**ARRAYS**

**1D Array**

**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]**

**A:** Yes, pointer can be used in array-style syntax. If ptr is pointing to a valid memory location, we can use ptr[10] to access the 10th element from the memory location.

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

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

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

**A:** Yes, ptr++ is valid. This increments the pointer ptr to point to the next memory location.

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

**A:** No, array++ is not valid. In C, the name of an array is a constant pointer to its first element, and we cannot change the address of a constant pointer.

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

**A:** sizeof(array) returns 400. The array array has 100 elements of type int, the size of an int is 4 bytes so the total size is 100 \* 4 = 400 bytes.

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

**A:** sizeof(ptr) returns 8. The pointer ptr is of type int \*, which occupies 8 bytes.

**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];**

**}**

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

* Elements arr[0] = 11, arr[1] = 22, arr[2] = 33.
* All other elements (arr[3] to arr[99]) are implicitly initialized to 0

2. arr1[MAX] = {0};

* All elements of arr1 (arr1[0] to arr1[99]) are initialized to 0.

3. static int arr2[MAX];

* All elements of arr2 (arr2[0] to arr2[99]) are initialized to 0 because arr2 is static and static arrays are zero-initialized 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.**

**A:**

**A screen shot of a computer program

Description automatically generated**

**OUTPUT:**

**A black screen with white text

Description automatically generated**

**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**

**A:**

**A screenshot of a computer program

Description automatically generated**

**OUTPUT:**

**A black screen with white text

Description automatically generated**

**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;**

**}**

**A: A computer screen shot of a program code

Description automatically generated**

**OUTPUT:**

****

**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**

**A:** **A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**OUTPUT:**

**A screen shot of a computer

Description automatically generated**