

## 9. C 言語特有の書き方

### (1) ちょっと便利な代入文

システム記述用言語というからには、その言語で書かれたプログラムの高速性が要求されます。このため、C 言語では他の言語とは変わった風変わりな表記法があります。たとえば、

```
A = B = C = 10;
```

のように記述することができます。この代入文では、右から評価されて、

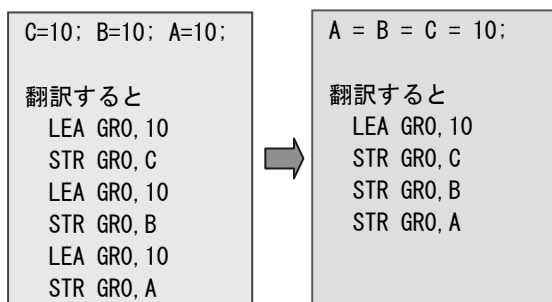
- まず変数 C に 10 が入り、
- その結果(10)が B に入り、
- さらにその結果(10)が A に入る。

と処理されます。代入文の結果は、代入された値そのものですから、

```
C = 10 ; B = 10; A = 10;
```

と同等になります。なぜ、こんな代入文が用意されているかというと、効率のよいオブジェクトを生成できるようにするためです。

仮想的なアセンブラで示すと、最適化処理をしなくても、次のように効率のオブジェクトを生成することができます。



なお、このアセンブラコードを理解する必要はありません。翻訳結果のコードが少なくなることを理解してください。

この他、以下のような便利な代入文もあります

No.	記述	意味
1	A++;	A = A + 1;
2	B = A++;	B = A; A = A + 1;
3	++A;	A = A + 1;
4	B = ++A;	A = A + 1; B = A;
5	A--;	A = A - 1;
6	B = A--;	B = A; A = A - 1;
7	--A;	A = A - 1;
8	B = --A;	A = A - 1; B = A;
9	A <OP>=B;	A = A <OP> B;

(9 の例)

```
A += B;    A = A + B;
A -= B;    A = A - B;
A *= B;    A = A * B;
A /= B;    A = A / B;
A %= B;    A = A % B;
```

[C 特有の代入文を使って書き直す]

```
#include "stdafx.h"
int _tmain(int argc, _TCHAR* argv[])
{ int A[10], i, total; i=0;
  /* 最初にデータを読み込んでおく */
  while(i<10)
  { printf("i=%d :", i);
    scanf("%d", &A[i]);
    i++;
  }
  total=0; i=0;
  while(i<10) /* 配列のデータを加算 */
  { total += A[i];
    printf("A = %d ( %d ) %n",
           A[i], total);
    i++;
  }
}
```

### (2) while 以外の繰返し

while 以外に以下のような繰返しがあります。

■後判定型繰返し(do …while)

```
文 1 ; while(式) 文 1 ;
```

は、次のように書くことができます。