# CS & IT ENGINEERING

## C Programming

## Functions

Lecture No.- 01

By- Satya sir

## Recap of Previous Lecture

- Arrays and Pointers

  - Pointers & 2-D arrays

- Let A be 2-D array

$(A+i) == *(A+i) == \& A[i][\ ]$

$*(A+i)+j == (A+i)+j == \& A[i][j]$

$*(*(A+i)+j) == A[i][j]$

## Topics to be Covered

- Array of Pointers
- Functions
  - Definition
  - Declaration Vs Definition Vs Calling
  - Caller, callee
  - Function Prototypes

## Topic : Functions, Prototypes

- Pointers & Arrays can be implemented together in 3 ways

1) Pointer to an individual Element of array — datatype *Pointer;

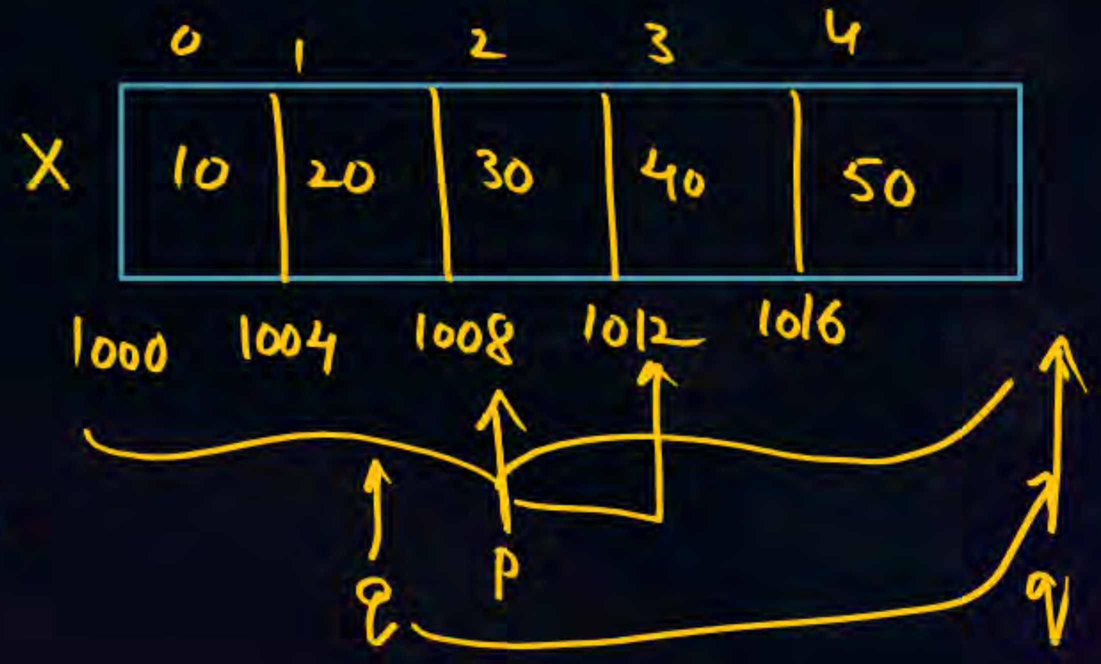
2) Pointer to whole array — datatype (*Pointer)[size];

3) Array of Pointers — datatype *Pointer[size];

Ex: int x[5] = {10, 20, 30, 40, 50};

```
int *P, (*q)[5];
    P = x[2];
    q = &x;
    P++; // P Points to x[3], 1012 address
    q++; // q Points to 1020
```

