2.6. 補数

数値を基準となる数値にするために加える数値のことを「補数」と呼びます。
組込み C 言語に必要な知識として、以下の補数について説明をします。

- 1 の補数
- 2 の補数

1 の補数

2 進数の数値に対して、足しても桁数が上がらない最大の数値を 1 の補数といいます。
1 の補数は、ビットを反転させることで求めることができます。

（例 1） 110 の 1 の補数 = 001

数値 110 の桁数は 3 桁とします。
110 の 1 の補数は以下のように求めます。

```
110
↓ 反転
001
```

※　001 は 110 に足しても桁数が上がらない最大の数値です。

```
 110
+001
 111
```

2 の補数

2 進数の数値に対して、足して全体の桁数が上がる最小の数値を 2 の補数といいます。
2 の補数は、1 の補数に 1 を足すことで求めることができます。

（例 2） 101 の 2 の補数 = 011

数値 101 の桁数は 3 桁とします。
101 の 2 の補数は以下のように求めます。

```
101
↓ 反転
010 (1 の補数)
↓ + 1
011
```

※　011 は 101 に足して全体の桁数が上がる最小の数値です。

```
 101
+011
 1000
```

1 の補数と 2 の補数の具体例

1 の補数と 2 の補数の具体的な例を以下に示します。

	1 の補数	2 の補数
00000000	11111111	00000000
01100001	10011110	10011111
11001100	00110011	00110100
11010010	00101101	00101110

※　数値の桁数は 8 桁とします。

符号ビット

2 進数の数値の正負を判断するためのビットのことを「符号ビット」と呼びます。
一般的なコンピュータには、最上位ビットを符号ビットとして扱える仕組みがあります。
符号ビットが 0 なら正の数、1 なら負の数として扱います。

10 進数 97 の 2 進表記

```
01100001
```

10 進数 -97 の 2 進表記

絶対値が同じで符号を反転した値(-1 をかけた値)は、2 の補数表現を使って算出します。

```
01100001 (10進数 97)
  ↓ 1 の補数
10011110 (10進数 -98)
  ↓ 2 の補数
10011111 (10進数 -97)
```

補数の特徴

基数の補数を負の数の表記法として採用すると、最上位桁からの桁上がり（オーバフロー）を無視すれば、通常の加算処理で負の数の加算（つまり正の数の減算）が行えることになります。
※　例えば、97 - 97 = 0 という減算処理を 97 + (-97) = 0 という加算処理で行うことを意味します。

この利点のため、2 の補数は多くのコンピュータで負の数の内部表現に採用されています。

（例 3） 97 + (-97) = 0

```
01100001 の 2 の補数 10011111
  01100001
+ 10011111
 100000000 (オーバフロー)
```

組込みソフトウェア開発では、メモリ領域を 8 進数、16 進数で確認することがあるため、負の数を読めるようにしておくと良いです。