Basicの基礎・サブプログラムと関数

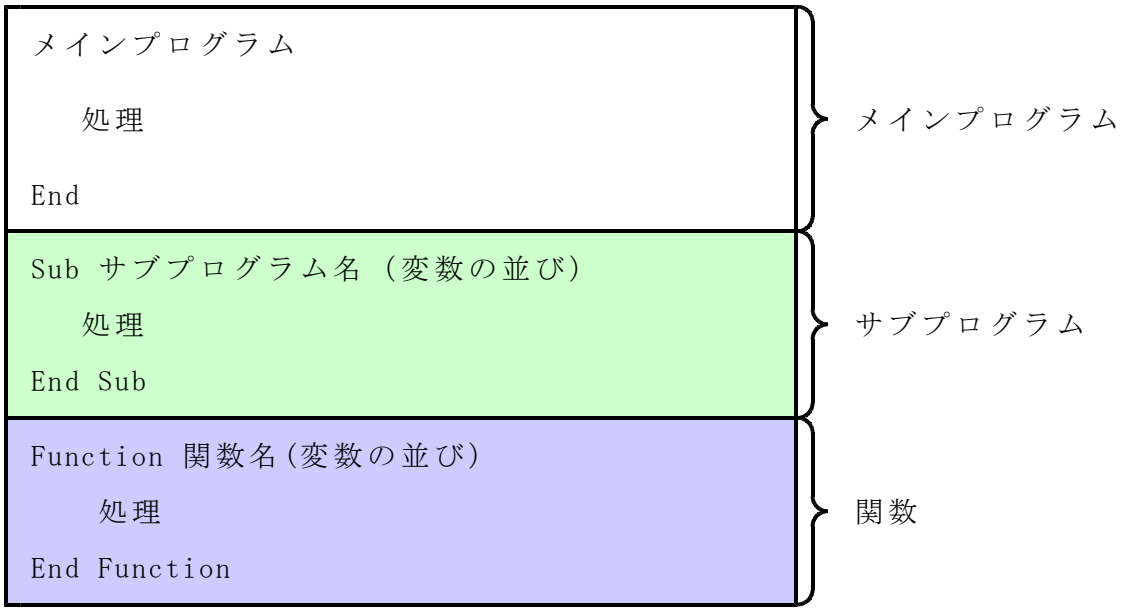
０．目次

７．サブプログラムと関数

７．１　サブプログラム・関数とは

プログラムは、メインプログラムと必要があればサブプログラムや関数で構成される。

メインプログラムは実行部分のプログラムをいい、サブプログラムは Sub～End Subで構成され、関数は Function～End Functionで構成される部分である。



プログラムは実行されると，メインプログラムが実行され、サブプログラムはメインプログラムからCall文で呼び出され、関数は数式内で呼び出される。
呼び出されるときに、変数の並びを使って、メインプログラムのデータがサブプログラムや関数に引き渡される。

サブプログラムや関数は、まとまりある処理を記述するときに用いられ、データを変えて同じ処理を何度も行うときに便利である。また、プログラムをわかりやすくする効果もある。

文	機能
Sub	・サブプログラムを定義する。 ・Call文で呼び出される。
Function	・関数を定義する。 ・数式内で呼び出される。 ・関数内の処理を行い、計算結果を返す。
Call	・サブプログラムを呼び出す。

サブプログラムの例（1からNまでの和を求める）を示す。

●プログラム(K711.bas)

```
1  ' << K711.bas >>
2  '
3  N=5
4  Call Sum(N)
5  '
6  Call Sum(10)
7  End
8  '
9  ' サブプログラム。
10 Sub Sum(N)
11   S=0
12   For I=1 To N: S=S+I: Next I
13   Print"Sum(";N;")=";S
14 End Sub
```

実行結果

```
Sum( 5)= 15
Sum( 10)= 55
OK
```

関数の例（1からNまでの和を求める）を示す。

●プログラム(K712.bas)

```
1  ' << K712.bas >>
2  '
3  N=5
4  Print"Sum(";N;")=";Sum(N)
5  '
6  Print"Sum(";N;")=";Sum(10)
7  End
8  '
9  ' 関数。
10 Function Sum(N)
11   S=0
12   For I=1 To N: S=S+I: Next I
13   Sum=S
14 End Function
```

