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

**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, a pointer can be used in array style syntax. Here ptr[10] is equal to \*(ptr+10), means accessing the value at the memory location ptr+10,

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

A)Yes, a Array be used in Pointer-style syntax. Here \*array is equal to array[0],

Dereferencing the first element in the array.

\*(array + 0) is the same as \*array (dereferencing the first element).

\*(array + 10) is equivalent to array[10], accessing the 11th element of the array.

Q3: is ptr++ valid?

A)Yes, ptr++ is valid.

The ptr++ expression increments the pointer ptr so that it points to the next element in memory The increment works because ptr is a pointer to int, and incrementing it moves it to the next int in memory, which is typically 4 bytes ahead .

Q4: is array++ valid?

A)No, array++ is not valid. The name of an array, Here it is array is a constant pointer to the first element of the array, and its address cannot be changed. This means you cannot increment the array itself. The expression array++ would attempt to modify this constant pointer, which is not allowed.

Q5: what is sizeof(array)?

A)The sizeof operator returns the size of the array in bytes.

sizeof(array) = 100 \* sizeof(int) = 100 \* 4 = 400 bytes

Q6: what is sizeof(ptr)?

A) ptr is a pointer to an int, so sizeof(ptr) gives the size of the pointer itself, not the size of the data it points to. The size of the ptr is 4 bytes or 8bytes.

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()

{

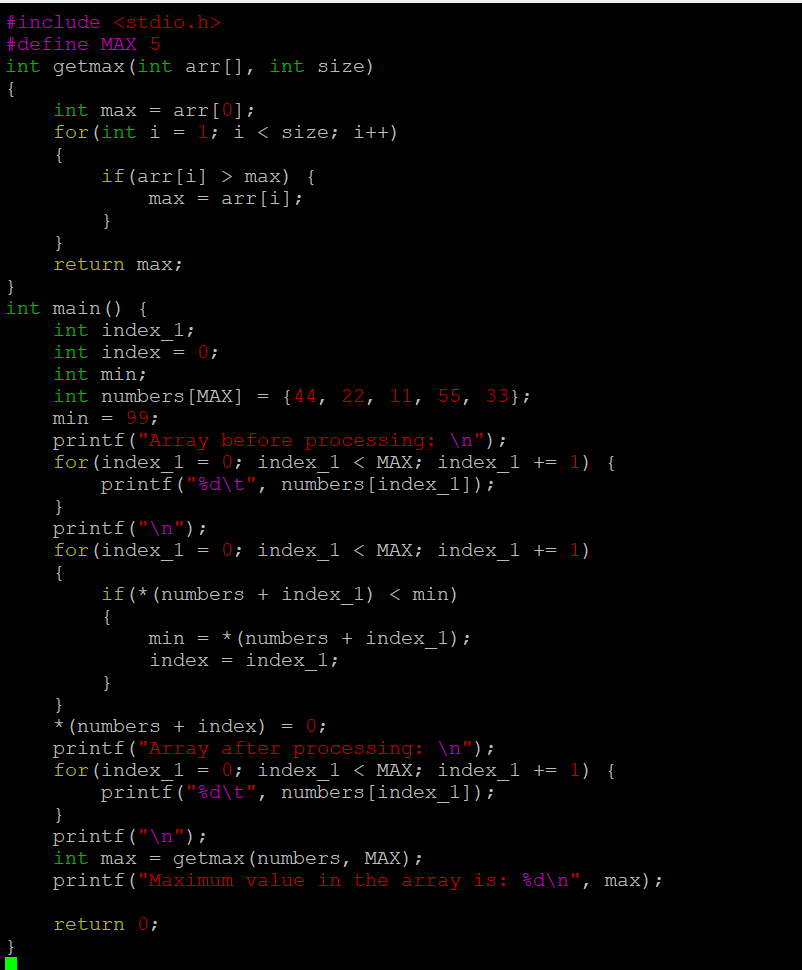
int arr[MAX] = {11,22,33};

int arr1[MAX]={0};

static int arr2[MAX];

}

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 screen shot of a computer

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 arrayA computer screen shot of a program

Description automatically generated

A screen shot of a computer

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 computer screen shot of a program code

Description automatically generated

A screen shot of a computer program

Description automatically generated

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 screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated