**1D, 2D, MultiDimensional Array Assignments**

**1. Refer the code snippet and answer the queries**

**int main()**

**{**

**int array[100];**

**int \*ptr;**

**// do something**

**}**

**Q1: Can pointer be used in Array-style syntax? e.g. ptr[10], ptr[0]**

**Ans:** Yes, pointers can be used with array-style syntax. For example, ptr[10] is equivalent to \*(ptr + 10).

**Q2: Can Array be used in Pointer-style syntax? e.g. \*array, \*(array + 0), \*(array + 10)**

**Ans:** Yes, arrays can be used with pointer-style syntax. For example, \*array is equivalent to array[0], and \*(array + 10) is equivalent to array[10].

**Q3: is ptr++ valid?**

**Ans:** Yes, ptr++ is valid. It increments the pointer to point to the next element in the array it is pointing to.

**Q4: is array++ valid?**

**Ans:** No, array++ is not valid. Arrays are not modifiable lvalues, meaning you cannot change the address of the array itself.

**Q5: what is sizeof(array)?**

**Ans:** sizeof(array) will give you the total size of the array in bytes. Since array is an array of 100 integers and assuming int is 4 bytes, sizeof(array) would be 100 \* 4 = 400 bytes.

**Q6: what is sizeof(ptr)?**

**Ans:** sizeof(ptr) will give you the size of the pointer itself, not the size of the array it points to. On most systems, this is typically 4 or 8 bytes, depending on whether the system is 32-bit or 64-bit.

**2. Refer the code snippet below. Comment on the other elements (other than those that are explicitly initialized) of all array variables in code snippet below.**

**#define MAX 100**

**int main()**

**{**

**int arr[MAX] = {11,22,33};**

**int arr1[MAX]={0};**

**static int arr2[MAX];**

**}**

**Ans:** 1. int arr[MAX] = {11, 22, 33};

* + The first three elements of arr are explicitly initialized to 11, 22, and 33.
  + The remaining elements (from arr[3] to arr[99]) are implicitly initialized to 0.

1. int arr1[MAX] = {0};
   * The first element arr1[0] is explicitly initialized to 0.
   * The remaining elements (from arr1[1] to arr1[99]) are implicitly initialized to 0.
2. static int arr2[MAX];
   * Since arr2 is declared as a static array, all elements are implicitly initialized to 0 by default.

**3. Refer the program “array\_pointer.c”. Add a function getmax() to find the maximum in the array and call in main() and display the result.**

Ans: #include <stdio.h>

#define MAX 100

// Function to find the maximum value in an array

int getmax(int \*array, int size) {

int max = array[0];

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

if (array[i] > max) {

max = array[i];

}

}

return max;

}

int main() {

int arr[MAX] = {11, 22, 33, 44, 55}; // Example array with some values

int size = 5; // Number of elements initialized in the array

// Call getmax() and display the result

int max\_value = getmax(arr, size);

printf("The maximum value in the array is: %d\n", max\_value);

return 0;

}

**4. Extend the code given below to read N and a start value from the user to perform the given operations.**

**#define MAX 100**

**int main()**

**{**

**int arr[MAX] = {11,22,33};**

**}**

**Add the following functions choosing proper input, output and return.**

**a. init() - Use the inputs to initialize the first N elements of the array with N consequetive values starting with given start value .**

**b. update() – increment value of every element in the array**

**c. display() – display the contents of array**

Ans**:** #include <stdio.h>

#define MAX 100

// Function to initialize the first N elements of the array

void init(int \*array, int N, int start\_value) {

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

array[i] = start\_value + i;

}

}

// Function to increment the value of every element in the array

void update(int \*array, int size) {

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

array[i]++;

}

}

// Function to display the contents of the array

void display(int \*array, int size) {

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

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

}

printf("\n");

}

int main() {

int arr[MAX] = {0}; // Initialize array with zeros

int N, start\_value;

// Read N and start value from the user

printf("Enter the number of elements to initialize (N): ");

scanf("%d", &N);

printf("Enter the start value: ");

scanf("%d", &start\_value);

// Initialize the array

init(arr, N, start\_value);

// Display the array after initialization

printf("Array after initialization: ");

display(arr, N);

// Update the array

update(arr, N);

// Display the array after update

printf("Array after update: ");

display(arr, N);

return 0;

}

**2D, MultiDimensional Arrays**

**1. Implement sort() to sort a given array. Refer the code snippet below.**

**int main()**

**{**

**char arr[]= “xaybz”;**

**sort(arr, sizeof(arr)/sizeof(arr[0]);**

**return 0;**

**}**

Ans: #include <stdio.h>

#include <string.h>

// Function to sort a given array

void sort(char \*array, int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = i + 1; j < size; j++) {

if (array[i] > array[j]) {

// Swap array[i] and array[j]

char temp = array[i];

array[i] = array[j];

array[j] = temp;

}

}

}

}

int main() {

char arr[] = "xaybz";

int size = sizeof(arr) / sizeof(arr[0]) - 1; // Subtract 1 to exclude the null terminator

sort(arr, size);

printf("Sorted array: %s\n", arr);

return 0;

}

**2. Refer the code snippet below.**

**int main()**

**{**

**char arr[][3] = {**

**sort(arr, sizeof(arr)/sizeof(arr[0]);**

**return 0;**

**}**

**Allow user to perform the following operations.**

**a. init() - initialize the array and return 0**

**b. search\_update() – search for a given element in array and if found update it to given value and return 0 else return 1**

**c. display() – traverse and display array contents**

**For the functions, pass array and other required arguments to functions and return as per requirement**

Ans: #include <stdio.h>

#include <string.h>

// Function to initialize the array

void init(char array[][3], int rows) {

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

printf("Enter 3 characters for row %d: ", i + 1);

scanf("%s", array[i]);

}

}

// Function to search for an element and update it if found

int search\_update(char array[][3], int rows, char search, char update) {

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

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

if (array[i][j] == search) {

array[i][j] = update;

return 0; // Element found and updated

}

}

}

return 1; // Element not found

}

// Function to display the contents of the array

void display(char array[][3], int rows) {

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

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

printf("%c ", array[i][j]);

}

printf("\n");

}

}

int main() {

char arr[5][3]; // Example array with 5 rows

int rows = sizeof(arr) / sizeof(arr[0]);

// Initialize the array

init(arr, rows);

// Display the array after initialization

printf("Array after initialization:\n");

display(arr, rows);

// Search and update an element

char search, update;

printf("Enter the character to search for: ");

scanf(" %c", &search);

printf("Enter the character to update it with: ");

scanf(" %c", &update);

int result = search\_update(arr, rows, search, update);

if (result == 0) {

printf("Element found and updated.\n");

} else {

printf("Element not found.\n");

}

// Display the array after update

printf("Array after update:\n");

display(arr, rows);

return 0;

}