10/02/2023

Arrays

A baggage which contains only similar type of elements. Array is a data structure.

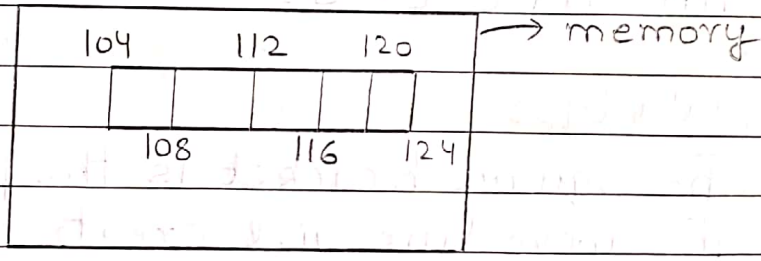
Data structure is an entity in which the data is stored. It is possible that way of storing data is same but at the end data is only getting stored.

If we can make an array of integers, then only integer values can be stored. No other data type such as float, char, string etc. can be stored in the integer array.

Behind the scene

array of 5 integers → 5 × 4 = 20 bytes.

The values of array are stored at contiguous memory locations.



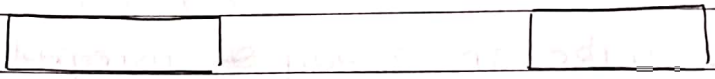
Why is array needed?

One application of array can be like if we want to find maximum number from n numbers, then we can make an array of n size and then find the maximum. This will easier than creating n variables & then comparing. This is one of the application of array & there are many other applications as well.

int arr [1000];
The above line is the syntax of creating an array of 1000 integers. By this we have reserved space for 1000 integers.

Is there memory wastage in case of contiguous allocation ? (Internal fragmentation in OS)

int arr [10]; → 4×10 = 40 bytes



24 byte 16 byte

Total memory available = 24 + 16 = 40 byte but it is not continuous, it is discrete & hence we can't make the array of 10 integers. Hence this is memory wastage.

Creating an array

variable name
int arr [10];
↑ ↑ size of array
data type

The square bracket is the part of syntax. The above line will create a continous memory location of 40 bytes.

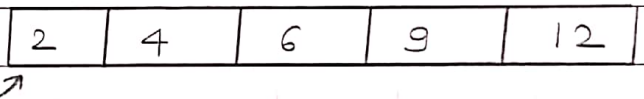
We have now known the declaration part of array i.e how to declare the array.

Note→ First address of array is also known as the base address.

```
int arr [7];
cout << arr << endl;
cout << &arr << endl;
```
> Both gives same result i.e the base address in hexadecimal value. This is from concept of pointers.

Initialization part
This means to put the values in the array.

```
int arr [ ] = {2, 4, 6, 9, 12};
```



arr
(Base address)

Also we can put size inside the square brackets during initialization also.

```
int arr [5] = {2, 4, 6, 9, 12};
int arr1 [10] = {2, 4, 6, 9, 12};
```
→ In this rest of 5 values will be 0.

```
int arr2 [4] = {2, 4, 6, 9, 12};
```
→ This will give an error as we have put more no. of elements than the initialized size.

All the above arrays are static arrays i.e fixed sized arrays. There are dynamic arrays also which we will cover later.

Note →
```
int n;
cin >> n;
int arr [n];
```
This will run on some compilers but this is a bad practice as there is a limit to store elements.

Indexing in arrays

int arr [5] = {10, 20, 30, 40, 50}
The array elements are accessed with the help of indexes.

10	0th index
20	1st index
30	2nd index
40	3rd index
50	4th index

Note → In arrays, 0-based indexing is done. If our array is of size = n, then index will lie from 0 to n-1 (both inclusive).

cout << arr [0]; → 10 is the output

Formulae

Significance of 0th index → (Base address + index * size of data ty

arr [0] → arr + 0 x 4 = arr → This means that give the value present at address arr which is the base address. Hence value present at the address arr is 10.

arr [1] → arr + 1 x 4 = arr + 4 → This means the next memory location to arr & hence 20 will be printed.

Printing the array values using for loop
Now if we have say 100 size array, so we won't be writing arr [0], arr [1] .. arr [10

We use for loop in printing array values.

```
for (int i=0; i<5; i++){
        cout << arr [i] << " ";
}
```

Output
10 20 30 40 50

Initialization of array using for loop

```
int arr [10];
cout<<"Enter input values in array "<<endl;
for (int i=0; i<10; i++){
        cin>> arr [i];
}
```

Ques 1 Take 5 elements input in array and print their doubles.

Ans 1

```
int arr [5];
cout << "Enter array values" <<endl;
for (int i=0; i<5; i++){
        cin >> arr [i]; → Taking i/p in array
}
for (int i=0; i<5; i++){
        cout << 2 * arr [i] << " ";
}
```

↳ Printing double of each number / array value.

Ques 2 Change all the array values to 1.

