

Prototype
declaration

```
/* Prototype declarations */  
int multiply (int multiplier, int multiplicand );  
  
int main (void)  
{  
    int product;  
    ...  
    product = multiply (6, 7);  
    ...  
    return 0;  
} /* main */
```

Call

Function
definition

```
int multiply (int x,  
             int y)  
{  
    ...  
    return x * y;  
} /* multiply */
```

42

values copied

x	6
y	7

```
/* Prototype Declarations */
```

```
void fun (int num1);
```

```
int main (void)
```

```
{
```

```
/* Local Definitions */
```

```
int a = 5;
```

```
/* Statements */
```

```
fun (a)
```

```
printf("%d\n", a);
```

```
return 0;
```

```
} /* main */
```

prints 5

```
void fun (int x)
```

```
{
```

```
/* Statements */
```

```
x = x + 3;
```

```
return;
```

```
} /* fun */
```

a 5

One-way
communication

x 5

Only a copy


MAIN PROGRAM MEMORY

4000



age

If you pass only a copy of 25 to a function, it is called “pass-by-value” and the function will not be able to change the contents of age. It is still 25 when you return.




MAIN PROGRAM MEMORY

4000



age

BUT, if you pass 4000, the address of age to a function, it is called “pass-by-reference” and the function will be able to change the contents of age. It could be 23 or 90 when you return.



4000

25

age

Argument in Calling Block

Value Parameter

The value (25) of the argument is passed to the function when it is called.

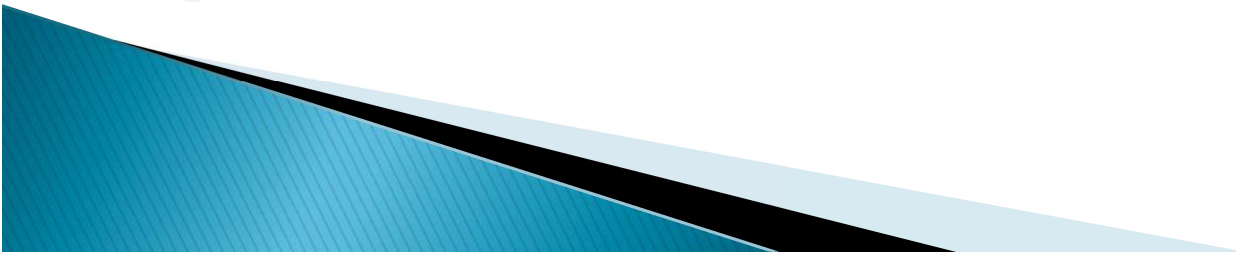
Reference Parameter

The memory address (4000) of the argument is passed to the function when it is called.

By default, parameters

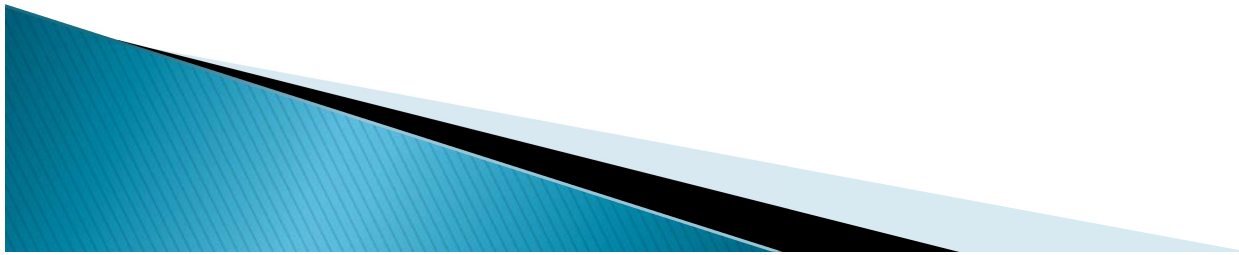
**are always value parameters,
unless you do something to
change that.**

