LECTURE - 34

POINTER ARITHMETIC AND ARRAYS (PART 03)

PROGRAMMING IN 'C'

File Edit Selection View Go Run Terminal Help

Code

arrays01.c arrays2.c

arrays01.c > main()

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main(){
    //defining an array
    int arr[] = {10,20,30,40,50};
    printf("%d \n", arr[0]); //10
    printf("%d \n", arr[4]); //50

    int newArr[3] = {52,48,63};
    printf("%d \n", newArr[0]);
    printf("%d \n", newArr[2]);
    // printf("%d \n", newArr[300]); //garbage value

    int arr2[3] = {15,21,36,47,92,41};
    printf("%d \n", arr2[0]);
    printf("%d \n", arr2[1]);
    printf("%d \n", arr2[2]);

    int arr3[3] = {12,24};
    printf("%d \n", arr3[0]);
    printf("%d \n", arr3[1]);
    printf("%d \n", arr3[2]);

    return 0;
}
```

Code

```
arrays01.c:15:32: warning: excess elements in array initializer
   15 |     int arr2[3] = {15,21,36,47,9
,41};
      |                                ^
arrays01.c:15:32: note: (near initialization for 'arr2')
arrays01.c:15:35: warning: excess elements in array initializer
   15 |     int arr2[3] = {15,21,36,47,92
,41};
      |
 ^~
arrays01.c:15:35: note: (near initialization for 'arr2')
10
50
52
63
15
21
36
12
24
0
PS>
```

0 0 Connect localHost sgm_2024 Ln 11, Col 32 Spaces: 4 UTF-8 CRLF C Win32 Prettier

17°C Clear ENG 8:47 PM 11/17/2024

File Edit Selection View Go Run Terminal Help

Code

arrays01.c | arrays2.c

arrays2.c > main()

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main(){
    int arr[5]; //declaration of an array -> blank array
    // inserting an element into an array
    arr[0] = 9999;
    arr[1] = 888;
    arr[2] = 77;
    arr[3] = 7;
    arr[4] = 6;
    printf("%d \n", arr[0]); //9999
    printf("%d \n", arr[2]); // 77
    printf("%d \n", arr[4]); // 6

    // update the values?
    arr[0] = 3333; //update the value of 0th index
    printf("%d \n", arr[0]); // 3333


    return 0;
}
```

```
9999
77
6
3333
PS>
```

Code

0 0 Connect localHost sgm_2024 Ln 7, Col 18 Spaces: 4 UTF-8 CRLF C Win32 Prettier

17°C Clear ENG 8:47 PM 11/17/2024

```
main()
{
    int arr[4] = {300, 302, 304, 308, 3016};

    printf("%d", arr[0]);

    arr[3] = 37;

    arr[2] = 73;

    printf("%d", arr[0] + arr[3]);

    printf("%d", 3[arr]);

    return 0;
}
```

Size = 4 integers

Excess : Warning

Ignore

→ 300

300 + 37 = 337

→ 37

Allowed → arr[3]

