**W1 -** PRACTICE

*Array Problem Solving*

## *At the end of this practice, you should be able to…*

* Access to **previous, next elements** in an array
* Be able to check if **everything is** XX or at **least something is** XX in a list of numbers
* Refactor a LOOP for **avoid breaks statements**
* Use **debugging** **techniques** to check for errors

## *How do we structure exercises?*

We organize each practice into 3 parts:

| ANALYSE | **Understand** existing codes, find the **bugs** or **complete** missing gaps |
| --- | --- |
| MANIPULATE | Ensure you can **apply the theory** with some basic challenges |
| CREATE | **Express your creativity** with more complex challenges |

## *Are you lost?*

You can read the following documentation to be ready for this practice

<https://www.w3schools.com/c/c_arrays.php>

**EX 1 (Check the occurrences)**

We want to check whether **2 arrays of 5 characters** contain the same amount of A, B and C.

| INPUT | Array of 5 characters  Array of 5 characters |
| --- | --- |
| OUTPUT | ‘Same’ OR ‘Different’ |

Examples

| INPUT | OUTPUT |
| --- | --- |
| { 'C', 'D', 'B', 'B', 'C' }  { 'C', 'C', 'B', 'B', 'N' } | Same  Explanation: Both arrays contain A (0), B (2) and C (2) |
| { 'C', 'D', 'A', 'B', 'C' }  { 'C', 'C', 'B', 'B', 'N' } | Different  Explanation: The first array contains 1 A and the second does not. |

**Q1 –** How many variables do you need for this game? *(you can add more rows…)*

| VARIABLE | TYPE | GOAL |
| --- | --- | --- |
| i | int | for the loop |
| arr | char | for user input |
| Arr | char | for user input |
| ARr | int | for counting the elements in arr |
| ARR | int | for counting the elements in Arr |
| true | int\bool | to decide if we print the output or not |

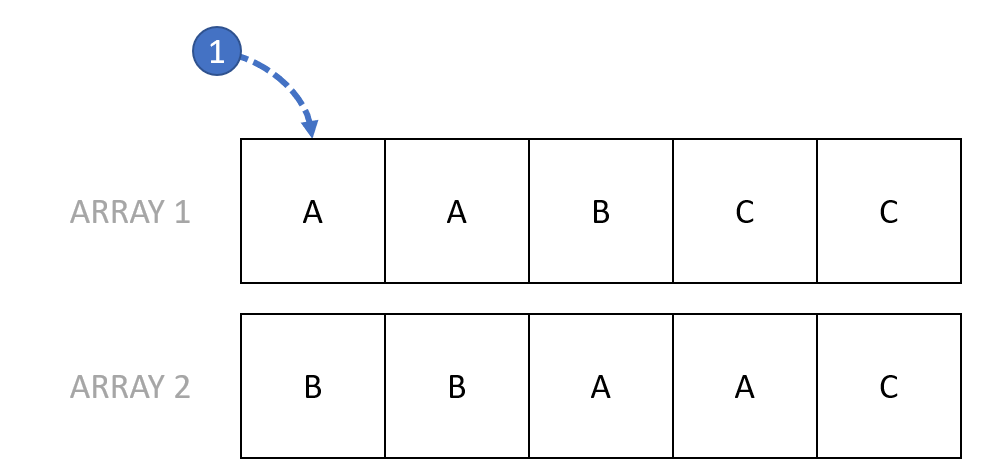
**Q2 –** How will you **iterate**? (Check the right choice and explain why)

* I will first loop on array 1 and then on array 2
* I will use only 1 loop and access to array1 and array2 elements in this loop

Why

i will loop and count both array, because it’s impossible to count 2 different array by using one loop(at least that’s what I’m thinking) i could use 2D array like arr[][] and loop it but i don’t want to do it in this case

**Q3 –** Using the bellow example, **represent your path** on each array and array elements:



*Indicate the step with a number, as show in blue*

**Q3 –** Write first the pseudo code.

//user input

set arr[5], Arr[5]

