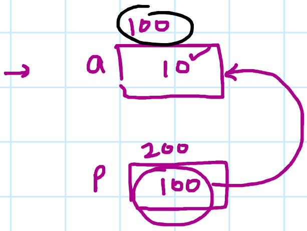
# Day 05

21 March 2025 08:37 AM

## Pointer :→

int a; →



int *p;

Referencing → p = &a;

Dereference → *p = 10;

printf("%d", *p); → 10

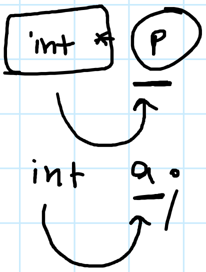
## Rules :→

### ① Type must be same :-

int a;

float b;

| int *p1; | float *p2; |
|---|---|
| ✓ p1 = &a; | p2 = &b; ✓ |
| ✗ p1 = &b; | p2 = &a; ✗ |



✗ p1 = (int*)&b; ✓

*p1 = 7.5;

printf("%f", b); → 0.000000

### ② size of a pointer :-

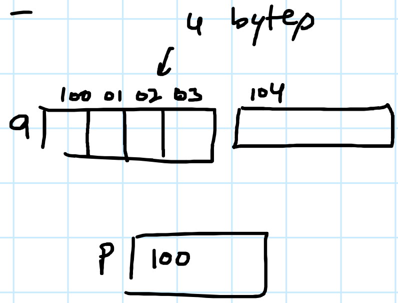
int *p1; printf("%d", sizeof(p1)); →

float *p2; printf("%d", sizeof(p2)); → } → same as int

char *p3; printf("%d", sizeof(p3)); →

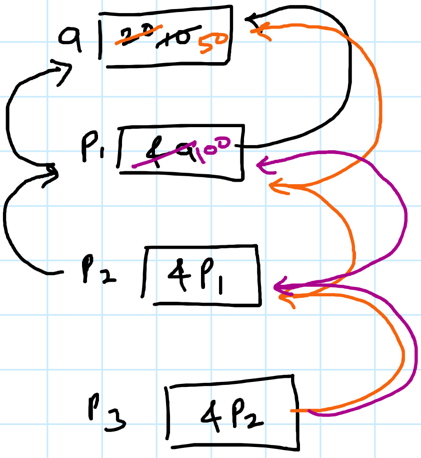
```
char *P3;     Printf("%d", sizeof(P3)); →  ⎤    int
double *P4;   Printf("%d", sizeof(P4)); →  ⎦
```

## ③ inc / dec of a pointer :-



```
int a;

int *p = &a;

p++;

printf("%d", p);  →104

p--;

printf("%d", p);  → 100
```

## ④ Pointer to variable :→



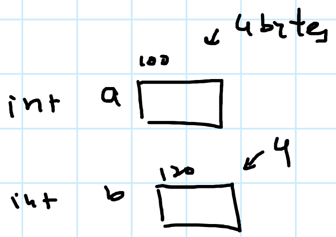
```
int a;

int *P1 = &a;

*P1 = 10;   → pointer to variable.

int **P2 = &P1;

**P2 = 20;    pointer to pointer to variable

int ***P3 = &P2;              int b;

***P3 = 50;

**P3 = ~~100~~ &b;
```

Allowed

- ✓ p++
- ✓ p--
- p += 2
- p -= 2
- $\boxed{p_1 - p_2}$

not allowed

- $p_1 + p_2$ ✗
- $p_1 * p_2$ ✗
- $p_1 / p_2$ ✗
- $p_1 \% p_2$ ✗



$p_1 = \&a$

$p_2 = \&b$

$5 \leftarrow p_1 - p_2$

$20/4 \rightarrow \boxed{5}$

## Types of pointer

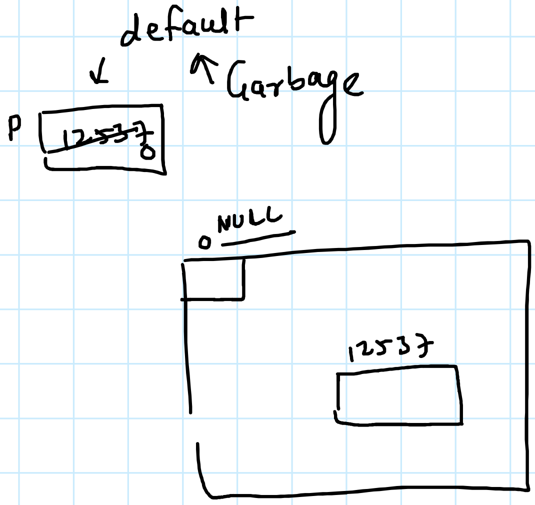
### ① void pointer :→

```
void *p;
int a;          | float b;
✓ p = &a;       | p = &b; ✓
✗ *p = 10       | *p = 7.5 ✗
```

