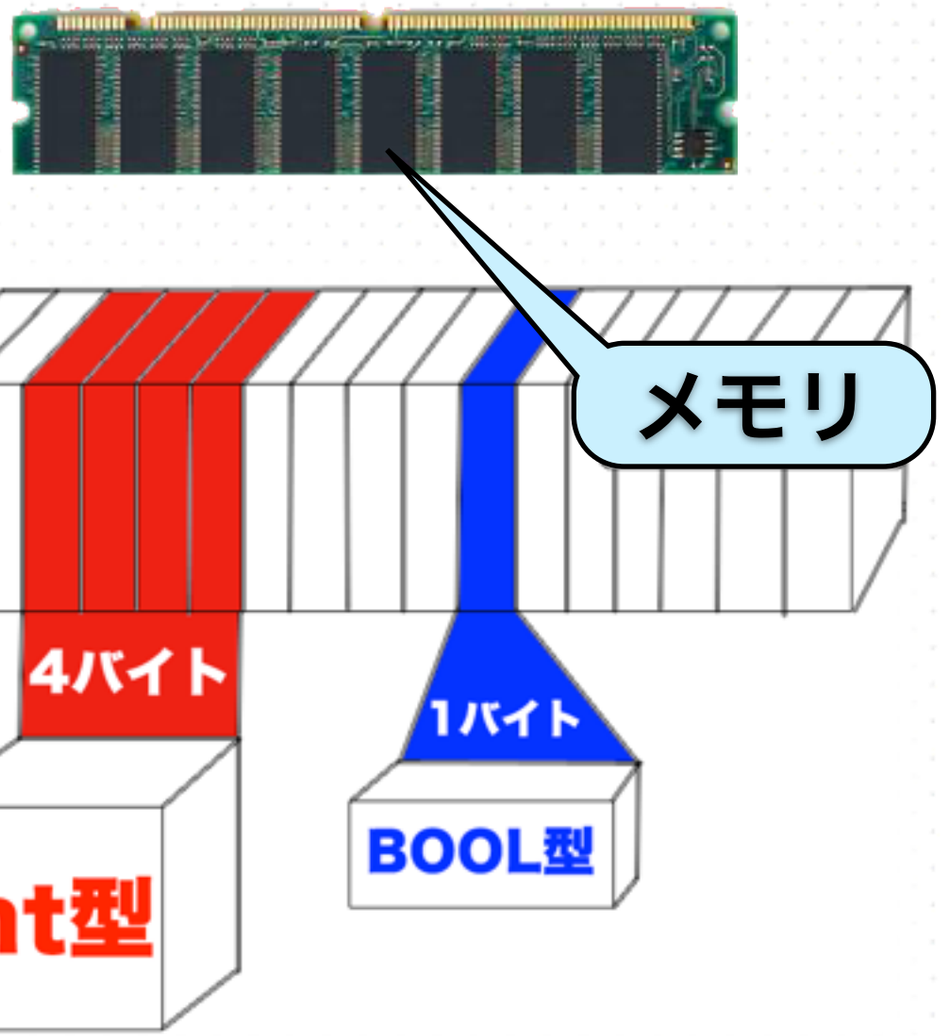
変数と
メソッド

変数

変数とは

変数とは、**値**を**記憶**しておく**入れ物**で、箱のようなイメージです。
コンピュータの**メモリ**という場所に、配置（確保）されます。





変数の型

変数を箱だとすると、変数の **型** は、その大きさを表します。
型には、 **基本型** と **参照型** の２種類があります。
よく使われる型には、次のようなものがあります。

