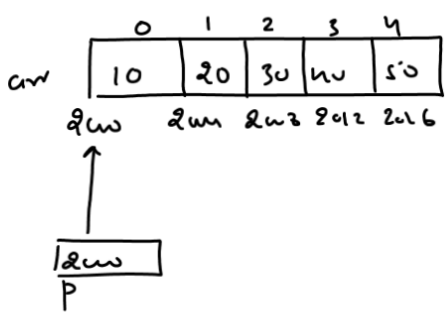
Accessing Array Using Pointer

```
int main()
{
    int a[5];
    int i;
    int *p;
    p = a;
    for(i=0; i<5; i++)
    {
        printf("Enter no:");
        scanf("%d", p+i);
    }
    for(i=0; i<5; i++)
    {
        printf("\n%d", *(p+i));
    }
    return 0;
}
```



p+i => p + i * sizeof(data type of p)

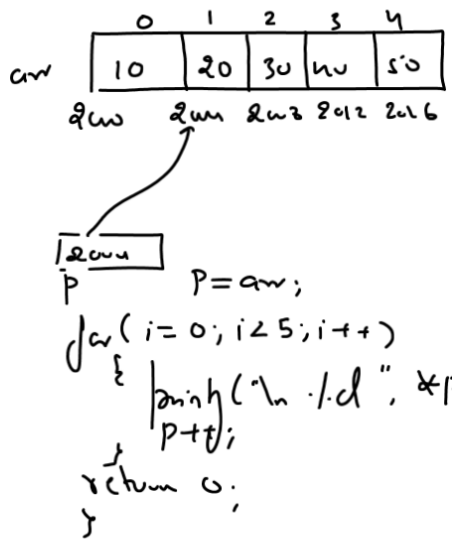
a. i=0, p+i => 2000+0 => 2000+0*4 => 2000

b. i=1, p+i => 2000+1 => 2000+1*4 => 2004

c. i=2, p+i => 2000+2 => 2000+2*4 => 2008

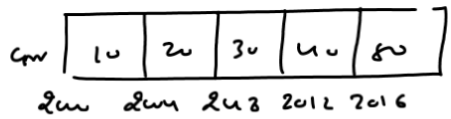
Accessing Array Using Pointer (Sec Soln)

```
int main()
{
    int arr[5];
    int i;
    int *p;
    p = arr;
for(i=0; i<5; i++)
  {
    printf("Enter no:");
    scanf("%d", p);
    p++;
  }
```



How Compiler Handles Arr Expressions ?

```
int arr[5] = {10, 20, 30, 40, 50};
```




```
printf("%d", arr[3]);
          40
printf("\n%d", 3[arr]);
          L,40
printf("\n%d", arr[-1]);
        Will become *(1996)
```

arr[i] => *(arr+i)

arr[3] => *(arr+3)
=> *(2cc0+3)
=> *(2012)

([ ]) *

```
int arr[5] = {10,20,30,40,50};
int i, *p;
p = arr;

for(i = 0; i < 5; i++)
{
    printf("\n %d", ?    );
}
```

	0	1	2	3	4
arr	10	20	30	40	50
	2000	2004	2008	2012	2016

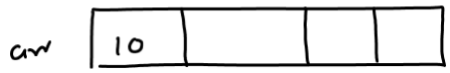
1. arr[i]
2. i[arr]
3. *(arr+i)
4. *(i+arr)
5. p[i]
6. i[p]
7. *(p+i)
8. *(i+p)
9. *p++

