CS & IT ENGINEERING

[redacted]

'C' Programming

Functions

Lecture No.- 02

By- Satya sir

Recap of Previous Lecture

- Functions
  - Function Definition – A single stmt (or) group that Performs a Task
  - Types of functions – Predefined, User-defined
  - Properties of function – Name, arguments, Return type, Body
  - Declaration, Definition, Calling
  - Function Prototypes

Arguments	Return Values
✓	✓
✓	✗
✗	✓
✗	✗

Topics to be Covered

- Call-by-Value Vs Call-by-Reference
- Recursion
- Types of Recursion
- Head, Tail Recursion

Topic : Recursion - 1

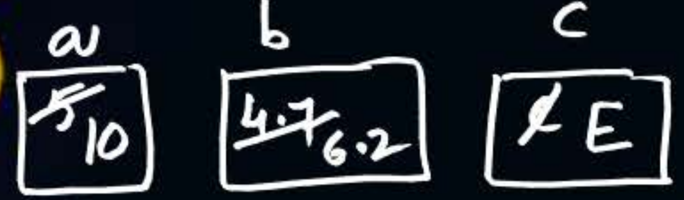
Arguments (or) Parameters

- Formal (or) Dummy arguments = Used while definition of function
- Actual arguments = Used while calling a function

Ex:

```
 void fun (int x, int y) //Callee
                  ──────────→ Formal arguments
{
    Printf("%d", x*y);
}
 Void main( ) // Caller
 {
    int i=7;
    fun( i, 5 );
 }       Actual arguments
```

Topic : Recursion - 1

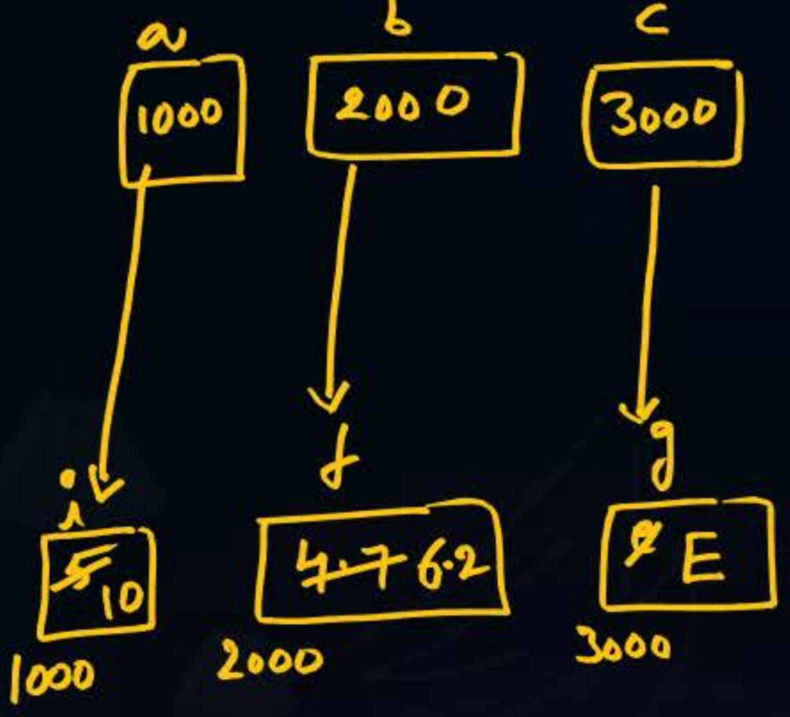


Call-by-Value ⇒ formal arguments will be Effected.

```
           5            4.7          'e'
Void fun(int a, float b, Char c)
{
    a = a*2;      // a=10
    b = b+1.5;    // b=6.2
    c = c-32;     // c='e'-32='E'
}
void main( )
{
    int i=5;
    float f=4.7;
    char g='e';
    Printf("%i, %f, %c", i,f,g);      // 5, 4.7, e
    fun(i,f,g); //Call-by-Value
    Printf("\n %i, %f, %c", i,f,g);   // 5, 4.7, e
}
```

