

LECTURE - 41

STRUCTURES IN C (PART 03)

'C' PROGRAMMING

- Declare
- Define ← Variables
- Nesting of Struct
- Size of Structure

↳ Define → struct StructureName {
 };
 ≡
 Other Methods

Methods to declare the Structure

(A)

```
struct student {  
    int rno;  
    char name [30];  
    int marks;  
};
```

(B)

Combined declaration & definition

```
struct student {  
    int rno;  
    char name [30];  
    int marks;  
} s1, s2;
```

Global Variables




(C) Using typedef keyword

```
typedef struct student {  
    int rno;  
    char name [30];  
    int marks;  
} s;
```

we can use 's' instead of student



```
int main() {  
    struct student Aditi;  
    s Aditi;  
}
```



① Anonymous Structure

```
type def struct {  
    int rno;  
    char name [30];  
    int marks;  
} Student;
```

No name declared (pointing to the opening curly brace of the struct definition)

pet name (pointing to the closing curly brace of the struct definition)

```
int main() {  
    Student s1 = { 2, "Arav", 85 };
```

Pet name (pointing to the variable name 's1')

Pure Anonymous (pointing to the struct definition)

```
struct {  
    int rno;  
    char name [30];  
    int marks;  
} s1, s2;
```

only two variables (pointing to the variable names 's1' and 's2')

```
int main() {  
    s1 = { 3, "Arpit", 75 };  
    s2 = { 4, "Akshansh", 65 };
```

(e) Inside function:

↳ we can define a structure within a fⁿ.

→ Local Scope of the structure.

```
void function ( ) {
```

```
    struct LocalStructure {
```

```
        int a;
```

```
        int b;
```

```
    };
```

```
    struct LocalStructure x = { a10, b20 };
```

```
    printf("%d, %d", x.a, x.b);
```

```
}
```


① Structure pointer :

```
Struct Employee {  
    int empid;  
    char name[30];  
    float Salary;  
};
```

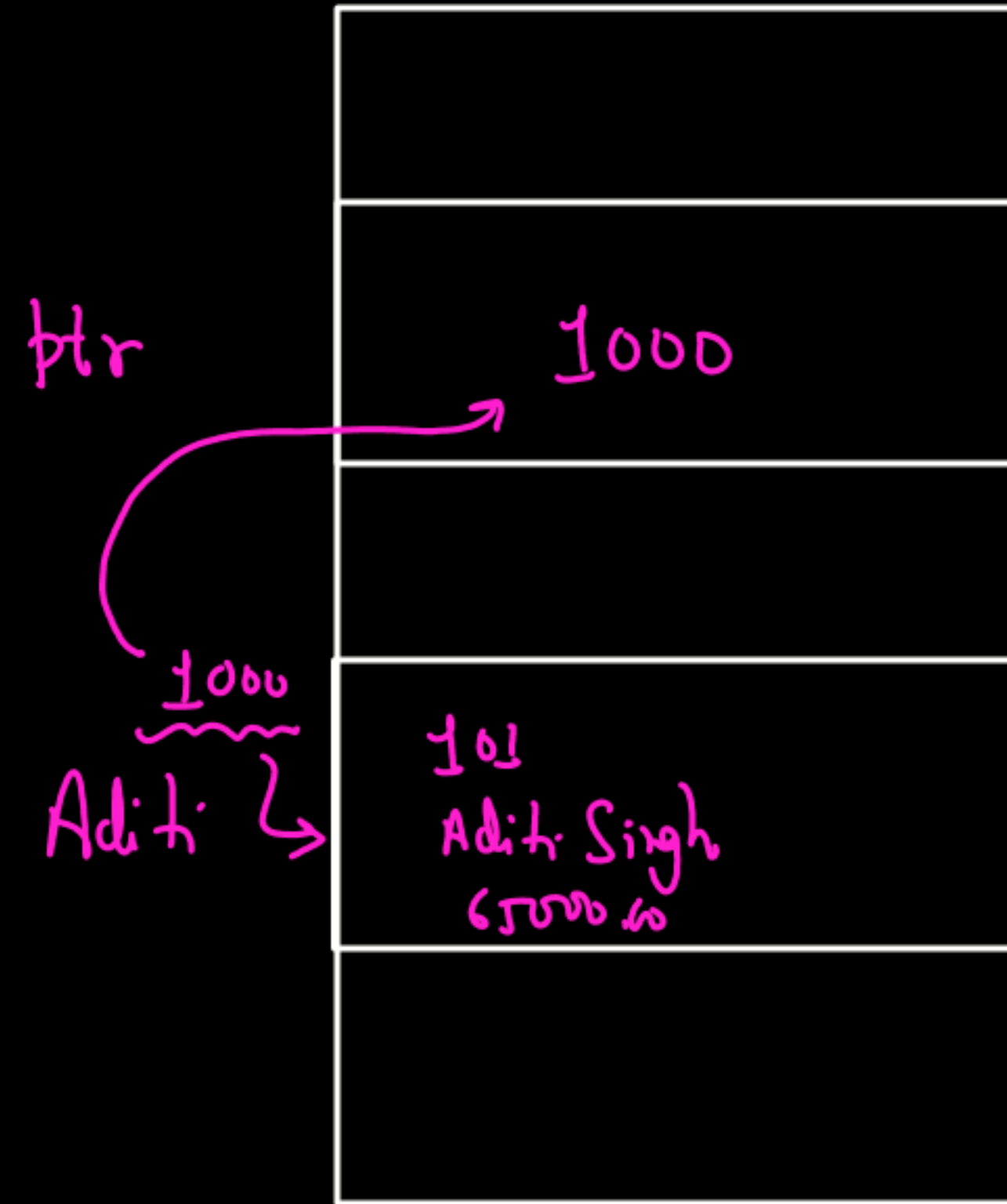
```
int main ( ) {
```

```
    Struct Employee *ptr; ← Garbage Value
```

```
    Struct Employee Aditi = { 101, "Aditi Singh", 65000.00 }
```

```
    ptr = &Aditi ← add. of Aditi
```

```
    :  
    :  
    : continue
```