*p++

v/s

(*p)++

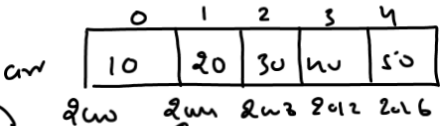
```
int main()
{
    int arr[5];
    int i;
    int *p;
    p = arr;
for(i=0; i<5; i++)
    {
        printf("Enter no:");
        scanf("%d", arr+i );
    }
```




```
addr
P
for(i=0; i<5; i++)
    {
        printf("\n%d", *(arr+i));
    }
return
}
```

```
int main()
{
    int arr[5];
    int i;
    int *p;
    p = arr;
for(i=0; i<5; i++)
    {
        printf("Enter no:");
        scanf("%d", arr );
        arr++;
    }
```

l-value reqd



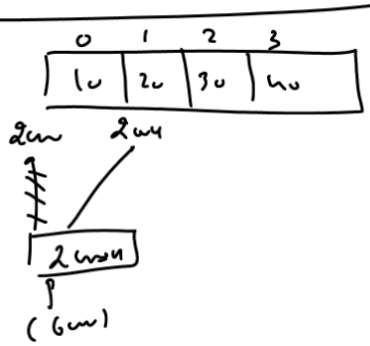

```
addr
P
p=arr;
for(i=0; i<5; i++)
    {
        printf("\n%d", *p);
        p++;
    }
return 0;
}
```

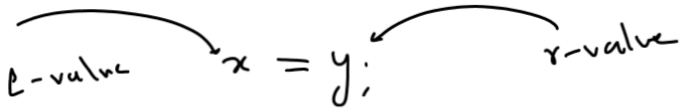
```
int i=10;
i++;
↳ i=i+1;
```

```
am++;
↳ am = am+1;  X
```

---

p++;
↓
p = p+1;



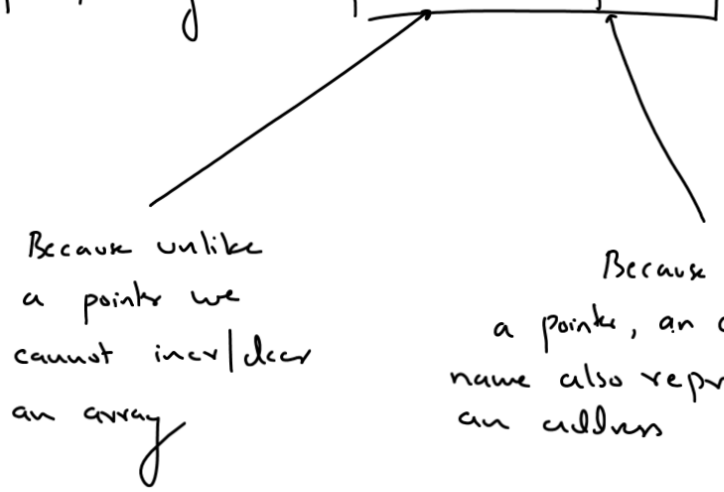


```
int a;
✓ a = 10;
  10 = a;  X
```

```
int x=10, y=20, z;
✓ z = x+y;
X x+y = z;
```

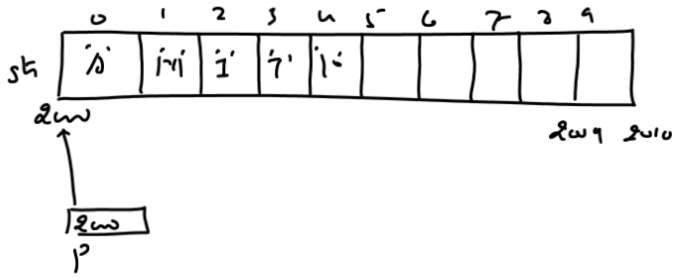
```
int x;
✓ x = pow(2,3);
X pow(2,3) = x;
```

