

多言語 プログラミング

～プログラミングにおける骨格～

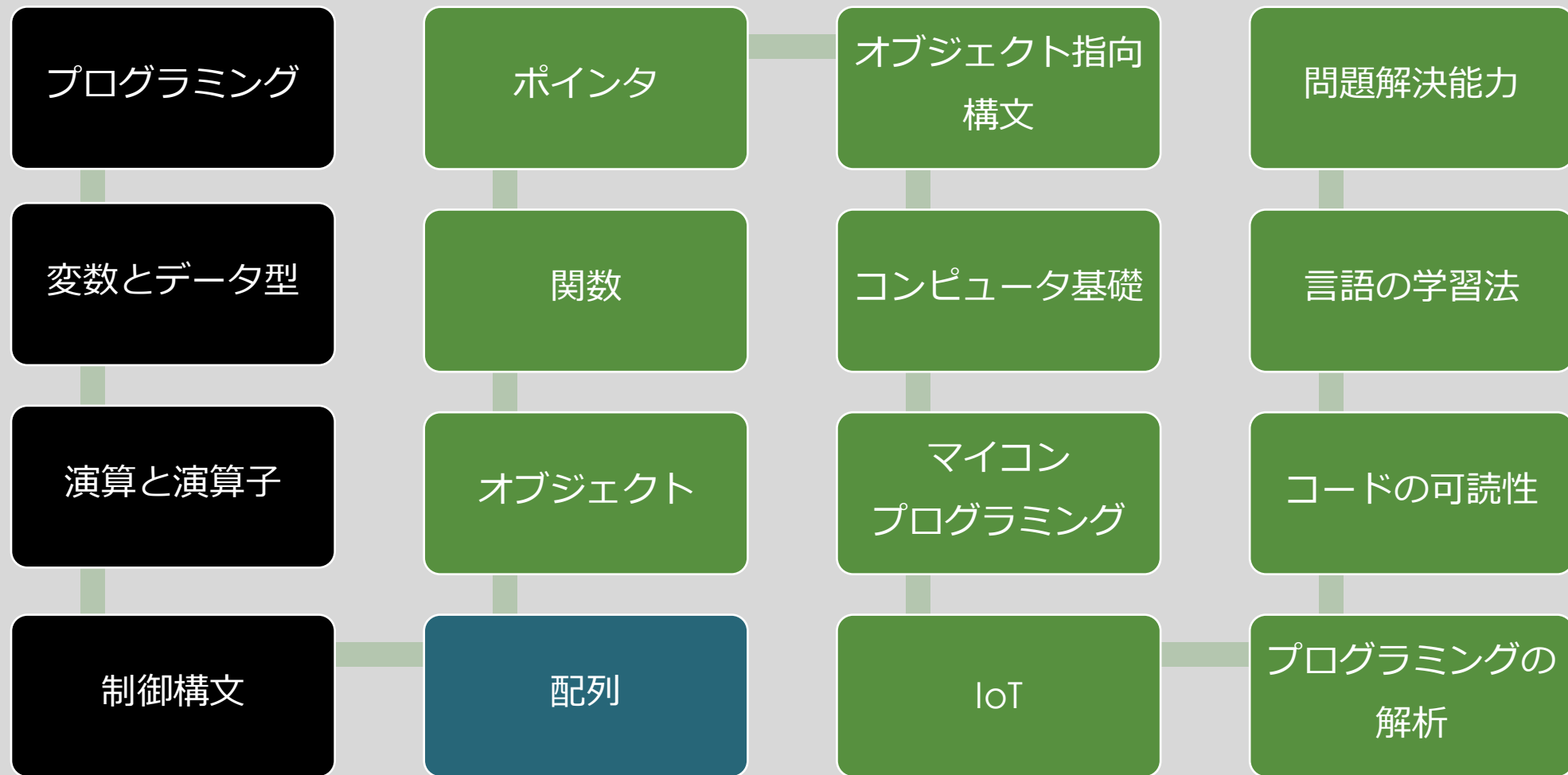




配列

～変数をまとめる～

進度



配列

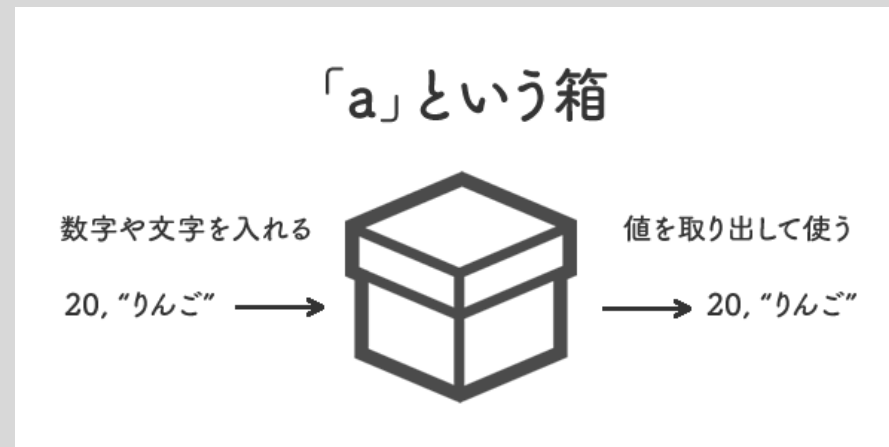
- 同じデータ型の変数をひとつに**まとめたもの**

- オブジェクト指向言語では配列を**オブジェクト**として扱う

➡ 後述

- 文字列はchar型の配列として扱う

- Cでは**静的**な配列、オブジェクト指向では**動的**な配列



基本要素

- 宣言

配列を作成すること

- 初期化

配列の作成とともに数値を割り当てること

- 代入

配列の値を上書きすること

宣言

```
int x[5];
```

```
let x;
```

- 。配列の宣言は言語によって様々（静的はC、動的はJSの例）

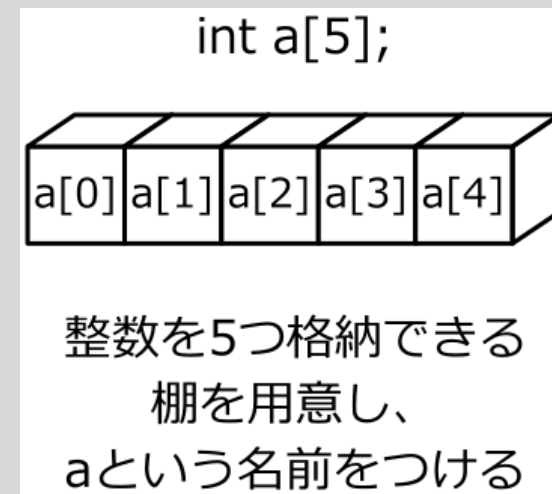
要素型 配列名[要素数]（静的）

let 配列名（動的）

- 。要素数は定数、要素型は要素に入れるデータ型
- 。動的配列は後述のインスタンス化によっても宣言可能
- 。添字演算子

配列名 + [先頭から何個後ろか] として配列にアクセス

[]を添字演算子と呼ぶ



初期化

```
int y[5] = {1, 2};  
//先頭から順に1, 2, 0, 0, 0
```


```
let x = [0, 1, "hoge"];  
//動的では個々に型が分かれる
```

- 宣言と共に値を割り当てる
 - 要素型 配列名[要素数] = { 初期値 } (静 的)
 - let 配列名 = [初期値] (動的) 静的はC、動的はJSの例
- 静的配列：要素数より少ない数を割り当てると、それ以外は0になる
 要素数を超えて割り当てると、エラーになる
- 動的配列：要素数も変動する

またJSの配列(arrayオブジェクト)では、複数のデータ型をまとめることが出来る

代入

```
x[0] = 3;  
//先頭から0番目(先頭含め1つ目)に3を代入  
x[4] = 2;  
//先頭から4番目(先頭含め5つ目)に2を代入
