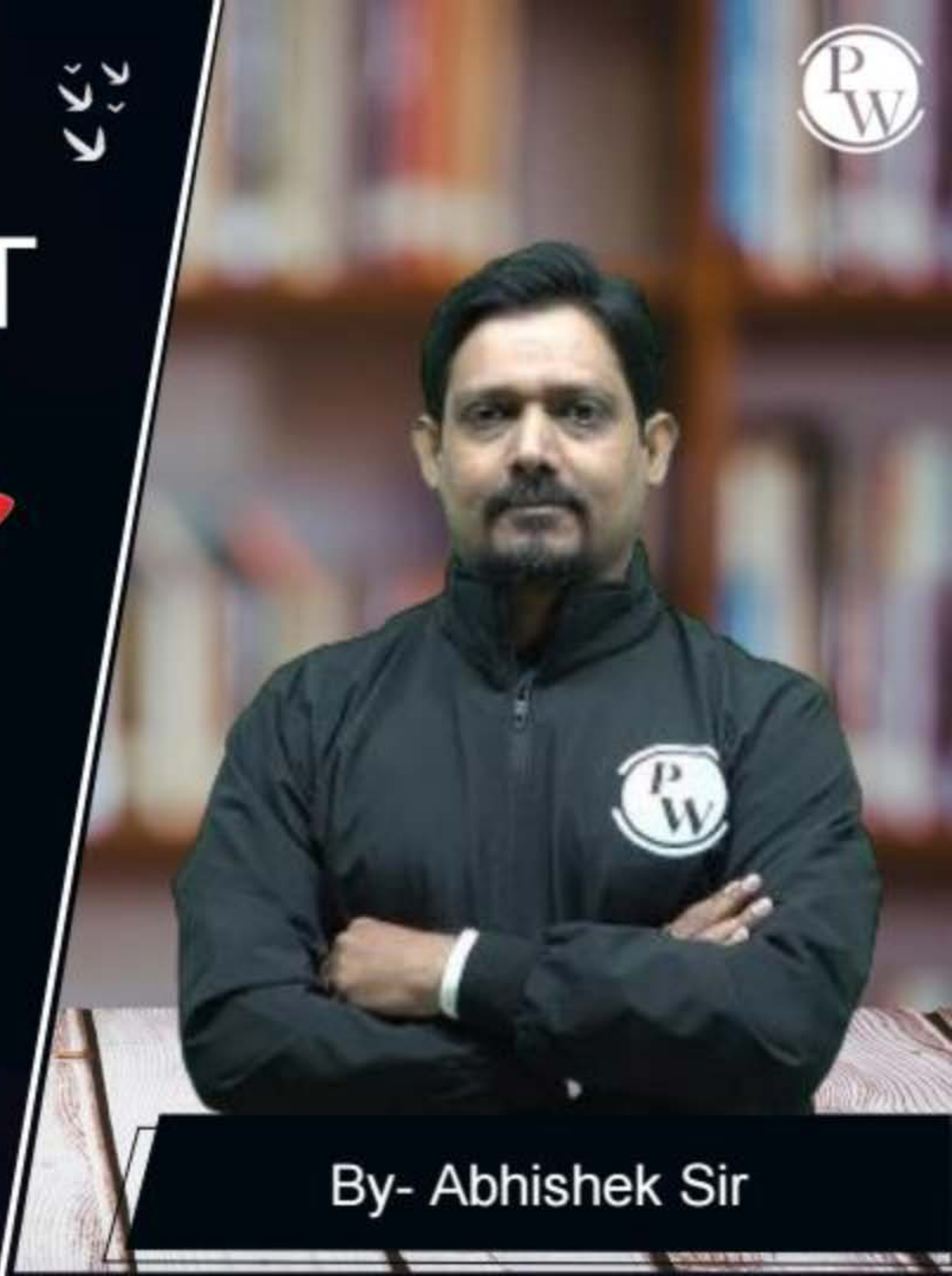
Computer Science & IT

C programming

Array & Pointers

Lecture No. 03



By- Abhishek Sir

Recap of Previous Lecture

Topic

Topic

Topic

Topic

Topic

pointer problem (Dangling reference,
uninitialized pointer

Array.

$a[i] = *(a+i) = *(i+a) = i[a]$

Slide

Topics to be Covered

Topic Data type

Topic problem

Topic 2D Array. ✓

Topic

Topic

Slide

Question

Consider the following program in C language:

```
#include<stdio.h>a
main(){
    int i;
    int*pi=&i;
    scanf("%d",pi);
    printf("%d\n",i+5);
}
```

Which one of the following statements is **TRUE**?

