Operations on Arrays

→ Traversal

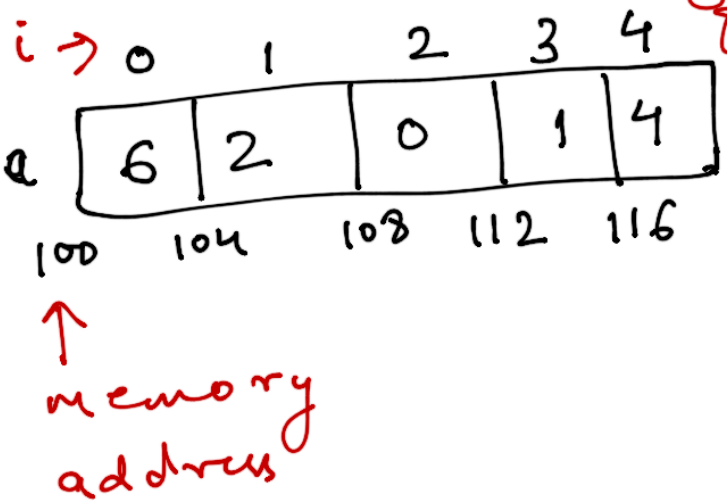
→ Insertion

→ Deletion

→ Searching

→ Sorting

Traversal : visiting every element of the array




```
int a[5];
for(i=0 ; i<5 ; i++)
{
cout<<"Enter a
    element<<endl;
cin >> a[i];
}
```

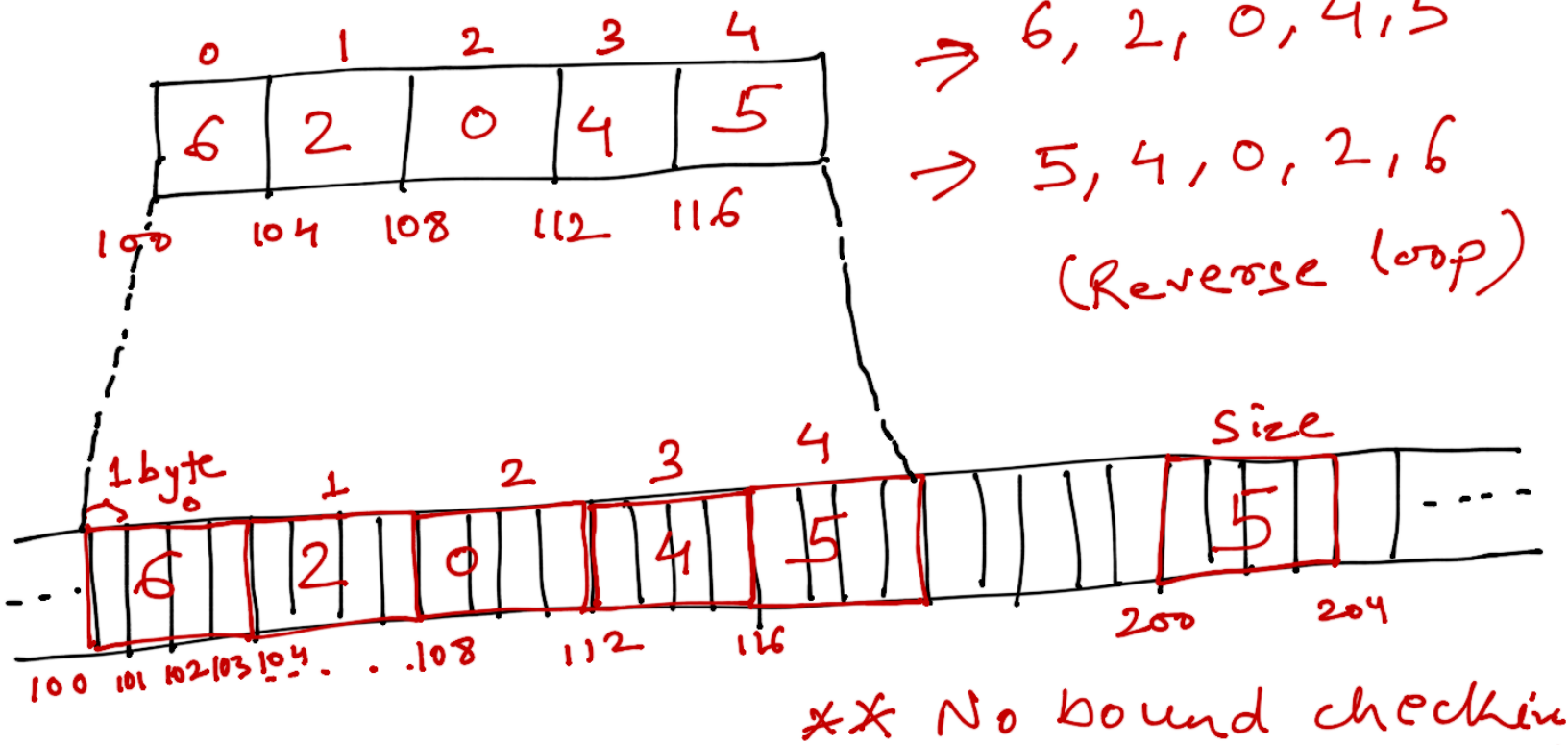
```
int a[50], size;
cout << "Enter size of an array";
cin >> size;
for (i=0; i < size; i++)
{
cout << "Enter a element" << endl;
cin >> a[i];
}
```

