今回やること

- for式
- while文

for式

処理を繰り返し行いたいときに使います

基本的な使い方

- kotlin

```kotlin
for (i in 0 until 5) {
  println(i)
}
```

- python

```python
for i in range(5):
  print(i)
```

- C, Java

```c
for (int i = 0; i <= 5; i++) {
  printf("%d\n", i)
}
```

- 出力

```
0
1
2
3
4
```

inより前についている特殊な構文

ループするための範囲は以下の構文を使用します。stepはループするカウントを示します

- until
    - 〜から〜を含まない数
        - 例) `0 until 5` -> 0からはじまり4まで

- ..
  - 〜から〜まで
    - 例) `0..5` -> 0からはじまり5まで
- downTo
  - 〜から〜まで(後ろから下がっていく)
    - 例) `5 downTo 0` -> 5からはじまり0まで
- step
  - 範囲後に使用することでカウント数を変更することができる
    - 例) `0..10 step 2` -> 0からはじまり2ずつ進んで10で終わる
  - カウントが割り切れる範囲内でなければ、範囲が超える前に終了する
    - 例) `0..9 step 2` -> 0からはじまり2ずつ進んで8まで

もんだい

- for式を使用して`*`を横に10個表示してみよう

forの中にforをかける(多重ループ)

```
for (i in 1..5) {
  for (j in 1..5) {
    println(i * j)
  }
}
```

- 結果

```
1
2
3
4
5  // ここで一区切り
2
4
6
8
10 // ここで一区切り
3
6
9
12
15 // ここで一区切り
4
8
12
16
20 // ここで一区切り
5
10
15
20
25 // ここで終わり
```

ここがfor式のつまづきやすい部分です。理解できましたか？

ちょいむずもんだい(その1)

多重ループのコード例を応用して九九の表を作ってみよう。結果は以下のようなものが好ましいです。難しい場合はまず結果を 1 行ずつ表示してみよう

```
1  2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
3  6  9 12 15 18 21 24 27
4  8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
```

```
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

Hint

- 文字はそれぞれ3文字おきにprintされています。以下のとっておき構文で3文字おきのprintをできます
  - `print(String.format("%3d", ここに計算結果をいれる))`
- println()は中身を書かないと単に改行だけ行います
- main関数を除くと合計6行で書けます

ちょいむずもんだい(その2)

多重ループを使用して高さ10、幅10の三角形を作ってみよう。結果は以下のようなものが好ましいです。九九の表のプログラムを応用すると描きやすいです。

```
*
**
***
****
*****
******
*******
********
*********
**********
```

むずかしいもんだい

上記のもんだいを基に全ての方向に対応した三角形を作ってみよう。結果は全部で3つあります。
めちゃめちゃ難しいと思うので時間内に完成しなくても大丈夫です。

```
**********
*********
********
*******
******
*****
****
***
**
*
```

```
**********
 *********
```

```
 ********
  *******
   ******
    *****
     ****
      ***
       **
        *
```

```
        *
       **
      ***
     ****
    *****
   ******
  *******
 ********
*********
**********
```

while文

for式と同じく、繰り返し処理を行いたいときに使います。
for式と異なる点はループの終了条件にあります。

```
var x = 5
while(x > 0) {
  println(x)
  x--
}
```

- 結果

```
5
4
3
2
1
```

while文は丸括弧内の条件に対して、真である場合に波括弧内の処理を実行します。
条件内で使用された変数にのちの処理が外れるように処理をしてあげなければ永遠にループします。

```
var = 5
while(x > 0) {  // 条件が外れることがないので永遠にループする
  println(x)
  x++
}
```

また、while文の先頭にdoをつけるとdo括弧内の処理を行った後に条件を比較します

```
var x = 5
do {
  println(x)  // ここから実行されるので、もし最初の条件で満たさなくても必ず一回は実行
される
  x--
} while(x > 0)
```

もんだい

- while文を使用して*を横に10個表示してみよう

次回

- kotlinコレクション

- Array, List, Map, Set