(A) Compilation fails.

(B) Execution results in a run-time error.

(C) On execution, the value printed is 5 more than the address of variable i.

(D) On execution, the value printed is 5 more than the integer value entered.

int *pi;

pi = &i;

Same

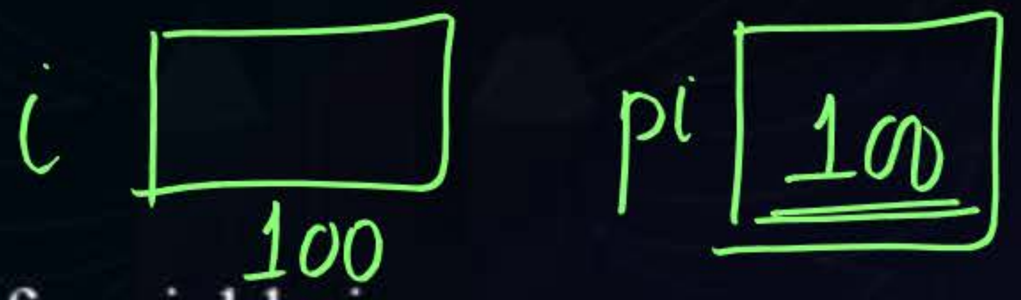
Scanf("%d", &i)

Same

Declaration & Initialize

int *pi = &i;

2014



&i = 100

Slide

Array

Data type in 1-D array

&a, a, *a

Problem

2d Array deflation & initialization

Meaning of a[i][j]

Datatype of a[i][j]

Slide

Question

Assume that array elements are stored from 1000, 1004, 1008, 1012, 1016

```
int main (){
        int a[] ={10,20,30,40,50};
        int i,*b,;
        b = a+4;          //1000 is assigned
        printf(“%d”, b[-1]);
        return 0;
}
```

The value printed by the program is ___40___?

10	20	30	40	50
1000	1004	1008	1012	1016

$$b = a+4 = 1000+4 = 1000+4\times4$$

$$= 1016$$

$$b \; \boxed{1016}$$

$$b[-1] = *(b-1) = *(1016-1)$$

$$= *(1016-1*4) = *(1012)$$

Slide

Array

1-D array data

$a \longrightarrow$

Array

1-D array data

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

a → Address of first element (Address of integer)

4B

*a → integer

Array Size

&a → Address of 1-D array

Slide

Array

```
#include <stdio.h>
int main(){
    int a[] = {1,2,3,4};

    printf("%u\n", a);
    printf("%d\n", *a);
    printf("%u\n", &a);

    printf("%u\n", a+1);
    printf("%d\n", *a+1);
    printf("%u\n", &a+1);

    return 0;
}
```

A[0]	A[1]	A[2]	A[3]	
11	12	13	14	
100	104	108	112	116

113
114
115

Assummed value

100

*(100) = 11

100

100+1 = 100+1×4 = 104

→ *(100)+1 = 11+1 = 12

100+1 = 100+1 (Size of array) = 100+16 = 116

Slide

Arithmetic Operation with Pointer

#Q

```
main (){
        int a[] ={10,20,30,40,50};
        int i,*b;
        b = &a[4]-4;
        for(i=0;i<=4;i++){
                printf("%d",*b);
                b++;

        }
return 0;
}
```

Output__________

$6 = 116 - 4 : \underline{100}$

$(116 - 4)$

a[4]

