

## 24 関数 (メソッド)

### 【ねらい】

メソッドの使い方について理解します。

### 1 メソッドとは

プログラムが長くなる場合に、まとまった処理部分を**メソッド**にすると読みやすいコードになります。メソッドは数学で言う「関数」のような働きをします。また、他のプログラミング言語の関数、プロシージャ、サブルーチンなどに相当します。C# ではクラスの一部として定義されるため、メソッドという呼び方が使用されます。

次の例は入力された整数の値を 2 乗した数値を出力するプログラムです。

<pre>class Program {     static void Main()     {         Console.WriteLine("x の値を入力してください。");         int x = int.Parse(Console.ReadLine());         int y = SquareNumber(x);         Console.WriteLine("x={0}、x^2={1}", x, y);          Console.ReadLine();     }      static int SquareNumber(int number)     {         int val = number * number;         return val;     } }</pre>	<p>Program クラスの定義。</p> <p>Main メソッドの定義。</p> <p>「x の値を入力してください。」を出力。 標準入力を数値に変換して x へ代入。 SquareNumber メソッドの呼び出し。 x と y の値を標準出力に出力。</p> <p>Enter キーの入力を待つ。</p> <p>SquareNumber メソッドの定義。</p> <p>number を 2 乗した値を val に代入。 val の値を戻り値として戻る。</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### サンプル プログラム

サンプル プログラムの「Application1」プロジェクトにこのサンプルがあります。

計算自体は SquareNumber メソッドで行います。その SquareNumber メソッドを Main メソッドの中から呼び出して使用しています。プログラムの中で 2 乗の計算が必要になるたびに SquareNumber を呼び出して、何度でも同じメソッドを使用することができます。

## 2 メソッドの定義

メソッドは次のような構文で定義します。

---

```
修飾子 戻り値の型 メソッド名 (引数の型 パラメータ, 引数の型 パラメータ)
{
    // 処理内容
    return 戻り値
}
```

---

以下、順に説明します。

### ・修飾子

修飾子の詳細については別途説明します。最初の例で指定している `static` は、このメソッドが静的なメソッドであることを示しています。静的なメソッドについてはクラスの所で説明しますが、今の段階では必ず `static` を付けてメソッドを定義するようにしてください。

### ・戻り値の型

メソッドは、呼び出し元に値を 1 つ返すことができます。これをメソッドの**戻り値**といいます。メソッド名の前にこのメソッドの戻り値の型を記述するようにします。戻り値の型には `int` 型や `string` 型など、任意の型を指定できます。また、メソッドが戻り値を返さない場合には、キーワードの `void` を代わりに指定するようにします。

### ・メソッド名

このメソッドの名前です。他と重複しない、わかりやすい名前を付けるようにします。

### ・引数の型

メソッドには、呼び出す側からメソッドへの入力値を渡すことができます。これを**引数**と言います。引数は複数指定することができ、カンマで区切って複数の引数を記述します。引数を 1 つも持たない場合にはかっこの中には何も書かないようにします。

メソッド定義の引数の型には、引数ごとの型を指定します。任意の型を指定できます。

### ・パラメータ

仮引数とも呼ばれます。メソッドに引数として渡された値を保持するための変数名を記述します。このパラメータを使用してメソッドの中で演算を行うことができます。

### ・戻り値

メソッドの最後にキーワード `return` に続けて、このメソッドが返す値 (戻り値) を指定します。メソッドの戻り値の型として指定されている型と一致している必要があります。if ステートメントなどの分岐命令を使用して、コードの途中で `return` ステートメントを書くこともできます。プログラムがメソッドの処理の最中に `return` ステートメントに到達した場合には、メソッドの以後のコードは実行せずに、呼び出し元に戻ります。メソッドが `void` として定義されている場合には `return` ステートメントは不要ですが、分岐の途中などで戻り値を伴わずに `return` キーワードのみを指定することもできます。

次のコードは 2 つの引数を受け取り、return ステートメントが 2 つあるメソッドの例です。

<pre>// 大きい方の値を返す static int MaxInt(int x, int y) {     if (x &gt; y)         return x;     else         return y; }</pre>	MaxInt メソッドの定義。  x が y より大きいか。 x の値を戻り値として戻る。  y の値を戻り値として戻る。
----------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------