Call-by-Reference // Actual arguments gets Effected.

```
Void fun(int *a, float *b, char *c)
{
    *a = *a * 2;
    *b = *b + 1.5;
    *c = *c - 32;
}
Void main( )
{
    int i=5;
    float f=4.7;
    char g='e';
    Printf("%d %f %c", i,f,g);   // 5, 4.7, e
    fun(&i, &f, &g); // Call-by-Reference
    Printf("\n %d %f %c", i,f,g); // 10 6.2 E
}
```

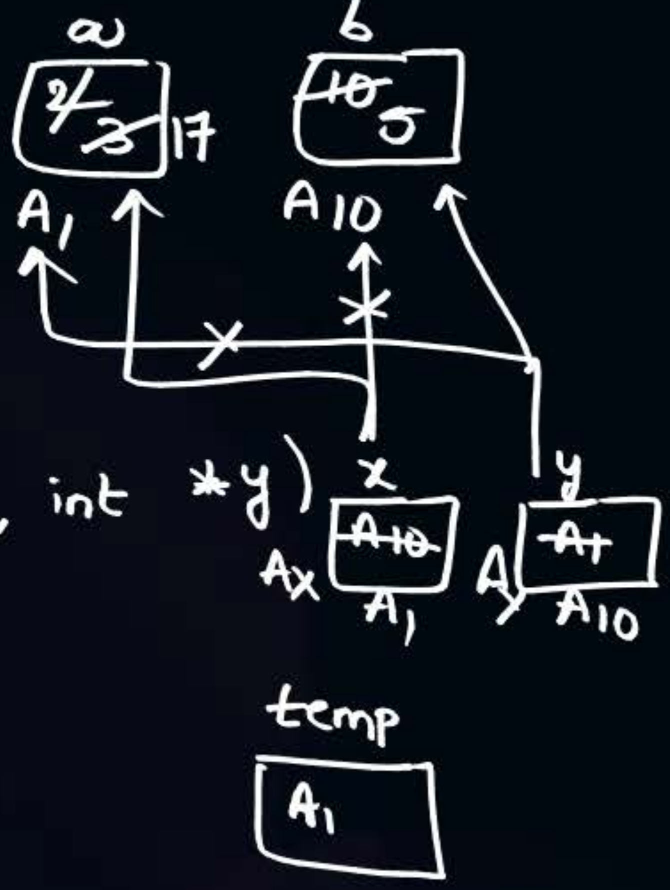


Topic : Recursion - 1

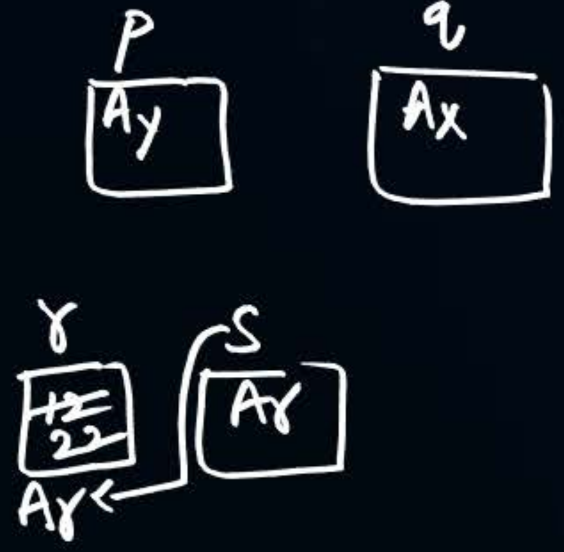
Examples :