$\underline{*(a+4)}$

10	20	30	40	50
100	104	108	112	116

$116 - 4\times4 = 116 - 16 = 100$

$\boxed{1020304050}$

Slide

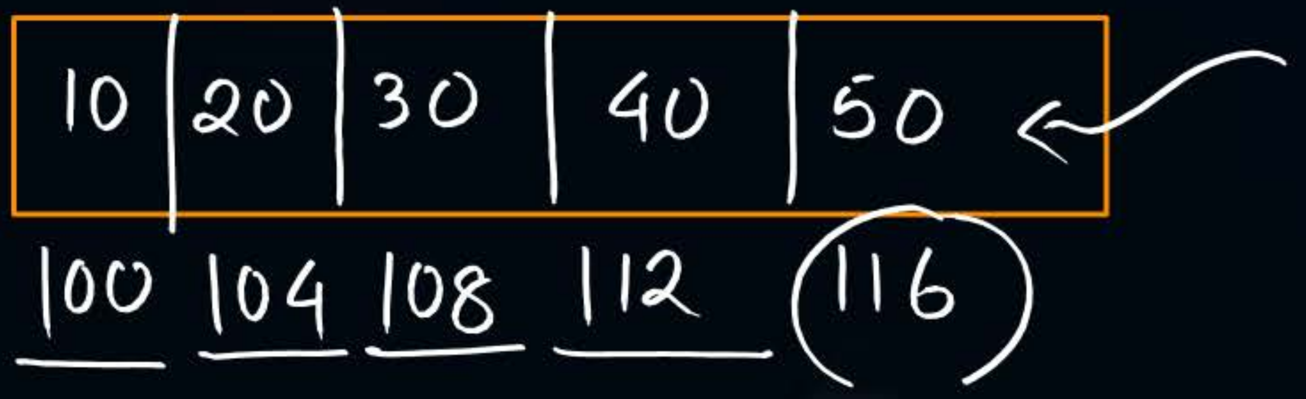
Arithmetic Operation with Pointer

#Q

```
main (){
        int a[] ={10,20,30,40,50};
        int i,*b;
        b = &a[4]-4;
        for(i=0;i<=4;i++){
                printf("%d",*b);
                b++;
        }
return 0;
}
Output__________
```

5 ← violating array bounday

a[4]   *(a+4)



10	20	30	40	50
100	104	108	112	116

*(b+i)

No provision for checking array boundry

Slide

Question

Out put of the program

```
#include<stdio.h>
     int main(){
     int i , b[] = {2, 3, 4, 5, 6};
     b++;
     printf ("%d\t" ,*b) ;
     }
```

b	1	2	3	4	5	6
	100					

(A) 2

(B) 3

(C) Address Increment

(D) Error

Array Name & Address Associated as constant

Constant value

Constant Can be Incremented.

$b \boxed{100}$ $i = b$

Slide

Question

#Q What is the output of the following program ?

```
#include<stdio.h>
int main(){
int i , b[] = {2, 3, 4, 5, 6}, *p ;
p = b ;
b++;
printf ("%d\t" ,*b) ;
}
```

(A) 1 3

(B) 2 3

(C) 2 4

(D) 3, 4

Slide

Question

#Q What is the output of the following program ?

```
#include<stdio.h>
int main(){
int i , b [] = {2, 3, 4, 5, 6}, *p ;
p = b ;
++*p ;
printf ("%d\t" ,*p) ;
p += 2 ;
printf ("%d" , *p);
}
```

(A) 1 3

(B) 2 3

(C) 2 4

(D) 3, 4

Slide

Question

#Q What is the output of the following program ?

```
#include<stdio.h>
int main(){
int i , b [] = {2, 3, 4, 5, 6}, *p ;
p = b ;
++*p ;
printf ("%d\t" ,*p) ;
p += 2 ;
printf ("%d" , *p);
}
```

++*(100) ✓

100	104	108	112	116
~~2~~ 3	3	4	5	6

(A) 1 3

(B) 2 3

(C) 2 4

[✓ (D) 3, 4]

3 4

p $\boxed{100}$

p: 100+2 = 100+2*4 = 108

Slide

Differen between two pointer variable

100	104	108	112	116
10	20	30	40	50

```
# include<stdio.h>
int main() {
    int a[] = {10,20,30,40,50}
    int *b, *b1, i;
    b = a;
    b1 = a+4;
    i = b1-b;
                    point("%d", i);
                }
```

$b = \boxed{100}$

$b_1 = \boxed{116}$

$i = 116 - 100$

$$\underline{\underline{4}}$$

$$116 - 100 = \boxed{16}$$

$$ptr_2 - ptr_1 = \frac{(ptr_2 - ptr_1)}{\boxed{Size}}$$

$$= \frac{16}{4} = \boxed{4}$$

$$\begin{array}{cc} \{20.0 , & 25.0\} \\ 100 & 108 \end{array}$$

$p = a$

$q = a + 1$

$$(q - p) = (108 - 100)$$

$$(*q . *p) \qquad \frac{8}{8} = 1$$

$$25.0 - 20.0 = \boxed{5.0}$$

$\underline{\text{double}}$ a[] = {20.0, 25.0}, *p, *q;

$\underline{100}$ 100 $\underline{108}$

p = a   100

q = a+1   108

$$(q - p) = \frac{108 - 100}{8} = \frac{8}{8} = \boxed{1}$$

int (q - p)

(*q - *p)

(25.0 - 20.0)

= (5.0)