← Struct Employee

int * ptr
int n = 200
ptr = &n

⋮
printf ("%f", Aditi. salary); ← 65000.000000

printf ("%f", ptr → salary); ← 65000.000000

↖ Arrow operator

↓
dereference the structure variable
through pointer

int x = 200
int *ptr = &x
printf ("%d", *ptr)
 ↓
 200
 =

```
1  #include<stdio.h>
2
3  struct Employee ✓
4  {
5      int empid; ✓
6      char name[30]; ✓
7      float salary; ✓
8  };
9
10 int main(){
11     struct Employee e1 = {1, "Pragyaan Rover", 50000.00};
12     struct Employee *ptr = &e1;
13     printf("Name: %s\n", e1.name);
14     printf("Name (using pointer): %s\n", ptr->name);
15     return 0;
16 }
```

Handwritten annotations:

- Red arrows point from the labels *empid*, *name*, and *Salary* to the corresponding values in the struct initialization: *empid* points to `1`, *name* points to `"Pragyaan Rover"`, and *Salary* points to `50000.00`.
- A red arrow points from the text *member Acc. op* to the `e1.name` access in line 13.
- A red arrow points from the text *Arrow operator* to the `ptr->` access in line 14.

```
ramming in C\Code> gcc structurepointer.c -o structurepointer ; if ($?) { .\structurepointer }
Name: Pragyaan Rover
Name (using pointer): Pragyaan Rover
PS C:\Users\sagar\OneDrive\Desktop\Daily Notes\Aditi Chand\Programming in C\Code> 
```

* Arrow operator is used to access the members of the Structure Variable

Pointer inside Structure:

← Hold every datatype

x
↓
int * return ← &r

```
struct Employee {  
    int empid;  
    int name;  
    int salary;  
    int * poi ;  
}
```

← poi is an integer pointer that can hold the address of a integer variable

```
int main() {  
    int r = 7;  
    struct Employee x;  
    x.empid = 101;  
    strcpy(x.name, "Deepak");
```

```
    x.salary = 30000.0;  
    x.poi = &r;  
    return 0;  
}
```

pointerinstructure.c > main()

```
1  #include<stdio.h>
2  #include<string.h>
3  struct Employee
4  {
5      int empid;
6      char name[30];
7      float salary;
8      int *roi;
9  };
10 int main(){
11     int r = 7;
12     struct Employee e1;
13     e1.empid = 5;
14     strcpy(e1.name, "Apoorva");
15     e1.salary = 5000;
16     e1.roi = &r;
17     printf("%d\n", e1.roi);
18     printf("%d\n", *(e1.roi));
19     // printf("%d\n", e1->roi); //invalid
20     // printf("%d\n", roi->e1); //invalid
21     return 0;
22 }
```

Variable (pointing to line 8)

add of 'r' (pointing to line 16)

value at 'r' (pointing to line 18)