Traversal

→ 6, 2, 0, 4, 5

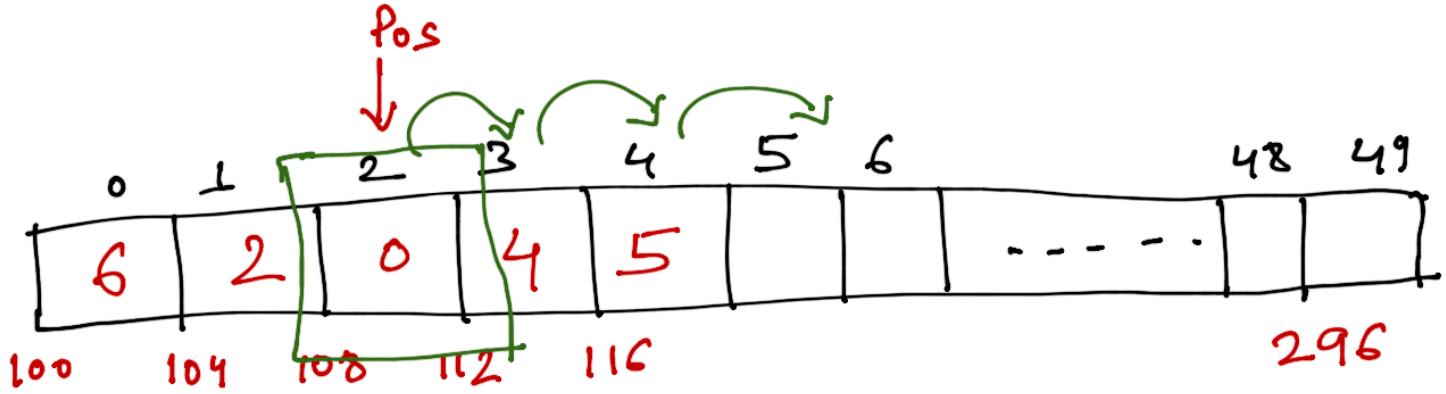
→ 5, 4, 0, 2, 6

(Reverse loop)



** No bound checking

int a[50]

