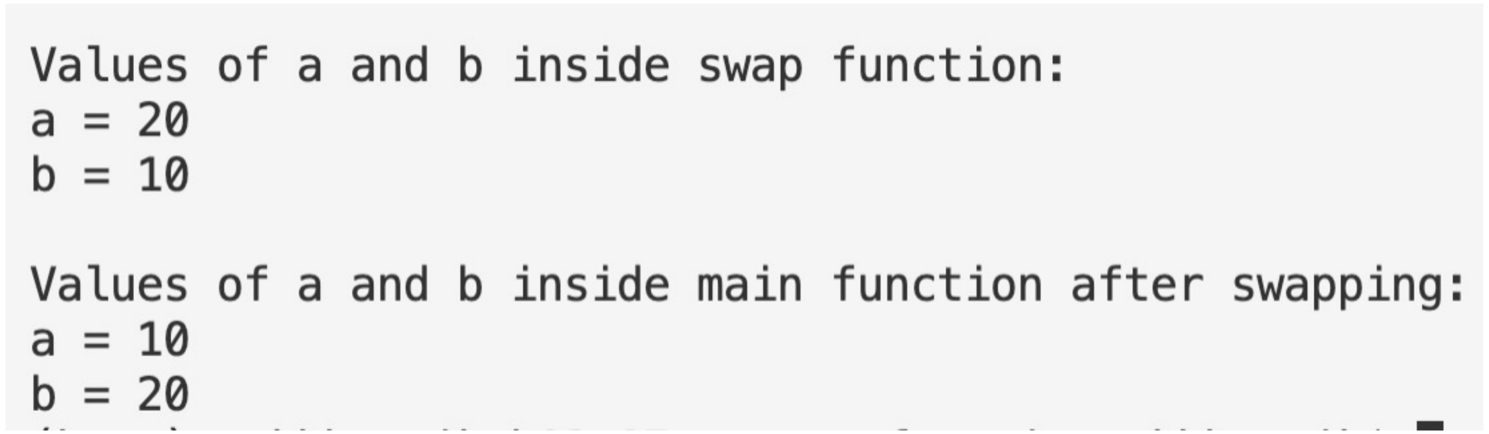
Review

```c
#include <stdio.h>

void swap(int a, int b);

int main(){
    int a, b;
    a = 10;
    b = 20;
    swap(a, b);

    printf("\nValues of a and b inside main function after swapping:\n");
    printf("a = %i\n", a);
    printf("b = %i\n", b);
}
```

```c
void swap(int a, int b){
    int temp;
    temp = a;
    a = b;
    b = temp;

    printf("\nValues of a and b inside swap function:\n");
    printf("a = %i\n", a);
    printf("b = %i\n", b);
}
```

Review



```
Values of a and b inside swap function:
a = 20
b = 10

Values of a and b inside main function after swapping:
a = 10
b = 20
```

Review

```c
#include <stdio.h>

void check_divisibility(int a);

int main(){
    int a = 12;

    check_divisibility(a);

    return 0;
}
```

```c
void check_divisibility(int a){
    if(a % 2 == 0){
        printf("%i is divisible by 2.\n", a);
        return;
    }

    if(a % 3 == 0){
        printf("%i is divisible by 3.\n", a);
        return;
    }
}
```

Review

- 12 is divisible by 2.

Functions and Arrays

Passing Array Elements to Function

- Possible to pass an array element or an entire array as argument to a function
- Passing a single array element to a function
  - printf(“%i\n”, array[i]);
  - sq_root_result = squareRoot (averages[i]);
- To pass an entire array, it is only necessary to list the name of the array, *without any subscript*
  - Let’s say gradeScore is an array containing 100 elements
  - minimum(gradeScore) passes the entire array as an argument

Finding minimum value of array

- minimum() returns value of type int
- parameter list contains an integer array of size 100

```
int  minimum (int  values[100])
{
     ...
     return minValue;
}
```

