4. Arrays

- There are two types of arrays: indexed arrays and associative arrays;
- An array can be created explicitly using the keyword **declare** (and can also be initialized at the same time), followed by the option **-a** for an indexed array, and **-A** for an associative array.

Example

```
declare -a indexedArray=("one" "two" "three" "four")
declare -A associativeArray=(['one']="un" ['two']="deux"
['three']="trois")
```

4. Arrays

- The display of an array is done with the syntax **${myArray[*]}** or **${myArray[@]}**. The difference between using **@** and * is identical to that between the special shell variables **$@** and **$***

Example

```
declare -a indexedArray=("one" "two" "three" "four")
echo ${indexedArray[@]}  # Output: one two three four

declare -A associativeArray=(['one']="un" ['two']="deux"
['three']="trois")
echo ${associativeArray[@]}  # Output: un deux trois
```

4. Arrays

- The use of braces is necessary to avoid conflicts with path expansions (where square brackets have a special meaning). It is possible to obtain the list of keys of an array using the syntax **${!array[@]}**. In the case of an indexed array, you get its indices.

Example

```
declare -a indexedArray=("one" "two" "three" "four")
echo ${!indexedArray[@]}  # Output: 0 1 2 3

declare -A associativeArray=(['one']="un" ['two']="deux"
['three']="trois")
echo ${!associativeArray[@]}  # Output: one two three
```

4. Arrays

- This allows you to check the indices assigned to the values during the initialization of the array;

Example

```
declare -a indexedArray=("fry" "leela" [42]="bender" "flexo")
echo ${!indexedArray[@]}  # Output: 0 1 42 43
declare -a indexedArray=("one" "two" "three" "four")
echo ${#indexedArray[@]}  # Output: 4

declare -A associativeArray=(['one']="un" ['two']="deux"
['three']="trois")
echo ${#associativeArray[@]}  # Output: 3
```

4. Arrays

- Reading an element is done using the syntax **array[index]**, but it always requires braces:

```
array=("one" "two" "three" "four")
echo ${array[2]}  # Outputs: three
```

- For associative arrays, an index is provided in the form of a key enclosed in double quotes:

```
declare -A assocArray=( ['one']="un" ['two']="deux" ['three']="trois" )
echo ${assocArray["two"]}  # Outputs: deux
```

- For indexed arrays, a negative index corresponds to an index starting at the maximum index of the array plus one (the last element is therefore -1). If no index is provided for the array, index 0 is accessed.

5. Exercises

1. Write a Bash script that iterates through numbers from 1 to 20. For each number, if it's divisible by 3, print "Fizz". If it's divisible by 5, print "Buzz". If it's divisible by both 3 and 5, print "FizzBuzz". Otherwise, print the number itself.
2. Write a Bash script that prompts the user to guess a secret number between 1 and 100. The script should generate a random number as the secret number. The user should continue guessing until they correctly guess the secret number. After each guess, the script should provide feedback on whether the guess is too high, too low, or correct.

5. Exercises

3. Create a script that asks the user to enter a grade and displays a message based on this grade:

- "excellent" if the grade is between 16 and 20;
- "good" when it is between 14 and 16;
- "fairly good" if the grade is between 12 and 14;
- "average" if the grade is between 10 and 12;
- "insufficient" if the grade is less than 10. Ensure that the program repeats until the user enters a negative grade.

4. Create a script that calculates and displays the factorial of a given number passed as a parameter.

5. Exercises

5. The exercise involves writing a shell script to compute the sum of numbers stored in an array. Here are the steps:

- Define an array of numbers.
- Initialize a variable to store the sum.
- Loop through the array and add each number to the sum.
- Print out the sum.