Functions

Lecture- 9

Raghav Garg

$y = f(x) = x^2 + 3$

x: input

y: output

Let's create new bigger patterns

Take a, b, c as input and print the following pattern:
a = 3, b = 4, c = 5

Let’s change the code!

What if we we want to make a same pattern for numbers?

A lot of code change! Duh!!!

Can we do it in a better way?

The importance of functions

↓

in projects we always tend to use same/similar piece of code in the project multiple times, but not always we are looking for continuous repetition.

↓

loops

Syntax for Functions

Wrapping the logic under a name

```
function_name {
    // function body
}
```

```cpp
#include<iostream>
using namespace std;
void greeting(){
    cout<<"Good Morning"<<endl;
    cout<<"Have a nice day"<<endl;
}
int main(){
    greeting(); // function calling
    greeting();
    greeting();
}
```

Output

- Good Morning
- Have a nice day
- Good Morning
- Have a nice day
- Good Morning
- Have a nice day
-

Syntax for Functions

The need for arguments

```
function_name(int a, int b, int c) {
    // function body
}
```

Syntax for Functions

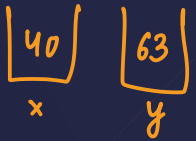
The need for a return type: Understanding with example

```
<void/int/float> function_name(int a, int b, int c) {
    // function body
}
```

Return Type :

```cpp
#include<iostream>
using namespace std;
int sum(int x, int y){
    return x+y;
}
int main(){
    cout<<sum(40,63);
}
```

103



Output

• 103

```cpp
#include<iostream>
using namespace std;
void starTriangle(int x){
    for(int i=1;i<=x;i++){
        for(int j=1;j<=i;j++){
            cout<<"*";
        }
        cout<<endl;
    }
}

int main(){
    starTriangle(3);
    starTriangle(4);
    starTriangle(5);
}
```

What is int main()?

main function

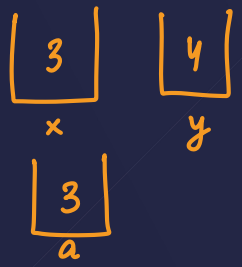
1) Sabse pehle yahi chalta hai

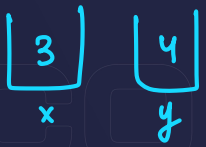
2) Ye ek hi baar chalta hai

Some inbuilt library functions

```
min(x, y)
max(x,y)
sqrt(x);
```

```
int mini(int x, int y){
    int a;
    if(x<y) a = x;
    else a = y;
    return a;
}
int main(){
    //cout<<sum(40,63);
    int x,y;
    cin>>x>>y;
    cout<<mini(3,4);
}
```





Output/Input

- 3 4
- 3

Ques : Combination and Permutation

Factorial → $n! = n \times n-1 \times n-2 \times \cdots 3 \times 2 \times 1$

$5! = 5 \times 4 \times 3 \times 2 \times 1$

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

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

```
int combination (int n, int r){
}
3
```

$n = 5$

$r = 2$

$n-r = 3$

$^nC_r = {}^5C_2 = \frac{5!}{2! \times 3!}$

Ques : Combination and Permutation

$nfact / rfact * nrfact = \frac{nfact}{rfact} * nrfact$

$nfact / (rfact * nrfact) = \frac{nfact}{rfact * nrfact}$

```cpp
int fact(int x){
    int f = 1;
    for(int i=2;i<=x;i++){
        f *= i;
    }
    return f;
}
int main(){
    int n;
    cout<<"enter n : ";
    cin>>n;
    int r;
    cout<<"enter r : ";
    cin>>r;
    int nfact = fact(n);
    int rfact = fact(r);
    int nrfact = fact(n-r);
    int ncr = nfact/(rfact*nrfact);
    cout<<ncr;
}
```

