C in One Shot

Part – 5

Functions in One Shot

What and Why?

$y = f(x)$ [functions in mathematics]

$y = f(x) = x^2 + 5$

1) Repetition (to avoid) → loops

2) Readable

```c
#include<stdio.h>
void greet(){
    printf("Good Morning\n");
    printf("How are you ?\n");
    return;
}
int main(){
    greet();
    greet();
    greet();
    return 0;
}
```

External function

khatam

call

- Good Morning
- How are you ?
- Good Morning
- How are you ?
- Good Morning
- How are you ?

[5-1]

DRY → do not repeat yourself

Basic syntax

```
fun(){

    // code

}
```

```
main()
return ;
```

starts from main()

→ khatam

How functions work : ek ke andar doosra, doosre ke andar teesra

[5-4]

[5-5]

Kaam ki baate :

1) main() ek hi baar aata hai.
2) Starts with main
3) unlimited functions

Return type : ~~Power function~~ se samajte hai

sum of 2 dea koraise

Sum of 2 numbers

return type

arguments

```
int add(int a, int b){
    return a+b;
}
```

return 7;

a = 3, b = 4

```
main()
{
    int a,b;
    scanf(a), b
    int sum = add(a,b);
}
```

a = 3, b = 4

[5-6]

[5-7]

Library functions [in build functions]

```
printf ("Hello");
scanf ("%d", &n);
sqrt (49);
pow (2,5);
```

[5-8]

2^5

[5-9]

Function Prototype - ~~useless~~

[5-10]-[5-13]

Ques : Combination and Permutation

$${}^{n}C_{r} = \frac{n!}{r! \times (n-r)!}$$

$n = 7$, $r = 3$ $\rightarrow$ $\frac{7!}{3! \cdot 4!}$

```
int combination (int n, int r){

    //code

}
```

[5-14]-[5-16]

$(n-r)!$ = $(n-r) \times (n-r-1)!$

↓ number

Ques : Combination and Permutation

$$ncr = n\,fact / r\,fact * n{-}r\,fact$$

$$= \frac{n\,fact}{r\,fact} \times n{-}r\,fact$$

$${}^{n}C_{r} = \frac{n!}{r! \times (n-r)!}$$

$${}^{7}C_{3} = \frac{7!}{3! \times 4!} = \frac{7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1}{3 \times 2 \times 1 \times 4 \times 3 \times 2 \times 1} = 35$$

*Ques : Pascal triangle

[n]	
0	1
1	1 _ 1
2	1 _ 2 _ 1
3	1 _ 3 _ 3 _ 1
4	1 _ 4 _ 6 _ 4 _ 1
5	1 _ 5 _ 10 _ 10 _ 5 _ 1
6	1 _ 6 _ 15 _ 20 _ 15 _ 6 _ 1

