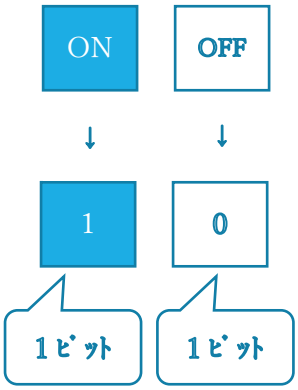
情報の表現

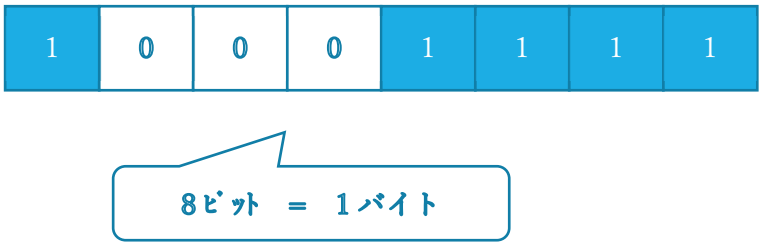
SOFT STUDIO
ソフトスタジオ

情報量の単位

コンピュータの内部では、すべての情報は電気信号の「ON」「OFF」のように 2 つの値で扱われているので、これを一般的に 2 進数の「 1 」と「 0 」に対応させて表現しています。

コンピュータで扱う最小の情報の単位を**ビット（bit）**といい、ビット 8 個で**バイト（Byte）**といいます。





接頭語		備　考
k （キロ）	1000	1 KB(キロバイト)＝１０００バイト
M （メガ）	K×1000	1 MB(メガバイト)＝ 1 KB×１０００　バイト
G （ギガ）	M×1000	1 GB(ギガバイト)＝ 1 MB×１０００　バイト
T （テラ）	G×1000	1 TB(テラバイト)＝ 1 GB×１０００　バイト

※携帯電話の速度、容量、パソコンのメモリ容量、ディスク容量などを表すときなどに使用されている

2 進数(数値の表現)

10 進数は、0～ 9 までの 10 種類の数字を使って 9 の次が 1 つ桁上がりします。

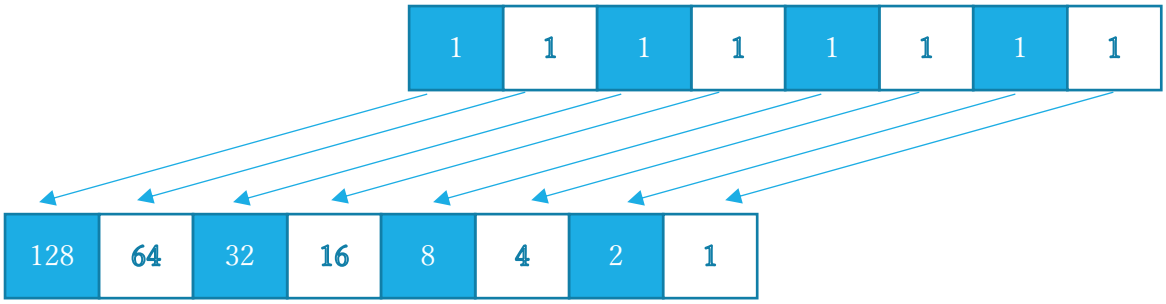
2 進数は 0 と 1 の 2 種類の数字を使って、1 の次が 1 つ桁上がりします。

2 進数は得た数が非常に長くなるため、人間が考えるときは、2 進数と簡単に変換できる 16 進数がよく使われます。

10 進数	2 進数	16 進数
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8

10 進数	2 進数	16 進数
9	1001	9
10	1010	**A**
11	1011	**B**
12	1100	**C**
13	1101	**D**
14	1110	**E**
15	1111	**F**
16	10000	10

※4 ビットで 0～15 の数値を表現できる。　8 ビット（1 バイト）では 0～255 の数値を表現できる。



※　1　+　2　+　4　+　8　+　16　+　32　+　64　+　128 = 255

👉計算機で計算してみてください

文字の表現

コンピュータの内部は 0 と 1 の 2 進数で表現されています。にもかかわらずコンピュータが文字を扱うことができるのは、文字の一つひとつに、特定の 2 進数が割り当てられていて、文字という情報をコード化しているからです。