基本型

整数	int
小数	float
文字(１文字)	char
真偽値	BOOL

**真偽値は、
YES または NO の
どちらかを持ちます。**

参照型

文字列	NSString *

**参照型は、変数名の前に
*(アスタリスク)がつきます。**

```
例： NSString *str ;
```

変数の使い方

変数は、まず、**型** を **宣言** してから使います。

例えば、int型（整数）の変数aは、次のように宣言します。

```
int a ;
```

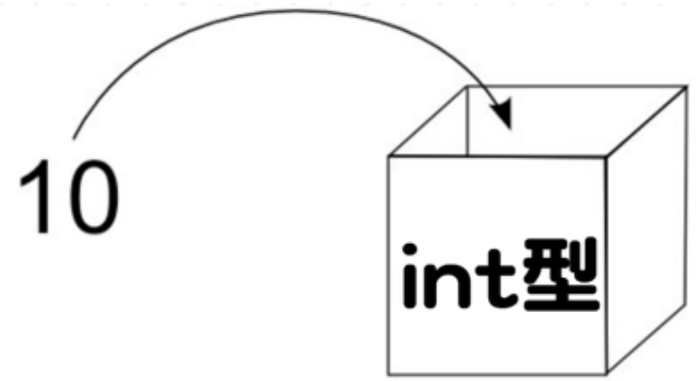
宣言した変数には、次のように値（データ）を入れられます。

```
a = 10 ;
```

宣言と同時に値を入れることもできます。

```
int a = 10 ;
```

これを
初期化 といいます。



文字列

参照型 の例として、文字列の`NSString*`型を挙げました。

`@" "` で囲んだ値を 文字列 といい、

次のように、ラベルのテキストの設定などに利用できます。

文字列

```
- (IBAction)push
{
    label.text = @"ボタンが押されました。";
}
```

文字列を変数に入れるためには、`NSString*`型を用います。

```
NSString *mojiretu = @"もじれつ。" ;
label.text = mojiretu ;
```

変数を含む文字列

カウントアプリで、ボタンを押した回数をラベルに表示しました。
次のように、**文字列に変数を含める**ことができます。

```
int count = 10 ;
label.text = [NSString stringWithFormat:@"%d 回", count ] ;
```

ここにcount変数の中身が入ります。

「 **%d** 」の部分には、整数（int型）の変数が入ります。
小数の場合は「 **%f** 」、文字列の場合は「 **%@** 」を使います。

```
float shosu = 5.6 ;
NSString *mojiretu = @"もじれつ。" ;
label.text = [NSString stringWithFormat:@"小数:%f 文字列:%@", shosu , mojiretu ] ;
```

変数のスコープ(1)

変数には、使える範囲があり、これを**スコープ**と呼びます。
次の例を見てください。変数number を viewDidLoadメソッドで宣言して、plusメソッドで使おうとしていますが…

ViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];
   // Do any additional setup after loading the view, typically from a nib.

    int number = 0 ;
}

- (IBAction)plus
{
    number = number + 1 ;
}
```

viewDidLoadで宣言

plusで使おうとする

エラー！

変数のスコープ(2)

「{」と「}」で囲まれた範囲を、**ブロック**と呼びます。
変数は、宣言したブロックの内側でしか使えないのです。

ViewController.m

```
- (void)viewDidLoad
{
    int number ;

    if (number == 0)
    {
        int i ;

        for (i = 0; i < 3; i = i + 1)
        {
            number = number + 1 ;
        }
    }
}
```

① numberのスコープ

② iのスコープ

前ページのプログラムがエラーになってしまった
理由を考えてみよう！

ヘッダファイルでの宣言

同じ変数を複数のメソッドで使うにはどうしたら良いでしょうか。
実は、拡張子が「.h」である**ヘッダファイルで宣言**すると、
「.m」ファイルの、どこでも使えるようになります。

ViewController.h

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
{
    ① ここに変数の宣言
}

    ② ここにメソッドの宣言

@end
```

