

# Recursion

## CHAPTER 28



**SURESH TECHS**

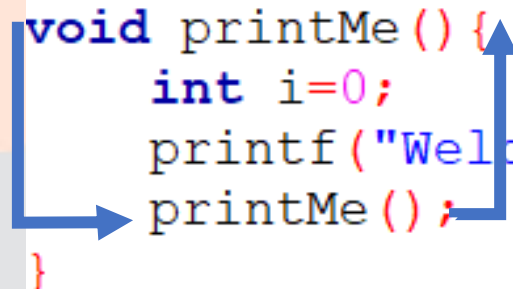
**C PROGRAMMING COURSE**

# Recursion

- When a function calls **itself** then this technique is known as **Recursion** and the function is known as **recursive function**
- While using recursion, you will have to define an **exit condition** on that function, if not then it will go into an **infinite loop**.

```
#include<stdio.h>
void printMe() {
    int i=0;
    printf("Welcome %d", i);
    printMe();
}

int main() {
    printMe();
    return 0;
}
```

A blue arrow originates from the 'printMe()' call within the 'printMe()' function body and points back to the start of the 'printMe()' function definition, illustrating a recursive call. Another blue arrow points from the 'main()' function to the 'printMe()' call, showing the initial invocation.

# Types of recursion

- Direction Recursion

```
#include<stdio.h>
void printMe() {
    int i=0;
    printf("Welcome %d",i);
    printMe();
}

int main() {
    printMe();
    return 0;
}
```

A diagram illustrating direct recursion. A blue arrow starts from the recursive call `printMe();` inside the `printMe()` function and points back to the function's definition, indicating that the function calls itself directly.

- Indirect Recursion

```
#include<stdio.h>
void sum() {
    printf("Sum Me\n");
    printMe();
}
void printMe() {
    int i=0;
    printf("Welcome %d",i);
    sum();
}

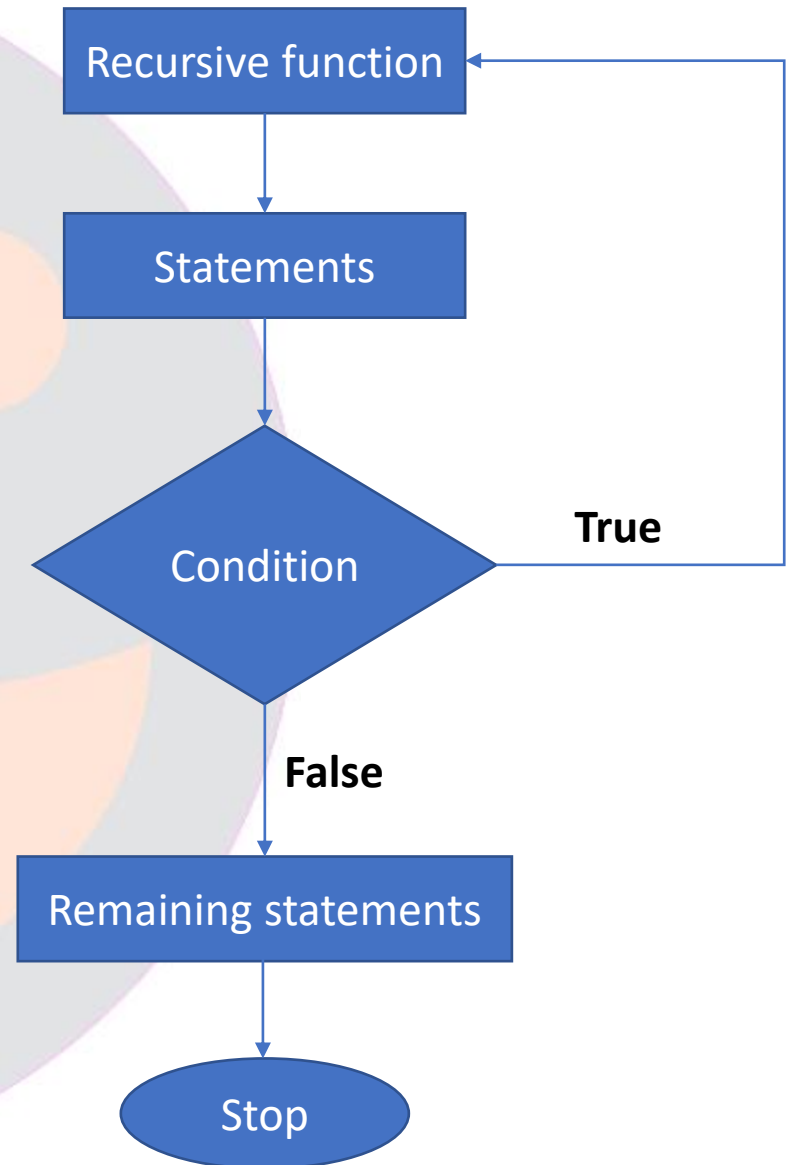
int main() {
    printMe();
    return 0;
}
```

A diagram illustrating indirect recursion. It shows two functions, `sum()` and `printMe()`. A blue arrow points from the `printMe();` call inside `sum()` to the `printMe()` function definition. Another blue arrow points from the `sum();` call inside `printMe()` back to the `sum()` function definition, showing a cycle of indirect calls.

# Exit condition

```
#include<stdio.h>
void printMe() {
    static int i=0;
    printf("Welcome %d\n",i);
    if(i<10){
        i++;
        printMe();
    }
}

int main() {
    printMe();
    return 0;
}
```



# Note

- You have to be **very careful** when you are using recursion in your program.
- You just **cannot use recursion in all your problems** because it will make your program more complex and difficult.

# What next?

- Decision making and looping