### ② NULL Pointer

```
int *p;
p = NULL;
```

$\boxed{\#define\ NULL\ 0}$



## Function :→

Pre defined                                        user defined

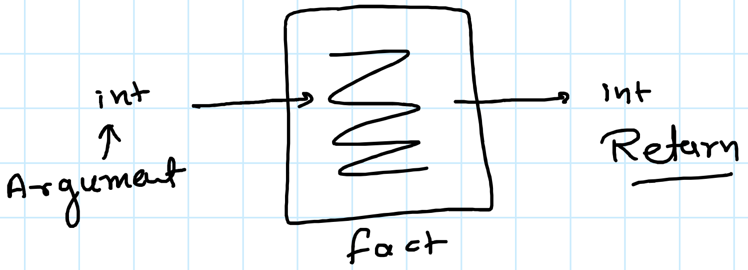
## User defined function →

steps

1. Function declaration
2. function calling
3. function Definition

sum of fact of 3 nums

a! + b! + c!



```
int  fact ( int ) ;      ← declaration
```

Return type, func name, Argument type

a 4
b 5
c 6

```
int main ()
{
        int a, b, c, sum;
        pri ______ );
        sca ______ &a, &b, &c);
                  24         120        720
        sum = ~~fact (a)~~ + ~~fact(b)~~ + ~~fact (c)~~ ;
                                         ↑ func Calling
        printf ("%d", sum);  → 864
        return 0;
}
```

4

24

```
    return 0;
}
int fact (int n)
{
    int temp = 1;
    while (n)
    {
        temp *= n;
        n--;
    }
    return temp;
}
```

4

24

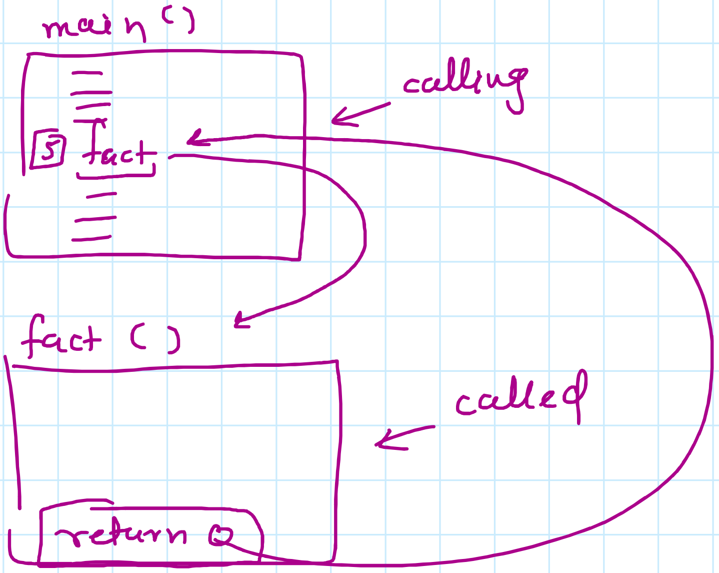
n 4

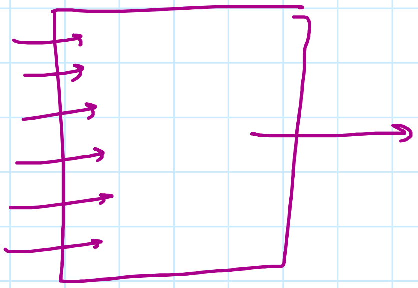
temp 24

← definition

return → Jump statement

Transfer control from called function to calling function





* A function can accept any number of arguments

* A function can accept any number of arguments but return max one value.

void

## swap 2 nums using function

```c
#include<stdio.h>
void swap(int a, int b)       //a=20  b=10
{
    int temp = a;
    a=b;
    b=temp;
    printf("%d %d",a,b);
    return;
}
int main()
{
    int a,b;
    printf("Enter 2 numbers:");
    scanf("%d%d",&a,&b);
    swap(a,b);       //a=10 b=20
    return 0;
}
```

formal parameters

call by value

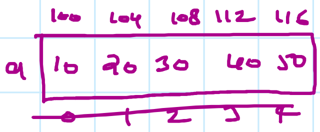
Actual parameters

## call by Address :

```c
#include<stdio.h>
void swap(int *p1, int *p2)      //p1=&a  p2=&b
{
    int temp = *p1;         //temp = a=10
    *p1 = *p2;
    *p2 = temp;
    return;
}
int main()
{
    int a,b;
    printf("Enter 2 numbers:");
    scanf("%d%d",&a,&b);
    swap(&a,&b);      //a=20 b=10
    printf("%d %d",a,b);
    return 0;
}
```