2 D Array

Declaration

```
int a[][]; ✗
int a[][3]; ✗
int a[4][]; ✗
int a[4][3]; ✓
```

Initilization

```
✗ int a[][]={1,2,3,4,5,6} ✗
✗ int a[4][ ]={11,12,13,14,15,16} ✗
✓ int a[][2]={60,70,80.40};
int a[2][2]={1,2,3,4}; ✓
int a[][]={{1,2},{3.4}}; HW
```

Slide

Slide

2 D Array

2-D array

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

Row colamn

	Column 0	Column 1
Row-0	1 a[0][0]	2 a[0][1]
Row-1	3 a[1][0]	4 a[1][1]

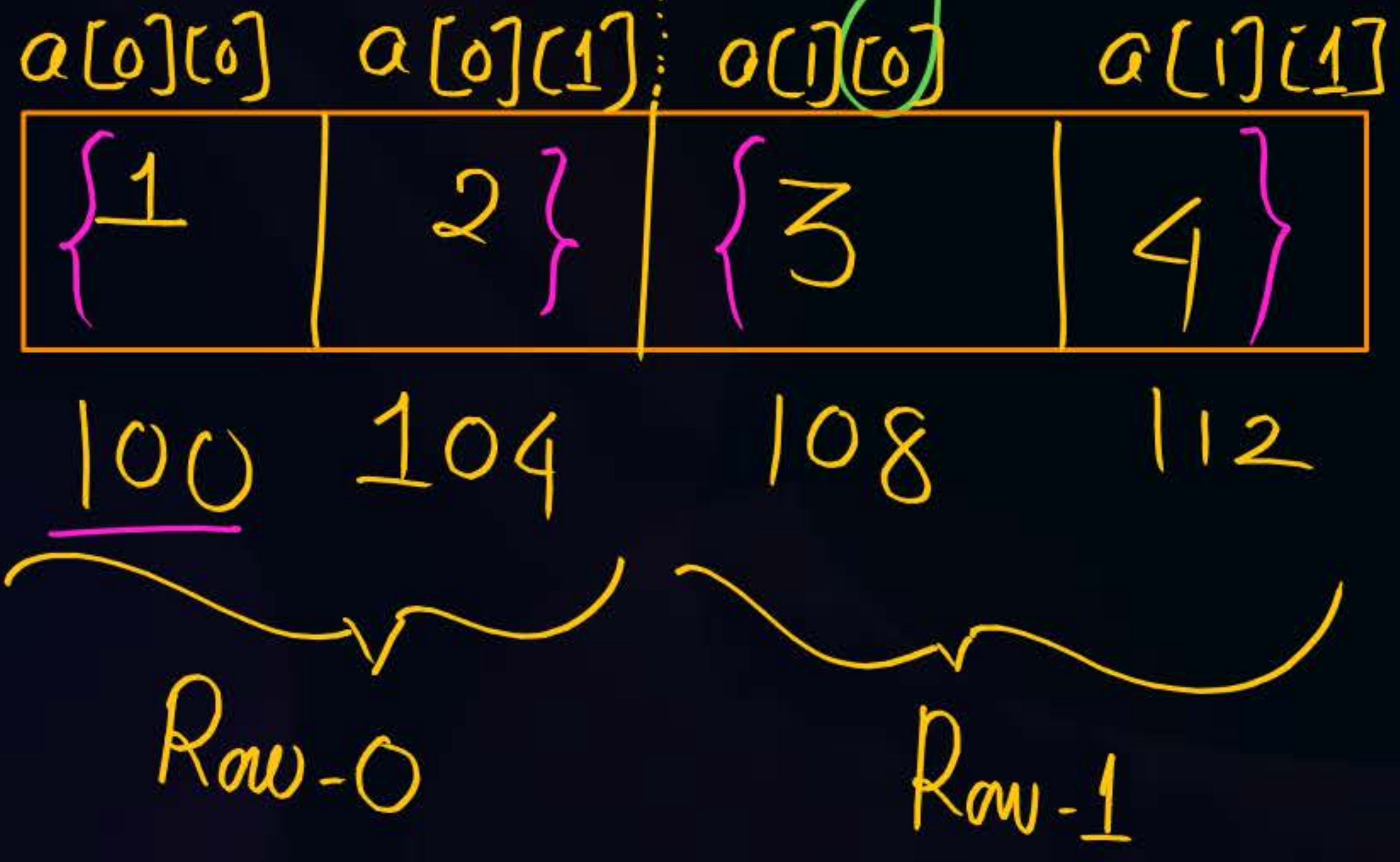
Logical Representation

Slide

2 D Array

$a[2][2] = \{\{1,2\},\{3,4\}\}$

2-D stored in memory in Row major order



a[0][0]	a[0][1]	a[1][0]	a[1][1]
{1	2}	{3	4}
100	104	108	112

$\underbrace{100 \quad 104}_{\text{Row-0}} \quad \underbrace{108 \quad 112}_{\text{Row-1}}$