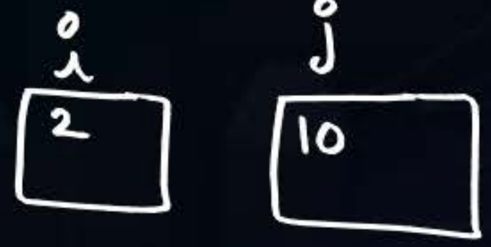
Ex.1:

```
Void f(int a, int b)
{
    a = a+3;
    b = b-5;
    g(&b, &a);
}
Void g(int **x, int *y)
{
    int *temp;
    temp = y;
    y = x;
    x = temp;
    fun(&y, &x);
}
```



```
fun(int **p, int **q)
{
    int r = 12, *s;
    s = &r;
    **q = **p + *s;
    *s = **q + **p;
}
Void main(   )
{
    int i = 2, j = 10;
    f(i, j);
    Printf("%d %d", i, j);
}
```





o/p: 2 10

Topic : Recursion - 1

Ex:2

a: ~~10~~ ~~10~~ — A1 — ~~-20~~ ~~-41~~
b: ~~20~~ — A10 — ~~-20~~ ~~-20~~ -23
c: ~~30~~ — A100 — ~~-200~~ ~~-20~~ -18

```
→ Void main(   )
{
    int a=10, b=20, C=30;
    f(&b, &a, &c); ✓
    g(&c, &b, &a); ✓
    h(&a, &c, &b); ✓
  Printf("%d, %d, %d", a, b, c);
}
Void f(int *a, int *b, int *c)
{
    *b = *c - *a;
    *a = *b - *c;
    *c = *a * *b;
}
```

$*b = 30-20 = 10$

$*a = 10-30 = -20$

$*c = -20 * 10 = -200$

a: $\boxed{A_{10}}$ b: $\boxed{A_1}$ c: $\boxed{A_{100}}$

```
Void g(int *a, int *b, int *c)
{
    *c = *b;
    *a = *c;
    *b = *a;
}
```

a: $\boxed{A_{100}}$ b: $\boxed{A_{10}}$ c: $\boxed{A_1}$

```
Void h(int *a, Int *b, int *c)
{
    *b = *b+2;        -20+2 = -18
    *c = *c - 3;      -20 -3 = -23
    *a = *b + *c;     -18 + -23 = -41
}
```

a: $\boxed{A_1}$ b: $\boxed{A_{100}}$ c: $\boxed{A_{10}}$

O/p : -41 -23 -18

Topic : Recursion - 1

Recursion

- In Programming, To Execute one or more statements repeatedly, It is Possible through either of 2 ways :

  ① Iteration [Control statements, Loops] Ex: fibonacci series, MCM, TSP

  ② Recursion Towers of Hanoi, Tree Traversals, Merge sort, Quick Sort - - -

- Based on Type of Problem, Iteration (or) Recursion is chosen.

- Recursion makes, Programming Simple, Comfortable.

Topic : Recursion - 1

Recursion ? = The Process of calling itself.

- A function, which call itself, is said to be Recursive function.

Ex:

```
Void main( )
{
    fun(5);
}
void fun( int x )
{   if(x>1)
    { Printf("%d", x);
      fun(x-1);
    }
    else return;
}
```

CAUTION: Infinite Calling need to be taken Care of.

Topic : Recursion - 1

Recursive functions, generally contains 2 Types of Code:

1) Base Case ⇒ The statement, Expression, written to terminate Recursive Calling.

2) Recursive Case ⇒ The statement / Expression, where Recursive Calling happens.

Ex:

```
void display( int x )
{
    if (x<0)            // Base Case
      return;
    Printf("HAI");      // Recursive Case
    display(x-1);
}
```

2 mins Summary

Types of Recursion :

2 Types :
- Direct Recursion (Calls itself)
  - Head Recursion
  - Tail Recursion
  - Tree Recursion
  - Nested Recursion
- Indirect Recursion (Calls itself through some other function)

To be contd . . .

THANK - YOU