```

- 配列に配列を代入することは不可
  後述のオブジェクトで代入することは可能
- 配列名[先頭からのカウント] = 値
 として添字演算子で配列の要素にアクセスし、代入する
- 配列を丸ごとコピーするには、制御構文を用いた工夫が必要

配列と制御構文

- 。列は制御構文を用いて操作をする

➡ 構造化プログラミングの手法

- 。代表的なコピーと反転について紹介
- 。大抵はfor文の繰り返し処理と組み合わせることが重要

配列のコピー

- for文で要素数分インクリメントして繰り返す
- コピーする側とされる側両方に 同じ要素を指定して代入
- 0からインクリメントすると先頭から順番にコピーされていく

```
int x[5] = {1,2,3,4,5};  
int y[5] = {0}  
  
for(int i = 0, i < 5; i++){  
    y[i] = x[i];  
}
```

配列の反転

- 要素数の半分だけ繰り返す
- 入れ替えるときに塗り替えられる値は
あらかじめ適当な変数に格納して保存しておく

```
int x[5] = {1,2,3,4,5};  
  
for(int i = 0; i < 2; i++){  
    int temp = x[i];  
    x[i] = x[4-i];  
    x[4-i] = temp;  
}
```

x[0]をtempに保存 ➡ x[0]にx[4]代入 ➡ x[4]にtemp代入

x[1]をtempに保存 ➡ x[1]にx[3]代入 ➡ x[3]にtemp代入

演習 5 “配列”①

1. Arduinoを用いて配列を作成し、
for文を使用して配列aから配列bにコピーする

2. Arduinoを用いて配列を作成し、
while文を使用して配列aを反転させる

Serial.println(引数)を用いてシリアル通信でモニタに表示

演習 5 “配列”① ヒント

```
int x[5] = {1,2,3,4,5};  
int y[5] = {0}  
  
for(int i = 0, i < 5; i++){  
    y[i] = x[i];  
}
```

