**Lab: Understanding Arrays and Pointers in C**

**Objective:**

This lab will focus on the relationship between arrays and pointers, memory allocation for pointers, and working with double pointers for two-dimensional arrays.

**1. Arrays and Pointers: Similarities and Differences**

**Concept:**

* An array name acts like a pointer to the first element.
* Pointer arithmetic can be used to traverse arrays.
* Arrays have a fixed size, while pointers can be dynamically allocated.

**Task:**

* Create an integer array and traverse it using both array indexing and pointer arithmetic.

c

#include <stdio.h>

int main() {

int arr[] = {10, 20, 30, 40, 50};

int \*ptr = arr; // Pointer to first element of array

printf("Using array indexing:\n");

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

printf("%d ", arr[i]);

}

printf("\nUsing pointer arithmetic:\n");

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

printf("%d ", \*(ptr + i));

}

return 0;

}

**2. Array of Pointers and Allocating Memory for Pointers**

**Concept:**

* An array of pointers stores addresses instead of values.
* Each pointer in the array can point to different locations.

**Task:**

* Create an array of char\* pointers and allocate memory for each pointer.

c

#include <stdio.h>

#include <stdlib.h>

int main() {

char \*arr[3]; // Array of pointers

// Allocating memory for each pointer

arr[0] = (char\*)malloc(10 \* sizeof(char));

arr[1] = (char\*)malloc(10 \* sizeof(char));

arr[2] = (char\*)malloc(10 \* sizeof(char));

// Assigning values

snprintf(arr[0], 10, "Hello");

snprintf(arr[1], 10, "World");

snprintf(arr[2], 10, "C");

// Printing values

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

printf("%s\n", arr[i]);

free(arr[i]); // Free allocated memory

}

return 0;

}

**3. Pointer to Array and Pointer Arithmetic**

**Concept:**

* A pointer to an array stores the address of an entire array.
* Pointer arithmetic works differently compared to normal pointers.

**Task:**

* Declare a pointer to an array and use pointer arithmetic to access elements.

c

#include <stdio.h>

int main() {

int arr[5] = {1, 2, 3, 4, 5};

int (\*ptr)[5] = &arr; // Pointer to an array of 5 integers

printf("First element using pointer to array: %d\n", (\*ptr)[0]);

printf("Second element using pointer arithmetic: %d\n", \*(\*ptr + 1));

return 0;

}

**4. Allocating Memory for Pointer to an Array**

**Concept:**

* Dynamically allocate memory for an array and assign it to a pointer.

**Task:**

* Allocate memory for an integer array using malloc() and store the pointer.

c

#include <stdio.h>

#include <stdlib.h>

int main() {

int \*arr = (int\*)malloc(5 \* sizeof(int));

if (arr == NULL) {

printf("Memory allocation failed\n");

return 1;

}

// Assign values

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

arr[i] = i \* 10;

}

// Print values

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

printf("%d ", arr[i]);

}

free(arr); // Free memory

return 0;

}

**5. Double Pointers and Allocating a 2D Array**

**Concept:**

* A double pointer (\*\*ptr) can be used to dynamically allocate a 2D array.

**Task:**

* Allocate memory for a 2D array using double pointers.

c

#include <stdio.h>

#include <stdlib.h>

int main() {

int rows = 3, cols = 4;

int \*\*arr;

// Allocate memory for row pointers

arr = (int\*\*)malloc(rows \* sizeof(int\*));

// Allocate memory for each row

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

arr[i] = (int\*)malloc(cols \* sizeof(int));

}

// Assign values

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

for (int j = 0; j < cols; j++) {

arr[i][j] = i \* cols + j;

}

}

// Print values

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

for (int j = 0; j < cols; j++) {

printf("%d ", arr[i][j]);

}

printf("\n");

}

// Free allocated memory

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

free(arr[i]);

}

free(arr);

return 0;

}

**Lab Questions**

1. How does the behavior of an array name differ from a pointer?
2. What happens if we try to modify the base address of an array?
3. What is the significance of (\*ptr)[size] notation when working with a pointer to an array?
4. Why do we need to allocate memory dynamically when using an array of pointers?
5. Explain the difference between a single pointer (\*ptr), a pointer to an array ((\*ptr)[size]), and a double pointer (\*\*ptr).
6. What happens if we forget to free dynamically allocated memory?