Ans 2

```
int arr [5] = {1, 2, 3, 4, 5};
for (int i=0; i<5; i++){
        arr [i] = 1;
}
```

```
for (int i=0; i<5; i++) {
        cout << arr[i] << " ";
}
```

Output

1 1 1 1 1

Note → (i) int arr[10]; → Initialized with garbage value
(ii) int arr[10] = {0}; → All values are initialized with 0.

Initializing array with a particular value
We use the memset function here.

Arrays and functions

In pass by value concept, copy is created & operations are performed on the copy & not the actual variable.

It is important to note that whenever the array is passed to function, address is passed & hence the operation will be performed on the actual array & not the copy. This concept is known as pass by reference.

Note → We have to follow a practice that whenever we pass array to the function then we need to pass the size along with it in the function.

Exc →
```
void incr (int arr[], int size) {
        arr[0] = arr[0] + 10;
}
```

```
int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    cout << arr[0] << endl;
    incr(arr, 5);
    cout << arr[0] << endl;
    return 0;
}
```

Output

1

11 → Actual array is updated.

Note→ Pass by reference means that the address is passed & hence a seperate copy is not created.

Why size to be passed along with array in functions?

arr[10] = {1, 2, 3};

$\frac{\text{size of (arr)}}{\text{size of (arr[0])}} = \frac{40}{4} = 10$ but no. of elements in the array is three, that's why we need to pass the size explicitly.

Linear Search in Array

int arr[ ] = {2, 9, 6, 7, 4, 12, 15};

We need to find whether 6 is present in arr or not?

We go to each index & see whether it is 6 or not.

(i) If not, move to next index & repeat

(ii) If present, then terminate search.

Terminate search if all the array elements are

checked.

Code

```
bool findElement (int arr[], int size, int key)
{
    for (int i=0; i< size; i++) {
        if (arr[i] == key) {
            return true;
        }
    }
    return false; → If not found even after
}                   traversing whole array.
```

Ques 3 Count 0s and 1s in array.
Just traverse the array & if element is zero, then do zeroCount ++ & if current element is one, then do oneCount ++;

Code

```
int arr[5] = {1,0,0,1,0};
int zeroCount = 0;
int oneCount = 0;

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

        if (arr[i] == 0)
            zeroCount ++;
        else → As our array only have 0 & 1.
            oneCount ++;
}
```

```
cout << "No. of zeroes are " <<zeroCount <<endl;
cout << "No of ones are " << oneCount <<endl;
```

Ques 4 Find maximum number in the array
Assume that maxi = arr[0] and traverse the array from i=1 to n-1 & compare with maxi & if element is greater than maxi, then update maxi. At the end print maxi which is the maximum number.

Note → Best practice is to initialize maxi with INT_MIN when finding maximum number in the array.

Code

```
int arr[5] = {3, 1, 2, 5, 4};
int maxi = INT_MIN;
for (int i=0; i<5; i++) {
        if (arr[i] > maxi) {
                maxi = arr[i];
        }
}
cout << "Maximum number is " << maxi;
```

Note → To use INT_MIN & INT_MAX, we need to include limits.h header file. This assumption of taking INT_MIN as initial maximum will work in all the cases.

Ques 5 Find out the minimum number in array.
Same logic as that of finding a maximum number in array.

Code

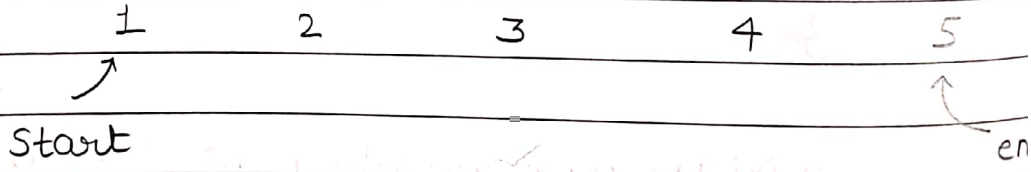
```
int arr[5] = { 3, 1, 2, 5, 4 };
int mini = INT_MAX;
for (int i=0; i<5; i++) {

        if (mini > arr[i]) {
            mini = arr[i];
        }
}
cout << "Minimum number is " << mini;
```

Ques 6 Extreme prints in array

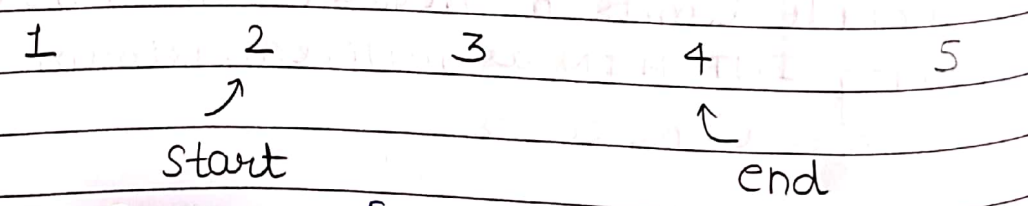
```
int arr[5] = { 1, 2, 3, 4, 5 };
```

O/p → 1 5 2 4 3

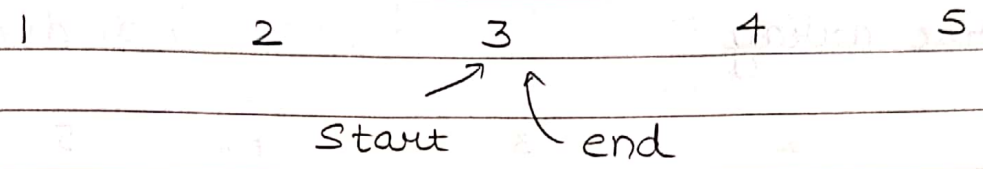
This is question of 2 pointer approach
Here take pointer as variable.