**To get a reference parameter
you need to use addresses as
parameters.**

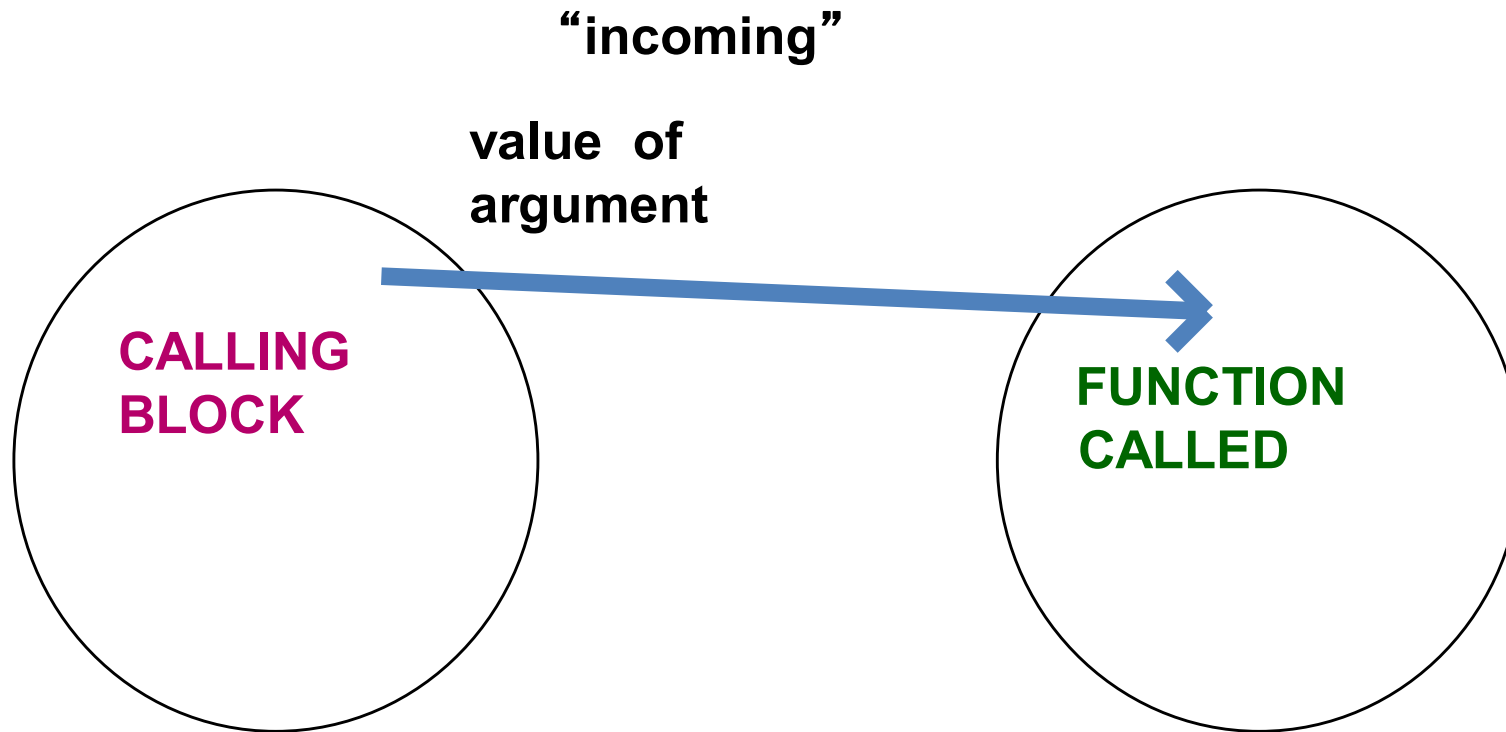


When to Use Reference Parameters

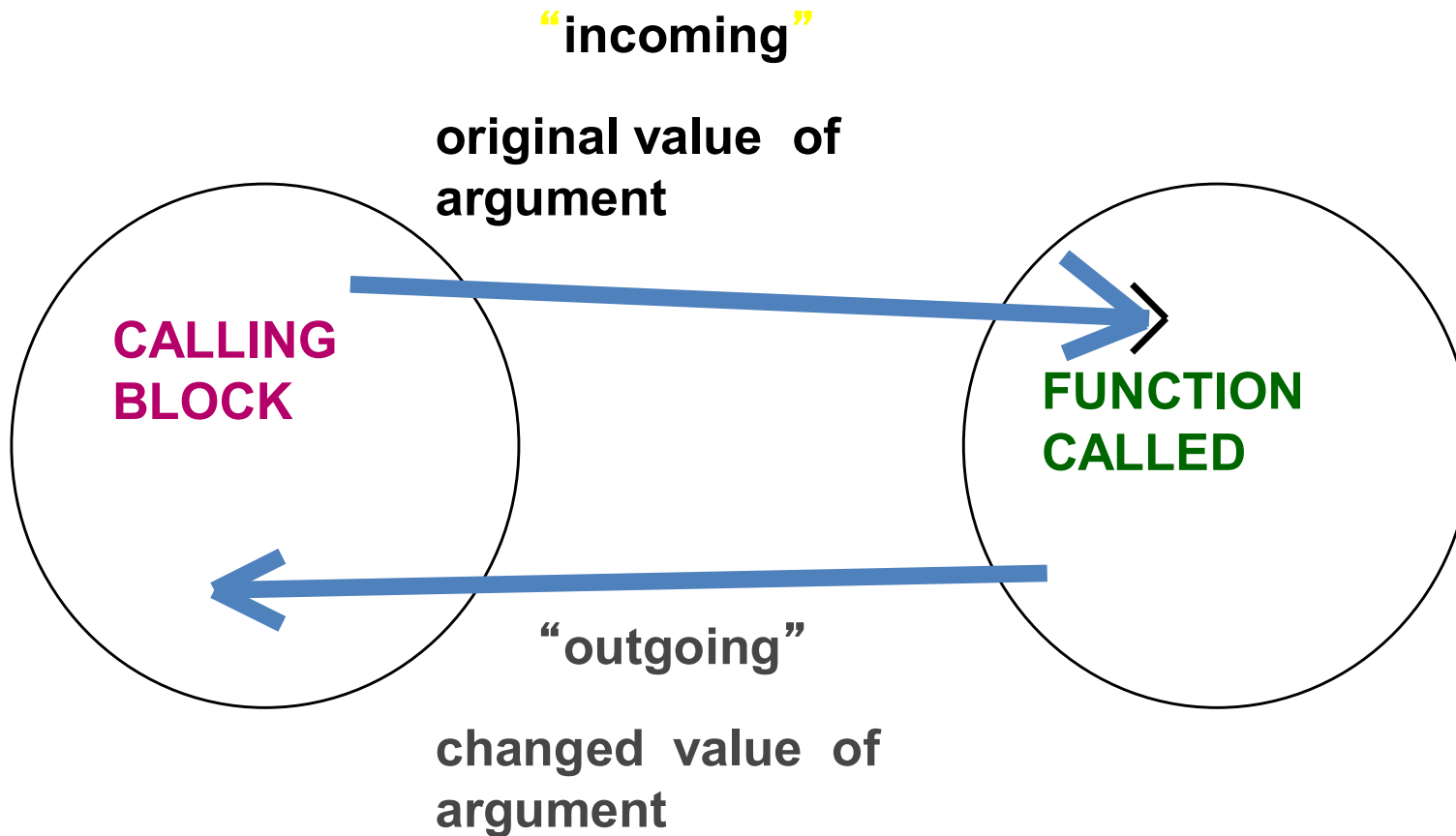
- ▶ **reference parameters should be used when you want your function to give a value to, or change the value of, a variable from the calling block.**
- ▶ **Return more than one variables.**