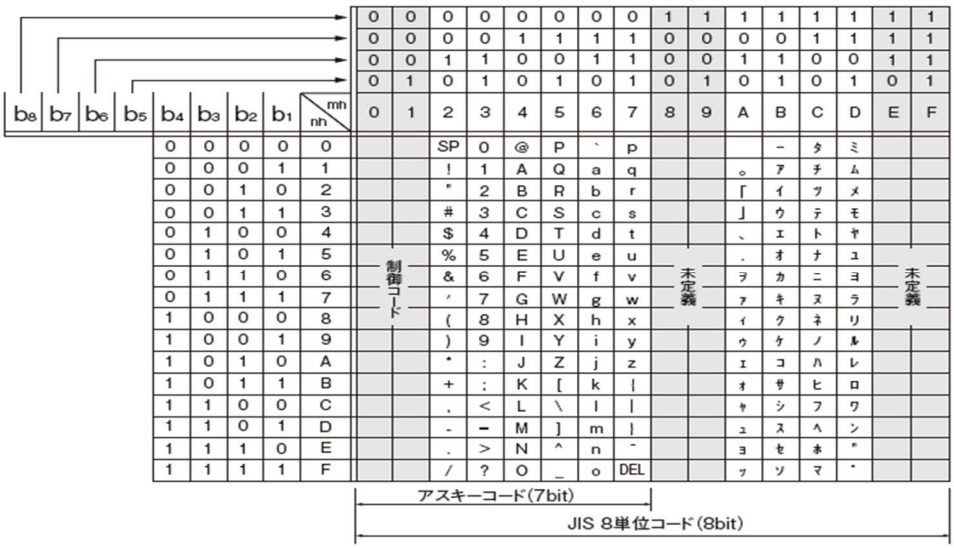


図　ASCII（アスキー）コード表

文字“A”のコードは　0 x 4 1　、文字”６“のコードは 0 x ３６、カタカナの”ｷ“は 0 x B 7

※「0 x 」は 16 進数であることを表現しています。

表　コード表の種類

コード名	概　要
ASCII コード	英数字と特殊文字のみ、漢字・かなに関する規定はない
シフト JIS コード	ASCII コードと漢字・かなが混在可能で、Windows や MacOS などに用いられている。
EUC	UNIX や Linux などで用いられる文字コード、漢字・かなが使える
UNICODE	政界の文字の多くを一つの体系で表現するコード、UTF-8 は、UNICODE の符号化方式（文字の振り方）の一つ

C#の変数

SOFT STUDIO
ソフトスタジオ

変数

変数は、プログラムの実行中にデータ（値）を一時的に保管するための入れ物です。
変数を使うには、予め変数の名前とデータ型（データの種類）を宣言しておきます。

変数の宣言

変数の宣言は下のような形で記述します。

```
データ型　変数名;
string　　test ;
```

変数の宣言と同時に、値を代入するには、次のように記述します。

```
データ型　変数名　=　　値;
 string　 test　　=　“テスト”;
 int　 i　　= 0 ;
```

Point

変数名で使用できる文字列は、アルファベット、「_」（アンダースコア）、数字、ひらがな、カタカナ、漢字ですが、**予約語（キーワード）**と同じ名前は使えません。また、変数名の先頭に数字を使うこともできません。