実行結果

```
Sum( 5)= 15
Sum( 10)= 55
OK
```

Sub文は一般につぎのように書く。

書き方	**Sub サブプログラム名（変数の並び）** **処理** End Sub
機能	・サブプログラム名は変数名と同じ規則に従う。 ・サブプログラム内で使われる変数は、外部の変数と独立に扱われる。 すなわち、同じ変数名でも異なるものと扱われる。 **局所変数**と呼ばれる。 ・変数の並びは,変数をカンマで区切って並べる。 配列変数の場合は、配列名()と書く。 この変数の並びを使って、サブプログラム内にデータが渡される。 ・Call文で変数を使うと，**変数引渡し**になり、式を使うと**値引渡し**になる。

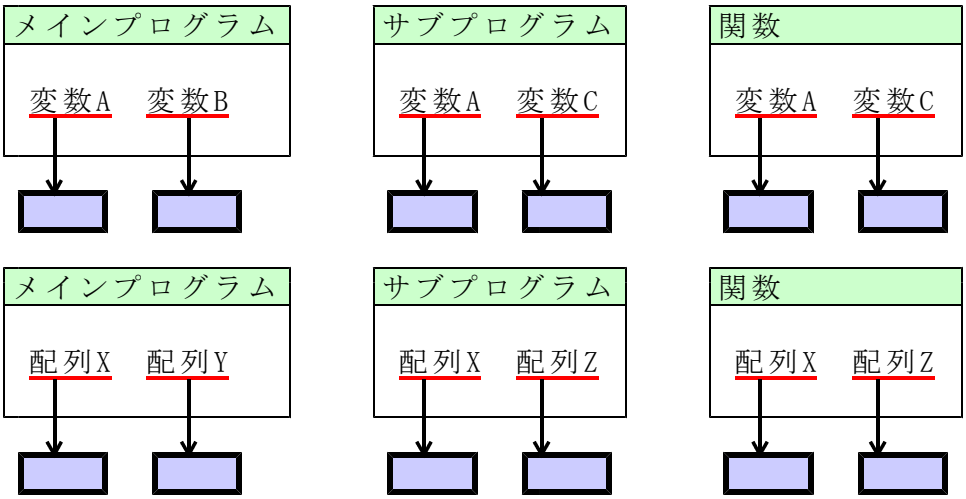
Function文は一般につぎのように書く。

書き方	**Function 関数名（変数の並び）** **処理** **関数名=計算結果** End Function
機能	・関数名は変数名と同じ規則に従う。 ・関数内で使われる変数は、外部の変数と独立に扱われる。 すなわち、同じ変数名でも異なるものと扱われる。 **局所変数**と呼ばれる。 ・変数の並びは,変数をカンマで区切って並べる。 配列変数の場合は、配列名()と書く。 この変数の並びを使って、関数内にデータが渡される。 ・変数の並びにおいて、変数は**値引渡し**、配列は**変数引渡し**になる。 ・関数名=計算結果 で、計算結果が戻される。

７．２　局所変数

局所変数の動作を確認する。

メインプログラムとサブプログラム（関数）で異なる名前の変数、配列があったとき、それぞれ独立に動作する。
メインプログラムとサブプログラム（関数）で同じ名前の変数、配列があったときも、それぞれ独立に動作する。



メインプログラムとサブプログラムや関数の作成時期が異なったり、異なる人でプログラムを共同開発したときに、この規則があれば、不用意なミスを防ぐことができる。

●変数の場合

サブプログラム側の更新がメインプログラム側に反映していないことに注意。

```
1  ' << K721.bas >>
2  ' メインプログラム。
3  A$="メインプログラム内"
4  Print"(メイン側：Call前)A$=";A$
5  '
6  Call Test()
7  '
8  Print"(メイン側：Call後)A$=";A$
9  End
10 '
11 ' サブプログラム。
12 Sub Test()
13   A$="サブプログラム内"
14   Print"(サブ側)        A$=";A$
15 End Sub
```