Pass-by-value

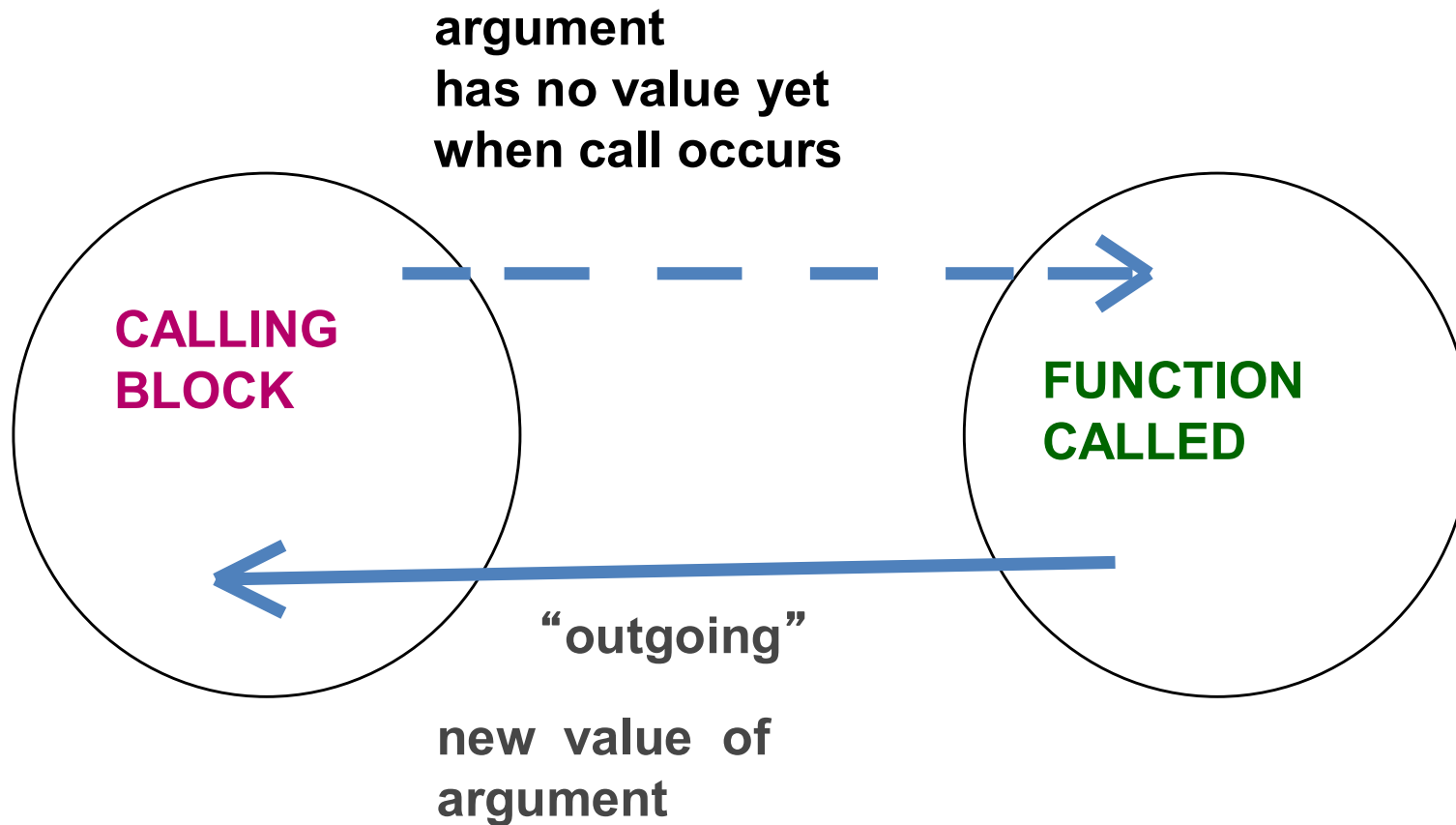


Pass-by-reference



OR,

Pass-by-reference



Example: bad_swap.c

```
/* Swap the values of two variables.
   */
```

```
void badSwap ( int a, int b )
{
    int temp;

    temp = a;
    a = b;
    b = temp;

    printf("%d %d\n", a, b);
}
```

```
int main()
{
    int a = 3, b = 5;

    printf("%d %d\n", a, b);
    badSwap ( a, b );
    printf("%d %d\n", a, b);

    return 0;
}
```

Output: 3 5

Example: bad_swap.c

```
/* Swap the values of two variables.
   */

void badSwap ( int a, int b )
{
    int temp;

    temp = a;
    a = b;
    b = temp;

    printf("%d %d\n", a, b);
}
```

```
int main()
{
    int a = 3, b = 5;

    printf("%d %d\n", a, b);
    badSwap ( a, b );
    printf("%d %d\n", a, b);

    return 0;
}
```

Output:

```
3 5
5 3
```

Example: bad_swap.c

```
/* Swap the values of two variables.
   */

void badSwap ( int a, int b )
{
    int temp;

    temp = a;
    a = b;
    b = temp;

    printf("%d %d\n", a, b);
}
```

```
int main()
{
    int a = 3, b = 5;

    printf("%d %d\n", a, b);
    badSwap ( a, b );
    printf("%d %d\n", a, b);

    return 0;
}
```

