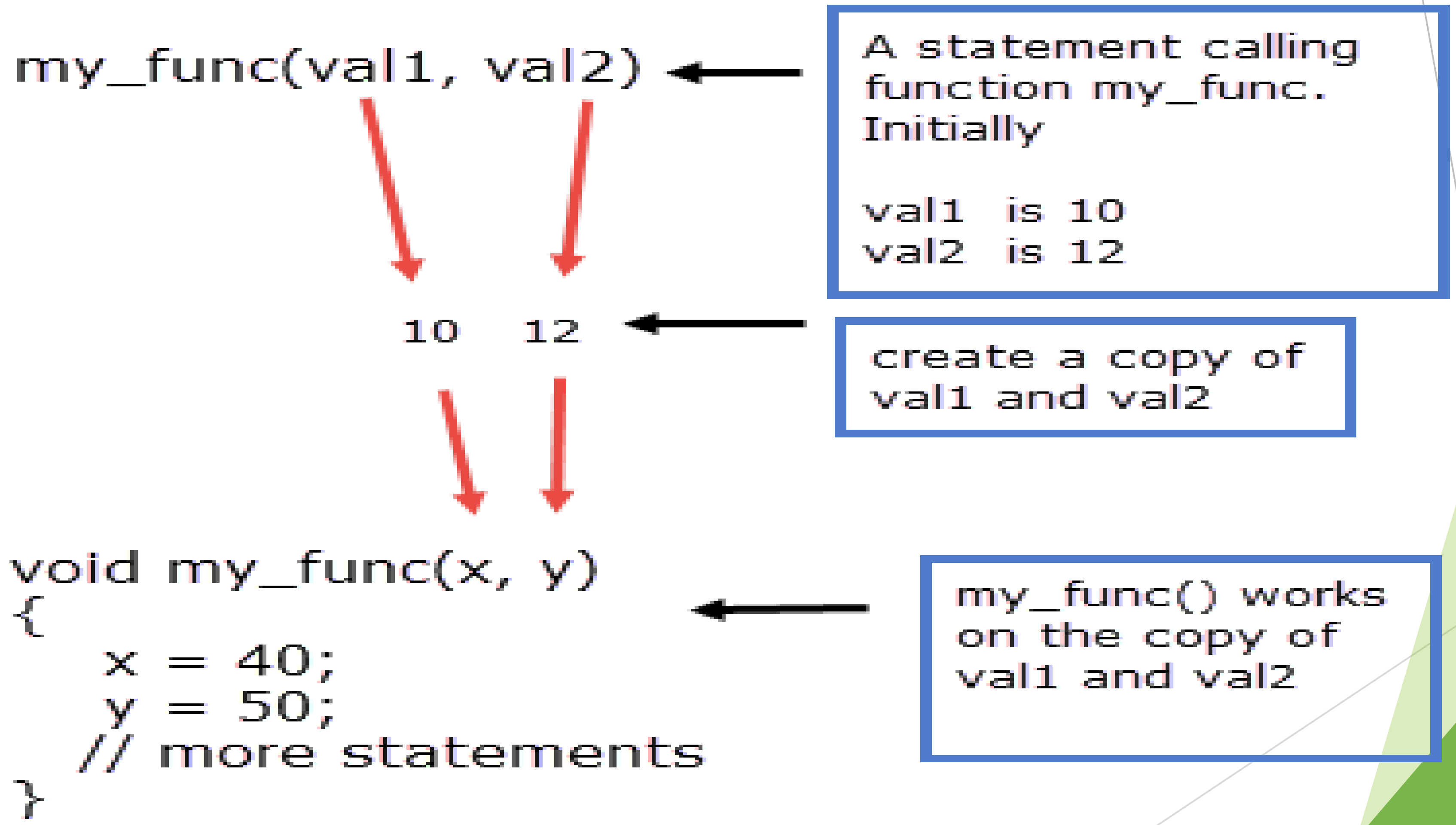
Pointers

Call by Reference

Call by value

In this method a copy of each of the actual arguments is made first then these values are assigned to the corresponding formal arguments.



```
my_func(val1, val2)
```

