**rW6 -** PRACTICE

*Pointers*

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

* Understand memory addresses and pointers
* Declare and initialize pointers
* Use the address (&) and indirection (\*) pointer operators
* Use pointers arithmetic to iterate through arrays
* Use pointers to pass arguments to functions by reference

## *How do we structure exercises?*

We organize this practice into 4 parts:

| GAME | Play a serious game to **grasp the concepts** |
| --- | --- |
| 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

To understand the basics:

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

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

<http://w3schools.com/c/c_pointers_arrays.php>

To learn a bit more about pointer and references:

<https://www.shiksha.com/online-courses/articles/difference-between-pointer-and-reference-blogId-155435>

**GAME**

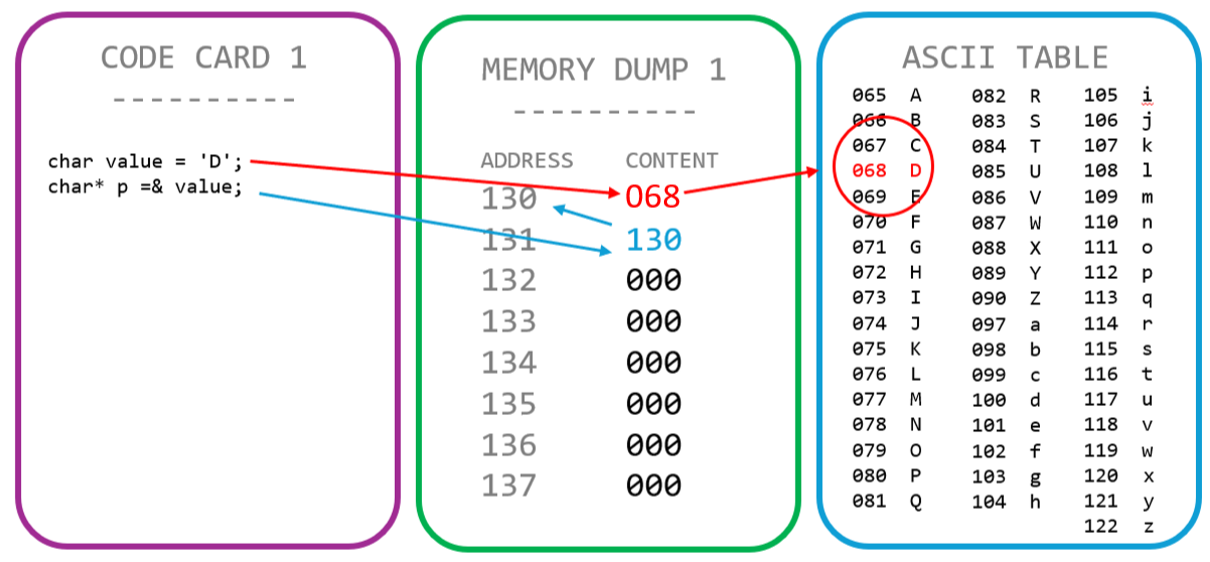
**THE MEMORY DUMP BATTLE**

*Teams of 3–4 players ! 5 Rounds*

**Goal**

Match a **card** code to a **memory dump card.**

*The memory dump shows how characters, arrays, and pointers affect memory at the end of the code execution*



**Some tips…**

* We visualize memory as 8 bytes (addresses 000-007)
* Each memory byte can contain either:
* A char (ASCII value)
* A pointer (an address they point to)

**ANALYSE**

**EX 1** **(From Code to Memory Dump)**

Look at the bellow code.

char src[] = {'h', 'e', 'l', 'l', 'o'};

char dest[5];

char i;

char\* ps = src;

char\* pd = dest;

for (i = 0; i < 5; i = i + 1) {

\*(pd + i) = \*(ps + i) + 1;