実行結果

```
(メイン側：Call前)A$=メインプログラム内
(サブ側)          A$=サブプログラム内
(メイン側：Call後)A$=メインプログラム内      A$の値は変わっていない。
OK
```

●配列の場合

サブプログラム側の更新がメインプログラム側に反映していないことに注意。

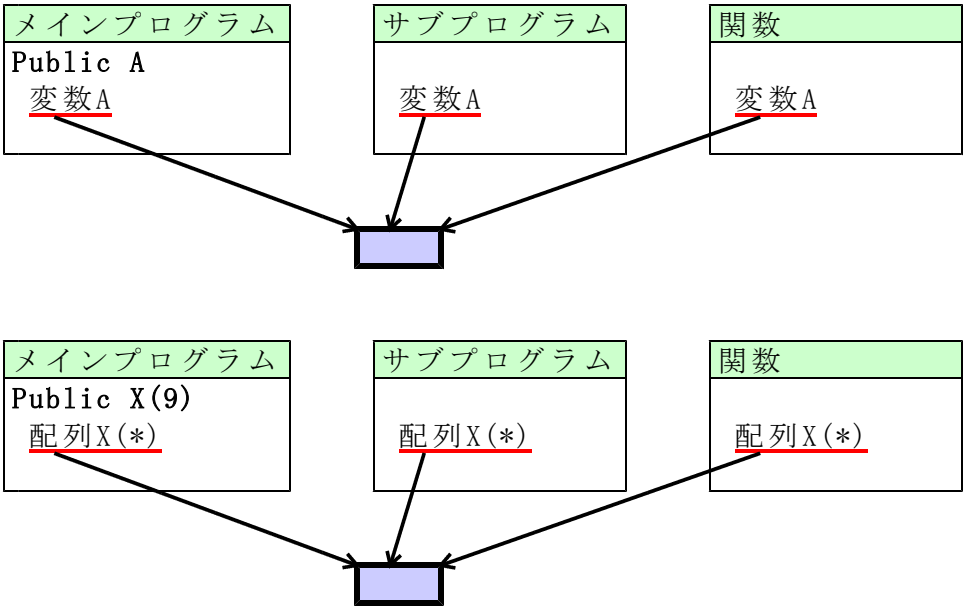
```
' << K722.bas >>
' メインプログラム。
Dim A(2)
'
A(1)=123: A(2)=456
'
Print"(メイン側：Call前) A(1)=";A(1);" A(2)=";A(2)
'
Call Test()
'
Print"(メイン側：Call後) A(1)=";A(1);" A(2)=";A(2)
End
'
' サブプログラム。
Sub Test()
  Dim A(2)
  A(1)=-123: A(2)=-456
  Print"(サブ側)           A(1)=";A(1);" A(2)=";A(2)
End Sub
```

実行結果

```
(メイン側：Call前) A(1)= 123 A(2)= 456
(サブ側)           A(1)=-123 A(2)=-456
(メイン側：Call後) A(1)= 123 A(2)= 456    A(1),A(2)の値は
                                          変わっていない。
OK
```

７．３　Public変数

Public変数とは、メインプログラム、サブプログラム、関数で共通に使える変数のことをいう。Public文で宣言する。



共通に使える変数があると便利である。

文	機能
Public	Public 変数名 メインプログラム、サブプログラム、関数で共通に使える変数となる

●変数の場合

サブプログラム側の更新がメインプログラム側に反映していることに注意。

```
1  ' << K731.bas >>
2  ' メインプログラム。
3  ' Public変数。
4  Public A,B$
5  '
6  A=123: B$="abc"
7  '
8  Print"(メイン側：Call前) A=";A;" B$=";B$
9  '
10 Call Test()
11 '
12 Print"(メイン側：Call後) A=";A;" B$=";B$
13 End
14 '
15 ' サブプログラム。
16 Sub Test()
```

```
17    Print"(サブ側)              A=";A;" B$=";B$
18    A=-A: B$=B$+B$
19    Print"(サブ側)              A=";A;" B$=";B$
20  End Sub
```

実行結果

```
1  (メイン側：Call前) A= 123 B$=abc
2  (サブ側)           A= 123 B$=abc
3  (サブ側)           A=-123 B$=abcabc
4  (メイン側：Call後) A=-123 B$=abcabc    A,B$の値が変わっている。
5  OK
```