A statement calling function my_func. Initially

val1 is 10
val2 is 12

```
10   12
```

create a copy of val1 and val2

```
void my_func(x, y)
{
   x = 40;
   y = 50;
  // more statements
}
```

my_func() works on the copy of val1 and val2

Call by Reference

- In this method addresses of the actual arguments are copied and then assigned to the corresponding formal arguments.
- Now formal and actual arguments both points to the same data (because they contain the same address).
- As a result, any changes made by called function also affect the actual arguments.
- To use call by reference we need to do two things:
  - Pass the addresses of the actual arguments instead of passing values to the function.
  - Declare the formal arguments of the function as pointer variables of an appropriate type.

Example - Passing Pointer to a Function in C

```
#include <stdio.h>
void salaryhike(int  *var, int b)
{
    *var = *var + b;
}
int main()
{
    int salary = 0, bonus = 0;
    printf("Enter the employee current salary:");
    scanf("%d", &salary);
    printf("Enter bonus:");
    scanf("%d", &bonus);
    salaryhike(&salary, bonus);
    printf("Final salary: %d", salary);
    return 0;
}
```

Output:

```
Enter the employee current salary:10000
Enter bonus: 2000
Final salary: 12000
```

Swapping two numbers using Pointers

```
#include  <stdio.h>
void swap(int *num1, int *num2);

int main( ) {
   int v1 = 11, v2 = 77 ;
   printf("Before swapping:");
   printf("\nValue of v1 is: %d", v1);
   printf("\nValue of v2 is: %d", v2);
   /*calling swap function*/
   swap( &v1, &v2 );

   printf("\nAfter swapping:");
   printf("\nValue of v1 is: %d", v1);
   printf("\nValue of v2 is: %d", v2);
}
```

```
void swap(int *num1, int *num2)
{
   int tempnum;
   tempnum = *num1;
   *num1 = *num2;
   *num2 = tempnum;
}
```

Output:

```
Before swapping:
Value of v1 is: 11
Value of v2 is: 77

After swapping:
Value of v1 is: 77
Value of v2 is: 11
```

Pointers as Function Parameters

- Sometimes, you want a function to return more than one variables
- Example, you want a function that computes the minimum AND maximum numbers in 2 integers.
- Method 1, use two global variables.
  - In the function, assign the minimum and maximum numbers to the two global variables.
  - When the function returns, the calling function can read the minimum and maximum numbers from the two global variables.
  - This is bad because the function is not reusable.
- Method 2, use pointers instead.

Pointers as Function Parameters

```
void min_max(int a, int b, int *min, int*max)
{
    if( a > b)
    {
          *max=a;
          *min=b;
    }
    else
    {
          *max=b;
          *min=a;
    }
}
```

```
int main()
{
    int x, y;
    int small, big;
    printf("Two integers: ");
    scanf("%d %d", &x, &y);

    min_max(x, y, &small, &big);

    printf("%d <= %d", small, big);
    return 0;
}
```

Homework

1. Create a function to return sum, difference and product of two numbers passed to it.

2. Write a loop that uses a pointer reference as the loop counter. In other words, convert the variable x to a pointer reference in the below code so that it works exactly the same :

```
int x;
for (x= 1; x < 5; x++)
    printf(“loop counter value is %d \n“,x);
```

a) What is the value of the loop counter once the for loop has finished executing? Write a printf statement to output this value using the pointer variable. Write another printf statement to output this value using the variable x.

b) What is the value of the pointer variable; not the value that it points to, but the value of the pointer itself (i.e. the address)? Write a printf statement to output this value using the pointer variable. Write another printf statement to output this value using the variable x.