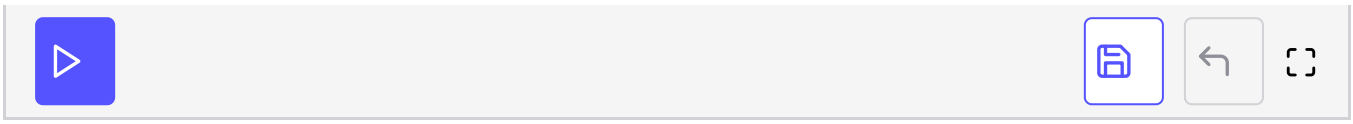
# Solution Review: Finding Fibonacci Numbers with Array

This lesson discusses the solution to the challenge given in the previous lesson.

```
package main
import "fmt"

var fib [10]int64       // global array containing Fibonacci values

func fibs() [10]int64{
        fib[0] = 1              // base cases for 0
        fib[1] = 1              // base case for 1

        for i:= 2; i <10; i++ {
                fib[i] = fib[i-1] + fib[i-2]    // recursive case without recursion using arr
        }
    return fib
}

func main() {

        arr := fibs()
        for i:=0; i < 10; i++ {
                fmt.Printf("The %d-th Fibonacci number is: %d\n", i, arr[i])
        }
}
```



Finding Fibonacci Numbers with Array

In the code above, at **line 4**, we create a global variable of type integer array called `fib` with length **10**. This array will contain the first 10 *Fibonacci* numbers. Now, look at the header of the function `fibs` at **line 6**. It returns the Fibonacci sequence in an array of type **int64**. We know that Fibonacci values for **0** and **1** both are **1**. So, we set the first two indexes of `fibs` array to **1**. Now, we have a *for* loop at **line 10**, which starts from **2** and ends at the **9** index. At **line 11**, we calculate the Fibonacci number for any value at index `i` as: `fibs[i] = fibs[i-1] + fibs[i-2]`. In the `main` function at **line 18**, we call `fibs` and store the result in separate array `arr`. Then, at the last, we have another

for loop at **line 19**, which prints all the Fibonacci values from `arr`.

---

That's it about the solution. In the next lesson, you'll study slices.