●配列の場合

サブプログラム側の更新がメインプログラム側に反映していることに注意。

```
1  ' << K732.bas >>
2  ' メインプログラム。
3  Public A(2)
4  '
5  A(1)=123: A(2)=456
6  '
7  Print"(メイン側：Call前) A(1)=";A(1);" A(2)=";A(2)
8  '
9  Call Test()
10 '
11 Print"(メイン側：Call後) A(1)=";A(1);" A(2)=";A(2)
12 End
13 '
14 ' サブプログラム。
15 Sub Test()
16   A(1)=-123: A(2)=-456
17   Print"(サブ側)              A(1)=";A(1);" A(2)=";A(2)
18 End Sub
```

実行結果

```
1  (メイン側：Call前) A(1)= 123 A(2)= 456
2  (サブ側)           A(1)=-123 A(2)=-456
3  (メイン側：Call後) A(1)=-123 A(2)=-456      A(1),A(2)の値が
4                                              変わっている。
5  OK
```

７．４　変数引渡し、値引渡し

メインプログラム内のデータをサブプログラムや関数に引き渡して処理を行う場合、変数、配列、式を通して行う。

メインプログラムCall文中の変数の並び（実引数と呼ぶ）とサブプログラムや関数の変数の並び（仮引数と呼ぶ）との対応付けや、サブプログラム内や関数内での処理結果がどのようにメインプログラムに返されるのかという点に注意が必要である。

７．４．１　サブプログラム

・変数の並びは，Call 文で変数を使って呼ぶと，変数引渡しになる。
　すなわち、変数引渡しでは、サブプログラム内での変更がメインプログラムへ影響を及ぼす。
　配列変数の引渡しも変数引渡しになる。

①変数引渡し（変数）

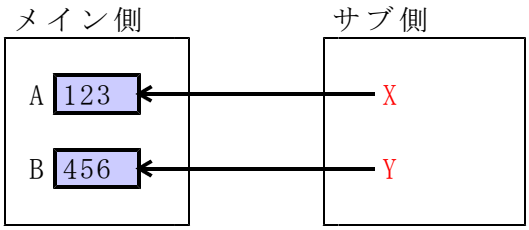
```
1  ' << K741a.bas >>
2  ' メインプログラム。
3  A=123: B=456
4  '
5  Print"（メイン側）A=";A;" B=";B
6  '
7  Call Test(A,B): ' 実引数は変数。
8  '
9  Print"（メイン側）A=";A;" B=";B
10 '
11 ' サブプログラム。
12 Sub Test(X,Y) ' 仮引数は変数。
13   Print"（サブ側）   X=";X;" Y=";Y
14   X=-X: Y=-Y
15   Print"（サブ側）   X=";X;" Y=";Y
16 End Sub
```

実行結果

```
1  （メイン側）A= 123 B= 456
2  （サブ側）  X= 123 Y= 456
3  （サブ側）  X=-123 Y=-456
4  （メイン側）A=-123 B=-456    A,Bの値が変化している。
5  OK
```