1) Print arr[start], arr[end].
start++, end--



2) Print arr[start], arr[end]
start++, end--



Now it gets printed twice. Just put a condition of start == end, then print once.

We have to stop when start goes beyond end.

Code

```
int arr [5]= { 1, 2, 3, 4, 5 };
int start = 0;
int end = 4;
while (start <= end) {
        if (start == end) {
                cout << arr [start] << " ";
                break;
        }
        cout << arr [start] << " ";
        cout << arr [end] << " ";
        start ++;
        end --;
}
```

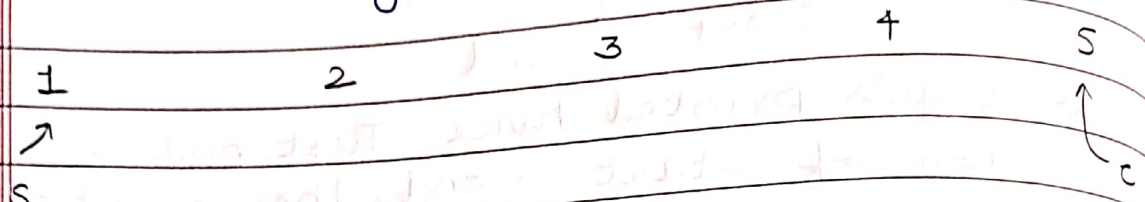
Ques 7 Reverse an array

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

o/p → { 5, 4, 3, 2, 1 }

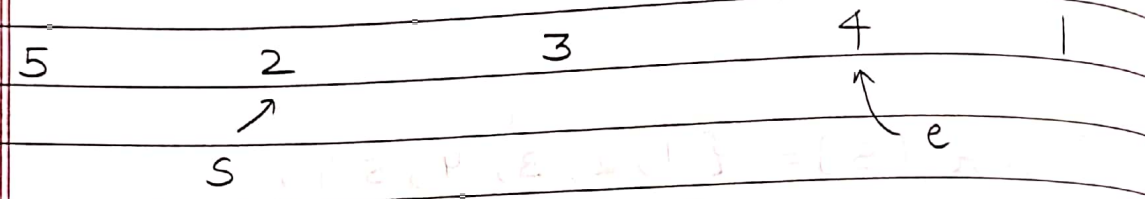
This can be done via 2 pointer approach by replacing arr [start] & arr [end] & rest is same as extreme prints. Extreme elements are swapped & hence we get the

reverse array:



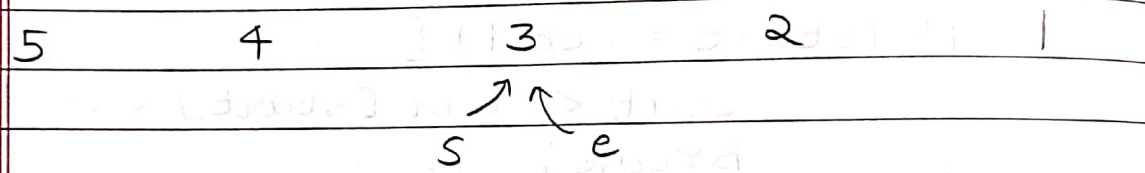
swap arr[s] & arr[e]

s++, e--



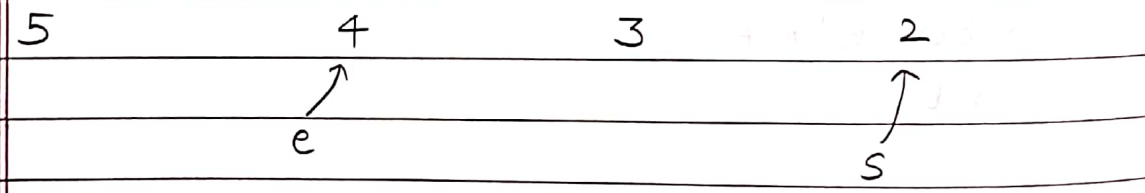
swap arr[s] & arr[e]

s++, e--



swap arr[s] & arr[e]

s++, e--



s > e → exit the loop

O/p → Reverse of the input array is what we got.

Code

```
int arr[5] = {1,2,3,4,5};
int s = 0;
int e = 4;
```

```
while (s<=e) {  → Inbuilt function
        swap (arr [s], arr [e]) ;
        s++;
        e--;
}
for (int i=0; i<5; i++) {
            cout << arr [i] << "  ";
}
```