今までスコープを気にしなくて良かったは、
ここに宣言していたからなんだ！

※ ヘッダファイルでは、変数への代入や初期化はできないので注意してください。

コンピュータの３大原則

2．プログラムは、**データ**と**命令**の集合体である。

変数と
メソッド

メソッド

起動したよ！

ボタンが押されたよ！

イベントドリブン

Viewが読み込まれたよ！

メソッドの作り方

ボタンが押されたときの動作は、次のように作ることができます。

```
- (IBAction)動作の名前
{
        動作の内容
}
```

この「動作」のことを、
「メソッド」と呼びます。

前ページで書いた動作は 「**push**」という名前で、
「**ラベルのテキストを変える**」メソッドだったことがわかります。

ViewController.m

```
- (IBAction)push
{
    label.text = @"ボタンが押されました。" ;
}
```

pushメソッド

配列

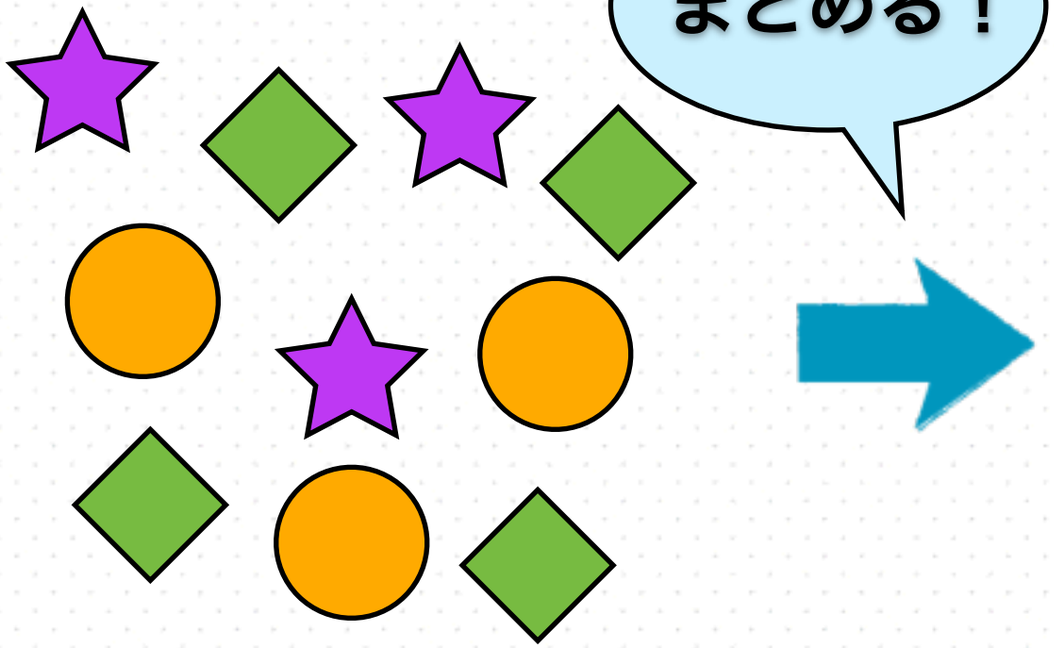
配列の使い方

配列とは何か、そして配列の使い方について復習しましょう。

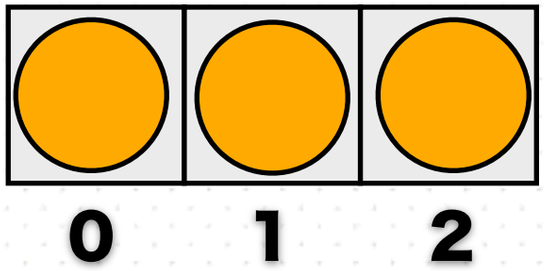
1. 配列とは
2. 配列の作り方
3. 配列への入れ方

配列とは

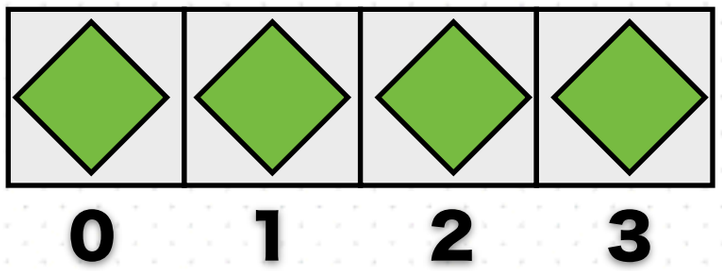
同じ種類のデータをまとめたものを **配列** といいます。



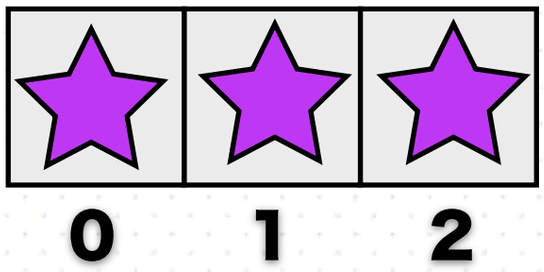
配列1（本棚）



配列2（本棚）



配列3（本棚）



配列の作り方

次のように、６冊の漫画をまとめる配列を作れます。
変数が１つにまとまり、プログラムが簡単になります。

＊配列を使わない場合

```
NSString *manga1 ;
NSString *manga2 ;
NSString *manga3 ;
NSString *manga4 ;
NSString *manga5 ;
NSString *manga6 ;
```