$^2C_2 = \frac{2!}{2!(2-2)!}$

$= \frac{2!}{2!.0!} = 1$

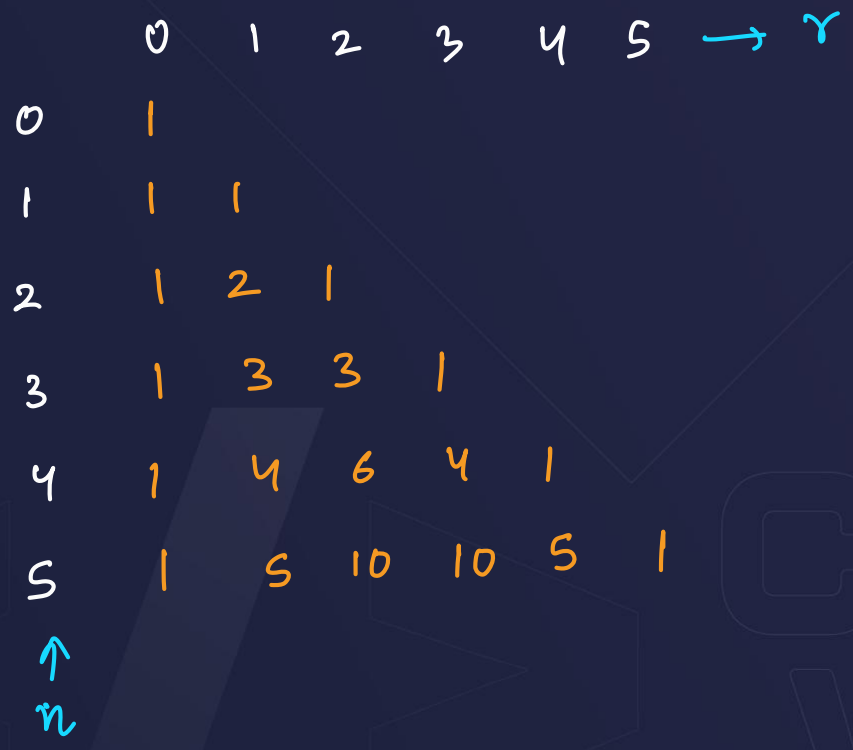
$0! = 1$

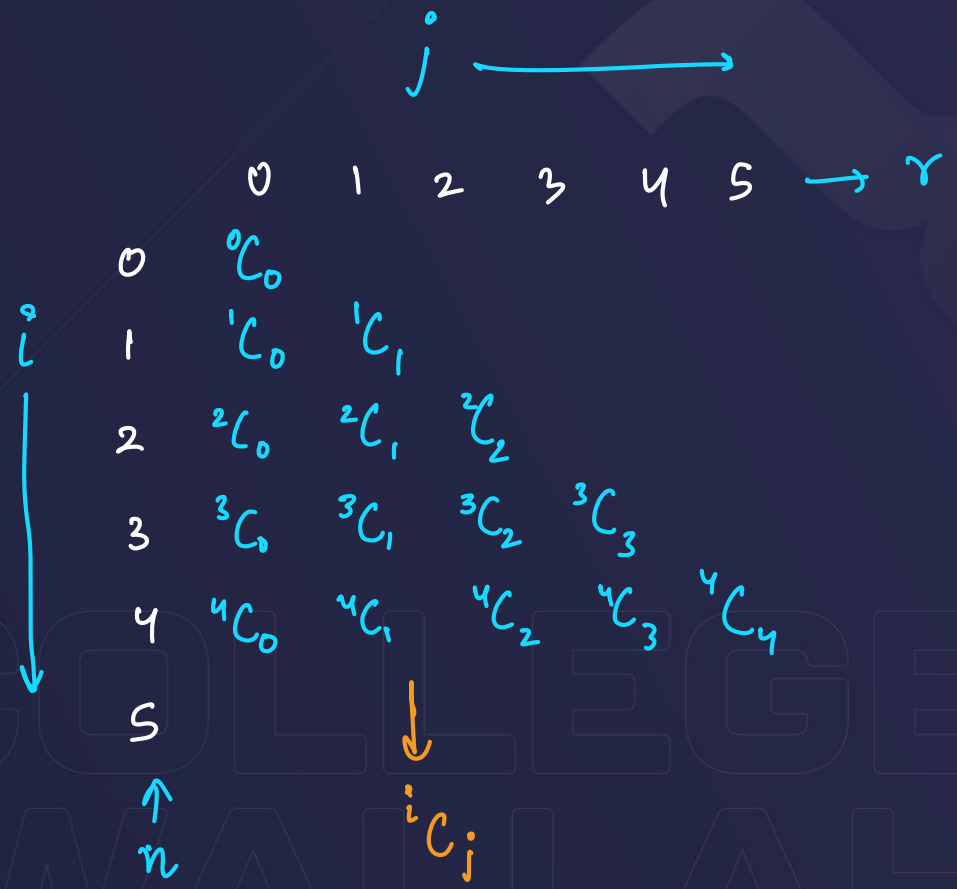
[5-17]-[5-18]

Homework : ^nP_r → function

$^nP_r = \frac{n!}{(n-r)!}$

Ques : Pascal triangle





Ques : Pascal triangle OPTIMISED

Ultimate Method - 'Maths'

$${}^nC_{r+1} = \frac{n!}{(r+1)! \times (n-r-1)!}$$

0 $\quad {}^0C_0$

1 $\quad {}^1C_0 \quad {}^1C_1$

2 $\quad {}^2C_0 \quad {}^2C_1 \quad {}^2C_2$

3 $\quad {}^3C_0 \quad {}^3C_1 \quad {}^3C_2 \quad {}^3C_3$

4 $\quad {}^4C_0 \quad {}^4C_1 \quad {}^4C_2 \quad {}^4C_3 \quad {}^4C_4$

$$\Rightarrow {}^nC_{r+1} = \frac{n!}{(r+1) \cdot r! \, \underbrace{(n-r)!}_{n-r}}$$

