Object Oriented and Programming Lab

Lab#14

Table of Contents

To Watch:

1. https://www.youtube.com/watch?v=c7VjS7ZZVWM
2. https://www.youtube.com/watch?v=4agL-MQq05E (FACTORIAL Example)
3. https://www.youtube.com/watch?v=NvVYd08NUXI

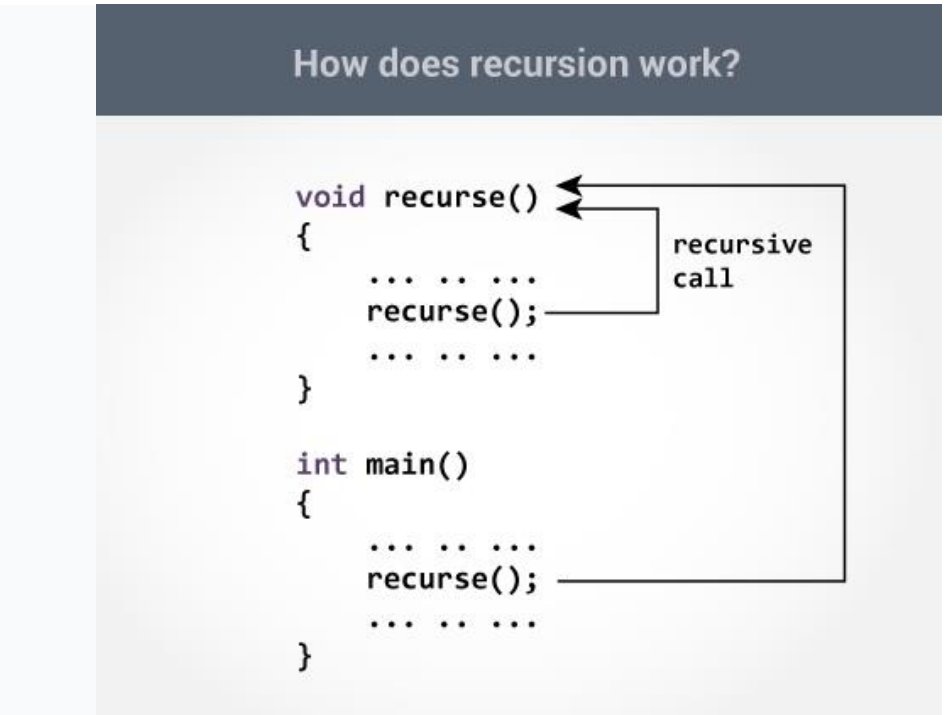
Recursion:

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, etc.

How recursion works in C++?

```
void recurse()

{

    ... .. ...

    recurse();

    ... .. ...

}


int main()

{

    ... .. ...

    recurse();

    ... .. ...

}
```

The figure below shows how recursion works by calling itself over and over again.



The recursion continues until some condition is met.

To prevent infinite recursion, if...else statement (or similar approach) can be used where one branch makes the recursive call and other doesn't.

What is base condition in recursion?

In the recursive program, the solution to the base case is provided and the solution of the bigger problem is expressed in terms of smaller problems.

```
int fact(int n)
{
    if (n < = 1) // base case
        return 1;
    else
        return n*fact(n-1);
```

```
}
```

In the above example, base case for n < = 1 is defined and larger value of number can be solved by converting to smaller one till base case is reached.

How a particular problem is solved using recursion?

The idea is to represent a problem in terms of one or more smaller problems, and add one or more base conditions that stop the recursion. For example, we compute factorial n if we know factorial of (n-1). The base case for factorial would be n = 0. We return 1 when n = 0.

Why Stack Overflow error occurs in recursion?

If the base case is not reached or not defined, then the stack overflow problem may arise. Let us take an example to understand this.

```
int fact(int n)
{
    // wrong base case (it may cause
    // stack overflow).
    if (n == 100)
        return 1;

    else
        return n*fact(n-1);
}
```

If fact(10) is called, it will call fact(9), fact(8), fact(7) and so on but the number will never reach 100. So, the base case is not reached. If the memory is exhausted by these functions on the stack, it will cause a stack overflow error.

