Array Level 2

Sunday, 10 September 2023 11:04 AM

```
int main() {
    int a = 5;
    solve(a);   } //fn. call
    cout << a << endl;
}

void solve(int a) {
    a++;
    cout << a << endl;
}
```

Call/Pass by value

humesha copy create hoga;

Output

>> 6 ∵ both a is diff.
>> 5

Ex-> 2:

```
int main()
{
    int mark = 90;
    mark++;
    solve(mark);   } //fn. call
    cout << mark;
}

solve(int m)
{
    m--;
    m = m * 10;
    cout << m << endl;
}
```

Output

>> 900
>> 91

Note: When we pass arguments as parameter in any fn. then it creates a copy of it or we can say it creates diff. memory loc. for it.

Ex 3:

```
int main(){
    int sundari = 100;
    sundari--;
    sundari -= 5;
    solve(sundari);   } //fn. call
    cout << sundari;
}

void solve(int jaadu){
    jaadu--;
    cout << jaadu + 10;
    return;
}
```

Output

>> 103
>> 94

// Call by reference

Ex1:
```
int main() {
    int a = 5;
    solve(a);
    cout << a;
}

void solve(int &a) {
    a++;
    cout << a << endl;
}
```

Output
>> 6
>> 6

-) pass by reference me copy create ni hoti hain hum actual memory location me kaam karte hain

Ex 2:

```
int main()
{
    int mark = 90;
    mark++;
    solve(mark);   // fn. call
    cout << mark;
}

solve(int &m)
{
    m--;
    m = m * 10;
    cout << m << endl;
}
```

Output
>> 900
>> 900

Ex3:
```
int main() {
    int sundari = 100;
    sundari--;
    sundari -= 5;
    solve(sundari);   // fn. call
    cout << sundari;
}

void solve(int &jaadu) {
    jaadu--;
    cout << jaadu * 10;
    return;
}
```

// in this we are giving different name to same memory loc. like sundari in main and jaadu in solve fn.

Output
>> 103
>> 93

→ Reference variable


```
int a = 50;
cout << a << endl;
// reference variable : same memory location with different names
   int b = &a;        Ex → Ganga dhan hi saktiman hain
   int &c = a;
```

Ex→
```
int main() {
   int arr[] = {1,2,4}
   int n = 3;
    Solve (arr, n)
   for (int i=0; i<n; i++) {
       cout << arr[i];
   }
}
```

```
solve (int arr[], int size)
{ arr[0] = 100;
}
```

Output
>> 100 2 4

Note : Array is by default Call / Pass by reference hota hain or array ko jab bhi fn. me pass karte hain toh uski copy ni vo array ki location pass hoti hain by default.

Q.) Find unique element
- → i/p : — , — , — , — , — .....
- → each element → occurs twice
- → except one , find out

we will use XOR , it gives 0 for same output and 1 for diff. output

→ Code:

```
int getUnique(int arr[], int n){
    int ans = 0; //why?
    for(int i=0; i<n; i++){
        ans = ans ^ arr[i];
    }
    return ans;
}
```

XOR

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

// kissi bhi no. ka zero ke sath XOR
gives that no.

```
int main(){
    int arr[] = {2,10,11,10,2,13,13,15,15};
    int n = 9;
    int final_Ans = getUnique(arr,n);
    cout << final_Ans;
    return 0;}
```

Output

2^10^11^10^2^13^13^15^15

⇒ 11

Q.) i/p → array → [ ] → {10, 20, 30}

↳ print all pairs like
(10,20), (10,30), (10,10)
(20,10), (20,30), (20,20)
(30,10), (30,20), (30,30)

We can solve it by using 2 loops.

```
for (int i=0 ; i< 3 ;i++){
    for (int j=0; j< 3 ;j++) {
        cout << arr[i] <<","<< arr[j]<<endl;
    }
}
```

Output

```
>> 10,10
   10,20
   10,30
   20,10
   20,20
   20,30
   30,10
   30,20
   30,30
```

H.W

```
for (int i=0 ; i<n ;i++){

    for (j=0 ; j<n ; j++)
    {
        cout << i<<" "<<j;
        cout << arr[i] <<" " << arr[j];
    }
}
// change the cond^n like n to n/2 , or do
   initialisation from i or j = 1 to n and
   then predict output by urself.
```

Q.) i/p → array → [1,2,3,4,5]
↳ print all triplets

in this we need to use 3 loops

```
n = 4;
for (int i=0 ; i<n ;i++) {
    for (int j=0; j<n ;j++) {
        for (int k=0; k<n ; k++) {
            cout << arr[i] << arr[j] << arr[k]<<endl;
        }
    }
}
```

H.W
Triplet Sum
or
Three Sum

[brute force → basic sol^n]

Q) i/p

0	1	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7	8

arr

Sort in ascending order

Dutch National flag problem

① Counting
② 2 pointer approach (H.W)
③ Sort() → predefined fn.

① by Counting
// count 0 and 1 and fill it on array

```
sort2zeroOne(int arr[], int n){
    int zeroCount = 0;
    int oneCount = 0;
//Counting zero and one
for(int i=0; i<n; i++){
    if(arr[i]==0)
        zeroCount++;
    else if(arr[i]==1)
        oneCount++;
  }

int i;
for(i=0; i<zeroCount; i++){
    arr[i]=0; //place all 0's first
  }
for(int j=i; j<n; j++){
    arr[j]=1; //placing all 1's
  }
```

//method 1

```
int index = 0;

while(zeroCount--){
    arr[index]=0;
    index++;
  }
while(oneCount--){
    arr[index]=1;
    index++;
  }
}
```

//method 2

```
int arr[] = {0,1,0,1,0,1,1,1,1,0,0,0,0,0};
int n = 14;

    sortZeroOne(arr, n);
  return 0
}
```

Output

>> | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

will get same output by both methods

2) Shift arrays element by 1

i/p -> | 10 | 20 | 30 | 40 | 50 | 60 |

o/p -> | 60 | 10 | 20 | 30 | 40 | 50 |

① temp = 60
② i = n-1 ⟶ i = 1
→ arr[i] = arr[i-1]

```
int main() {
    int temp;
  for(int i = n-1; i>=1; i--){
          temp = arr[i];

      arr[i] = arr[i-1];
  }
    arr[0] = temp;
  return 0;
}
```

H.W
do Left shift like
20 30 40 50 60 10

Output

>> | 60 | 10 | 20 | 30 | 40 | 50 |

→ Here there is only one element which coming first from last but what if many elements shift to starting indexes from last

| 10 | 20 | 30 | 40 | 50 | 60 |

both will go to starting index

① temp[] = {50, 60}
② Shifting
③ copy temp in starting indexes

H.W