SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23ITT101- PROBLEM SOLVING & C PROGRAMMING

I YEAR /I SEMESTER

Unit IV – FUNCTIONS AND POINTERS

Topic : Recursion

Topics Covered

- Recursion
- Pass arrays to a function in C
- Pass strings to a function in C

Recursion

- Recursion is the process which comes into existence when a function calls a copy of itself to work on a smaller problem.
- Any function which calls itself is called recursive function, and such function calls are called recursive calls.

Recursion

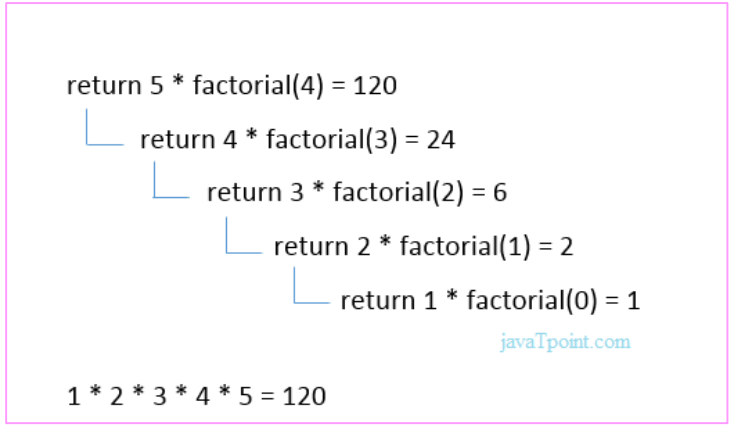
- Recursion involves several numbers of recursive calls.
- However, it is important to impose a termination condition of recursion.
- Recursion code is shorter than iterative code however it is difficult to understand.

Recursion

```
#include <stdio.h>

// Function prototype
int factorial(int n);
int main() {
    int num;
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Factorial of a negative number is undefined.\n");
    } else {
        printf("The factorial of %d is %d\n", num, factorial(num));
    }
    return 0;
}
// Recursive function to calculate factorial
int factorial(int n) {
    if (n == 0 || n == 1) { // Base case: factorial(0) = 1 and factorial(1) = 1
        return 1;
    }
    return n * factorial(n - 1); // Recursive step
}
```



Recursive Function

- Explanation:

Base Case: If n is 0 or 1, the function directly returns 1. This stops further recursive calls.

Recursive Step: For n > 1, the function calls itself with n-1 and multiplies the result by n.

Recursive Function

- A recursive function performs the tasks by dividing it into the subtasks.
- There is a termination condition defined in the function which is satisfied by some specific subtask.
- After this, the recursion stops and the final result is returned from the function.

Recursive Function

- The case at which the function doesn't recur is called the base case whereas the instances where the function keeps calling itself to perform a subtask, is called the recursive case.
- All the recursive functions can be written using this format.

