Array Initialization Zeroed

int main()
{
   int arr[10] = {0};
   for(int x:arr){cout<<x<<" ";}
   return 0;
}
does this opint zero in al the values of array?

Yes, the line `int arr[10] = {0};` **initializes all elements of the array** `arr` **to zero**.

Explanation:

In C++, when you use **aggregate initialization** like this:

```cpp
int arr[10] = {0};
```

It means:

- The **first element** `arr[0]` is explicitly set to 0.
- The **rest of the elements** are **value-initialized**, which means they are also set to **0**.

Output:

Your code:

```cpp
int arr[10] = {0};
for(int x : arr) { cout << x << " "; }
```

Will output:

```
0 0 0 0 0 0 0 0 0 0
```

So yes, it **initializes all values of the array to zero**.

but what if I do arr[10] = {1] so will all the elements be initialized to 1?

Good question, Vaibhav!

If you do this:

```cpp
int arr[10] = {1};
```

Then:

- `arr[0]` will be initialized to **1** (the first value you specified).
- **All other elements** (`arr[1]` to `arr[9]`) will be initialized to **0** — **not 1**.

This is how aggregate initialization works in C++. Only the explicitly listed values are used, and the rest are default-initialized to 0.

Example:

```cpp
int arr[10] = {1};
for(int x : arr) { cout << x << " "; }
```

Output:

```
1 0 0 0 0 0 0 0 0 0
```

If you want all elements to be `1`:

You need to do it manually, like:

```cpp
int arr[10];
for(int i = 0; i < 10; ++i) {
```

```
    arr[i] = 1;
}
```

Or use `fill` from the STL:

```cpp
std::fill(arr, arr + 10, 1);
```

Let me know if you want sample code with input/output too!