Let B = 1000, 1 int = 4 Bytes



## Topic : Functions, Prototypes

### Array of Pointers

Ex: `int i=5, j[3]={11,22,33}, K=-1, x[5]={10,20,30,40,50};`

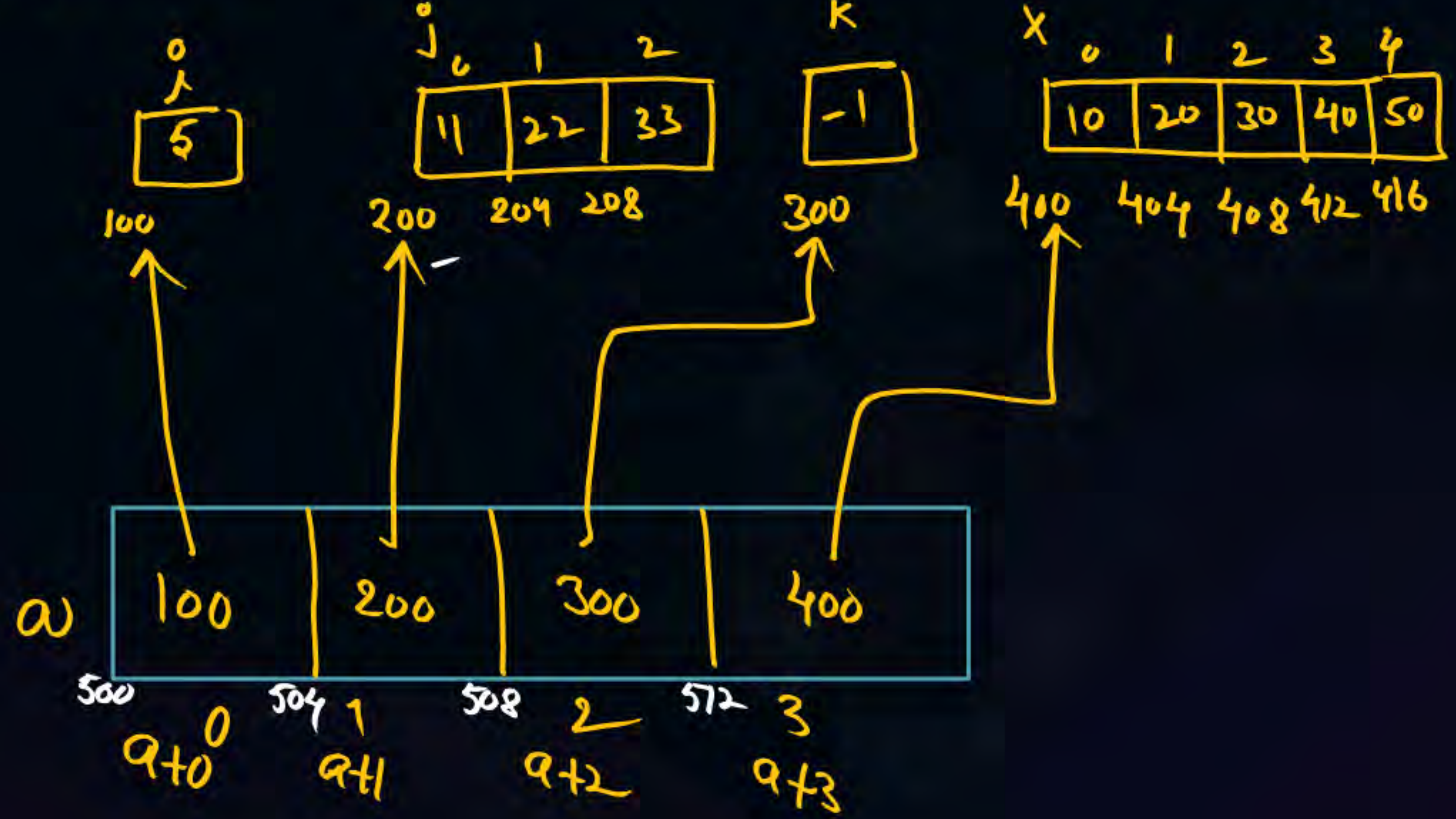
`int *a[4]; // Array of 4 Pointers`

```
a[0] = &i;
a[1] = &j[0];
a[2] = &k;
a[3] = &x[0];
```

$\&j[1] = *(a+1)+1$

$j[1] = *(*(a+1)+1)$

$\&x[3] = *(a+3)+3$



## Topic : Functions, Prototypes

## Functions

Definition : A single statement (or) group of statements that performs specific Sub Task, is called as function.

Example :

Task: Get GATE AIR < 100

1. Attend all subjects classes regularly
2. understand all concepts
3. Attempt DPPS, WTs
4. Practice of PYQ's, Test Series
5. Periodic Revision
6. Doubts resolution
7. Subject-wise Tests
8. Full length Mock Tests
9. Preparing Lecture Notes
10. Preparing short Notes
11. Perform well in Exam

## Topic : Functions, Prototypes

Example :

