



# RECURSION IN C

BY: DR. SHEFALI ARORA CHOUHAN

# RECURSIVE FUNCTION

- ▶ A recursive function is a function that calls itself again
- ▶ Activation record is maintained on a stack
- ▶ Return address of call & local variables are maintained

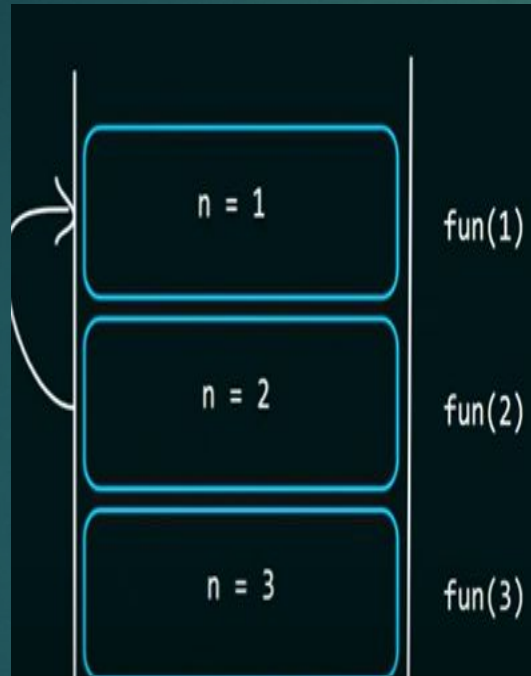
```
int fun()
{
    ...
    fun();
}
```

The diagram illustrates recursive calls between two functions. On the left, a block of code for `fun()` contains `//some code`, a call to `fun2();`, and `//some code`. On the right, a block of code for `fun2()` contains `//some code`, a call to `fun();`, and `//some code`. A white arrow points from the `fun2();` line in the `fun()` block to the `fun2()` block. Another white arrow points from the `fun();` line in the `fun2()` block back to the `fun()` block, showing the return path.

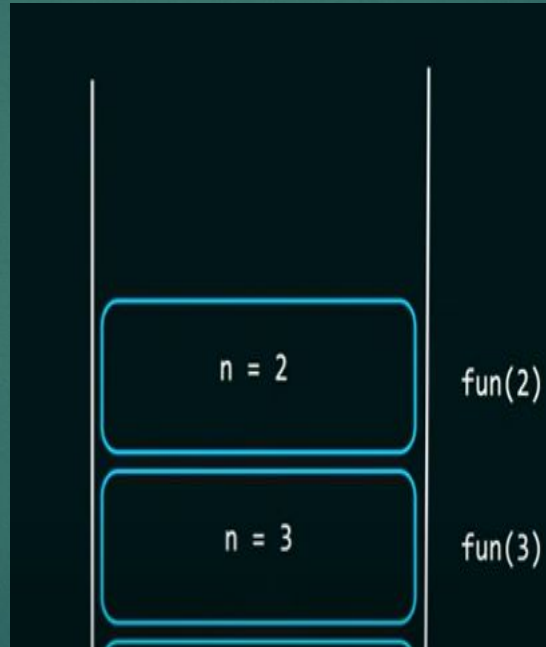
```
int fun(int n)
{
    if( n==1 )
        return 1;
    else
        return 1 + fun( n-1 );
}

int main() {
    int n = 3;
    printf("%d", fun(n));
    return 0;
}
```

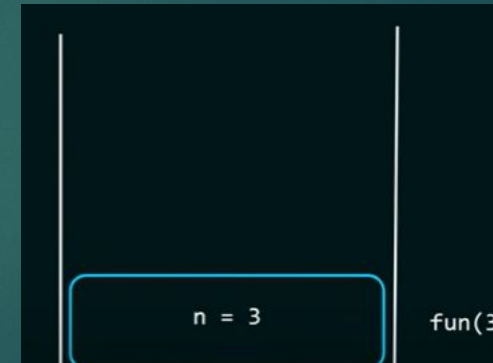
# USING STACKS: THE PUSH & POP CONCEPT



1



1+1



1+1+1

# CALCULATE FACTORIAL USING RECURSION

```
Fact( )  
{  
    if( )  
    {  
        ...  
    }  
    else  
    {  
        ...  
    }  
}
```

Base Case

Recursive  
Procedure

```
Fact(int n)  
{  
    if( n == 1 )  
    {  
        return 1;  
    }  
    else  
    {  
        return n * Fact(n-1);  
    }  
}
```

# HEAD VS TAIL RECURSION

- ▶ A function is tail recursive if recursion is the last thing done by the function in a program

```
void fun(int n) {  
    if(n == 0)  
        return;  
    else  
        printf("%d ", n);  
    return fun(n-1);  
}  
int main() {  
    fun(3);  
    return 0;  
}
```

fun(0)	return;
fun(1)	Act f1
fun(2)	Act f2
fun(3)	Act f3
main()	Act m

# HEAD VS TAIL RECURSION

- ▶ A function is non-tail recursive if there is something left to evaluate

```
void fun(int n) {  
    if(n == 0)  
        return;  
    fun(n-1);  
    printf("%d ", n);  
}  
int main() {  
    fun(3);  
    return 0;  
}
```

fun(0)	return;
fun(1)	Act f1
fun(2)	Act f2
fun(3)	Act f3
main()	Act m