Array

Wednesday, 6 September 2023 9:18 PM

* What is Array?

-> list of similar elements

-> collection of elements of similar datatype

-> Data structure

-> Continuous Memory block

Use case

-> Instead of creating hundredd of variable we can use an array of size hundred to store all that data like `int arr[1000]`

we created memory location which will store data of same datatype.

array ke andar first integer type data store hoga

-> `int brr[10];` // 10 x 4 = 40 bytes space is allocated

-> continuous memory location allocated when array created.

-> this type of array creation is called static array creation

* Creation of an Array :

```
int money[127];   -> 127 x 4 bytes
char alpha[26];   -> 26 x 1 bytes
bool flags[27];   -> 27 x 1 bytes
long num[100];
short snum[1000];
```

-> `int a = 5;` means symbol table me variable a ke corresponding jo address hain usme 5 store hoga.

table where variable and their corresponding address is present

→ We can check address by using '&' operator.

address of

```
int a = 5;
cout << &a;
```

↳ address of a print hoga

```
int arr[10];
cout << &arr;
cout << arr;
```

address of array arr.

both will give same output ∵ arr points to the address of array.

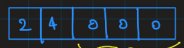
```
int a = 5;  int arr[10];
cout << sizeof(a);        // 4
cout << sizeof(arr);      // 10x4 = 40
```

* array initialisation

→ int arr[] = {1, 2, 3, 4, 5};

→ int brr[5] = {1, 2, 3, 4, 5};

→ int crr[5] = {2, 4}



baaki jagah me zero store hoga

→ int drr[2] = {1, 2, 3, 4, 5};
this will give error since array size is less than the no. of element given

→ int n;
cin >> n;
int arr[n];

bad practice
Ex→ 10 byte memory is allocated to the program but if user input n = 20 than the size of program becomes 80 byte, than program will show bad behaviour

→ To counter this we use dynamic array

* Indexing in Array

0-base indexing

→ first element of array is always in 0^{th} index and last element is in $(n-1)^{th}$ index

```
int arr[10];
```

arr →

3	5	8	9	12	10	6	7	1	2
0	1	2	3	4	5	6	7	8	9

```
arr[2] = 8
arr[5] = 10
arr[3] = 9
arr[8] = 1
arr[0] = 3
```

→ So for accessing all elements inside array instead of writing each index we can use loop like,

```
// Printing an array
int n=10;
for(int i=0 ; i<n ; i++){
    cout << arr[i];
}
```

Output

>> 3 5 8 9 12 10 6 7 1 2

//taking input in an array

```
int arr[5];
```

if there is no element in this array than garbage value will be stored

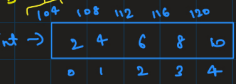
```
cin >> arr[0];
cin >> arr[1];
.
.
cin >> arr[4];
```

instead of this we can us loop

```
int n=5;
for(int i=0; i<n; i++){
    cin >> arr[i];
}
```

→ Formula:

arr[i] => value at (Base address + (datatype size * index))



arr[2] => value at (104 + (4 * 2))

=> value at (104 + 8)

=> value at (112)

so arr[2] => value at 112 i.e. 6

Q.) i. 10 size array
ii. take i/p in that array
iii. double-up each value of that array.

=)
```
int arr[10];
for (int i=0; i<10; i++){
        cin >> arr[i];
      }
// doubling value
for (int i=0; i<10; i++){
        arr[i] = arr[i] * 2;
      }
// printing array
for (int i=0; i<10; i++){
      cout << arr[i];
     }
```

Q.) i. 5 size array
ii. 5 - i/p
iii. total sum print

```
int arr[5];
for (int i=0; i<5; i++){
      cin>> arr[i];
     }
//total sum
 int sum =0;
for (int i=0; i<5; i++){
      sum = sum + arr[i];
     }
cout << sum;      // 10 as output
                  (4+3+2+1+0)
```

* Linear Search in an Array

→ used to search an element in an array

```
Ex→ int arr[5] = {2,4, 6, 8, 10};
    int target = 10;
    int n = 5;
    bool flag = 0;   // 0 → not found
                     // 1 → found

        for (int i=0; i<n; i++){
            if (arr[i] == target){
                cout << "found";
                break;
                flag = 1;
                }

        // So if loop ends but we did not found
        that our element i.e. target is in the array
        or not. So, that's where we use flag.
    }
    if (flag == 1)
        cout << "found";
    else
        cout << "Not found";

    return 0;
    }
```

→ Output

>> Not found

* Arrays and functions

```
int main()
{
    int arr[5];
    solve(arr, size);   //this is the way to pass array in a function
}

solve(arr, size){
    - - -
    //body
    - - -
}
```

this size means no. of elements in an array.

