実践ロボットプログラミング

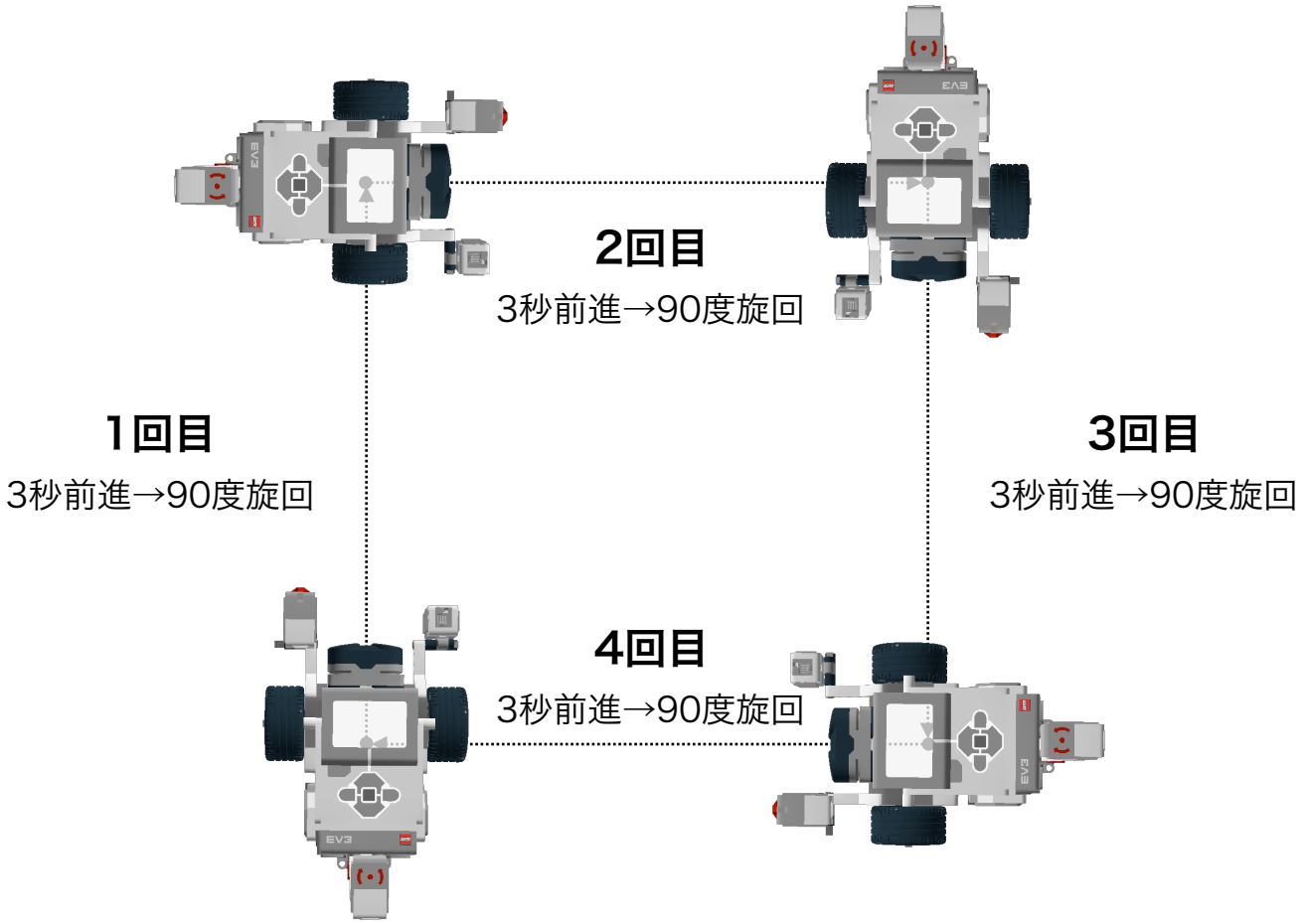
LEGO Mindstorms EV3 で目指せロボコン！

WEB：http://www.robot-programming.jp/
著者：藤吉弘亘, 藤井隆司, 鈴木裕利, 石井成郎
E-mail：support@robot-programming.jp

04: 関数化 (p.63〜)