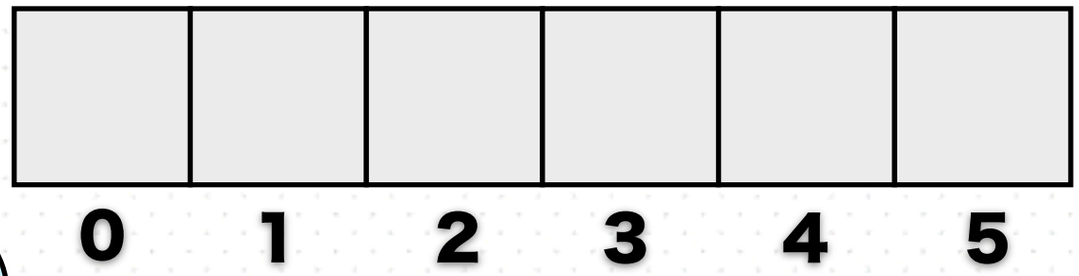
簡単に
書ける！

＊配列を使う場合

```
NSString *mangaArray[6] ;
```

データ型　　配列名　　要素数

mangaArray配列（中身はまだなし）

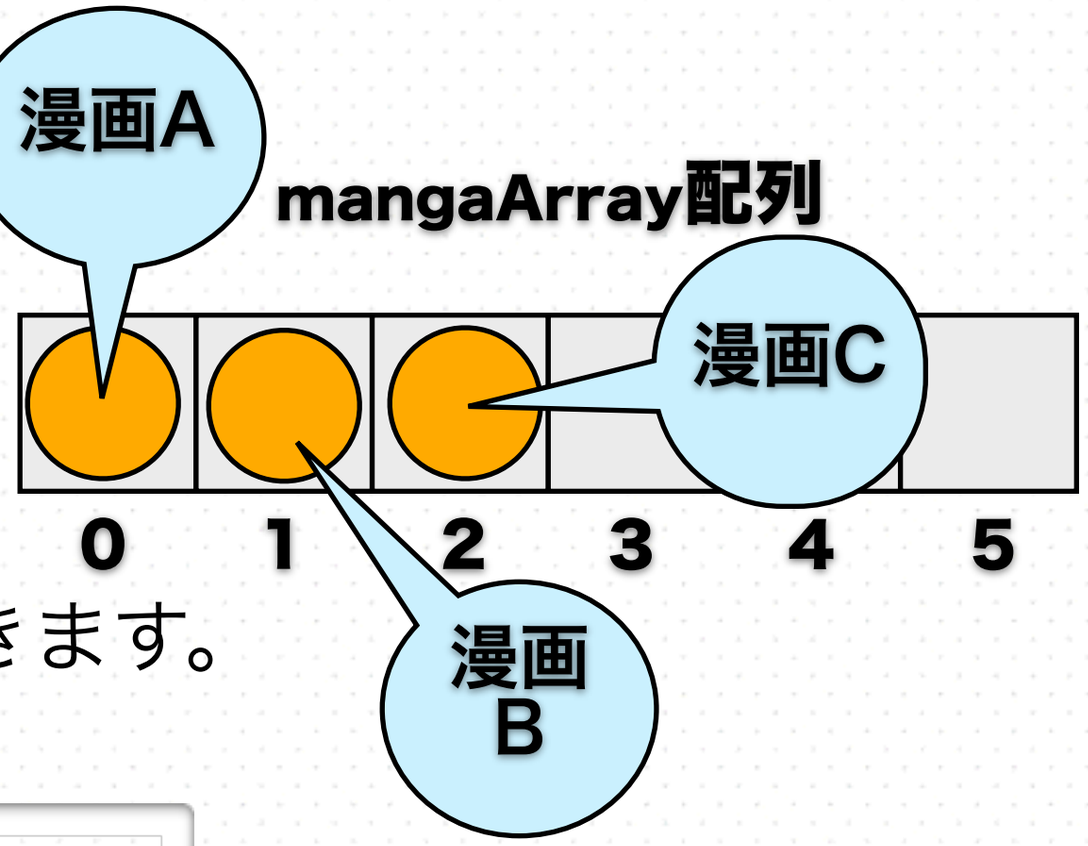


配列への入れ方

次のように、各要素にデータを入れることができます。

要素番号を指定して、漫画のタイトルを入れています。

```
mangaArray[0] = @"漫画A" ;
mangaArray[1] = @"漫画B" ;
mangaArray[2] = @"漫画C" ;
```