1. 配列表示にはprintArray関数を使用してください
2. 1は配列2つ宣言⇒for文でコピー⇒表示
3. 2は配列1つ宣言⇒表示⇒for文で反転⇒反転後の配列表示

```
void printArray(int *array, int num){  
    for(uint8_t i = 0; i < num; i++){  
        Serial.print(array[i]);  
        Serial.print(",");  
    }  
    Serial.print("¥n");  
}
```

```
int x[5] = {1,2,3,4,5};  
  
for(int i = 0; i < 2; i++){  
    int temp = x[i];  
    x[i] = x[4-i];  
    x[4-i] = temp;  
}
```

演習 5 “配列”① 回答

```
int a[5] = {1,2,3,4,5};
int b[5] = {0};

Serial.print("a : ");
printArray(a, 5);

for(int i = 0; i < 5; i++){
    b[i] = a[i];
}

Serial.print("b : ");
printArray(b, 5);
```

```
[Starting] Openi
a : 1,2,3,4,5,
b : 1,2,3,4,5,
```

```
int a[5] = {1,2,3,4,5};

Serial.print("a : ");
printArray(a, 5);

for(int i = 0; i < 2; i++){
    int temp = a[i];
    a[i] = a[4 - i];
    a[4 - i] = temp;
}

Serial.print("a : ");
printArray(a, 5);
```

```
[Starting] Openi
a : 1,2,3,4,5,
a : 5,4,3,2,1,
```

多次元配列

- 配列を配列として縦に並べると...

次元の多い配列が出来る



- 配列を要素とした配列と捉える ← 多次元配列

- 添字演算子が1つ増えるだけの違い

1次元配列

0	12
1	24
2	18

要素数 3

2次元配列

0	0	12
	1	24
	2	18
1	0	31
	1	9
	2	27

要素数 2

3次元配列

0	0	0	12
	1	2	18
1	0	1	31
	1	2	27
1	0	0	4
		1	30
	1	0	12
		1	15

要素数 2

多次元配列基本要素

- 要素型 配列名[行要素][列要素] =
 { { 初期値} , { 初期値} ... }
- let 配列名 = [[初期値],[初期値],...] 静的はC、動的はJSの例
- 配列名[行][列]
 で配列の要素にアクセスでき、代入が可能
- 多次元配列は行列的なので2重ループを使って処理をすることが殆ど

演習 5 “配列”②

1. $3*4$ の多次元配列を2つ作成し、全て乱数値で満たし
その合計値の配列を表示する
2. $8*8$ の多次元配列に0か1で座標データを入力し
1の時に*を描画して図形を表示させる

演習 5 “配列”② ヒント

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

```
Serial.print(x[0][1]);  
//2が表示される
```

1. 多次元配列の宣言例
2. 多次元配列からの値取得例
3. 多重ループの作り方
4. 1は3つの2次元配列宣言⇒2重ループで3つ目の配列に和を代入
5. 2は図形を決める⇒座標を配列で初期化⇒2重ループで表示

0の時 : Serial.print(" ");

1の時 : Serial.print("*");

行末 : Serial.print("¥n");

```
for(let i = 0; i < 5; i++){  
    for(let j = 0; j < 3; j++){  
        console.log(i * j);  
    }  
}
```

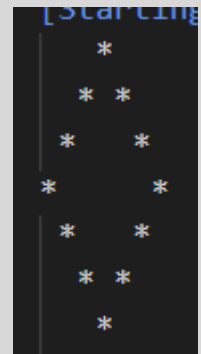
演習 5 “配列”② 回答

```
int x[3][4] =
    {{1,2,8,2},{23,234,4563,6321},{234,2134,6321,3465}};
int y[3][4] =
    {{2354,3425,574,854},{234,572,532,345},{435,623,578,48}};
int res[3][4] = {0};

for(int i = 0; i < 3; i++){
    for(int j = 0; j < 4; j++){
        res[i][j] = x[i][j] + y[i][j];
    }
}

Serial.print("result : ¥n");
for(int i = 0; i < 3; i++){
    Serial.print(i+1);
    Serial.print(" : ");
    printArray(res[i], 4);
}
```

```
result :
1 : 2355,3427,582,856,
2 : 257,806,5095,6666,
3 : 669,2757,6899,3513,
```



```
int zahyo[8][8] =
    {{0,0,0,1,0,0,0,0},{0,0,1,0,1,0,0,0},
     {0,1,0,0,0,1,0,0},{1,0,0,0,0,0,1,0},
     {0,1,0,0,0,1,0,0},{0,0,1,0,1,0,0,0},
     {0,0,0,1,0,0,0,0},{0,0,0,0,0,0,0,0}};

for(int i = 0; i < 8; i++){
    for(int j = 0; j < 8; j++){
        if(zahyo[i][j] != 0){
            Serial.print("*");
        }else{
            Serial.print(" ");
        }
    }
    Serial.print("¥n");
}
```

配列のまとめ

- 。配列にも変数と同じように宣言、初期化、代入の操作が存在
- 。配列丸ごと操作することは出来ず、制御構文を用いて操作する
- 。配列を要素とすることで多次元配列を生成可能に