[5-19]

$$\Rightarrow {}^nC_{r+1} = \frac{n! \cdot (n-r)}{r! \cdot (n-r)! \cdot (r+1)}$$

$$(n-r)! = (n-r) \times (n-r-1)!$$

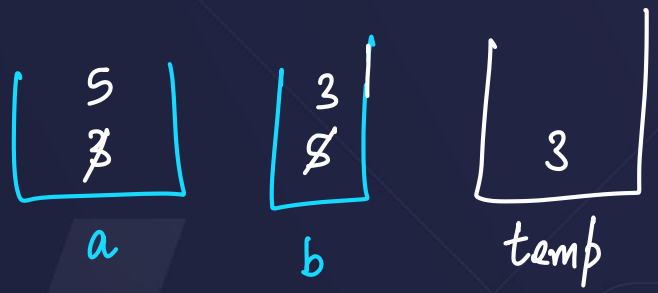
$$\Rightarrow \frac{(n-r)!}{n-r} = (n-r-1)!$$

$$\Rightarrow {}^nC_{r+1} = {}^nC_r \times \left(\frac{n-r}{r+1}\right)$$

$${}^iC_{j+1} = {}^iC_j \times \frac{i-j}{j+1}$$

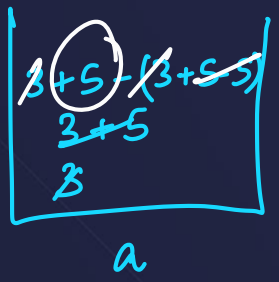
Ques : Swap 2 numbers

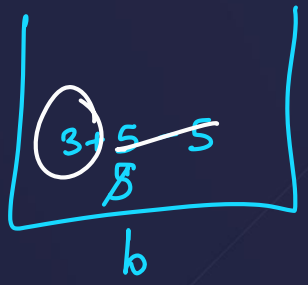
```
int a = 3;
int b = 5;
```




```
int temp;
temp = a;
a = b;
b = temp;
```

Ques : Swap 2 numbers without using a third variable



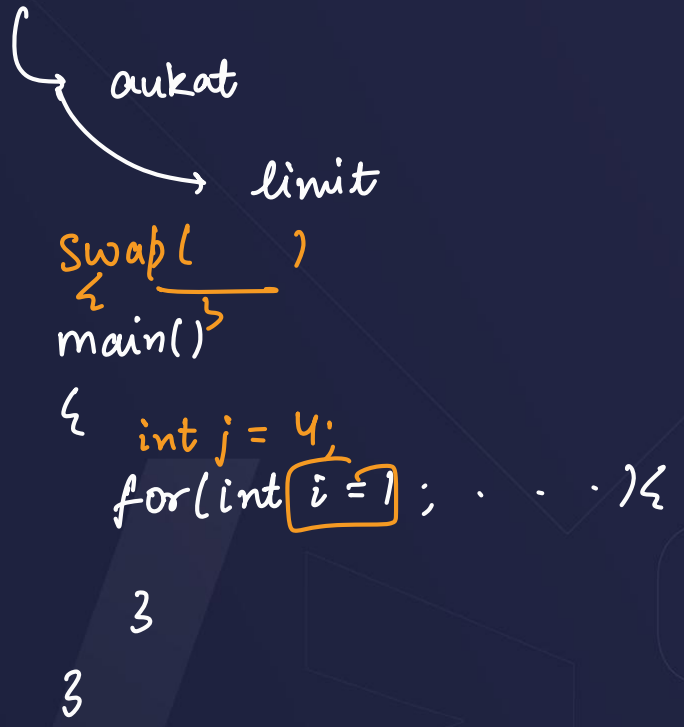



```
a = a + b;

b = a - b;

a = a - b;
```

Scope of variable



Formal parameters and Actual Parameters

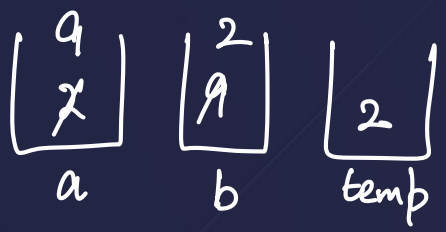
```
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
    return;
}
int main(){
    a = 2;
    b = 9;
    swap(a,b);
}
```