要素番号に変数を使うこともできます。

```
int index = 0 ;
mangaArray[index] = @"漫画A" ;
```

復習⑤

制御文

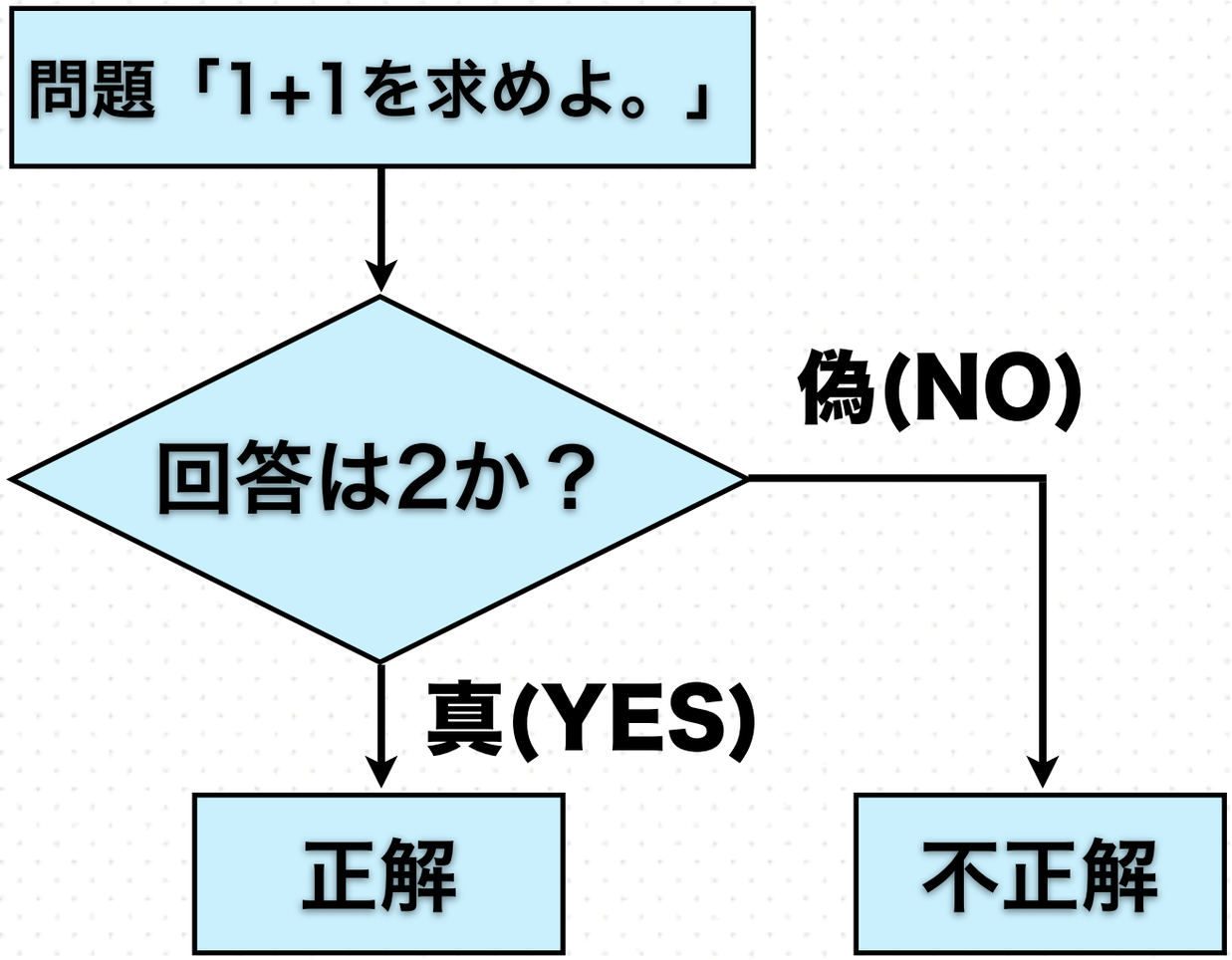
制御文

if文などの条件分岐や**for文**などのループを、**制御文**といいます。
これらについて、復習しましょう。

1. **if**文とは
2. **if**文の使い方
3. ループ＝繰り返し
4. **while**文の使い方
5. **for**文の使い方

if文とは

プログラムでは、条件によって処理を変えることがあります。
これを実現するのが、`if`文です。
`if`文は、条件が**真(YES)**か**偽(NO)**かで、処理を変えます。