Output

Enter n : 5

Enter r : 2

Ques : Pascal triangle

$0! = 1$

${}^{n}C_{0} = 1$

${}^{n}C_{1} = n$

	0	1	2	3	4	5
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

$i C_j$

$i C_{j+1} = i C_j \times \left(\frac{i-j}{j+1}\right)$

	0	1	2	3	4	5 → j
0	${}^{0}C_{0}$					
1	${}^{1}C_{0}$	${}^{1}C_{1}$				
2	${}^{2}C_{0}$	${}^{2}C_{1}$	${}^{2}C_{2}$			
3	${}^{3}C_{0}$	${}^{3}C_{1}$	${}^{3}C_{2}$	${}^{3}C_{3}$		
4	${}^{4}C_{0}$	${}^{4}C_{1}$	${}^{4}C_{2}$	${}^{4}C_{3}$	${}^{4}C_{4}$	
5	${}^{5}C_{0}$	${}^{5}C_{1}$	${}^{5}C_{2}$	${}^{5}C_{3}$	${}^{5}C_{4}$	${}^{5}C_{5}$

i

Ques : Pascal triangle Optimised

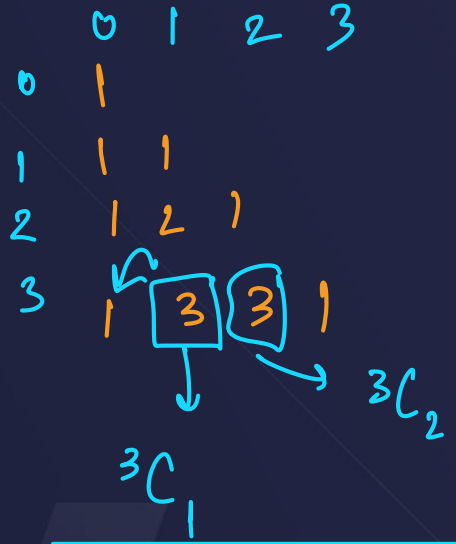
$$^{n}C_{r+1} = \frac{n!}{(r+1)! \times [n-(r+1)]!} = \frac{n!}{(r+1)\cdot r! \cdot (n-r-1)!}$$

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

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

$$\boxed{^{n}C_{r+1} = {}^{n}C_{r} \cdot \left(\frac{n-r}{r+1}\right)}$$

Ques : Pascal triangle Optimised



$$next = curr \times \left(\frac{i-j}{j+1}\right)$$

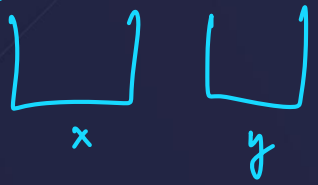
Ques : Pascal triangle Optimised

$$^{n}C_{r+1} = {}^{n}C_{r} \cdot \left(\frac{n-r}{r+1}\right)$$

Are arguments passed actually the same?

Printing out the actual address of variables in the functions...

```
int fun(int x, int y){
    ____________
}
int main(){
    int x;
    fun(x, y);
}
```



Are arguments passed actually the same?

Printing out the actual address of variables in the functions...

C Drive → Movies → Action → Spider Man

Formal parameters and Actual Parameters

```cpp
#include<iostream>
using namespace std;
void fun(int x, int y){
    cout<<"address of fun x "<<&x<<endl;
    cout<<"address of fun y "<<&y<<endl;
}
int main(){
    int x = 3;
    int y = 7;
    cout<<"address of main x "<<&x<<endl;
    cout<<"address of main y "<<&y<<endl;
    fun(x,y); // fun(3,7);
}
```

formal parameters

actual parameters

Scope of variable → Limits

```cpp
#include<iostream>
using namespace std;
void fun(int x, int y){
    cout<<"address of fun x "<<&x<<endl;
    cout<<"address of fun y "<<&y<<endl;
}
int main(){
    int x = 3;
    int y = 7;
    cout<<"address of main x "<<&x<<endl;
    cout<<"address of main y "<<&y<<endl;
    fun(x,y);
}
```

int a = 4;

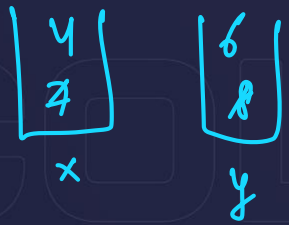
Default values of Arguments

```
void fun(int x = 7, int y = 8){
    cout<<x<<" "<<y;
}
int main(){
    fun();
}
```

7 is now default value of x

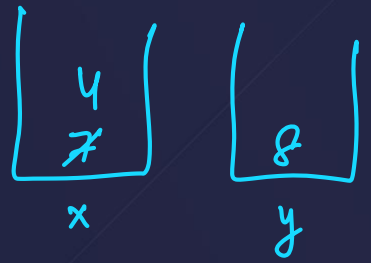
8 is default value of y

```
void fun(int x = 7, int y = 8){
    cout<<x<<" "<<y;
}
int main(){
    fun(4,6);
}
```