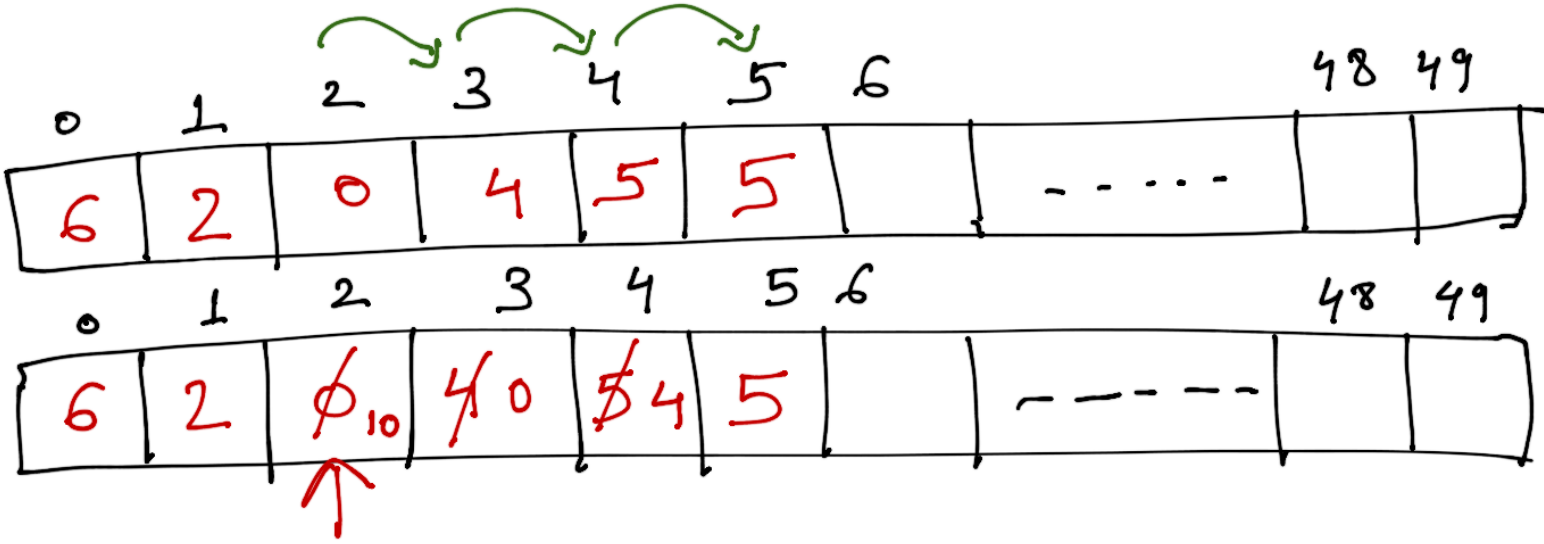


Insertion
- At a specific position
- At beginning
- At end of array

Position at 3

Insert num = 10

at pos = 3

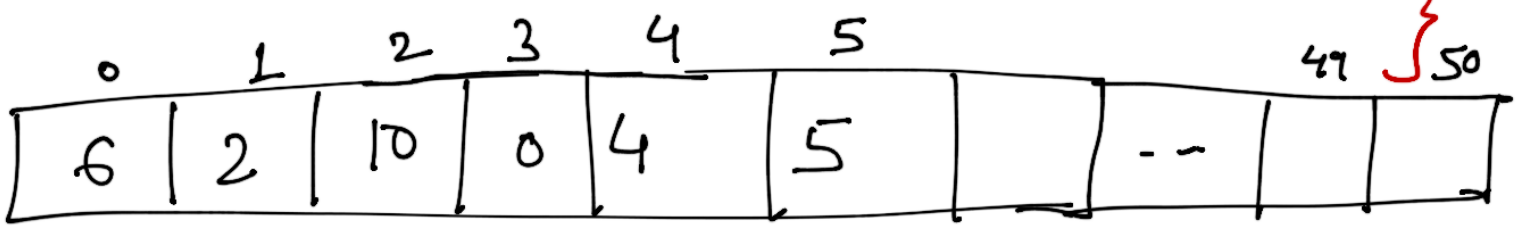


```
cout << "Enter data to be inserted";
cin >> num;
cout << "Enter position";
cin >> pos;
```