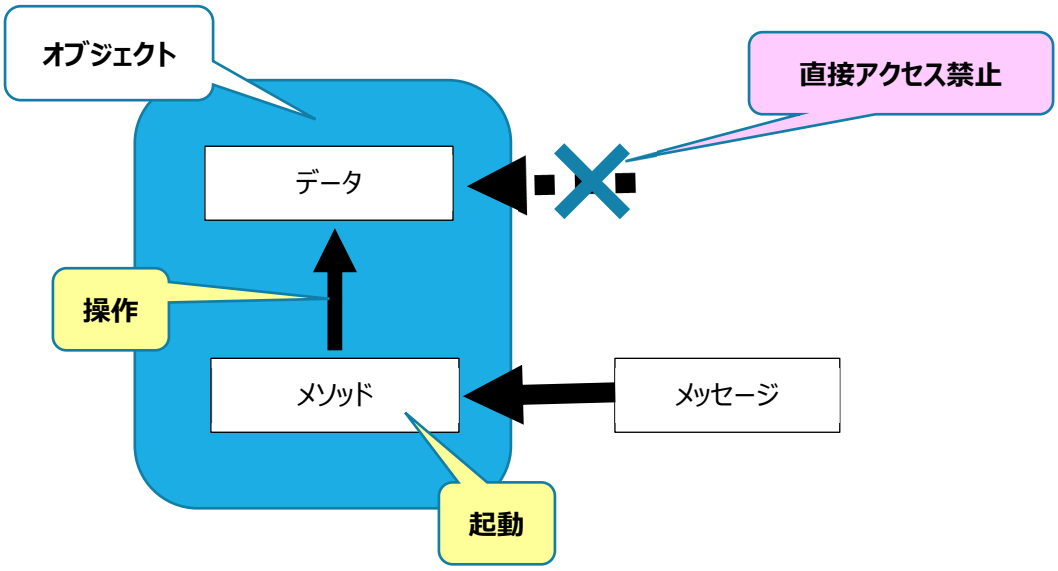
データ型

プログラムでは、文字や数値など様々な種類のデータを扱います。このデータの種類をデータ型（または型）といいます。
例えば、プログラムでは、**文字データは string 型、数値データは int 型と**して扱います。それぞれのデータ型は、サイズと扱う値の範囲が決まっています。

型	用途とサイズ	値の範囲	.NET Framework 型	規定値
sbyte	符号付 8 ビット整数	-128~127	System.Sbyte	0
byte	符号なし 8 ビット整数	0~255	System.Byte	0
short	符号付 16 ビット整数	-32768~32767	System.Int16	0
ushort	符号なし 16 ビット整数	0~65535	System.Uint16	0
int	符号付 32 ビット整数	-2147483648~2147483647	System.Int32	0
uint	符号なし 32 ビット整数	0~4294967295	System.Uint32	0
long	符号付 64 ビット整数	-9223372036854775808~	System.Int64	0L
ulong	符号なし 64 ビット整数	0~1844674407370955615	System.Uint64	0
float	32 ビット浮動小数点数	±1.5e-45~±3.4e38	System.Single	0.0F
double	64 ビット浮動小数点数	±5.0e324~±1.7e308	System.Double	0.00
decimal	128 ビット 10 進数	±1.0×10-28to±7.9×1028	System.Decimal	0.0M
char	Unicode16 ビット文字	U+0000~U+ffff	System.Char	‘¥0’
string	Unicode 文字（可変）	約 20 億文字まで	System.String	
bool	真偽	true(真)または false(偽)	System.Boolean	false
object	オブジェクト参照	任意のデータ型	System.Object	

オブジェクト

データ（属性）とそれを操作するメソッド（手順）を一体化したものをオブジェクトといいます。



クラスとインスタンス

- **クラス**

  データとメソッドを持ったオブジェクトのひな形を定義したもの

- **インスタンス**

  クラスをもとに生成されたオブジェクト

ローカル変数

メソッド内でのみ使うことのできる変数で、メソッド内で宣言し、宣言されたメソッドを実行中のみ有効となります。メソッドの実行を終えると、破棄されます。

メンバー変数（フィールド）

メンバー変数はクラス内全体で使う変数で、クラス内（メソッドの外）出宣言し、宣言されたクラス内のすべてのメソッドで利用できます

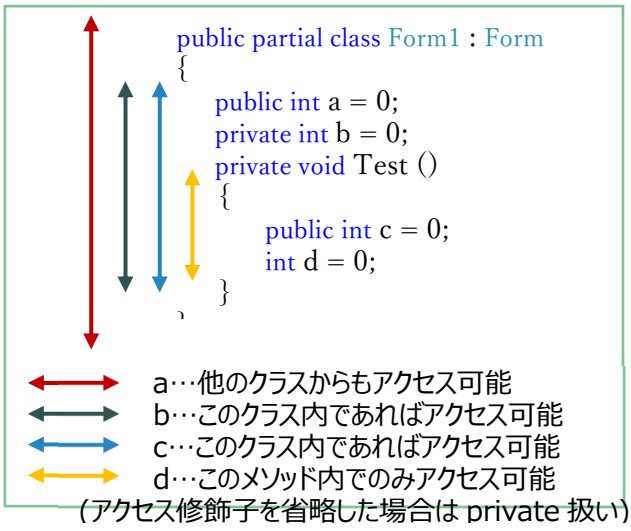
メンバー変数を宣言するときは、型名の前にアクセス修飾子を記述できます。

アクセス修飾子は、そのメンバー変数がどこから使用できるかというスコープ（有効範囲）を示します。

```
アクセス修飾子  データ型  変数名 1, 変数名 2, … ;
    private    string    test,      test2,    … ;
```