Factorial of a Number Using Recursion

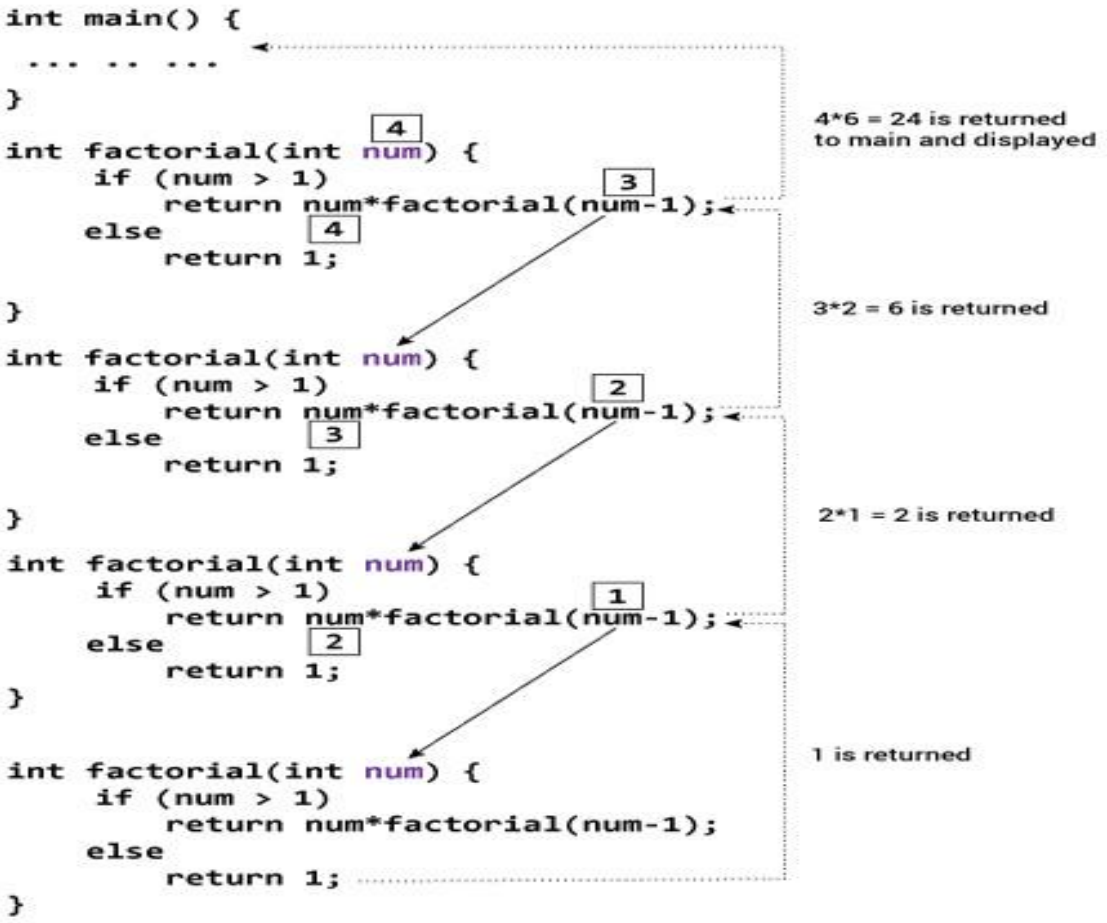
```
/ Factorial of n = 1*2*3*...*n
#include <iostream>
using namespace std;
int factorial(int);

int main()
```

```
{
    int n;
    cout<<"Enter a number to find factorial: ";
    cin >> n;
    cout << "Factorial of " << n <<" = " << factorial(n);
    return 0;
}
int factorial(int n)
{
    if (n > 1)
    {        return n*factorial(n-1); }
    else
    {        return 1; }
}
```

Output

```
Enter a number to find factorial: 4
Factorial of 4 = 24
```

Explanation: How this example works?



Suppose the user entered 4, which is passed to the `factorial()` function.

1. In the first `factorial()` function, test expression inside if statement is true. The `return num*factorial(num-1);` statement is executed, which calls the second `factorial()` function and argument passed is `num-1`which is 3.

2. In the second `factorial()` function, test expression inside if statement is true. The `return num*factorial(num-1);` statement is executed, which calls the third `factorial()` function and argument passed is `num-1` which is 2.

3. In the third `factorial()` function, test expression inside if statement is true. The `return num*factorial(num-1);` statement is executed, which calls the fourth `factorial()` function and argument passed is `num-1` which is 1.

4. In the fourth `factorial()` function, test expression inside if statement is false. The `return 1;` statement is executed, which returns 1 to third `factorial()` function.