Task: To Print Welcome Message

Function

```
Printf("Welcome To Programming");
```

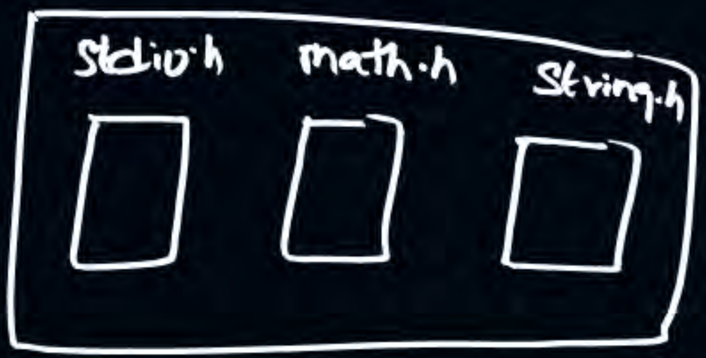
Task : To find factorial of given number

Function

```
1. int n, result, i;
2.  Scanf("%d", &n);
3.  for(i=1; i<=n; i++)
4.      result = result * i;
5.  Printf(" Factorial is %d", result);
```

## Topic : Functions, Prototypes

C Library



Types of Functions : 2 Types of functions

① Pre-defined (or) System-defined (or) Library functions

Ex: Printf( ), scanf( ), Strlen( ), Strcpy( ), sizeof( ), Pow( ), abs( ), clrscr( ), draw( ) - - -

② User-defined functions : Functions created by user.

Ex: main( ), foo( ), fun( ), f1( ), f2( ) - - -

- Name of function must be a Valid identifier

```
Printf (str) ;
int Printf(        )
{
    ___
    ___
    ___
}   ___
```

## Topic : Functions, Prototypes

// To avail function Services

**Function Declaration**

Returntype Name (arguments type);

Ex:

```
1) int X (void);
2) void fun( int, int );
```

**Function Definition**

```
Returntype Name (arguments)
{
    // Body
}
```

Ex:

```
      void fun (int a, int b)
      {
Block    int c ;
         c = a/b;
         Printf("%d", c);
      }
```

**Function Calling**

Name (arguments);

```
fun (x, y);
fun ( x, 5 );
fun( 9, 5 );
```

Every Function comprises of 4 Properties

1. Name
Parameters (or) 2. Arguments ( inputs ) // optional always
3. Body // Code inside Function block
4. Returntype (output type)

## Topic : Functions, Prototypes

//Declaration is Mandatory, when a function is called before, it's Definition. (Declare any where before calling).

Ex:

```
Void fun(int, int);
 Void main(    ) //Caller
{
      int i=5, j=7;
 →    fun(i,j);
}
 Void fun(int x, int y) //Callee
 {
      int k;
      k = x+ y %x;
      Printf(" k=%d", k);
 }
```

Actual arguments

Formal arg (or) Dummy arguments

```
Void fun( int x, int y) //Callee
{  int k;
   k= x + y % x;
   Printf(" k= %d", k);
}
Void main(  ) //Caller
{  int i=5, j=7;
   fun(i,j);
}
```

[A function can be either Caller or Callee or both]

Caller function (or) Calling function
- The function, that calls another

Callee function (or) Called function
- The function, that is called by another

Formal Parameters: Parameters used while definition

Actual Parameters: Parameters, used while calling

$$\begin{tikzcd}\text{Caller} \arrow[r, bend left, "\text{Arguments}"] & \text{Callee} \arrow[l, bend left, "\text{Return Values}"]\end{tikzcd}$$

## Function Prototypes

1. Functions with arguments, with return Values
2. Functions with arguments, without return Values
3. Functions without argument, without return Values
4. Functions without argument, with return Values.

## Examples

```
                          //callee
Void fun(int x, int y)
{
      int z;
      z = x+y;
      Printf("%d", z);
}
void main( ) //Caller
{
      fun(5,6);
}
```

With argument, without return value

```
void fun( ) //callee
{
   Printf("welcome");
}

void main( ) //caller
{
     fun( );
}
```

Without arguments, without return values

```
int fun(int x, int y) //callee
{
     int z;
     z = x+y;
     return z;
}
void main( ) //caller
{ int i=5, j=6;
  Printf("%d", fun(i, j));
}
```

With arguments, with return value

```
int fun( ) //callee
{ int x, y, z;
  Scanf("%d%d", &x, &y);
  z = x+y;
  return z;
}

void main( ) //Caller
{
  Printf("%d", fun( ));
}
```

Without arguments, with return values.

## 2 mins Summary

- Array of Pointers
- Functions
  - Declaration
  - Definition
  - Calling
  - Prototypes

To be contd . . .

THANK - YOU