2 D Array

int a[2][2] = $\{\underline{\overbrace{\{1,2\}}}, \overbrace{\{3,4\}}\}$;

Each element of 2 D array is = $\underline{\text{1 D array}}$

Each element of 1-D array = int

Slide

2 D Array

To Access element of 2 D array

$\underline{a[0][0]} = 1$

$a[i]$

$= *(a+i)$

$\underline{a[i]}[j] = \underbrace{*(a+i)}[j]$

$= \underbrace{*(*(a+i)+j)}$

Slide

2 D Array

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

$\boxed{\text{repeat}}$

a → Address of first element of array which is 1-D array

*a → Address of first element of 1-D array

integer

**a → integer

Slide

2 mins Summary

Topic

Topic

Topic

Topic

Topic

1-D array

pointer Substraction

2-D array.

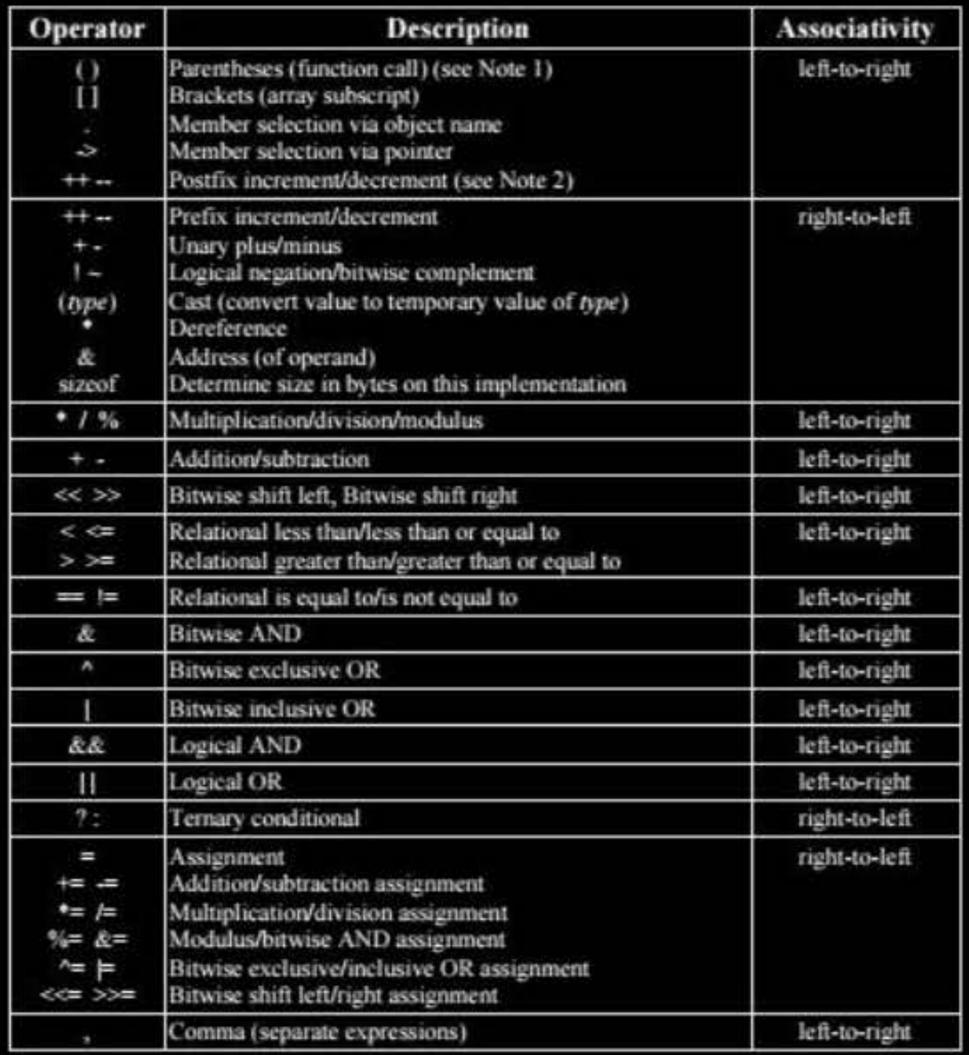
Slide

4B

8 bits

1B

THANK - YOU



Operator	Description	Associativity
( ) [ ] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of *type*) Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right