arr[0]	arr[1]	arr[2]	arr[3]
300	302	304 → 73	308 → 37

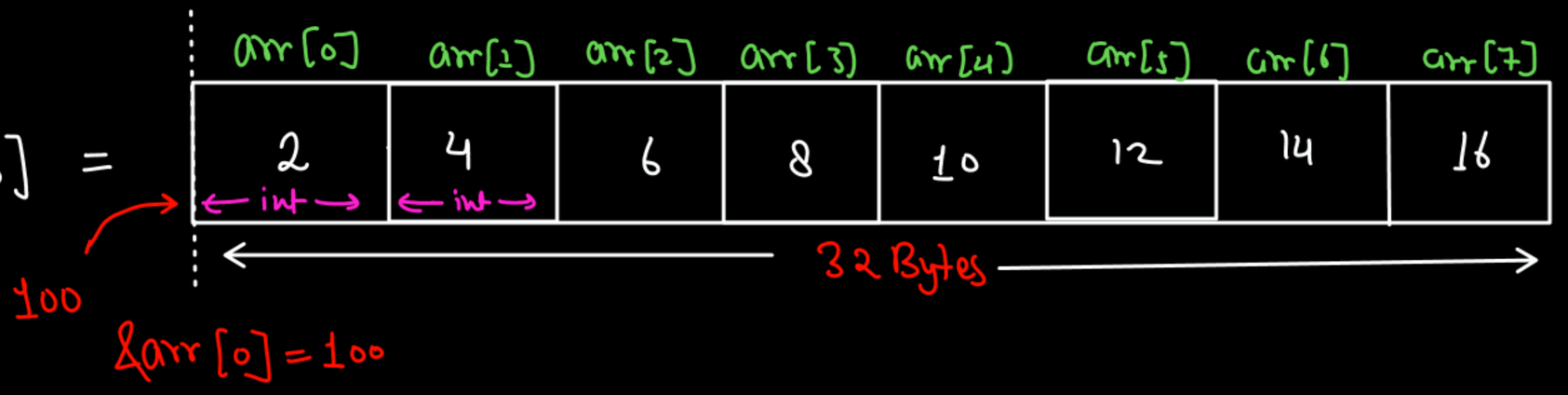
arr =

3[arr] = arr[3] ← Same

Array Addressing

int = 4 Bytes (64bit)

```
int arr[8] = { 2, 4, 6, 8, 10, 12, 14, 16 }
```



RAM

Byte Addressable

[1 cell capacity = 1 Byte]

int → 4 cells (int = 4 Bytes)

Size of array (arr) = No. of elements * (size of datatype)

8 * 4 = 32 bytes

```
printf("%d", arr);
```

↓

Address of 1st element

arr = &arr[0]

↓

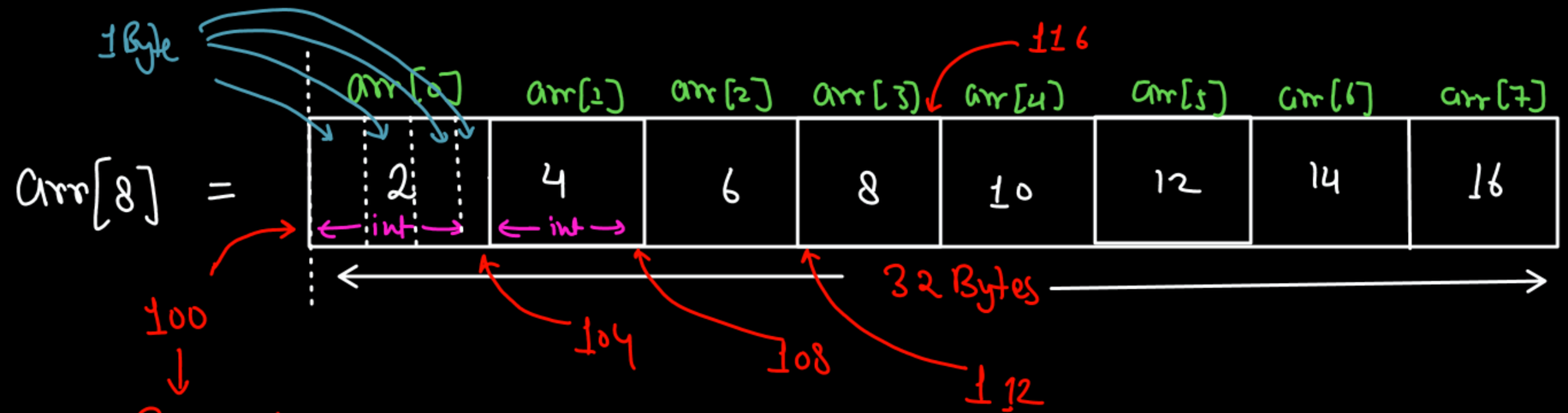
Base address

Address of 1st element of array (arr)

