C++ Functions

⇒ Functions

- Functions are also called Procedures or Subroutines or methods and they are often defined to perform a specific task.
- Can be called anytime without writing whole code again & again.

⇒ Functions Parameters

- Functions receives info that is passed to them as parameter.

→ Formal parameter

- Variables declared in function is called Formal parameters.
- Eg :

```
int sum (int a, int b) {
```

a, b → Formal parameters

```
}
```

→ Actual parameters / Arguments

- Variables that are passed to functions are called actual parameters.

- Eg:

```
int sum(int a, int b);
int main(){
  int num1 = 5;
  int num2 = 6;
  int sum(num1, num2);
}
```

num1, num2 → Actual parameters

⇒ Function Call

→ Call by value

- Eg: Swap function

```
void swap(int a, int b){
  temp = a;
  a = b;
  b = temp; }
```

Doesn't affect Actual values

call = swap(x, y);

→ Call by pointer

- Address of Actual parameters is sent to formal parameters

- Eg: Swap function

```
Void swap(int *a, int *b){
    temp = *a;
   *a = *b;
   *b = temp; }
```

call = swap(&x, &y);

→ call by Reference

- Reference of Actual parameters is sent to formal parameters.

- Eg: Swap fn.

```
void swap(int &a, int &b){
    int temp = a;
     a = b;
     b = temp; }
```

Call = swap(x,y);

→ Default Arguments

- Those values which are used by function if we don't pass it as parameter.

- Eg:

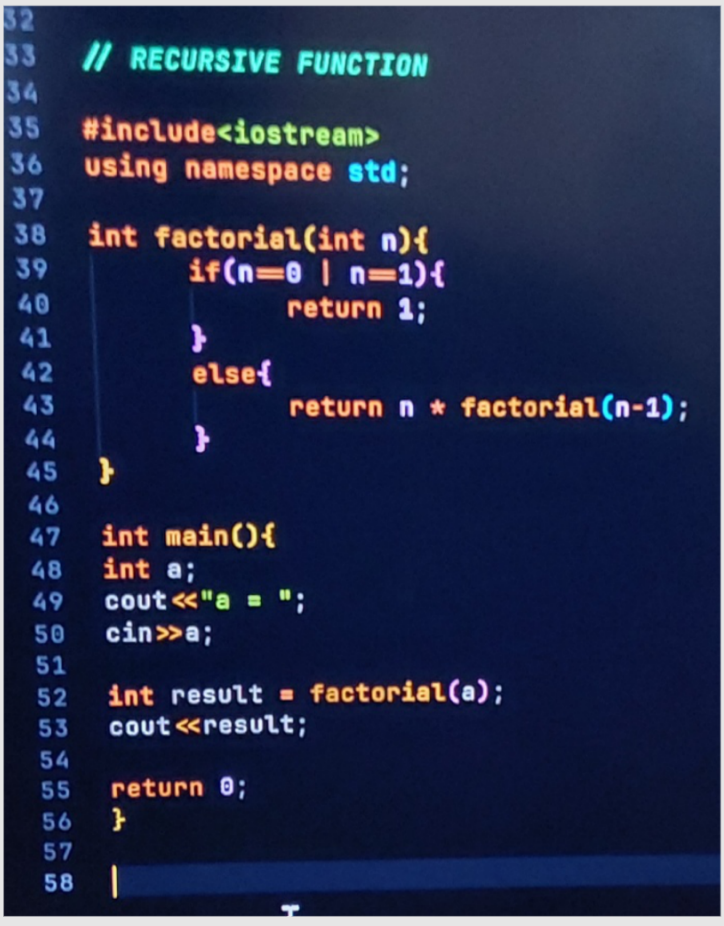
```
4
5   #include<iostream>
6   using namespace std;
7
8   int sum(int a, int b = 5){
9         return a+b;
10  }
11
12  int main(){
13
14      int result = sum(10);
15      cout<<result;
16
17  return 0;
18  }
19  // OUTPUT : 15
```

⇒ Recursion

- When a function calls itself, it is called recursion

- Recursive functions must be designed with a base case to make sure recursion stops.

Eg: Factorial of number



```
32
33  // RECURSIVE FUNCTION
34
35  #include<iostream>
36  using namespace std;
37
38  int factorial(int n){
39        if(n==0 | n==1){
40              return 1;
41        }
42        else{
43              return n * factorial(n-1);
44        }
45  }
46
47  int main(){
48  int a;
49  cout<<"a = ";
50  cin>>a;
51
52  int result = factorial(a);
53  cout<<result;
54
55  return 0;
56  }
57
58
```