Finding minimum value of array

```c
int  minimum (int  values[10]);

int main ()
{
    int  scores[10], i, minScore;

    printf ("Enter 10 scores\n");

    for ( i = 0;  i < 10;  ++i )
        scanf ("%i", &scores[i]);

    minScore = minimum (scores);
    printf ("\nMinimum score is %i\n", minScore);

    return 0;
}
```

```c
int  minimum (int  values[10])
{
    int  minValue, i;

    minValue = values[0];

    for ( i = 1;  i < 10;  ++i )
        if ( values[i] < minValue )
            minValue = values[i];

    return minValue;
}
```

Finding minimum value of array

- *values* is used to reference the elements of the array inside the function
    - declared to be an array of 10 integers
- *minValue* is used to store the minimum value in the array
    - initially set to values[0]
- for loop sequences through the remaining elements in the array
- comparing each element in turn against the value of *minValue*
- If the value of *values[i]* is less than *minValue*, a new minimum in the array has been found
- In such a case, the value of *minValue* is reassigned to this new minimum value and the scan through the array continues
- When the execution of the for loop, *minValue* is returned to the calling routine

Finding minimum value of array

- this general-purpose *minimum()* function can find the minimum value of any array containing 10 integers
- If you have five different arrays containing 10 integers each, you could simply call the minimum function five times and find the minimum of each array
- Won’t work on arrays holding more or less than 10 integers
- We can have it take the number of elements In the array as an argument
- In that case, the number of the elements of the array doesn’t need to be specified in function declaration

Finding minimum value of array

```c
#include <stdio.h>

int  minimum (int  values[], int  numberOfElements);

int main (void)
{
    int  array1[5] = { 157, -28, -37, 26, 10 };
    int  array2[7] = { 12, 45, 1, 10, 5, 3, 22 };

    printf ("array1 minimum: %i\n", minimum (array1, 5));
    printf ("array2 minimum: %i\n", minimum (array2, 7));

    return 0;
}
```

```c
int  minimum (int  values[], int  numberOfElements)
{
    int  minValue, i;

    minValue = values[0];

    for ( i = 1;  i < numberOfElements;  ++i )
        if ( values[i] < minValue )
            minValue = values[i];

    return minValue;
}
```

Finding minimum value of array

- *minimum()* is defined to take two arguments
- array whose minimum you want to find
  - The open and close brackets inform the compiler that *values* is an array of integers
- number of elements in the array

Arrays as Function Parameters

- Arrays are not copied when passed to functions
- Changing the value of an element in an array passed to a function changes the element in the original array

Arrays as Function Parameters

```c
#include <stdio.h>

// After this function returns, each element at each index i of the given
// array will store i * i.
void fill_with_squares(int array[], size_t size);

// Prints each element of the given array. The elements will be separated by
// spaces in the output.
void print_array(const int array[], size_t size);

int main() {
    int array[] = {1, -1, 10, -50, 46, 3};

    printf("The largest element in the array is %i\n", max_value(array, 6));

    fill_with_squares(array, 6);

    printf("Here are the new contents of the array:\n");
    print_array(array, 6);
    printf("\n");

    return 0;
}
```

```c
void fill_with_squares(int array[], size_t size) {
    for (size_t i = 0; i < size; i++) {
        array[i] = i * i;
    }
}

void print_array(const int array[], size_t size) {
    for (size_t i = 0; i < size; i++) {
        printf("%i ", array[i]);
    }
}
```

Arrays as Function Parameters

- Entire contents of the array are not copied into the formal parameter array
- The function gets passed information describing where in the computer’s memory the array is located.
- Any changes made to the array by the function are made to the original array passed to the function, and not to a copy of the array.

const keyword

- Use the keyword const if elements should not not change
  - The compiler will complain if you accidentally write code that changes the value of an element
- An array parameter marked as const communicates to people using the function that the function will not alter the array elements

size_t

- Unsigned integer type used to represent sizes
- Wide enough range to represent the largest possible array on a system
- Also an appropriate type to use for array indexes
- Use %zu with printf() and scanf()

Array Functions

- Write three functions that calculate from the values in an array the:
  - Minimum
  - Sum
  - Average
- Grading
  - 1 - Program Successfully Compiles
  - 1 - No loop in average()
  - 1 - Proper Style
  - 3 - Passes all tests