マニュアルに戻る。

アニメーション

今まで学んだ繰り返し

コンピュータはある動作を繰り返し実行するのが得意である。「はじめてのプログラミング」では、**繰り返し**を使うことで、プログラムの一部を複数回実行できることを学んだ。繰り返されるプログラムは、一瞬で実行される。次のプログラムでは、10回繰り返している動作は見えず、画面には10回繰り返された後の結果が表示される。

```
かめた＝タートル！作る。
「かめた！20　歩く」！10　繰り返す。
```

「ペイントソフトを作ろう」では、**ボタン**オブジェクトを使うことで、特定の動作を手動で実行できることを学んだ。次のプログラムでは、画面のボタンを何度も押すことにより、繰り返し実行させることができる。

```
かめた＝タートル！作る。
前進ボタン＝ボタン！"前進"　作る。
前進ボタン：動作＝「かめた！20　歩く」。
```

タイマーオブジェクト

タイマーオブジェクトを使うと、一定間隔の繰り返しを行うことができる。タイマーは画面に表示されないオブジェクトで、他のオブジェクトに命令を繰り返して伝えるために使われる。

次のプログラムでは、「時計」という名前のタイマーオブジェクトを作り、実行したい内容をブロックで渡して実行している。このプログラムを実行すると、かめたは一定の間隔で少しずつ前進する。

```
かめた＝タートル！作る。
時計＝タイマー！作る。
時計！「かめた！20　歩く」実行。
```

作成と実行を1行で書くこともできる。この例では、タイマーに「時計」といった名前を付けずに実行している。

```
かめた＝タートル！作る。
タイマー！作る「かめた！20　歩く」実行。
```

タイマーには、繰り返す**間隔**と**回数**を指定できる。次のプログラムでは、「時計」という名前のタイマーに間隔と回数を指定している。標準では0.1秒間隔で100回の繰り返しを行う。[1)]

```
かめた＝タートル！作る。
時計＝タイマー！作る　1秒　間隔　5回　回数。
```

```
時計！「かめた！20　歩く」実行。
```

回数は実行時に指定することも可能である。

```
かめた＝タートル！作る。
時計＝タイマー！作る　1秒　間隔。
時計！「かめた！20　歩く」5回　実行。
```

回数の代りに時間を指定することもできる。次のプログラムでは、1秒間隔で5秒間の実行を行う。実行される回数は「5秒間÷1秒＝5回」である。

```
かめた＝タートル！作る。
時計＝タイマー！作る　1秒　間隔　5秒　時間。
時計！「かめた！20　歩く」実行。
```

次のプログラムは、ゆっくりと円を描くアニメーションの例である。

```
かめた＝タートル！作る。
時計＝タイマー！作る　0.1秒　間隔　36秒　時間。
時計！「かめた！1　歩く　1　右回り」実行。
```

次のプログラムは、図形が回転しながら動くアニメーションの例である。

```
かめた＝タートル！作る。
四角＝「かめた！30 歩く 90 左回り」！4 繰り返す（赤）図形を作る。
時計＝タイマー！作る。
時計！「四角！30　右回り　2　2　移動する」実行。
```

表に、タイマーの命令の一部をまとめておく。

タイマーの命令（一部）

命令	用途	使用例
間隔	実行間隔を指定する。単位は秒	時計！0.5　間隔。
回数	実行回数を指定する	時計！10　回数。
時間	実行時間を指定する。単位は秒	時計！5　時間。
実行	タイマーを実行する	時計！「かめた！10　歩く」実行。

複数のオブジェクトを動かす

実行するブロックに複数の文を書くことで、複数の動作を同時に行うことができる。次のプログラムは、「かめた」と「かめきち」という2つのタートルオブジェクトをタイマーの中で同時に動かしている。

```
かめた＝タートル！作る。
かめきち＝タートル！作る。
時計＝タイマー！作る　0.1秒　間隔　36秒　時間。
時計！「かめた！1　歩く 1 右回り。かめきち！1 歩く 1 左回り」実行。
```

同様に、違う種類のオブジェクトを同時に動かすことも可能である。次のプログラムでは、タートルオ

ブジェクト（かめた）と図形オブジェクト（四角）を同時に動かしている。

```
かめた＝タートル！作る。
四角＝「かめた！30 歩く 90 左回り」！4 繰り返す（赤）図形を作る。
時計＝タイマー！作る　0.1秒　間隔　36秒　時間。
時計！「かめた！1　歩く　1　右回り。四角！1　左回り」実行。
```

タイマーで複数の実行を行う

タイマーに複数の実行を送ると、それらは順次実行される。次のプログラムではタートルオブジェクト（かめた）を操作している。最初は右回りに動かし、続いて左回りに動かし、最後に前進させている。実行の前に数値を指定した場合には、その回数だけ実行される。

```
かめた＝タートル！作る。
時計＝タイマー！作る。
時計！「かめた！1　歩く　1　右回り」実行。
時計！「かめた！1　歩く　1　左回り」実行。
時計！「かめた！10　歩く」10　実行。
```

タイマーの終了を待ってから実行する

通常、プログラムは上から順に1行ずつ実行され、直前の行の実行が終るのを待って次の行が実行される。一方、タイマーで繰り返されるプログラムは、他のプログラムと並行して動作する。これは、タイマーは通常「数秒間から数分間」という長い時間動作を続けるため、その終了を待っていると、他の動作が行えなくなってしまうためである。このように、ひとつのプログラムの中で同時に複数の処理を並行して実行する仕組みを**スレッド**と呼ぶ。

しかし、ときにはタイマーの実行が終了するのを待ってから次に進みたいことがある。たとえば、ゲームの終了を待って得点を表示するときなどが考えられる。ここで、「タートルをその場で回転させてから、前進する」プログラムを考える。次のプログラムを実行すると、タイマーの実行が終る前にプログラムは次の行に進んでしまい、結果として回転する前にタートルが歩いてしまう。

```
かめた＝タートル！作る。
時計＝タイマー！作る。
時計！「かめた！10　右回り」実行。
かめた！100　歩く。
```

最後に実行を使うと、タイマーの実行が終了するのを待ってから実行する。次のプログラムを実行すると、タイマーの実行が終了するのを待ってから次の行に進むので、タートルは回転してから歩くという正しい動作を行う。

```
かめた＝タートル！作る。
時計＝タイマー！作る。
時計！「かめた！10　右回り」実行。
時計！「かめた！100　歩く」最後に実行。
```

1)

「0.1秒×100回＝10秒」**であることから、**実行**に要する時間は10秒間である。**