ロボットを一周させる時

- 同じ処理が何度も繰り返される → **関数化**



関数化1 (p.64 func.c)

```
#include "./jissenPBL.h"
int main()
{
  int i ;
  OutputInit();
  OnFwdEx(OUT_BC, 50,0);
  Wait(3000);
  OnFwdEx(OUT_B, 50,0);
  OnRevEx(OUT_C, 50,0);
  Wait(500);
  Off(OUT_BC);
}
```

関数化

→

```
#include "./jissenPBL.h"

void forward(){
  OnFwdEx(OUT_BC, 50,0);
  Wait(3000);
}

void turn_right(){
  OnFwdEx(OUT_B, 50,0);
  OnRevEx(OUT_C, 50,0);
  Wait(500);
}
int main()
{
  OutputInit();
  forward();
  turn_right();
  Off(OUT_BC);
}
```

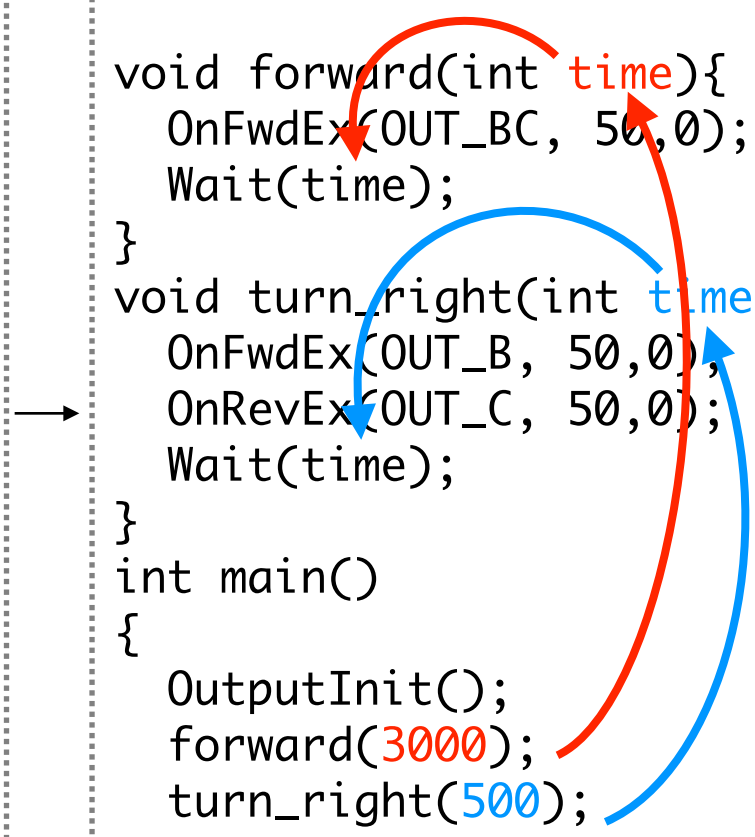
呼び出し

関数化2(p.64 func.c)

```
#include "./jissenPBL.h"

void forward(){
  OnFwdEx(OUT_BC, 50,0);
  Wait(3000);
}
void turn_right(){
  OnFwdEx(OUT_B, 50,0);
  OnRevEx(OUT_C, 50,0);
  Wait(500);
}
int main()
{
  OutputInit();
  forward();
  turn_right();
  Off(OUT_BC);
}
```

→

```
#include "./jissenPBL.h"

void forward(int time){
  OnFwdEx(OUT_BC, 50,0);
  Wait(time);
}
void turn_right(int time){
  OnFwdEx(OUT_B, 50,0);
  OnRevEx(OUT_C, 50,0);
  Wait(time);
}
int main()
{
  OutputInit();
  forward(3000);
  turn_right(500);
  Off(OUT_BC);
}
```



引数を持つ関数

- 関数の書式：void 関数名(引数){ }
- 引数は関数内のみ有効

#defineによる定義 (p.65)

```
#include "./jissenPBL.h"

#define TURN90 500

void forward(int time){
  OnFwdEx(OUT_BC, 50,0);
  Wait(time);
}
void turn_right(int time){
  OnFwdEx(OUT_B, 50,0);
  OnRevEx(OUT_C, 50,0);
  Wait(time);
}
```