//

set ARr[3], ARR[3], true = 1;

for (set i to 0 from 0 to 4)

if arr[i] = a

ARr[0] = ARr[0] + 1;

if arr[i] = b

ARr[1] = ARr[1] + 1;

if arr[i] = c

ARR[2] = ARr[2] + 1;

for (set i to 0 from 0 to 5)

if Arr[i] = a

ARR[0] = ARR[0] + 1;

if Arr[i] = b

ARR[1] = ARR[1] + 1;

if Arr[i] = c

ARR[2] = ARR[2] + 1;

for(set i to 0 from 0 to 2)

if arr[1] is equal to Arr[i]

print(is different);

true = 0;

break;

if true

print(same)

**Q4 –** Once you are **clear with your pseudo code**, code your algorithm in C.

#include <stdio.h>

int main() {

char arr[5] = {'a' , 'a', 'b', 'b', 'c' };

char Arr[5] = {'a' , 'a', 'b', 'b', 'a' };

int ARr[3] = {0}, ARR[3] = {0}, true = 1;

for (int i= 0; i<5;i++){

if(arr[i]== 'a'){

ARr[0] += 1;

}else if(arr[i] == 'b'){

ARr[1] += 1;

}else if(arr[i] == 'c'){

ARr[2] += 1;

}

}

for (int i= 0; i<5;i++){

if(Arr[i]== 'a'){

ARR[0] += 1;

}else if(Arr[i] == 'b'){

ARR[1] += 1;

}else if(Arr[i] == 'c'){

ARR[2] += 1;

}

}

for(int i = 0; i<3; i++){

if(ARr[i] != ARR[i]){

printf("Different\nelements in arr 1 : A(%d) B(%d) C(%d)\nelements in arr 2 : A(%d) B(%d) C(%d)", ARr[0], ARr[1], ARr[2], ARR[0], ARR[1], ARR[2]);

true = 0;

break;

}

}

if (true){

printf("Same : A(%d) B(%d) C(%d)", ARr[0], ARr[1], ARr[2]);

}

return 0;

}

**EX 2 (Check the occurrences of AB)**

We have an array of 10 characters.

We want to know how many times we can identify **the occurrence of "AB"**

| INPUT | Array of 10 characters |
| --- | --- |
| OUTPUT | Number of AB |

Examples

| INPUT | OUTPUT |
| --- | --- |
| {'A','B','C','A','C','A','B','B','A','B’} | 3 |
| {'A','B','C','A','C','B','B','B','A','C’} | 1 |
| {'A','A','A','A','A','A','A','A','A','A’} | 0 |

**Q1 –** How many variables do you need for this problem? *(you can add more rows…)*

| VARIABLE | TYPE | GOAL |
| --- | --- | --- |
| found | int | for counting the amount of a is next to b |
| i | int | for loop |
| arr | char | for user input |

**Q2 –** How do you plan to loop on the array elements? Complete the table

| I will **start** at index | i = 0 |
| --- | --- |
| I will **increment** of | i++ |
| I will **stop** at index | i < 9 |

**Q3 –** Write first the pseudo code.

set fount to 0

set arr[10]

user input

for (set i to 0 from 0 to 8)

if (arr[i] is equal to a and arr[i+1] is equal to b)

found is equal to found plus 1

print found

**Q4 –** Once you are **clear with your pseudo code**, code your algorithm in C.

#include <stdio.h>

int main(){

int found = 0;

char arr[10];

printf("Enter 10 char : ");

scanf("%10s", arr);

for(int i = 0; i < 9; i++){

if (arr[i] == 'a' && arr[i +1] == 'b'){

found += 1;

}

}

printf("The amount of a is next to b is : %d\n", found);

return 0;

}

**EX 3 (Check the occurrences of ABC)**

We have an array of 10 characters.