arr ← array → sizeof (arr)
↳ 32 Bytes

arr[0] ← array element
↳ sizeof (arr[0])
↳ 4 Bytes

The value of arr & arr[0] is same but size is different

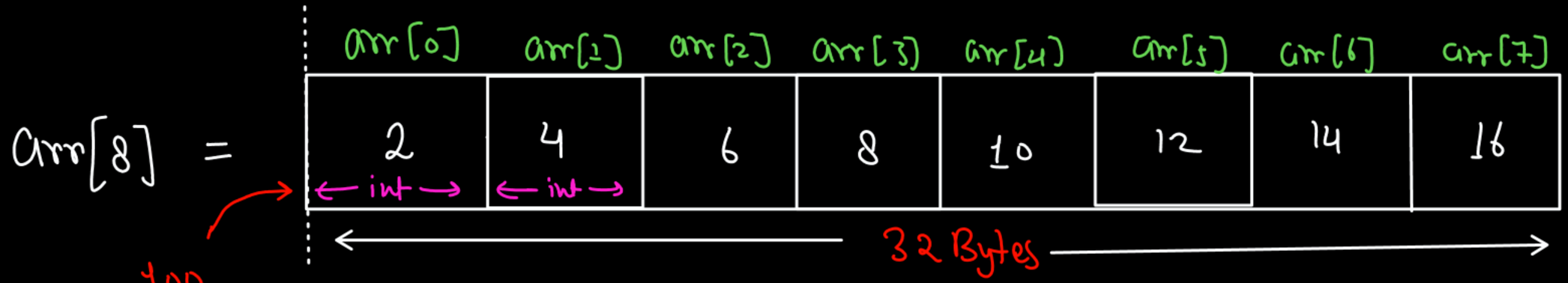


↓

Base address = &arr[0]

1st element = 100

2nd element ⇒ 1st element से 4 Bytes दूर होगा



&arr[0] = 100

index ← offset ← दूरी (Base Address)

&arr[0] = 100

&arr[1] = &arr[0] + sizeof(int)

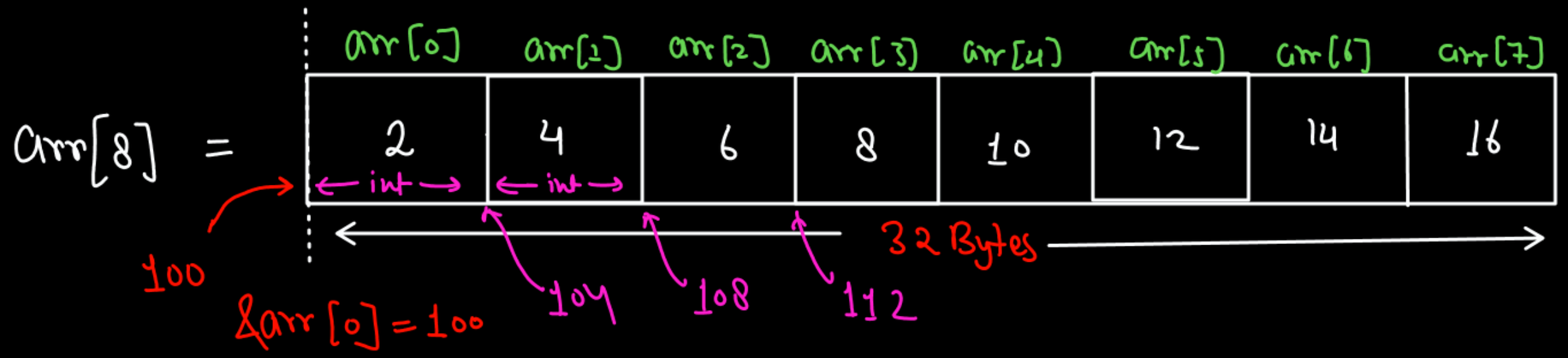
100 + 4 = 104

&arr[2] = &arr[1] + sizeof(int)

104 + 4 = 108

⋮

&arr[n] = &arr[n-1] + sizeof(datatype)



address of element = Base Address + offset (index)

&arr[3] = 100 + 3 (Pointer Arithmetic Rule)

⇓

100 + 3 * sizeof(datatype of arr)

100 + 3 * 4

100 + 12

&arr[3] = 112

arr = 100

&arr[0] = B.A + offset (index)

100 + 0

= 100 + 0 * 4

= 100

arr = &arr[0]

File Edit Selection View Go Run Terminal Help

Code

arrays01.c | arrays2.c | array3.c

array3.c > main()

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main(){
    // int arr[] ={16,156,51,46,67,41,86};
    // printf("%d\n", arr); //base address
    // printf("%d\n", &arr[0]); //address of 1st element (index 0)
    int arr[4] = {40,80,120,160};
    printf("%d \n", arr); //print the address of 1st element
    printf("%d \n", &arr[0]); //print the address of 1st element
    printf("%d \n", &arr[0]+1);
    printf("%d \n", arr+2); // base address + 2 = base address + 2 * size(type)
                            // 6422048 + 2*4 = 6422048 + 8 = 6422056

    return 0;
}
```

```
6422048
6422048
6422052
6422056
PS>
```

Address of (n^{th}) index of an array = Base Address + offset Value (n) * size of (arr[0])

= Base Address + n * size of (arr[0])