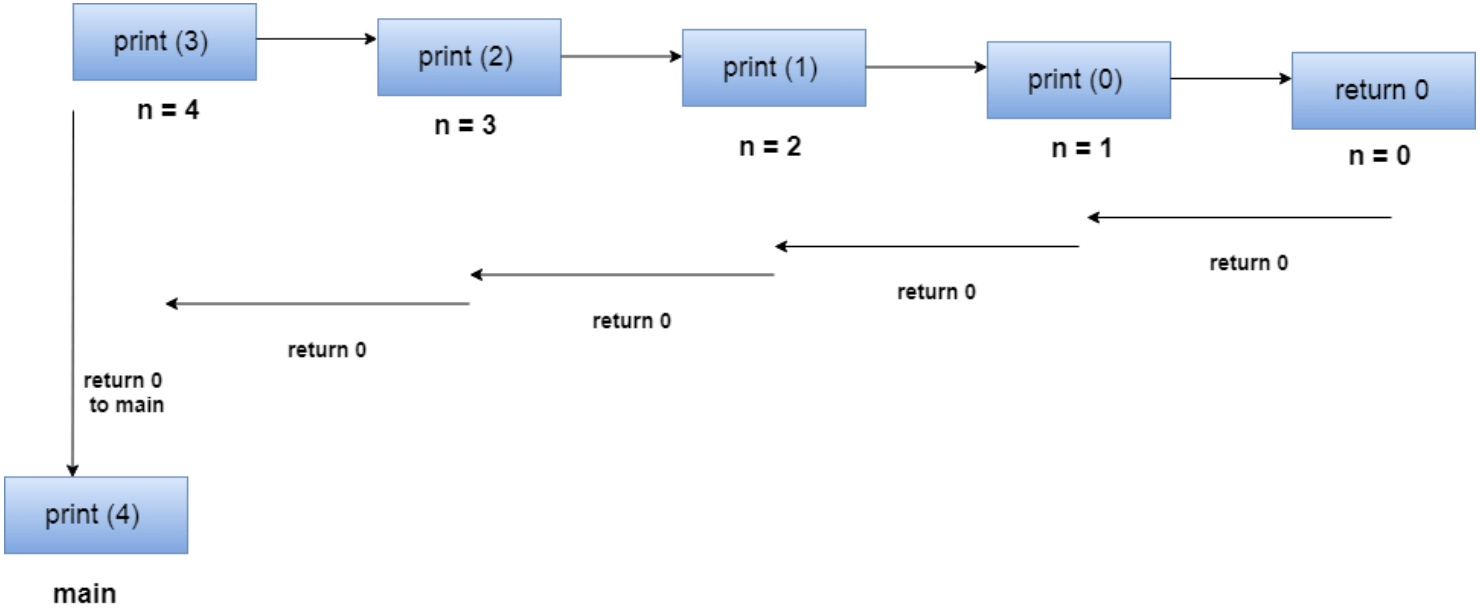
Memory allocation of Recursive method

- Each recursive call creates a new copy of that method in the memory.
- Once some data is returned by the method, the copy is removed from the memory.
- Since all the variables and other stuff declared inside function get stored in the stack, therefore a separate stack is maintained at each recursive call.
- Once the value is returned from the corresponding function, the stack gets destroyed.
- Recursion involves so much complexity in resolving and tracking the values at each recursive call.
- Therefore we need to maintain the stack and track the values of the variables defined in the stack.

Memory allocation of Recursive method

- Let us consider the following example to understand the memory allocation of the recursive functions.

```
int display (int n)
{
   if(n == 0)
      return 0; // terminating condition
   else
   {
      printf("%d",n);
      return display(n-1); // recursive call
   }
}
```



Stack tracing for recursive function call

Pass arrays to a function in C

- In C programming, you can pass an entire array to functions.
- To pass an entire array to a function, only the name of the array is passed as an argument.

```
// Program to calculate the sum of array elements by passing to a function

#include <stdio.h>
float calculateSum(float num[]);

int main() {
  float result, num[] = {23.4, 55, 22.6, 3, 40.5, 18};

  // num array is passed to calculateSum()
  result = calculateSum(num);
  printf("Result = %.2f", result);
  return 0;
}
```

```
float calculateSum(float num[]) {
  float sum = 0.0;

  for (int i = 0; i < 6; ++i) {
    sum += num[i];
  }

  return sum;
}
```

Output:
Result = 162.50

Pass arrays to a function in C

- **Pass two-dimensional arrays**

```
#include <stdio.h>
void displayNumbers(int num[2][2]);

int main() {
  int num[2][2];
  printf("Enter 4 numbers:\n");
  for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 2; ++j) {
      scanf("%d", &num[i][j]);
    }
  }

  // pass multi-dimensional array to a function
  displayNumbers(num);

  return 0;
}
```

```
void displayNumbers(int num[2][2]) {
  printf("Displaying:\n");
  for (int i = 0; i < 2; ++i) {
    for (int j = 0; j < 2; ++j) {
      printf("%d\n", num[i][j]);
    }
  }
}
```

Output:

Enter 4 numbers
2
3
4
5

Displaying:

2
3
4
5

Pass strings to a function in C

```
#include <stdio.h>
void displayString(char str[]);

int main()
{
    char str[50];
    printf("Enter string: ");
    fgets(str, sizeof(str), stdin);
    displayString(str);     // Passing string to a
function.
    return 0;
}
```

```
void displayString(char str[])
{
    printf("String Output: ");
    puts(str);
}
```

Summary

- The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples of such problems are Towers of Hanoi (TOH), Inorder/Preorder/Postorder Tree Traversals, DFS of Graph


Thank
you