We want to know how many times we can identify **the occurrence of "ABC"**

| INPUT | Array of 10 characters |
| --- | --- |
| OUTPUT | Number of ABC |

Examples

| INPUT | OUTPUT |
| --- | --- |
| {'A','B','C','A','C','A','B','B','A','B’} | 1 |
| { 'A','B','C', 'A','B','C', 'A','B','C' } | 3 |
| {'A','A','A','A','A','A','A','A','A','A’} | 0 |

**Q1 –** How many variables do you need for this game? *(you can add more rows…)*

| VARIABLE | TYPE | GOAL |
| --- | --- | --- |
| i | int | for the loop |
| arr | char | for the user input |
| found | int | to count the amount of abc that’s next to each other |

**Q2 –** How do you plan to loop on the array elements? Complete the table

| I will **start** at index | i = 0 |
| --- | --- |
| I will **increment** of | i++ |
| I will **stop** at index | i < 8 |

**Q3 –** Write first the pseudo code.

set fount to 0

set arr[10]

user input

for (set i to 0 from 0 to 7)

if (arr[i] is equal to a and arr[i+1] is equal to b and arr[i + 2] is equal to c)

found is equal to found plus 1

print found

**Q4 –** Once you are **clear with your pseudo code**, code your algorithm in C.

#include <stdio.h>

int main(){

int found = 0;

char arr[10];

printf("Enter 10 char : ");

scanf("%10s", arr);

for(int i = 0; i < 8; i++){

if (arr[i] == 'a' && arr[i +1] == 'b'&& arr[i + 2] == 'c'){

found += 1;

}

}

printf("The amount of a is next to b and c is : %d\n", found);

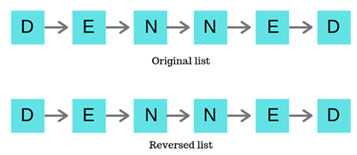
return 0;

}

**EX 4 (palindromic list)**

We want to know if an array of 6 numbers is palindromic.

A sequence is palindromic if it reads the same forwards as backwards



| INPUT | 10 numbers (int []) |
| --- | --- |
| OUTPUT | True or False *(palindromic or not)* |

Example

| INPUT | OUTPUT |
| --- | --- |
| {2, 5, 8, 8, 5, 2} | True |
| {2, 5, 8, 7, 5, 2} | False |
| {1, 1, 1, 1, 1, 1} | True |

**Q2 –** How do you plan to **loop** on the array elements? Complete the table

| I will **start** at index | i = 0 |
| --- | --- |
| I will **increment** of | i++ |
| I will **stop** at index | i < 5 |

**Q3 –** Write first the pseudo code.

* Test it using the following example: {2, 5, 8, 8, 5, 2}

**Q4 –** Once you are **clear with your pseudo code**, code your algorithm in C.

#include <stdio.h>

int main(){

int found = 1;

char arr[10];

printf("Enter 10 integers : ");

scanf("%10s", arr);

for(int i = 0; i < 5; i++){

if (arr[i] != arr[9 - i]){

printf("This array is not palindromic\n");

found = 0;

break;

}

}

if (found){

printf("This array is palindromic\n");

}

return 0;

}

**EX 5 (Sum of the 2 previous elements)**

We have as input an array of 6 numbers.

We want to check if all numbers (starting from the 3rd one) are equal to the sum of the 2 previous elements.

If yes the array is valid, if not the array is not valid.

| **INPUT** | int [6] | The array of 6 numbers |
| --- | --- | --- |
| **OUTPUT** | bool | VALID  *(if all elements are equal to the sum of their 2 predecessors)*  NOT VALID  *(otherwise)* |

Examples

| **INPUT** | **OUTPUT** | **EXPLANATION** |
| --- | --- | --- |
| {1, 2, 3, 5, 8, 13} | VALID | 1 + 2 = 3  2 + 3 = 8  … |
| {1, 2, 3, 5, 8, 14} | NOT VALID | 5 +8 != 14 |
| {10, 2, 12, 14, 26, 40} | VALID | 10 + 2 = 12  2 + 12 = 14  … |

#include <stdio.h>

int main(){

int found = 1;

int arr[10];

printf("Enter 10 integers :\n");

for(int i = 0; i< 10; i++){

scanf("%d", &arr[i]);

}

for(int i = 1; i < 9; i++){

if (arr[i] != arr[i-1] + arr[i-2]){

printf("Not Valid\nbecause %d + %d is not equal to %d\n", arr[i-2], arr[i-1], arr[i]);

found = 0;

break;

}

}

if (found){

printf("Valid\n");

}

return 0;

}