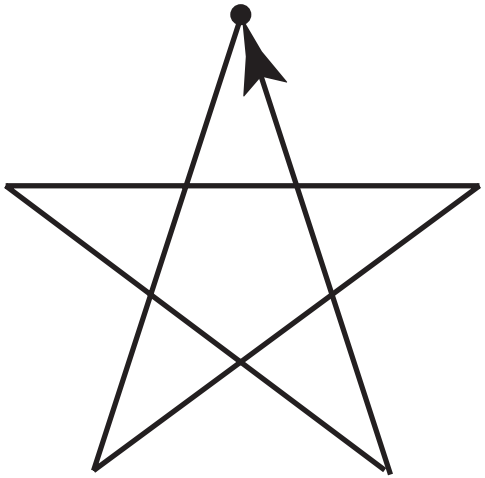
```
int main()
{
  int i;
  OutputInit();
  for(i=0; i<4; i++){
    forward(3000);
    turn_right(TURN90);
  }
  Off(OUT_BC);
}
```

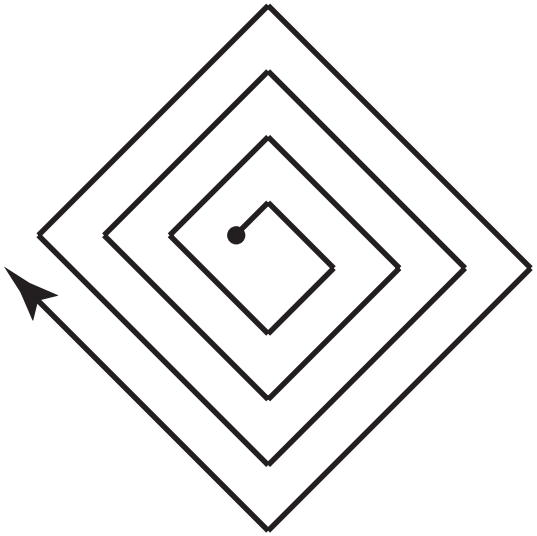
旋回角度（時間制御）を調整するには、TURN90の値を変るだけで全てに反映

■■　チャレンジ！　■■

.EV3

・ 星形やスパイラルの軌跡を描くロボットを実現してみよう！





05: 演算と変数 (p.66〜)

50cm前進するには？

- モータの回転角を変化させたときの直進距離を測定し、法則をみつける

```
RotateMotor(OUT_BC, 50, 180);
```

角度：
180～1440度

角度[°]	距離[cm]	1cmあたりの回転角度
180	9.0	20.0
360	17.5	20.6
720	34.5	20.8
1080	52.5	20.5
1440	69.5	20.4

中央値

※1cmあたりの回転角度 ＝回転角度[°] / 距離[cm]

回転角度 = 直進したい距離 × 1cmあたりの回転角度

20.5度

50cm前進するには？ (forward50cm.c)

- 実数型(double)を使用

```
#include "./jissenPBL.h"
int main()
{
  double angle;
  OutputInit();

  angle=50*20.5;
  RotateMotor(OUT_BC, 50, angle);
  Off(OUT_BC);
}
```

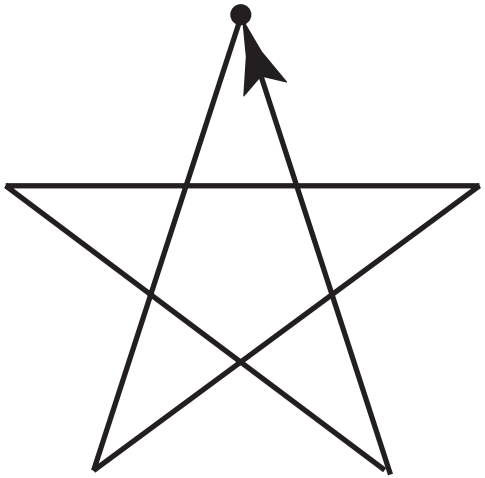
50cm前進するには？ (forward50cm.c)