if文の使い方(1)

if文は、次のように使うことができます。

```
if ( 条件① )
{
        条件①がYESのときの処理

} else if ( 条件② ) {
        条件①がNOで、条件②がYESのときの処理

} else {
        全ての条件がNOのときの処理

}
```

else if は、いくつでもつなげることができます。無くてもOK。

else { ... } も、無くてもOK。

if文の使い方(2)

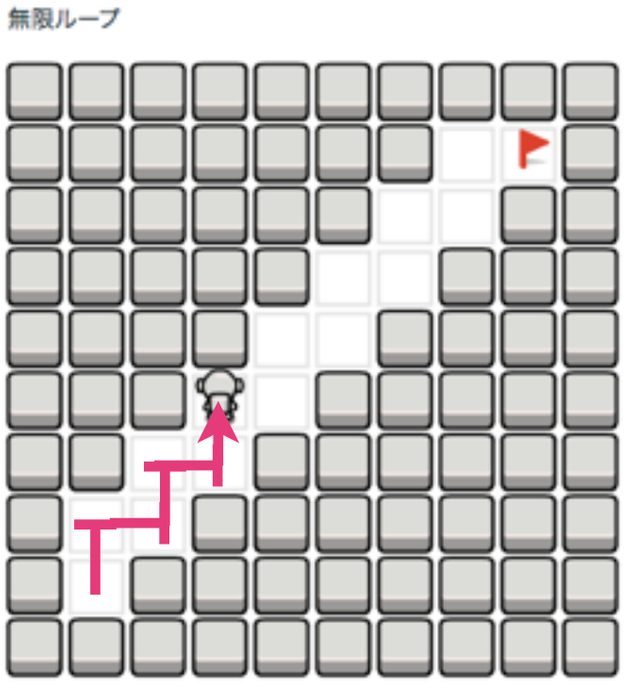
if文を使った例を見てみましょう。
次の例では、**else if** と、**else** も使っています。