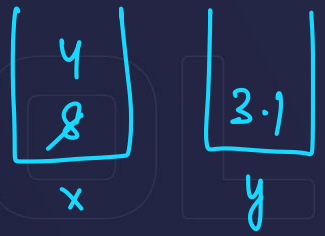


Default values of Arguments

```
void fun(int x = 7, int y = 8){
    cout<<x<<" "<<y;
}
int main(){
    fun(4);
}
```



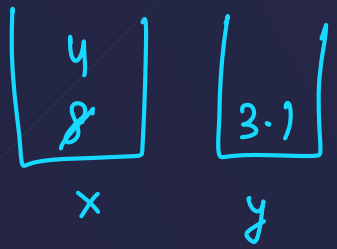

```
void fun(int x = 8, float y = 3.1){
    cout<<x<<" "<<y;
}
int main(){
    fun(4);
}
```



Default values of Arguments

```
void fun(int x = 8, float y = 3.1){
    cout<<x<<" "<<y;
}
int main(){
    fun(4.7);
}
```

float



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

x	y
1	1
24	60

1, 2, 3, 4, 6, 12

HCF (x, y) ≤ min(x, y)

↓

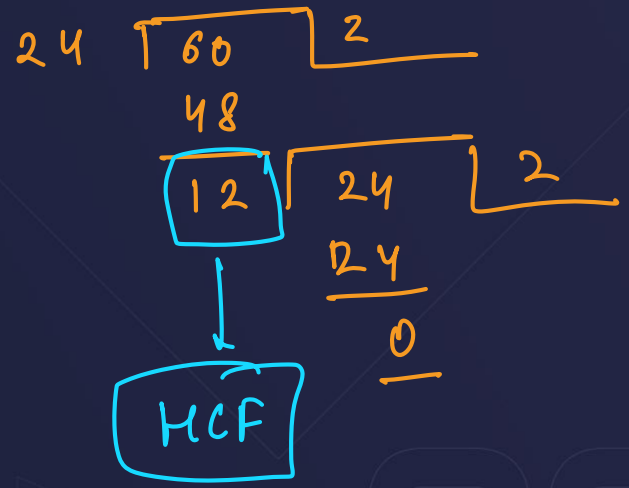
highest, common factor

↓

a number

divisible by both

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



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, but not for void necessarily

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. False

5) A function may contain more than one return statements. True , Only one can hit.

6) Each return statement in a function may return a different value. True

Bonus Ques : Print the factorials of first n numbers

$1! = 1$

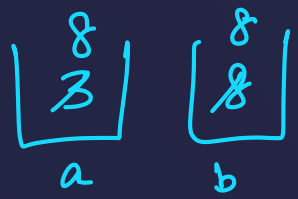
$2! = 2 \times 1! = 2$

$3! = 3 \times 2! = 6$

$4! = 4 \times 3! = 24$

Ques : Swap 2 numbers

```
int a,b;
cin>>a>>b;
a = b;
b = a;
cout<<a<<" "<<b;
```

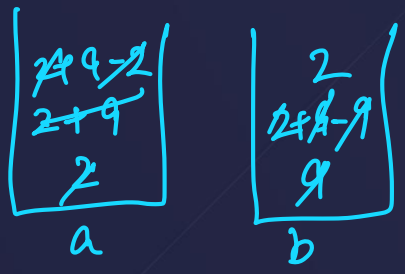


Output

- 3 8
- 8 8

Ques : Swap 2 numbers

```
int a,b;
cin>>a>>b;
a = a + b;
b = a - b;
a = a - b;
cout<<a<<" "<<b;
```



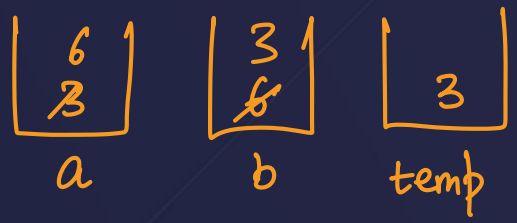
Output

2 9

Why does this not work?

Functions

```
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
    return;
}
int main(){
    int a,b;
    cin>>a>>b;
    swap(a,b);
    cout<<a<<" "<<b;
}
```



Output

- 3 6
- 3 6

Is there a way to solve this?

What if we are able to store or pass the actual address inside functions?

Next Lecture

Understanding the memory aspects of programming

Working with memory addresses using Pointers!