### 3 メソッドの呼び出し

メソッドを利用するにはメソッド名と引数を指定してメソッドを呼び出します。

<pre>int y = SquareNumber(x);</pre>	SquareNumber メソッドの呼び出し。
-------------------------------------	-------------------------

引数の数や型は、メソッド定義に書かれているパラメータの数や型と一致する必要があります。呼び出す側で引数を変数で指定する場合、この変数はメソッドのパラメータで指定した変数名と同じ名前である必要はありません。

メソッドの戻り値を演算や代入に使う場合は、メソッドの呼び出し自体を変数名と同じように使用して、演算や代入に使用することができます。

### 4 値渡しと参照渡し

メソッド内でパラメータの値を変更した場合に、呼び出し元で引数として指定した変数の値が影響を受けるかどうかは、変数が値型か参照型かに依存します。int 型などの値型の変数を引数にしてメソッドを呼び出した場合には、値がコピーされてメソッドが呼び出されるため、呼び出した先のメソッドでパラメータの値を変更しても呼び出し元の変数の値は変わりません。

<pre>static void Main() {     int n = 1;     ChangeInt(n); }</pre>	Main メソッドの定義。  変数 n に 1 を代入する。 変数 n を引数として ChangeInt を呼ぶ。
<pre>static void ChangeInt(int x) {     x = 5;     return; }</pre>	ChangeInt メソッドの定義。  パラメータ x に 5 を代入する。 呼び出し元に戻る。

## サンプル プログラム

サンプル プログラムの「Application2」プロジェクトにこのサンプルがあります。

このコードでは、変数 `n` の値を受け取ったメソッドのパラメータ `x` に、メソッド内で `5` を代入しています。しかしこのメソッドから戻った後でも変数 `n` の値は `1` のままで変わりません。パラメータ `x` には、値である `5` がコピーされて渡されているだけだからです。

これに対して、参照型のパラメータの値をメソッド内で変更した場合には、呼び出し元の変数の値も変更されます。参照型の場合には、参照がパラメータにコピーされるため、呼び出した先のメソッドでパラメータの値を変更すると、その参照先の値が書き変わるためです。

<pre>static void Main() {     int[] n = { 1 };     ChangeInt(n); }</pre>	Main メソッドの定義。  配列を、値 <code>1</code> で作成して <code>n</code> に代入。 変数 <code>n</code> を引数として <code>ChangeInt</code> を呼ぶ。
<pre>static void ChangeInt(int[] x) {     x[0] = 5;     return; }</pre>	ChangeInt メソッドの定義。  パラメータ <code>x</code> の要素 <code>0</code> に <code>5</code> を代入する。 呼び出し元に戻る。

## サンプル プログラム

サンプル プログラムの「Application3」プロジェクトにこのサンプルがあります。

このコードは、先ほどのコードの変数 `n` を、`int` 型から `int` 型の配列に変更したものです。配列は参照型のため、メソッド呼び出しの際にパラメータ `x` にコピーして渡されるのは、実際の配列の値ではなく、配列への参照になります。したがってメソッド内でこの参照先の要素の値を `5` に変更した場合には、結果として元の `n` の配列の要素の値も `5` に変更されることになります。

引数に `ref` キーワードを付けてメソッドを定義すると、値型の変数であっても参照渡しにすることができます。

```
static void Main()
{
    int n = 1;
    ChangeInt(ref n);
}
```

Main メソッドの定義。

変数 `n` に 1 を代入する。  
変数 `n` を引数として `ChangeInt` を呼ぶ。

```
static void ChangeInt(ref int x)
{
    x = 5;
    return;
}
```

ChangeInt メソッドの定義。

パラメータ `x` に 5 を代入する。  
呼び出し元に戻る。

### サンプル プログラム

サンプル プログラムの「Application4」プロジェクトにこのサンプルがあります。

この場合には最初の例とは異なり、`n` の値は 5 に更新されます。このような `ref` を付けたパラメータは、メソッドからの戻り値として、`return` ステートメントで戻す本来の 1 つの値だけではなく、複数の値をメソッド内で変更して呼び出し元に戻したい時などによく使用されます。

### 演習課題

- ・ 円の半径を `double` 型の引数として受け取り、その円の面積を `double` 型で返すメソッドを作ります。また、このメソッドを使用して、半径を標準入力から受け取り、円の面積を出力するプログラムを作ります。

### 解答例

サンプル プログラムに演習の解答例があります。