▼アクセス修飾子の種類とスコープ

種類	スコープ
public	外部のクラスからの参照も可能 もっとも広いスコープを持つ
protected	クラス内と派生クラスからのみ参照可能
internal	同一アセンブリ内からのみ参照可能
private	同一クラス内からのみ参照可能 最も狭いスコープを持つ



データ型を変換する

SOFT STUDIO ソフトスタジオ

データ型のキャスト

変数にデータ型が異なる値を代入する場合など、データ型を明示的に変換するには()演算子、あるいは as 演算子を使って記述します。例えば、long 型の変数 x の値を int 型に変換するには「(int)x」と記述します。このように、データ型を変換することを**キャスト**といいます。

なお、double 型や float 型から int 型へのキャストのように、変換元の値が変換先の値の範囲を超えるような場合は、超えた部分が失われてしまうことがあるので注意が必要です。

また、int 型と string 型のように互換性がない場合は、()演算子では変換できません。このような場合は **Parse メソッド**、**TryParse メソッド**、**ToString メソッド**を使います。

▼Parse メソッド

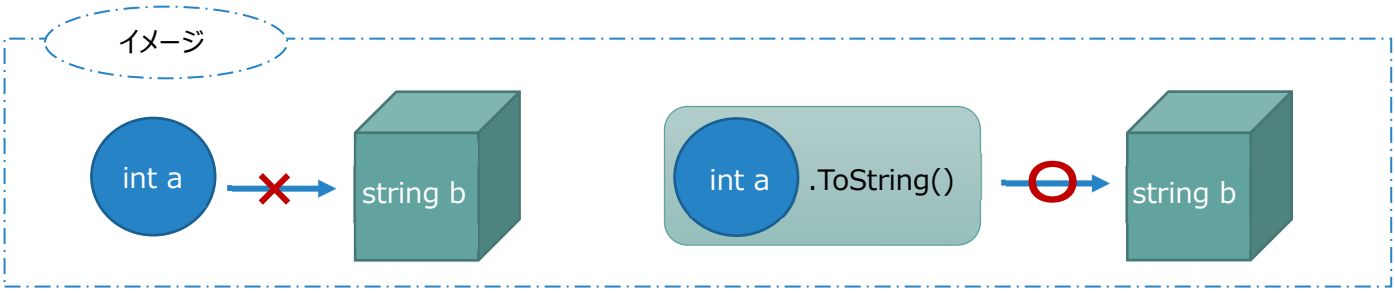
文字列の数字を int 型などの数値として扱ったり、計算したりするには、数字を数値に変換してから行います。数字を数値に変換するには、Parse メソッドを使います。

```
変換後の型.Parse（変換したい値）
```

▼ToString メソッド

数値などを文字列に変換するには、ToString メソッドを使います。引数に書式指定子を使って表示形式を指定することもできます。

```
値.ToString（書式指定子）
```



▼TryParse メソッド

TryParce メソッドは変換できる場合には True、変換不可の場合は False を返します。

下の例では、String 型の変数 wStr を Int 型に変換できる場合は変換したものを変数 wInt に代入し変数 Total に加算しています。例 1 では、文字列“10”を数値に変換することができるので、Total に 10 が加算され、Total は最終的に 110 になります。また、例 2 では文字列“ABC”は数値に変換することができないので、Total には何も加算されません。（この場合、wInt は 0 のままになります。）

int 型以外にも使用できるので、変換可能かあいまいな場合は TryParce メソッドを使用した方が安心です。

【例 1】

```
string wSrt = ”10”
int wInt = 0 ;
int Total = 100 ;

If ( int.TryParce( wSrt , out wInt ))
{
      Total += wInt;
}
```

【例 2】

```
string wSrt = ”ABC”
int wInt = 0 ;
int Total = 100 ;

If ( int.TryParce( wSrt , out wInt ))
{
      Total += wInt;
}
```