Addr

Value

```
Code> gcc pointerinstruct
terinstructure ; if ($?)
structure }
6422060 ✓
7 ✓
PS C:\Users\sagar\OneDr
ly Notes\Aditi Chand\Pro
Code> 
```

```
struct data {
```

```
  int x; ← int
```

```
  float y; ← float
```

```
  struct data * ptr; ← Pointer to the structure
```

```
}
```

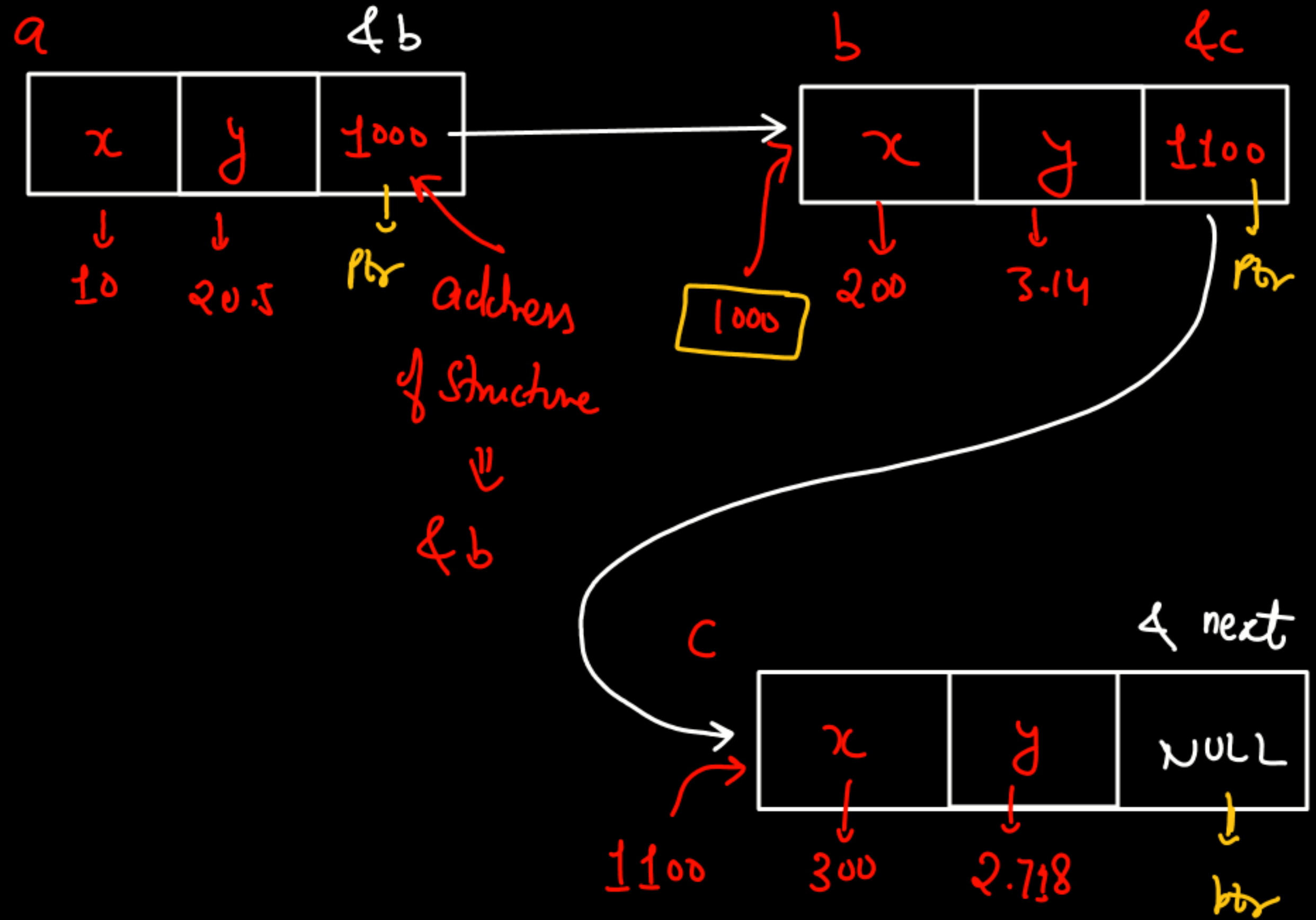
Self Referencing
Structure pointer

Next Course ← Data Structure
↓
Linked list

```
struct data a;
```

```
struct data b;
```

```
struct data c;
```



$a \xrightarrow{b} y : a.ptr \rightarrow y$

$a \xrightarrow{c} y : a.ptr \rightarrow ptr \rightarrow y \leftarrow 2.718$

tomorrow → 2 Jan 25

DP

↳ maximum

⇒ DMA