## Pointer - array :→

int a[5] = {10, 20, 30, 40, 50};



a[2] → *(a + 2)

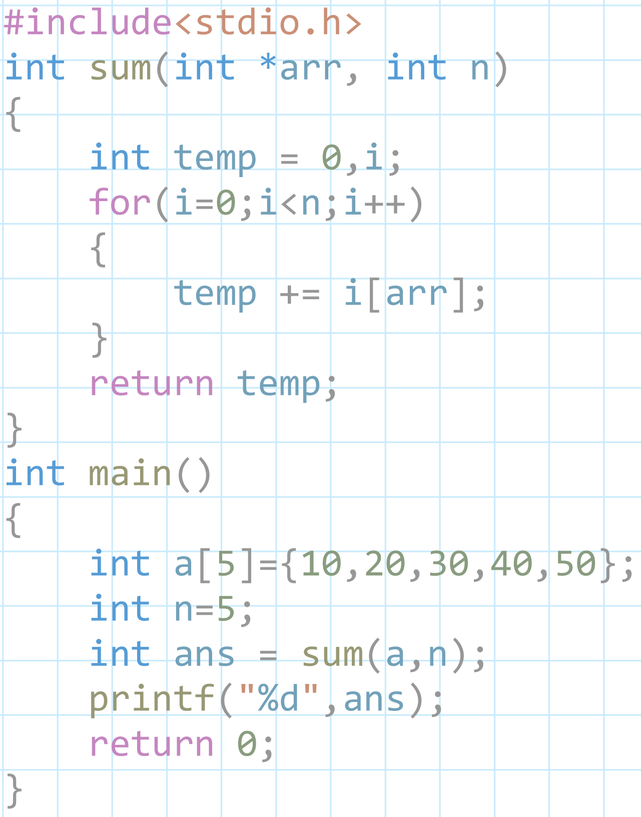
100 + 2 → *(108)

int *p = a;

p[2] → *(p+2)

100 +2 → *(108)

## sum of an array using function

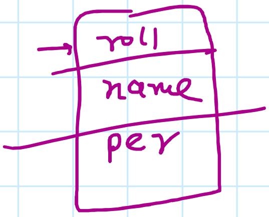


```
#include<stdio.h>
int sum(int *arr, int n)
{
    int temp = 0,i;
    for(i=0;i<n;i++)
    {
        temp += i[arr];
    }
    return temp;
}
int main()
{
    int a[5]={10,20,30,40,50};
    int n=5;
    int ans = sum(a,n);
    printf("%d",ans);
    return 0;
}
```

## structure :→

```
struct structure-name
{
        member1;
        member2;
           :
};
```

student



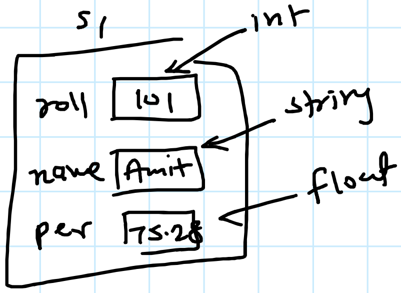
```
struct student
{
    int roll;
    char name[20];
    float per;
};
```