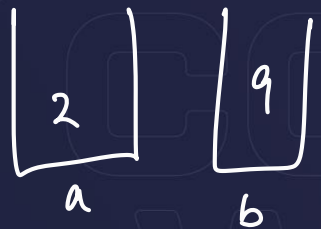
a & b are formal

2 & 9 are actual

Pass by value & Pass by reference

```
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
    return;
}
int main(){
    a = 2;
    b = 9;
    swap(a,b);
}
```



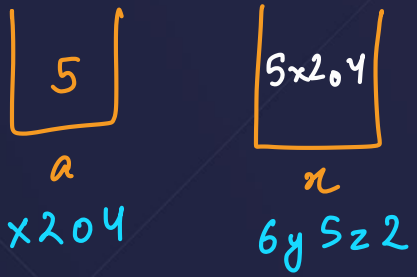


*Pointers

Notes

variable ka address store

```
int a = 5;
int* x = &a
*x = 7;
```



printf("%p", x); → 5x204

printf("%p", &x); → 6y5z2

printf("%d", *x); → 5

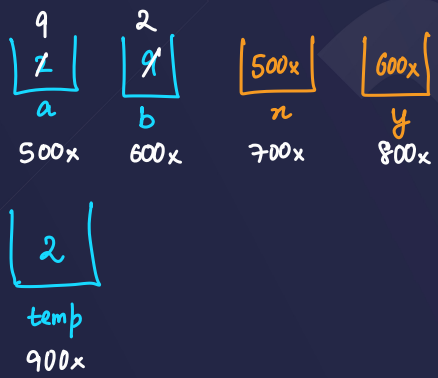
*Pointers

Swap 2 numbers 'pass by reference'

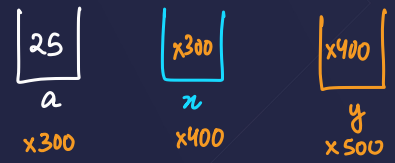
```
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
    return;
}
int main(){
    a = 2;
    b = 9;
    swap(&a, &b);
}
```

a & b are formal
↓
2 & 9 are actual

```c
#include<stdio.h>
void swap(int* x, int* y){
    int temp;
    temp = *x; // temp = 2
    *x = *y; // a = 9
    *y = temp; // *y = 2 -> b = 2;
    return;
}
int main(){
    int a = 2;
    int b = 9;
    swap(&a,&b);
    printf("The value of a is %d\n",a);
    printf("The value of b is %d",b);
    return 0;
}
```



```c
#include<stdio.h>
int main(){
    int a = 25;
    int* x = &a; //int*
    // VVIP ->   *x = 7;
    int** y = &x; // int
    printf("%d\n",a);
    printf("%d\n",*x);
    printf("%d\n",**y);
    return 0;
}
```



*x → 25 [%d]

*y → x300 [%p]

**y → 25 [%d]

1) Gate Wallah → C Programming

2) C++ → pointers

Homework

Ques : A positive integer is entered through the keyboard. Write a function to obtain the prime factors of this number.

(Hint : use seperate function to check if a number is prime or not)

Ques : Write a function to compute the greatest common divisor of two given numbers

a, b GCD → HCF

↓

highest common factor

24 → 1, 2, 3, 4, 6, 8, 12, 24

60 → 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60

```
for ( int i = 1 ; i < min(a,b) ; i++ ){
```

3

State TRUE or FALSE :

1) The variables commonly used in C functions are available to all the functions in a program. 'False'
2) To return the control back to the calling function we must use the keyword return. True
3) The same variable names can be used in different functions without any conflict. True

State TRUE or FALSE :

4) Every called function must contain a return statement. True
5) A function may contain more than one return statements. True → Only one should hit
6) Each return statement in a function may return a different value. True

State TRUE or FALSE :

7) A function can still be useful even if you don't pass any arguments to it and the function doesn't return any value back. True

8) Same names can be used for different functions without any conflict. False

State TRUE or FALSE :

9) A function may be called more than once from any other function True

10) It is necessary for a function to return some value.

No. False

HW : Print the factorials of first ‘n’ numbers

→ Using function

HW : Print first ‘n’ fibonacci numbers.

↳ using functions