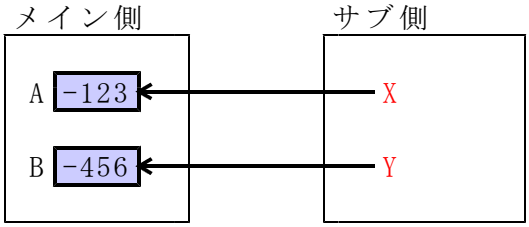
次のような処理が行われる。

（１）サブプログラム開始時



サブ側変数X,Yとメイン側変数A,Bが同一視される。
すなわち、サブ側変数X,Yに同じ記憶場所が割り当てられる。

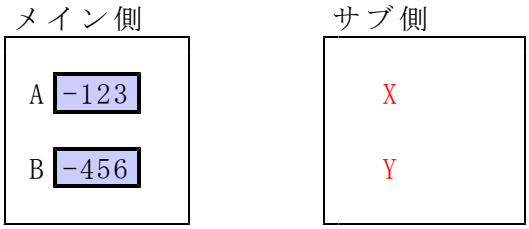
（２）サブプログラム内処理



サブ側変数X,Yとメイン側変数A,Bが同一視されるので、
サブ側変数X,Yの変更がメイン側変数A,Bに連動する

（３）サブプログラム終了時

サブ側変数X,Yとメイン側変数A,Bの対応が解消される。



②変数引渡し（配列）

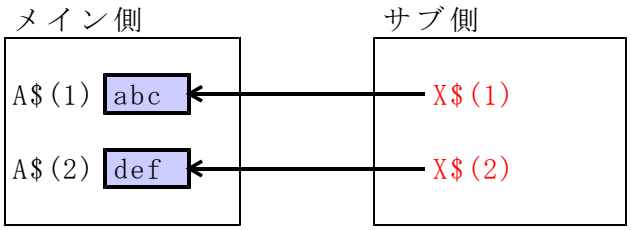
```
1  ' << K741b.bas >>
2  ' メインプログラム。
3  Dim A$(2)
4  '
5  A$(1)="abc": A$(2)="def"
6  '
7  Print"（メイン側） A$(1)=";A$(1);" A$(2)=";A$(2)
8  '
9  Call Test(A$()): ' 実引数は配列。
10 '
11 Print"（メイン側） A$(1)=";A$(1);" A$(2)=";A$(2)
12 '
13 ' サブプログラム。
14 Sub Test(X$()) ' 仮引数は配列。
15   Print"（サブ側）   X$(1)=";X$(1);" X$(2)=";X$(2)
16   X$(1)="ABC": X$(2)="DEF"
17   Print"（サブ側）   X$(1)=";X$(1);" X$(2)=";X$(2)
18 End Sub
```

実行結果

```
1  （メイン側） A$(1)=abc A$(2)=def
2  （サブ側）   X$(1)=abc X$(2)=def
3  （サブ側）   X$(1)=ABC X$(2)=DEF
4  （メイン側） A$(1)=ABC A$(2)=DEF      A$(1),A$(2)の値が
5  OK                                    変わっている。
```

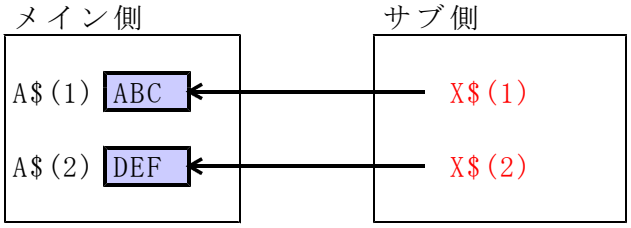
次のような処理が行われる。

（１）サブプログラム開始時



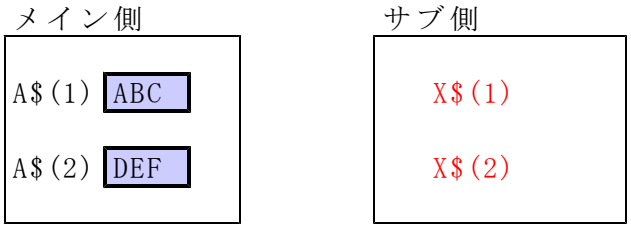
サブ側配列X$とメイン側配列A$が同一視される。
すなわち、サブ側配列X$に同じ記憶場所が割り当てられる。

（２）サブプログラム内処理



サブ側変数X$とメイン側変数A$が同一視されるので、
サブ側配列X$の変更がメイン側配列A$に連動する

（３）サブプログラム終了時



サブ側配列X$とメイン側配列A$の対応が解消される。

③値引渡し(変数)

・変数の並びは，Call 文で式を使って呼ぶと，値引渡しになる。

値引渡しでは、サブプログラム内での変更がメインプログラムへ影響を及ぼさない。

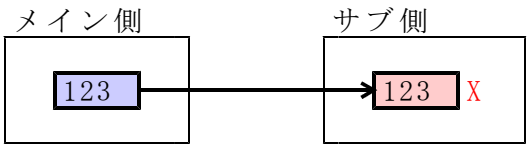
```
1  ’ << K741c.bas >>
2  ’ メインプログラム。
3  A=123
4  ’
5  Print"(メイン側：Call前)A=";A
6  ’
7  Call Test(1*A): ’ 実引数が式であることに注意。
8  ’
9  Print"(メイン側：Call後)A=";A
10 End
11 ’
12 ’ サブプログラム。
13 Sub Test(X) ’ 仮引数は変数。
14   X=-X
15   Print"(サブ側)          X=";X
16 End Sub
```