No memory

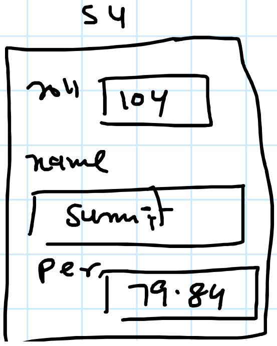
```
struct student s1, s2, s3;
```

objects

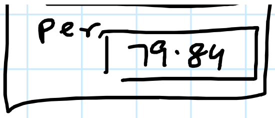
Data type

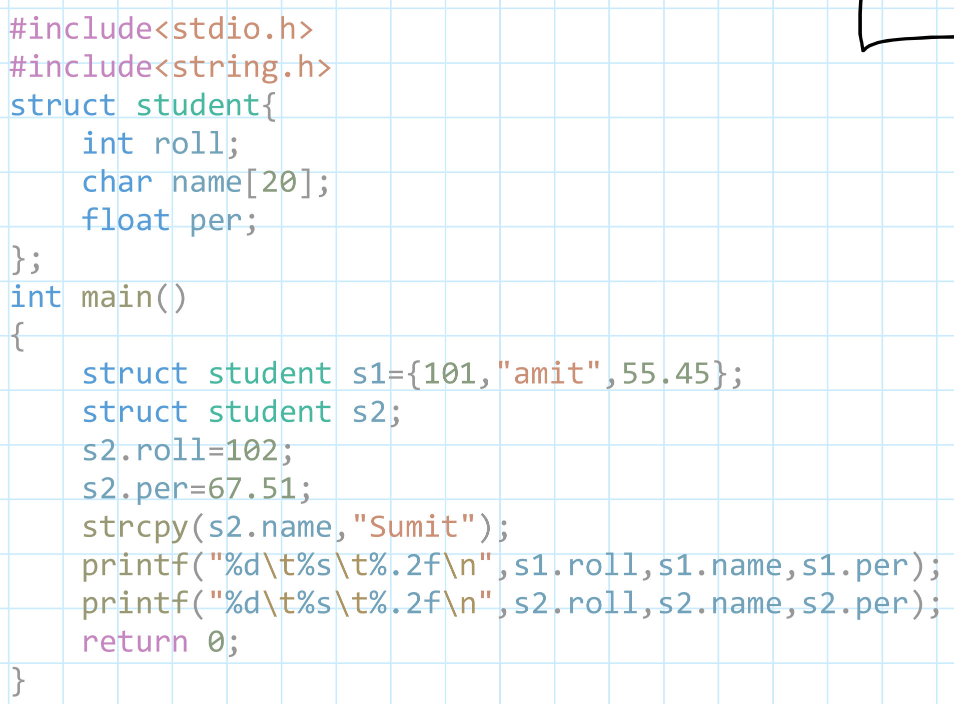


```
s1.roll = 101;
s1.name = "Amit"; ✗
✓ strcpy(s1.name, "Amit");
s1.per = 75.28;
struct student s4 = {104, "Sumit", 79.84};
```



```
#include<stdio.h>
```





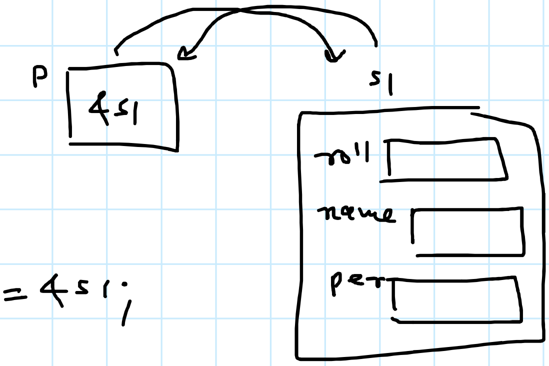
```
#include<stdio.h>
#include<string.h>
struct student{
    int roll;
    char name[20];
    float per;
};
int main()
{
    struct student s1={101,"amit",55.45};
    struct student s2;
    s2.roll=102;
    s2.per=67.51;
    strcpy(s2.name,"Sumit");
    printf("%d\t%s\t%.2f\n",s1.roll,s1.name,s1.per);
    printf("%d\t%s\t%.2f\n",s2.roll,s2.name,s2.per);
    return 0;
}
```

```
#include<stdio.h>
#include<string.h>
struct student{
    int roll;
    char name[20];
    float per;
};
struct student input()
{
    struct student temp;
    printf("Enter roll, name and per of a student");
    scanf("%d",&temp.roll);
    fflush(stdin);
    gets(temp.name);
    scanf("%f",&temp.per);
    return temp;
}
int main()
{
    struct student s1,s2;
    s1 = input();
    s2 = input();

    printf("%d\t%s\t%.2f\n",s1.roll,s1.name,s1.per);
    printf("%d\t%s\t%.2f\n",s2.roll,s2.name,s2.per);
    return 0;
```

}

## Pointer

struct student s1



struct student *p = &s1;

s1.roll → object to member

p->roll → pointer to member

(*p). roll

```c
#include<stdio.h>
#include<string.h>
struct student{
    int roll;
    char name[20];
    float per;
};
void input(struct student *p)
{
    printf("Enter roll, name and per of a student");
    scanf("%d",&p->roll);
    fflush(stdin);
    gets(p->name);
    scanf("%f",&p->per);
}
int main()
{
    struct student s1,s2;
    input(&s1);
    input(&s2);

    printf("%d\t%s\t%.2f\n",s1.roll,s1.name,s1.per);
    printf("%d\t%s\t%.2f\n",s2.roll,s2.name,s2.per);
    return 0;
}
```

| i | | | | | | | | | | j | K (i-1) | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5-5 | 1-7 (4) | 5-5 |
| 4 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 5-4 | 1-5 (3) | 4-5 |
| 3 | 5 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | 5-3 | 1-3 (2) | 3-5 |
| 2 | 5 | 4 | 3 | 2 | 2 | 2 | 3 | 4 | 5 | 5-2 | 1-1 (1) | 2-5 |
| 1 | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 | 5-1 | — | 1-5 |
| 2 | 5 | 4 | 3 | 2 | 2 | 2 | 3 | 4 | 5 | | | |
| 3 | 5 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 5 | | | |
| 4 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | | | |
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | | | |

j K L

i = n to 1
--

i = 2 to n
++

j = n to i

l = i to n
++

K = 1 to $(i-1)*2-1$ ++