Functions With Arrays as Arguments

This lesson discusses functions and arrays in Rust!

We'll cover the following

It is often necessary to pass arrays as arguments to functions. Rust allows the programmer to pass arrays either by value or by reference.

Pass by Value

Arrays can be passed to a function by value. What that means is that a copy of the array from the calling function is made to the called function.

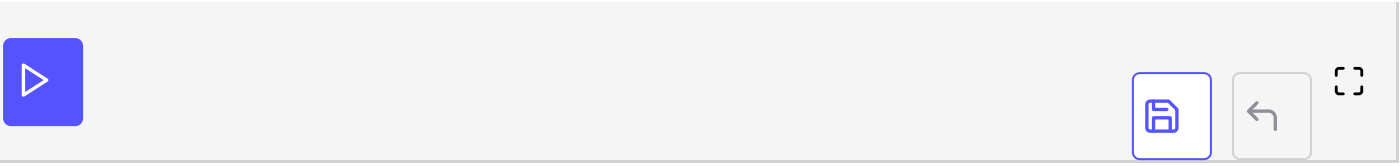
The general syntax for passing an array by value to a function is:

```
fn function_name( mut array_name:[datatype;size])
```

Example 1

The following example takes the array `arr` by value in the function parameter.

```
fn main() {
   let arr = [1, 2, 3, 4, 5];
   modify_my_array(arr);
   println!("Array in Driver Function : {:?}", arr);
}
fn modify_my_array(mut arr:[i32;5]){
   arr[2] = 8;
   arr[3] = 9;
   println!("Array in my Function : {:?}", arr);
}
```

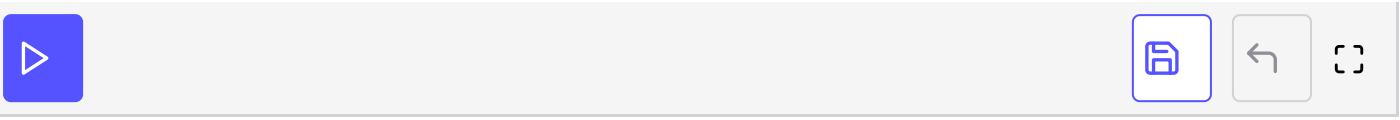


> **Note:** The `mut` **keyword when passing an array by value is optional**. It is written along with the array name if it is desired to make local changes. It can be omitted otherwise.

Example 2

The following example makes a function `calculate_mean` which calculates the mean of values in an array by first taking a summation inside a `for` loop and then dividing the result by 5.

```
fn main() {
   let arr = [1, 2, 3, 4, 5];
   println!("Array in Driver Function : {:?}", arr);
   calculate_mean(arr);
}
fn calculate_mean(arr:[i32;5]){
   let mut sum = 0;
   for i in 0..5 {
       sum += arr[i];
   }
   println!("Mean of array values: {}", sum/5);
}
```



Pass by Reference

Arrays can be passed by reference in the function parameter. In other words, changes are made in the original array and no copy is made when passed by reference in the function.

The general syntax for passing an array by reference to a function is:

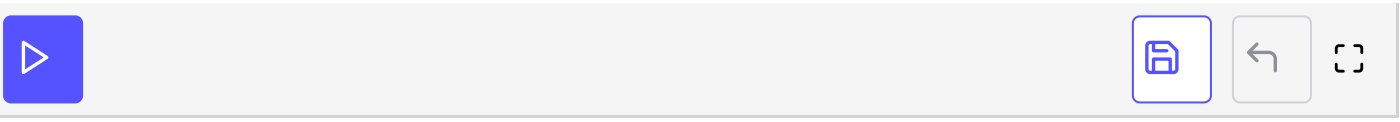
```
fn function_name(array_name:&mut [datatype;size])
```

Example

The following example takes the array `arr` by reference in the function parameter.

```
fn main() {
   let mut arr = [1, 2, 3, 4, 5];

   modify_my_array(&mut arr);
   println!("Array in Driver Function : {:?}", arr);
}
fn modify_my_array(arr:&mut [i32;5]){
   arr[2] = 8;
   arr[3] = 9;
   println!("Array in my Function : {:?}", arr);
}
```



Return an Array

Arrays can be returned from the function.

The general syntax for returning an array from a function is:

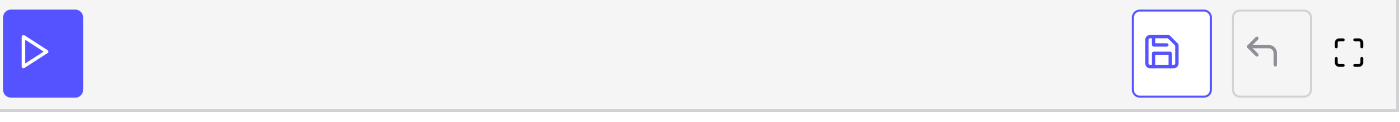
```
fn function_name()->[datatype;size]
```

> **Note:** Here the parameters can also be passed.

Example

The following example takes the array `arr`, modifies it within the function and returns it.

```
fn main() {
   let arr = [1, 2, 3, 4, 5];
   modify_my_array(arr);
   println!("Array in Driver Function : {:?}", arr);
   println!("Array after Function Call : {:?}", modify_my_array(arr));
}
fn modify_my_array(mut arr:[i32;5])->[i32;5]{
   arr[2] = 8;
   arr[3] = 9;
   arr
}
```



Now that you have learned all about functions, let's check your knowledge in the upcoming challenge.