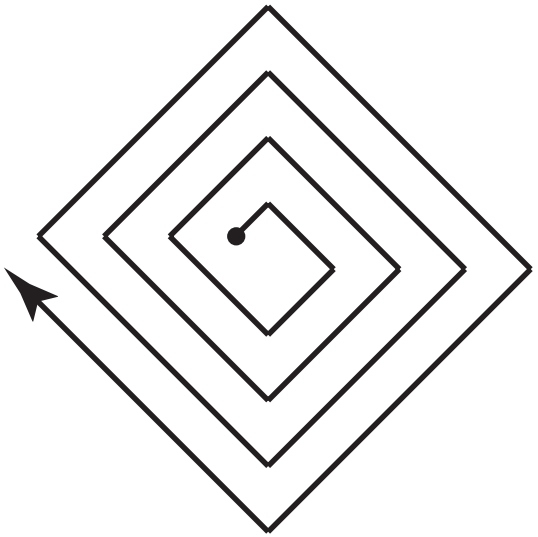
- 関数化

```
#include "./jissenPBL.h"

void forward_cm(double cm){
  double angle;
  angle=cm*20.5;
  RotateMotor(OUT_BC, 50, angle);
}

int main()
{
  OutputInit();

  forward_cm(50.0);
  Off(OUT_BC);
}
```

スパイラル軌跡に挑戦しよう

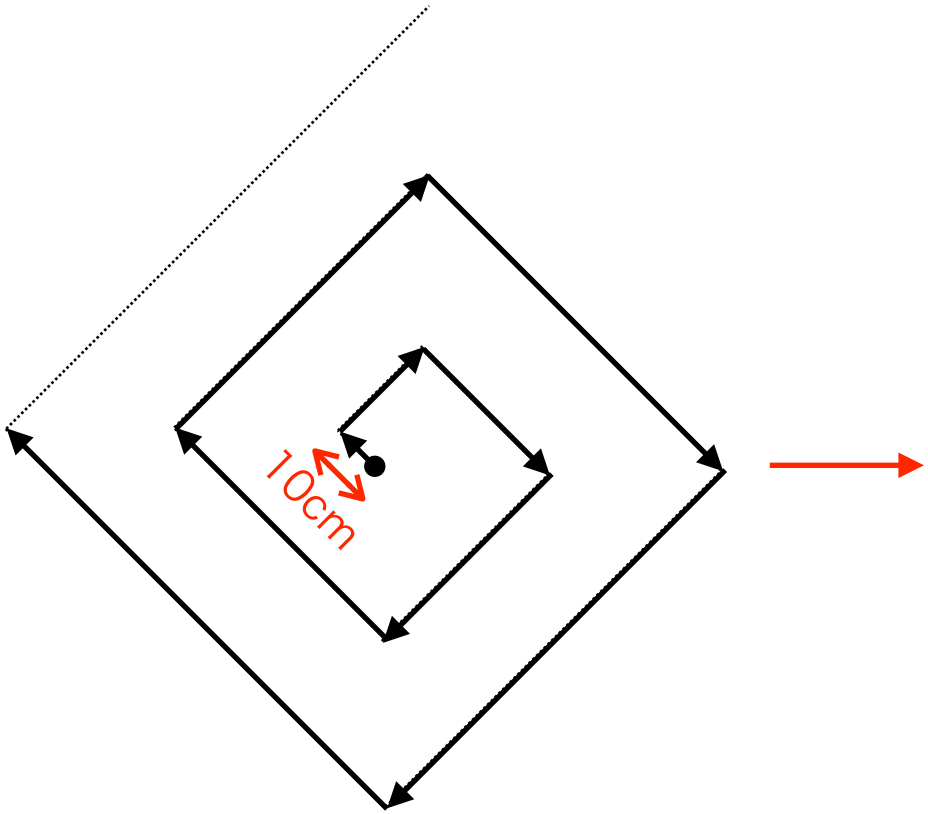
- 星形やスパイラルの軌跡を描くロボットの動きを実現してみよう！





実現したい動きの規則性

- 繰り返し回数ごとのロボットの動きを表にしてみよう



回数	前進	右回転
1	10cm	90度
2	20cm	90度
3	30cm	90度
4	40cm	90度
5	50cm	90度
6	60cm	90度
7	70cm	90度
8	80cm	90度
9	90cm	90度

実現したい動きの規則性

- 規則性を数式で表現

回数	前進	右旋回
1	10cm	90度
2	20cm	90度
3	30cm	90度
4	40cm	90度
5	50cm	90度
6	60cm	90度
7	70cm	90度
8	80cm	90度
9	90cm	90度