Output:

```
3 5
5 3
3 5
```

Example: bad_swap.c

```
/* Swap the values of two variables. */
```

```
void badSwap ( int a, int b )  
{  
    int temp;  
  
    temp = a;  
    a = b;  
    b = temp;  
  
    printf("%d %d\n", a, b);  
}
```

```
int main()  
{  
    int a = 3, b = 5;  
  
    printf("%d %d\n", a, b);  
    badSwap ( a, b );  
    printf("%d %d\n", a, b);  
  
    return 0;  
}
```

Output:

```
3 5  
5 3  
3 5
```

Example: swap.c

```
/* Swap the values of two variables. */

void Swap ( int *a, int *b )
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;

    printf("%d %d\n", *a, *b);
}
```

```
int main()
{
    int a = 3, b = 5;

    printf("%d %d\n", a, b);
    Swap ( &a, &b );
    printf("%d %d\n", a, b);

    return 0;
}
```

Output:

3 5

Example: swap.c

```
/* Swap the values of two variables.
   */

void Swap ( int *a, int *b )
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;

    printf("%d %d\n", *a, *b);
}
```

```
int main()
{
    int a = 3, b = 5;

    printf("%d %d\n", a, b);
    Swap ( &a, &b );
    printf("%d %d\n", a, b);

    return 0;
}
```

Output:

```
3 5
5 3
```


Example: swap.c

```
/* Swap the values of two variables.
   */

void Swap ( int *a, int *b )
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;

    printf("%d %d\n", *a, *b);
}
```

```
int main()
{
    int a = 3, b = 5;

    printf("%d %d\n", a, b);
    Swap ( &a, &b );
    printf("%d %d\n", a, b);

    return 0;
}
```

Output:

```
3 5
5 3
5 3
```

Example: swap.c

```
/* Swap the values of two variables.
   */

void Swap ( int *a, int *b )
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;

    printf("%d %d\n", *a, *b);
}
```

```
int main()
{
    int a = 3, b = 5;

    printf("%d %d\n", a, b);
    Swap ( &a, &b );
    printf("%d %d\n", a, b);

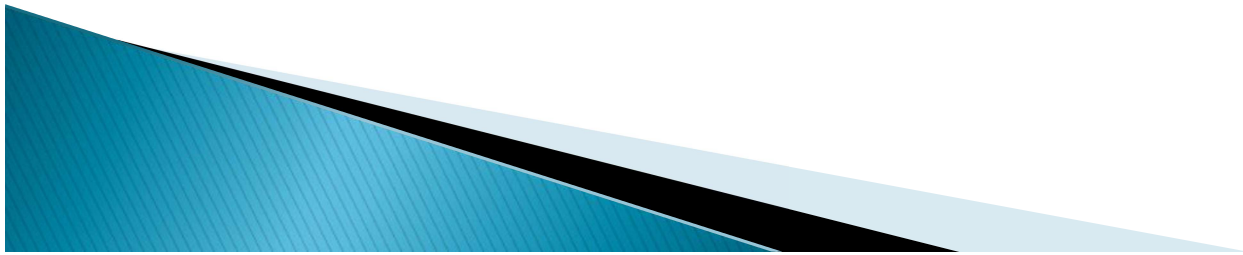
    return 0;
}
```

Output:

```
3  5
5  3
5  3
```

Example of Pass-by-Reference

We want to find 2 real roots for a quadratic equation with coefficients a,b,c. Write a prototype for a void function named `GetRoots()` with 5 parameters. The first 3 parameters are type float. The last 2 are reference parameters of type float.



```
#include <stdio.h>
#include <math.h>
void GetRoots(float, float, float, float*,
float*);
void main ( )
{
    float a, b, c, first, second;
    printf(" Enter a b c : ");
    scanf("%f %f %f", &a , &b, &c );
    GetRoots(a, b, c, &first, &second);
    printf(" Roots are %.2f %.2f \n " , first ,
second );
}

// function GetRoots goes here
.....
```

Function Definition

```
void GetRoots( float a, float b, float c,  
float *root1, float *root2)  
{  
    float temp;                // local variable  
  
    temp = b * b - 4.0 * a * c;  
  
    *root1 = (-b + sqrt(temp) ) / ( 2.0 * a );  
  
    *root2 = (-b - sqrt(temp) ) / ( 2.0 * a );  
  
    return;  
}
```