5. The third `factorial()` function returns 2 to the second `factorial()` function.

6. The second `factorial()` function returns 6 to the first `factorial()` function.

7. Finally, the first `factorial()` function returns 24 to the `main()` function, which is displayed on the screen.

Another Example

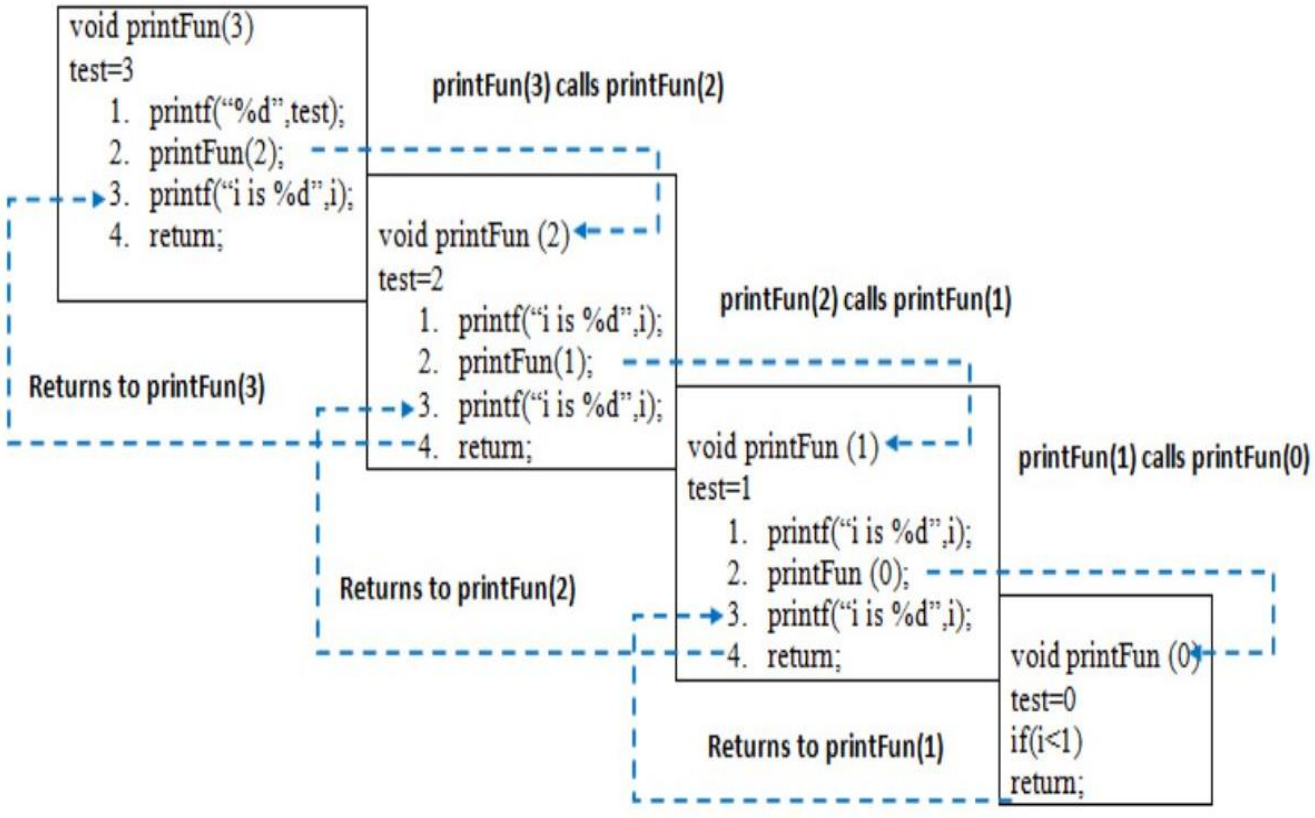
```
#include<>
using namespace std;

void printFun(int test)
{
   if (test < 1)
      return;
   else
   {
      cout << test << " ";
      printFun(test-1);   // statement 2
      cout << test << " ";
      return;
   }
}

int main()
{
   int test = 3;
```

```
    printFun(test);
}
```

Output :

```
3 2 1 1 2 3
```

When **printFun(3)** is called from main(), memory is allocated to **printFun(3)** and a local variable test is initialized to 3 and statement 1 to 4 are pushed on the stack as shown in below diagram. It first prints '3'. In statement 2, **printFun(2)** is called and memory is allocated to **printFun(2)** and a local variable test is initialized to 2 and statement 1 to 4 are pushed in the stack.
Similarly, **printFun(2)** calls **printFun(1)** and **printFun(1)** calls **printFun(0)**. **printFun(0)** goes to if statement and it return to **printFun(1)**. Remaining statements of **printFun(1)** are executed and it returns to **printFun(2)** and so on. In the output, value from 3 to 1 are printed and then 1 to 3 are printed. The memory stack has been shown in below diagram.



Reference

https://www.geeksforgeeks.org/recursion/

https://beginnersbook.com/2017/08/cpp-recursion/

https://www.programiz.com/cpp-programming/recursion