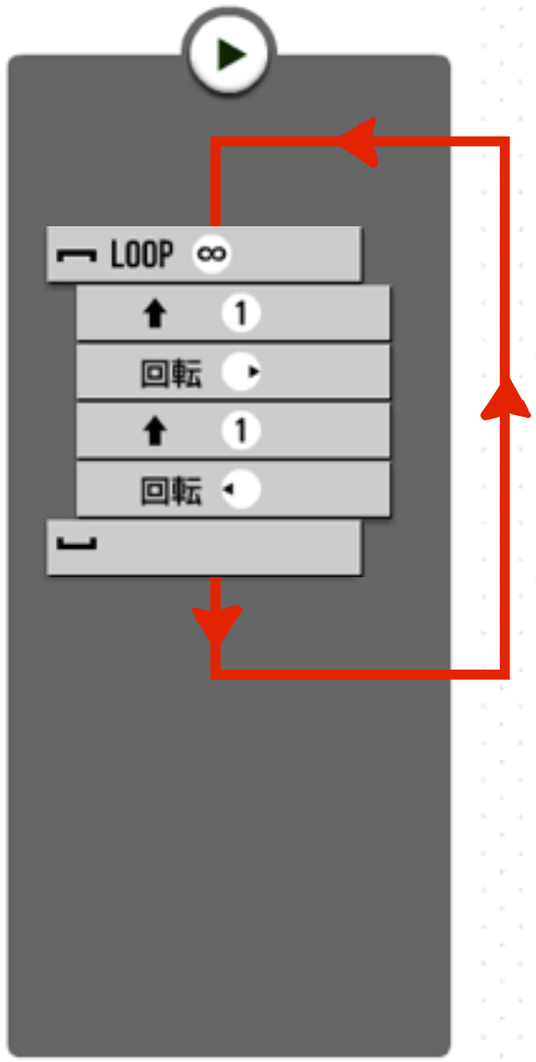
```
if (kotae == 2)
{
    label.text = @"その通り、1+1は 2 だな。" ;
} else if ( kotae == 11 ) {
    label.text = @"違うぞ、1+1が 11 なわけないだろう！" ;
} else {
    label.text = @"違うぞ、なにを言っているんだ。" ;
}
```

ループ＝繰り返し

第２回の講義で、アルゴロジック2 (※) で遊びました。

あの時、**ループ（LOOP）**を体験しています。





① 前に進む。

② 右を向く。

③ 前に進む。

④ 左を向く。

の**繰り返し**で旗が取れました。

※ http://home.jeita.or.jp/is/highschool/algo/prm/index2.html

while文の使い方

ループを使う方法に、**while**文と**for**文というものがあります。

while文は、次のように使うことができます。

ループをしない場合

```
int i = 0 ;

i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
i = i + 1 ;
```

while文を使う場合

```
int i = 0 ;

while ( i < 10 )    // i < 10 の間、繰り返す
{
    i = i + 1 ;     // iの値を1増やす
}
```