Q) make a program for linear search using functions.

```
// linear search fn.
bool linearsearch (int arr[], int size, int target){
    for(int i=0; i<size; i++){
        if (arr[i] == target) {
            //found
            return true;
        }
    }
    //not found
    return false;
}
```

true or false

// ye statement tabhi execute hoga jab if wali condition true ni hogi

→ Count 0's and 1's in an Array

arr

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

```
int c2ero=0, cOne=0;

for (int i=0; i<8; i++){
    if (arr[i]==0)
        c2ero++;
    else if (arr[i]==1)
        cOne++;
```

Output

» c2ero = 3
» cOne = 5

→ Minimum no. in an Array

arr

20	4	15	2	6	8	11
0	1	2	3	4	5	6

In case of signed integer range is $[-2^{31} \rightarrow 2^{31}-1]$

-2^{31} → INT_MIN

$2^{31}-1$ → INT_MAX

to use this we include `#include <limits.h>` header file

→ Best Practice

- Min. no. → INT_MAX
- Max no. → INT_MIN

// we do this so that when we need to find minimum no. that no. cannot be greater that INT_Max. similarly, to find max no. we compare with INT_MIN beCoz no. cannot be less than INT_MIN

Ex → int minAns = INT_MAX;

```
//tarika 1
for ( int i =0 ; i<n ; i++){
    if (arr[i] < minAns){
        minAns = arr[i];
    }
}
```

```
//tarika 2
// using min fn. (predefined fn.)
for(int i= 0; i<n ; i++){
    minAns = min (arr[i], minAns);
}
```

Q.) Make programs for both methods

H.W → Find maximum no. in an array

→ Reverse an Array

i/p →

arr	10	20	30	40	50	60
	0	1	2	3	4	5

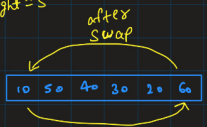
o/p →

arr	60	50	40	30	20	10
	0	1	2	3	4	5

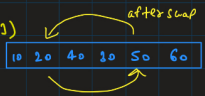
arr	60	50	40	30	20	10
	0	1	2	3	4	5

left = 0, right = 5

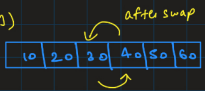
swap(arr[left], arr[right])
left++;
right--;



left = 1, right = 4
swap(arr[left], arr[right])
left++;
right++;



left = 2, right = 3
swap(arr[left], arr[right])
left++;
right++;



left = 3, right = 2

Now, stop.

H.W

implement swap fn. using
- → inbuilt fn.
- → add/sub method
- → using temp variable
- → using XOR

→ void reverseArray (int arr[], int size){

```
int left = 0;
int right = size-1;
while (left<=right){
  swap(arr[left],arr[right])
{   left++;
    right--;
}
```

//using predefined swap fn.

→ While loop

```
int i=1; //initialize
  while (i<=n) { //cond^n check)
      cout<<i;  //body
        i++; //updation
    }
```

Output

let n=5

>> 1 2 3 4 5

Now reverseArray by using for loop

```
for(int left=0, right=size-1; left<=right; left++, right++)
{
        swap(arr[left], arr[right]);
}
```

→ both codes using while and for loop will work for even as well as odd size array.

→ Extreme print in an Array] → H.W

i/p arr

10	20	30	40	50	60
0	1	2	3	4	5

o/p arr

10	60	20	50	30	40
0	1	2	3	4	5

```
void extremePrint(int arr[], int size){
    int left=0;
    int right=size-1;
    while(left<=right){
        cout<<arr[left++]<<endl;
        cout<<arr[right--]<<endl;
    }
}
```

→ Output

```
» 10
  60
  20
  50
  30
  40
```

→ this is working correctly for even size array but for odd size array it is printing middle no. twice, as you can see below

→ arr[5] = {10,20,30,40,50};

Output

>> 10
40
20
50
30
20

// adding an if cond^n.

→ void extremePrint (int arr[], int size){
int left = 0;
int right = size-1;
while (left <= right){
if(left == right)
cout<< arr[left];
else {
cout << arr[left++] <<endl;
cout<< arr[right--]<<endl;
}
}
}

// after adding if cond^n. this is giving right ouput for an odd size array.

Output

>> 10
50
20
40
30