



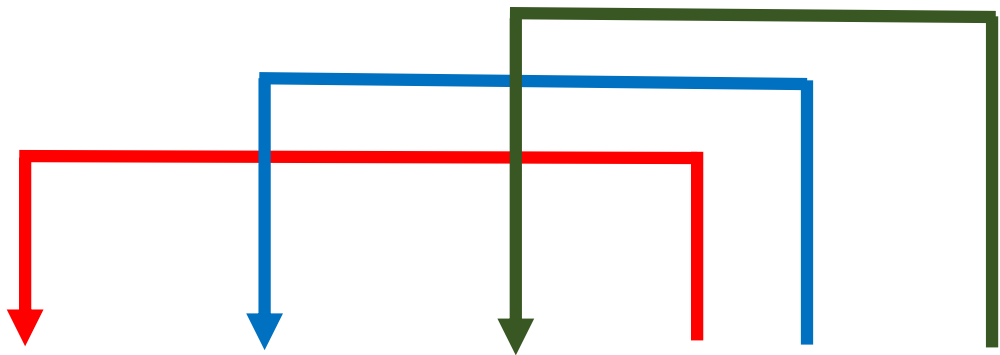
# 演算と変数（前半）

基本編2日目

# C#言語の演算子

| 演算子 | 読み方    | 意味                 | 使用例   |
|-----|--------|--------------------|-------|
| +   | プラス    | 足し算を行う演算子          | 5 + 5 |
| -   | マイナス   | 引き算を行う演算子          | 7 - 3 |
| *   | アスタリスク | 掛け算を行う演算子          | 7 * 3 |
| /   | スラッシュ  | 割り算を行う演算子          | 7 / 3 |
| %   | パーセント  | 剰余（じょうよ）演算子。割り算の余り | 7 % 3 |

# パラメータの表示



```
Console.WriteLine("{0} + {1} = {2}", 5, 2, 5 + 2);
```

「5 + 2」の計算結果の「7」がここに入る

{ }の中の番号が後で、で区切られたパラメータを表示するための番号を示す。

# 定数と変数

- Sample202.cs参照
- 値が定まった数のことを**定数**と言う
  - 2
  - 1.07
  - “Hello”
  - 5 + 2
  - など
- 値を常に変えることができる数が存在しこれを**変数**という

# 変数の宣言と代入

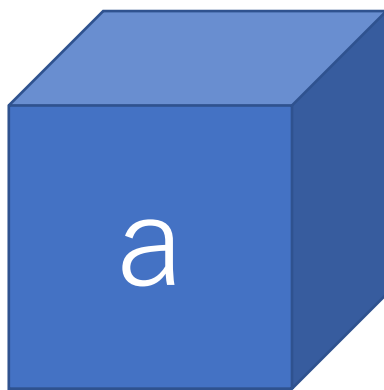
- 変数を使うには**宣言（せんげん）**が必要
- 変数に値を入れることを**代入（だいにゅう）**と言う
- 変数に最初の値を入れることを初期化（しょきか）と言う
- **a=6**とすれば、aは別の値が代入されるまで6
- 変数は原則的に何度も値を変えることが可能

## 変数の宣言と代入

```
int a;    ← 変数の 宣言(aという変数を用意)  
a = 6;    ← 代入（変数に値を入れる）
```

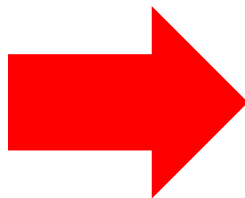
# 変数の宣言と代入のイメージ

**int a;**



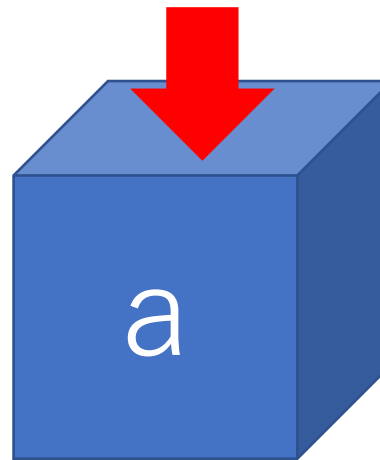
**宣言**

intの値が入るという  
器（変数）を用意する



**a=6;**

**6**



**代入**

用意される器に値  
(6) を入れる

# 演算の優先順位

- 括弧を用いると計算の優先順位を変更することが出来る

括弧が無い場合の計算

$$1 + 2 * 3 \rightarrow \text{結果は7}$$

括弧を用いた計算

$$(1 + 2) * 3 \rightarrow \text{結果は9}$$

- 基本的な数値計算に関しては、数学と同じルールに従う

# ()を使用した場合と、使用しない場合の演算の処理

$$1 + 2 * 3$$

①  $2 * 3$  の計算  
②  $1 + 6$  の計算

- ①  $2 * 3$  の計算 (=6)
- ②  $1 + 6$  の計算 (=7)

$$(1 + 2) * 3$$

①  $1 + 2$  の計算  
②  $3 * 3$  の計算

- ①  $1 + 2$  の計算 (=3)
- ②  $3 * 3$  の計算 (=9)



# データ型

| データ型           | 説明                               | ビット | 範囲   |
|----------------|----------------------------------|-----|--|
| <b>byte</b>    | 符号なし整数                           | 8   | 0 ~ 255  |
| <b>sbyte</b>   | 符号付き整数                           | 8   | -128 ~ 127   |
| <b>int</b>     | 符号付き整数                           | 32  | -2,147,483,648 ~ 2,147,483,647                       |
| <b>uint</b>    | 符号なし整数                           | 32  | 0 ~ 4294967295                                       |
| <b>short</b>   | 符号付き整数                           | 16  | -32,768 ~ 32,767                                     |
| <b>ushort</b>  | 符号なし整数                           | 16  | 0 ~ 65535  |
| <b>long</b>    | 符号付き整数                           | 64  | -922337203685477508 ~ 922337203685477507             |
| <b>ulong</b>   | 符号なし整数                           | 64  | 0 ~ 18446744073709551615                             |
| <b>float</b>   | 単精度浮動小数点型                        | 32  | -3.402823e38 ~ 3.402823e38                           |
| <b>double</b>  | 倍精度浮動小数点型                        | 64  | -1.79769313486232e308 ~ 1.79769313486232e308         |
| <b>char</b>    | 単一 Unicode 文字                    | 16  | テキストで使われる Unicode 記号                                 |
| <b>bool</b>    | 論理ブール型                           | 8   | true または false                                       |
| <b>object</b>  | 他のすべての型の基本型                      |     |  |
| <b>string</b>  | 文字列                              |     |  |
| <b>decimal</b> | 29 の有効桁数で 10 進数を表現できる正確な小数または整数型 | 128 | $\pm 1.0 \times 10^{28} \sim \pm 7.9 \times 10^{28}$ |

# 初期化

- 変数は、宣言時に値を代入（**初期化**）することが出来る

変数の宣言と初期化を同時に行う

```
int a = 6; ← 変数の宣言と同時に値を代入（初期化）
```

複数の変数を同時に宣言

```
int a,b;    ← 変数a,bを宣言  
int a=1,b=2; ← 変数a,bを初期化  
int a,b=1; ← 変数a,bを宣言、bのみを初期化
```