

関数とサブルーチンの違いは、結果を呼び出し元に返すかどうかだけ。

```
関数      返す
サブルーチン 返さない
```

関数

```
J=FUNC(2) /* 関数っぽく呼び出せる
SAY J     /* 4
EXIT

FUNC: PROCEDURE
ARG N
RETURN N*N
```

変数のスコープ

関数・サブルーチンの中では、変数スコープは完全に独立している。
(逆に言えば、呼び出し元と同じ変数を使用しても影響を与えない)

引数は、値渡し、参照渡しいずれも可能。

値渡し

- 基本は値渡し
- 配列を渡すことはできない

```
I=10
CALL SUB I
SAY I /* 10
EXIT

SUB: PROCEDURE
ARG I
SAY I /* 10
I=50 /* Iを変更
SAY I /* 50
RETURN
```

参照渡し

- 配列を渡すことも可能
- 参照渡しをする変数名が固定されてしまうので、サブルーチンの可搬性が低下し、美しくない

```
I=10
ARRAY.=0
CALL SUB
SAY I /* 50 10 ではない
SAY ARRAY.2 /* 100 0 ではない
EXIT

SUB: PROCEDURE EXPOSE I ARRAY. /* I と ARRAY. を参照渡しとする (名前はここで決まってしまう)
SAY I /* 10
I=50 /* Iを変更
SAY I /* 50

ARRAY.2=100
RETURN
```

参照渡しをする変数名を、実行時に指定するようにすることも可能ではある。
この例では、参照渡しする変数の名前を REFERENCE という変数が示す。
でも REFERENCE という名前は、サブルーチンで固定されてしまうので、結局可搬性はあんまりよくない。

```
I=100
REFERENCE='I'
CALL SUB
SAY I /* 500

K=200
REFERENCE='K'
```

```
CALL SUB  
SAY K /* 500  
EXIT
```

```
SUB: PROCEDURE (REFERENCE) /* 参照渡しする変数名を示す変数は REFERENCE という名前にすると決める  
REF=WORD(REFERENCE,1) /* REFERENCE の最初のワードが、参照渡しされた変数名  
SAY VALUE(REF) /* 最初の呼び出しでは 100, 2番目では 200  
VALUE(REF,500)  
RETURN
```