ループを続ける条件

条件を満たしていれば

ループする内容

for文の使い方

for文は、① ループを始める前に変数を初期化し、
② 内容を繰り返すたびに、変数の値を変化させます。

ループをしない場合

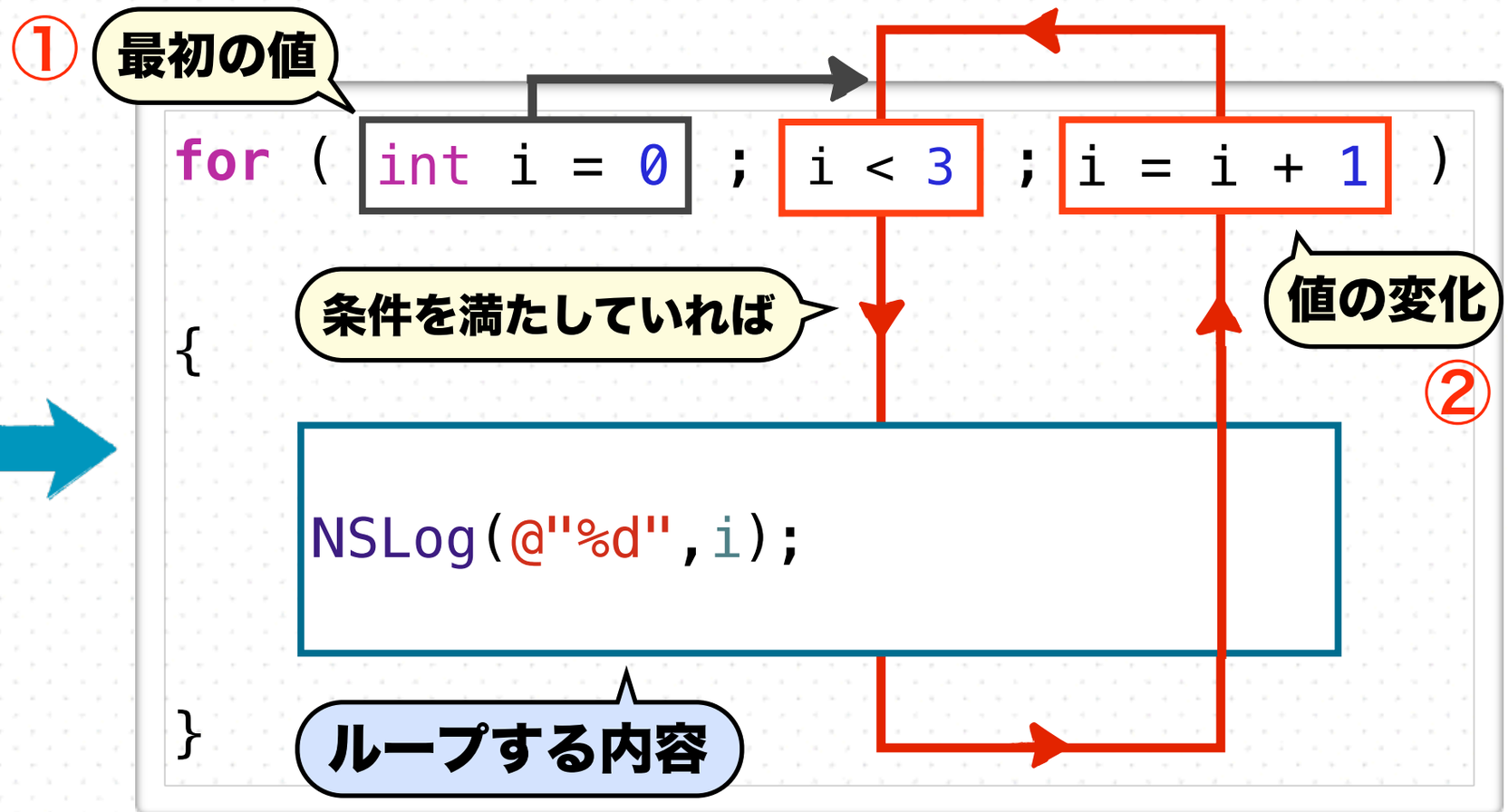
```
int i = 0 ;

NSLog(@"%d",i);
i = i + 1 ;

NSLog(@"%d",i);
i = i + 1 ;

NSLog(@"%d",i);
i = i + 1 ;
```

for文を使う場合



綺麗なコード

誰かが読むことを意識する。

４つの原則

１．簡潔性

２．明瞭性

３．一般性

４．自動化

変数の名前の付け方

スコープの広い変数ほど名前から伝わる情報量を増やす。

- グローバル変数はわかりやすい名前。

```
NSString *navTitle; // ナビゲーションバーに表示するタイトル
```

- ローカル変数は短い名前。

```
MGLine *line = section.topLines[i];
```

コメントをつけるとなお良い

インデント

- インデントを整える
  control + i　（Xcodeの場合）
- マジックナンバー
- コメント

突然ですが試験です

Countアプリを

作ってください

これから

MVC Countアプリに

進化させます

MVCとKVO

バージョン 1

バージョン 2

非同期な通知