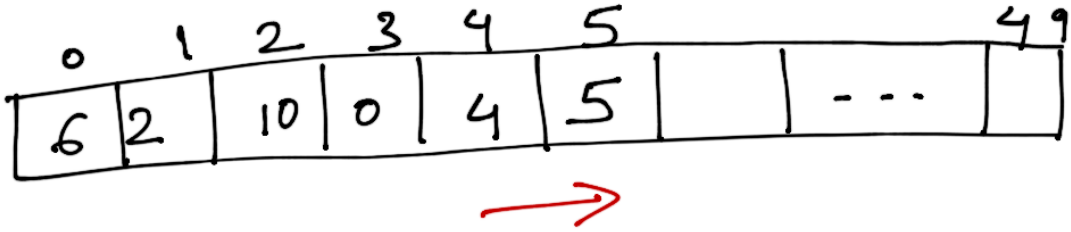
Else

```
for (i = size-1; i >= pos-1; i--)
{
    a[i+1] = a[i];
}
a[Pos-1] = num;
size++;
```

```
if (pos<=0 || pos >size+1)
{ invalid position;
}
```



Insertion at beginning



Just keep shifting towards right

```
for (i=size-1 ; i>=0 ; i--)
{
    a[i+1] = a[i];
}
a[0] = num;
size++;
```

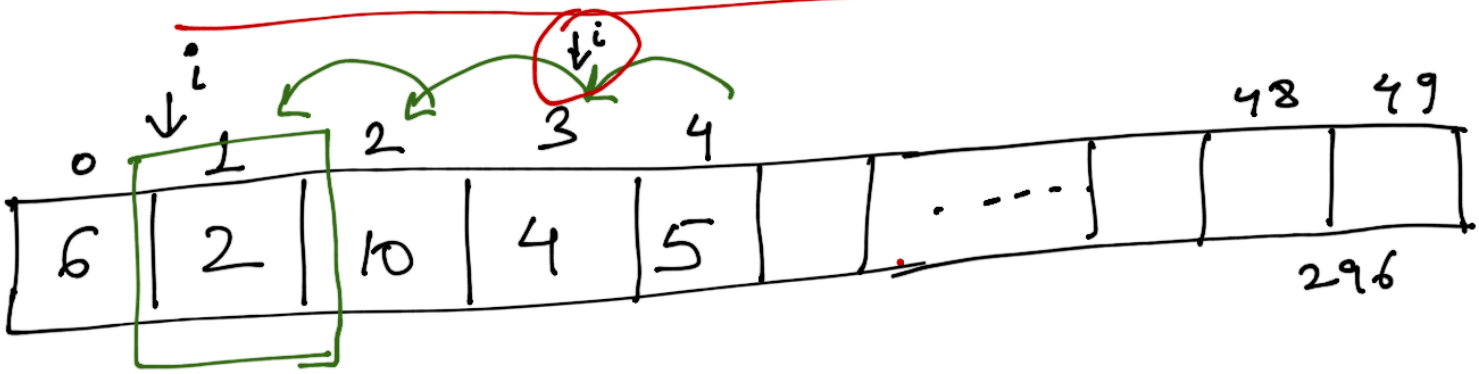
Time Complexity

Begin → O(n)

End → O(1)

position p → O(n-p)

Deletion in an Array



position 2 (position & index are different)

i = 1

```
a[1] = a[2];
a[2] = a[3];
a[3] = a[4];
```

Size = 5

↓ Decrease after deletion

Size = 4

```
Cout << " Enter position to be deleted";
Cin >> pos;
if ( pos <=0  || pos > size)
        {  invalid pos;
           item = a[pos-1]; } // for printing deleted item
else
{     for(i = pos-1; i< size-1 ; i++)
        { a[i] = a[i+1]; }  size--; }
```

If delete from beginning

```
for(i=0 ; i<size-1;i++)
    { a[i] = a[i+1];
    }
size--;
```

for unsorted array

→ Directly pick last element & place at position from where you deleted the element

O(1)