1

10

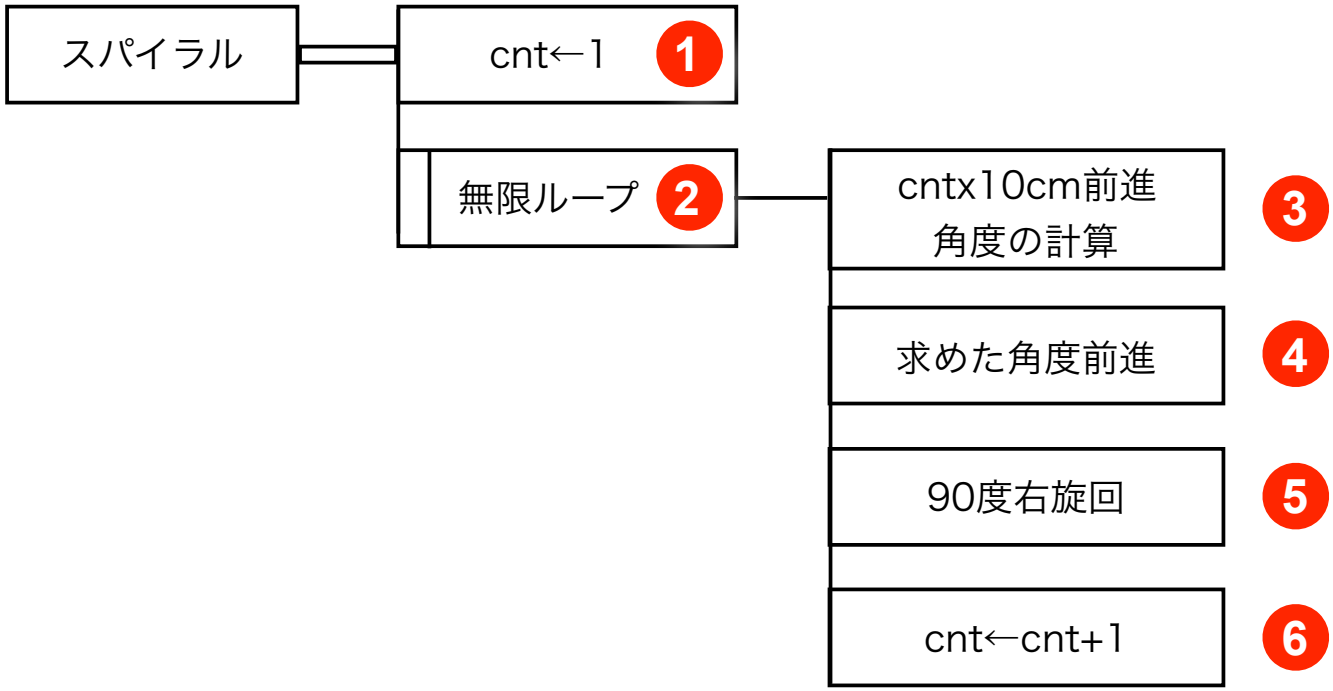
回数←1

前進する距離 = 10cm×回数

回数←回数+1

スパイラル状の軌跡のPAD

- 繰り返し回数cntと演算によりスパイラル軌跡を実現



スパイラル状の軌跡 (p.69 spiral.c)

```
#include "./jissenPBL.h"

#define TURN90 500

void forward_cm(double cm){
  double angle;
  angle=cm*20.5;                    3
  RotateMotor(OUT_BC, 50, angle);   4
}

void turn_right(int time){
  OnFwdEx(OUT_B, 50,0);
  OnRevEx(OUT_C, 50,0);
  Wait(time);
}
```

```
int main()
{
  int cnt=1 ;   1
  double angle;
  ButtonLedInit();
  OutputInit();                                  2
  while(true){
    forward_cm(cnt*10.0);   3  4
    turn_right(TURN90);     5
    cnt=cnt+1;              6
    //force-quit
    if(ButtonPressed(BTN1)) break;
  }
  Off(OUT_BC);
}
```

while文を用いたループ

- while(true){処理} とすると無限ループ

```
while (true)
{
        繰り返す処理

        //force-quit
        if(ButtonPressed(BTN1))    break;
}
Off(OUT_BC);
```

無限ループのプログラムを作成する時は：
プログラムの緊急停止（無限ループ脱出）のための処理を必ず入れておくこと
無限ループ脱出後の処理として、モータを停止する命令を追加すること