For compiler, array is a constant pointer



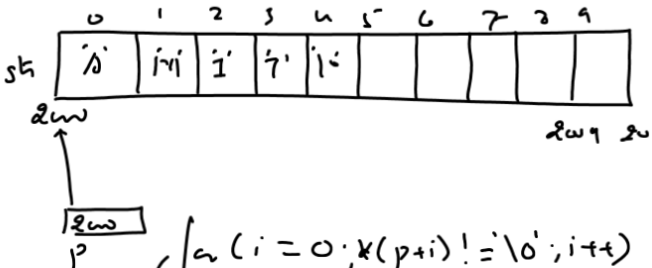
Accessing Character Array Using Pointer

```
int main()
{
    char sh[10];
    char *p;
    p = sh;
    printf("Enter your name:");
    scanf("%s", p);
    printf("Hello %s", p);
    return 0;
}
```



Traversing A Character Array Using Pointer

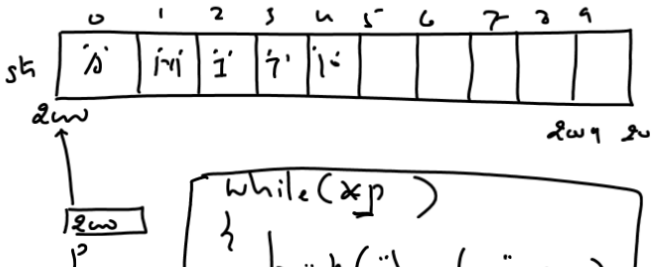
```
int main()
{  int i;
   char sh[10];
   char *p;
   p = sh;
printf("Enter your name:");
scanf("%s", p);
```




```
for(i=0; *(p+i)!='\0'; i++)
    printf("\n%c", *(p+i));

return 0;
}
```

```
int main()
{
   char sh[10];
   char *p;
   p = sh;
printf("Enter your name:");
scanf("%s", p);
```




```
while(*p)
{
   printf("\n%c", *p);
   p++;
}
return 0;
}
```

①
```
int a, b, c;
a = 10;
b = 10;
c = 10;
OR
a = b = c = 10;
```

②
```
int i = 10;
i = i+1;
or
i += 1;
or
i++;
or
++i;
```

Two New Shortcuts

①
```
if( a != 0 )
or
if( a )
```
Same

```
if( a % 2 != 0)
{

}
if( a % 2 )
{

}
```
Same

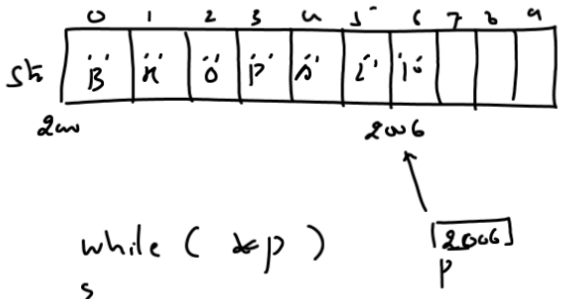
```
while ( x+y )  ==========>  while ( x+y != 0 )
{                           {

}                           }
```

(2)

```
if ( a == 0 )                 |   if ( a % 2 == 0 )
                              |   {
   or                         |
                              |   }
if ( ! a )                    |   OR
                              |   if ( ! ( a % 2 ) )
                              |   {
                              |
                              |   }
```

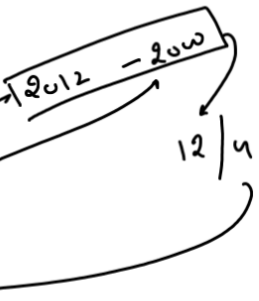
```
int main()
{
    char str[10];
    char *p;
    p = str;
printf("Enter a string:");
scanf("%s", p);
```




```
while (*p)
{
    p++;
}
printf("length = %d", p - str);
return 0;
}
```

⑥

```
int an[5] = {10, 20, 30, 40, 50};
int *p, *q;
p = an;        ——> 2000
q = &an[3];    ——> 2012
printf("%d", q - p);
```



③

If p and q are pointers, then

$p + q$ X

$p * q$ X

p / q X

$p \cdot / \cdot q$ X

$p - q$ ✓]

$q - p$ ✓]