実行結果

```
1  (メイン側：Call前)A= 123
2  (サブ側)          X=-123
3  (メイン側：Call後)A= 123    Aの値は変わっていない。
4  OK
```

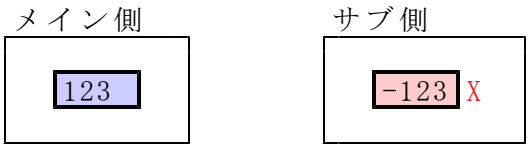
次のような処理が行われる。

（１）サブプログラム開始時



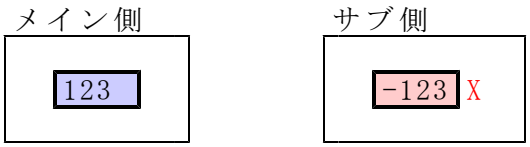
サブ側変数Xとメイン側変数Aは独立したものとして扱われる。
メイン側からサブ側にデータがコピーされる。

（２）サブプログラム内処理



関数側変数Xの変更がメイン側変数Aに影響しない。

（３）サブプログラム終了時



サブ側からメイン側に仮引数のデータがコピーされることはない。

７．４．２　関数

・Function文の変数の並びにおいて、変数は値引渡し、配列は変数引渡しになる。

値引渡しでは、関数内での変更がメインプログラムへ影響を及ぼさない。
変数引渡しでは、関数内での変更がメインプログラムへ影響を及ぼす。

①値引渡し(変数)

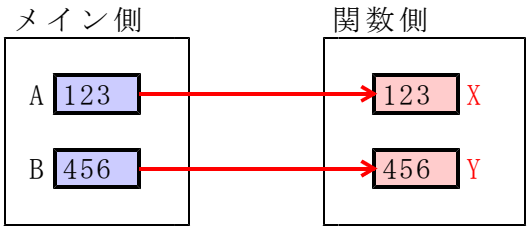
```
1  ' << K742a.bas >>
2  ' メインプログラム。
3  A=123: B=456
4  '
5  Print"（メイン側：呼び出し前）A=";A;" B=";B
6  '
7  C=Test(A,B): ' 実引数は変数。
8  '
9  Print"（メイン側：呼び出し後）A=";A;" B=";B
10 '
11 ' 関数。
12 Function Test(X,Y) ' 仮引数は変数。
13   Print"（関数側）              X=";X;" Y=";Y
14   X=-X: Y=-Y
15   Print"（関数側）              X=";X;" Y=";Y
16 End Function
```

実行結果

```
1  （メイン側：呼び出し前）A= 123 B= 456
2  （関数側）              X= 123 Y= 456
3  （関数側）              X=-123 Y=-456
4  （メイン側：呼び出し後）A= 123 B= 456    A,Bの値は変わっていない。
5  OK
```

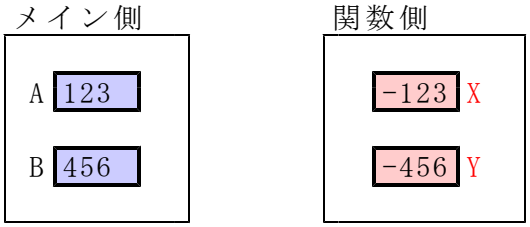
次のような処理が行われる。

（１）関数開始時



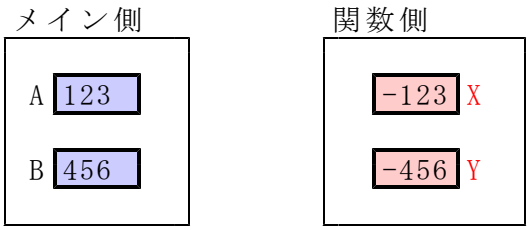
関数側変数X,Yとメイン側変数A,Bは独立したものとして扱われる。
メイン側から関数側にデータがコピーされる。