}

**QUESTION –** Complete the memory dump **at the end** of the code execution (first rows provided...)

| ADDRESS | CONTENT | COMMENT |
| --- | --- | --- |
| 100 | 104 | ‘h’ |
| 101 | 101 | ‘e’ |
| 102 | 108 | ‘l’ |
| 103 | 108 | ‘l’ |
| 104 | 111 | ‘o’ |
| 105 | 105 | ‘i’ |
| 106 | 102 | ‘f’ |
| 107 | 109 | ‘m’ |
| 108 | 109 | ‘m’ |
| 109 | 112 | ‘p’ |
| 110 | 060 | 4 |
| 111 | 100 | src[] = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’} |
| 112 | 105 | dest[] = {‘i’, ‘f’, ‘m’, ‘m’, ‘p’} |

**EX 2** **(Fix Buggy Code)**

Look at the bellow code.

1 int a = 42;

2 int b = 10;

3 int\* p = a;

4 int\* q = &b;

5 \*p = &b;

6 \*q = \*a + \*q;

**QUESTION –** you need to find 3 **bugs !** Complete the table below

| BUG LINE | BUG DESCRIPTION | FIX |
| --- | --- | --- |
| 3 | pointer p point to value a | \*p = &a; |
| 5 | value of pointer p is the address of b? | delete it |
| 6 | there’s no pointer of a | \*q = \*p + \*q |

**EX 3** **(Fix Buggy Code)**

Look at the bellow code.

1 int x = 5;

2 int y = 8;

3 const int\* ptr = &x;

4 \*ptr = 10;

5 ptr = &y;

6 int\* const p2 = &y;

6 p2 = &x;

6 \*p2 = \*ptr + 1;

**QUESTION –** you need to find 2 **bugs !** Complete the table below

| BUG LINE | BUG DESCRIPTION | FIX |
| --- | --- | --- |
| 4 | can’t change the value of the pointer because it is a constant value | delete const on line 3 |
| 7 | can’t change the value of the pointer because we declared it as a constant value | delete sonst |

**MANIPULATE**

**EX 1 (THE SWAP FUNCTION)**

In this exercise, you will implement a function that **swaps** two integers using pointers.

| FUNCTION NAME | swap | |
| --- | --- | --- |
| FUNCTION DESCRIPTION | Swap 2 values, by passing the value by reference to the function | |
| PARAMETERS | Int\* | The first value |
| Int\* | The second value |
| RETURN | void |  |

As an output, print the memory address and value of the 2 variables (before and after the swap

Before the swap

X = 10 (address 0x100200)

Y = 20 (address 0x100200)

Before the swap

X = 20 (address 0x100200)

Y = 10 (address 0x100200)

*#include <stdio.h>*

*void swap(int\* firstValue, int\* secValue){*

*int temp = \*firstValue;*

*\*firstValue = \*secValue;*

*\*secValue = temp;*

*}*

*void print(int x, int y){*

*printf("%d\t(%p)\n%d\t(%p)\n", x, &x, y, &y);*

*}*

*int main (){*

*int x = 10, y = 20;*

*printf("Before swap :\n");*

*print(x, y);*

*swap(&x, &y);*

*printf("After swap :\n");*

*print(x, y);*

*return 0;*

*}*

**EX 2 (THE LOG FUNCTION)**

Write a program in C to print all elements in a array and their memory address.

You cannot use [] to access to elements

| FUNCTION NAME | log | |
| --- | --- | --- |
| FUNCTION DESCRIPTION | Print all array elements and their address | |
| PARAMETERS | Int\* | Pointer to the array |
| Int | Array size |
| RETURN | void |  |

**#include <stdio.h>**

**void logFunction (int\* ptrArr, int size){**

**for(int i = 0; i < size ; i++){**

**printf("%d\t(%p)\n", \*(ptrArr + i), (ptrArr + i));**

**}**

**}**

**int main() {**

**int size = 0;**

**scanf("%d", &size);**

**int arr[size];**

**for(int i = 0; i < size ; i++){**

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

**}**

**int\* ptrArr = arr;**

**logFunction(ptrArr, size);**

**return 0 ;**

**}**