（２）関数内処理



関数側変数X,Yの変更がメイン側変数A,Bに影響しない。

（３）関数終了時



関数側からメイン側に仮引数の値がコピーされることはない。

②変数引渡し(配列)

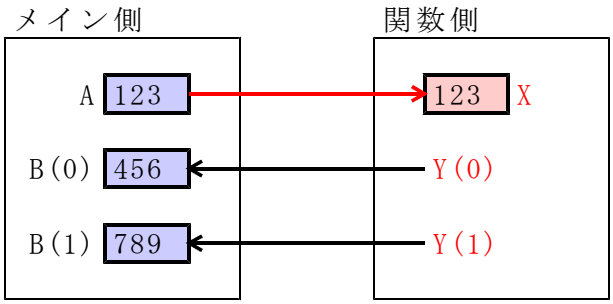
```
1  ' << K742b.bas >>
2  ' メインプログラム。
3  A=123
4  '
5  Dim B(1)
6  '
7  B(0)=456: B(1)=789
8  '
9  Print"(メイン側 : 呼び出し前)A=";A;" B(0)=";B(0);" B(1)=";B(1)
10 '
11 R=Func(A,B()): ' 実引数は、変数と配列。
12 '
13 Print"(メイン側 : 呼び出し後)A=";A;" B(0)=";B(0);" B(1)=";B(1)
14 End
15 '
16 ' 関数。
17 Function Func(X,Y()) ' 仮引数は、変数と配列。
18   X=-X: Y(0)=-Y(0): Y(1)=-Y(1)
19   Print"(関数側)              X=";X;" Y(0)=";Y(0);" Y(1)=";Y(1)
20   Func=X+Y(0)+Y(1)
21 End Function
```

実行結果

```
1  (メイン側 : 呼び出し前)A= 123 B(0)= 456 B(1)= 789
2  (関数側)              X=-123 Y(0)=-456 Y(1)=-789
3  (メイン側 : 呼び出し後)A= 123 B(0)=-456 B(1)=-789  Aの値は
4                                                    変わっていない。
5                                                    B(0),B(1)の値は
6  OK                                                変わっている。
```

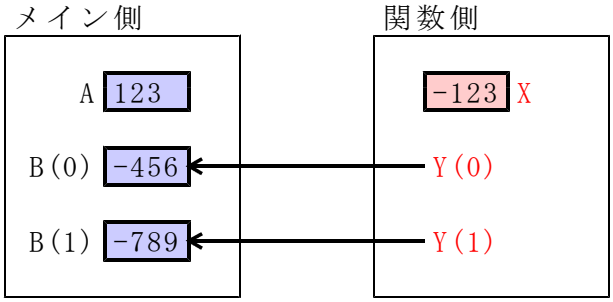
次のような処理が行われる。

（１）関数開始時



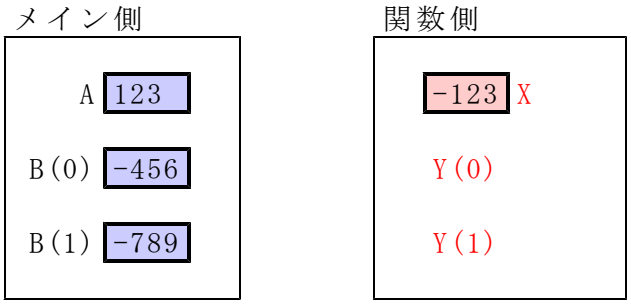
関数側変数Xとメイン側変数Aは独立したものとして扱われる。
メイン側からサブ側にデータがコピーされる。
関数側配列Yとメイン側配列Bが同一視される。
すなわち、関数側配列Yに同じ記憶場所が割り当てられる。

（２）関数内処理



関数側変数Xの変更がメイン側変数Aに影響しない。
関数側配列Yの変更がメイン側配列Bに連動する

（３）関数終了時



サブ側からメイン側に仮引数のデータがコピーされることはない。
サブ側配列Y$とメイン側配列B$の対応が解消される。