GeeksforGeeks

Related Articles

Tail Recursion

Difficulty Level : Easy • Last Updated : 13 Apr, 2021

What is tail recursion?

A recursive function is tail recursive when recursive call is the last thing executed by the function. For example the following C++ function print() is tail recursive.

C

```
// An example of tail recursive function
void print(int n)
{
    if (n < 0)  return;
    cout << " " << n;

    // The last executed statement is recursive call
    print(n-1);
}
```

Java

```
// An example of tail recursive function
static void print(int n)
{
    if (n < 0)
      return;

    System.out.print(" " + n);

    // The last executed statement
      // is recursive call
```

Python3

```
# An example of tail recursive function
def prints(n):

    if (n < 0):
        return
    print(" " + str(n),end='')

    # The last executed statement is recursive call
    prints(n-1)

    # This code is contributed by Pratham76
```

C#

```
// An example of tail recursive function
static void print(int n)
{
    if (n < 0)
      return;

    Console.Write(" " + n);

    // The last executed statement
      // is recursive call
    print(n - 1);
}

// This code is contributed by divyeshrabadiya07
```

Why do we care?

The tail recursive functions considered better than non tail recursive functions as tail-recursion can be optimized by compiler. The idea used by compilers to optimize tail-recursive functions is simple, since the recursive call is the last statement, there is nothing left to do in the current function, so saving the current function's stack frame is of no use (See this for more details).

function. Although it looks like a tail recursive at first look. If we take a closer look, we can see that the value returned by fact(n-1) is used in fact(n), so the call to fact(n-1) is not the last thing done by fact(n)

C++

```
#include<iostream>
using namespace std;

// A NON-tail-recursive function.  The function is not tail
// recursive because the value returned by fact(n-1) is used in
// fact(n) and call to fact(n-1) is not the last thing done by fact(n)
unsigned int fact(unsigned int n)
{
    if (n == 0) return 1;

    return n*fact(n-1);
}

// Driver program to test above function
int main()
{
    cout << fact(5);
    return 0;
}
```

Java

```
class GFG {

    // A NON-tail-recursive function.
    // The function is not tail
    // recursive because the value
    // returned by fact(n-1) is used
    // in fact(n) and call to fact(n-1)
    // is not the last thing done by
    // fact(n)
    static int fact(int n)
    {
        if (n == 0) return 1;

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

```
    {
        System.out.println(fact(5));
    }
}

// This code is contributed by Smitha.
```

Python3

```
# A NON-tail-recursive function.
# The function is not tail
# recursive because the value
# returned by fact(n-1) is used
# in fact(n) and call to fact(n-1)
# is not the last thing done by
# fact(n)
def fact(n):

    if (n == 0):
        return 1

    return n * fact(n-1)

# Driver program to test
# above function
print(fact(5))
# This code is contributed by Smitha.
```

C#

```
using System;

class GFG {

    // A NON-tail-recursive function.
    // The function is not tail
    // recursive because the value
    // returned by fact(n-1) is used
    // in fact(n) and call to fact(n-1)
    // is not the last thing done by
    // fact(n)
    static int fact(int n)
    {
```

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

    // Driver program to test
    // above function
    public static void Main()
    {
        Console.Write(fact(5));
    }
}

// This code is contributed by Smitha
```

PHP

```
<?php
// A NON-tail-recursive function.
// The function is not tail
// recursive because the value
// returned by fact(n-1) is used in
// fact(n) and call to fact(n-1) is
// not the last thing done by fact(n)

function fact( $n)
{
    if ($n == 0) return 1;

    return $n * fact($n - 1);
}

    // Driver Code
    echo fact(5);

// This code is contributed by Ajit
?>
```

Javascript

```
<script>

// A NON-tail-recursive function.
// The function is not tail
// recursive because the value
```

```
// fact(n)
function fact(n)
{
    if (n == 0)
        return 1;

    return n * fact(n - 1);
}

// Driver code
document.write(fact(5));

// This code is contributed by divyeshrabadiya07

</script>
```

Output :

```
 120
```

The above function can be written as a tail recursive function. The idea is to use one more argument and accumulate the factorial value in second argument. When n reaches 0, return the accumulated value.

C++

```
#include<iostream>
using namespace std;

// A tail recursive function to calculate factorial
unsigned factTR(unsigned int n, unsigned int a)
{
    if (n == 0)  return a;

    return factTR(n-1, n*a);
}

// A wrapper over factTR
unsigned int fact(unsigned int n)
{
   return factTR(n, 1);
}
```

```
    cout << fact(5);
    return 0;
}
```

Java

```
// Java Code for Tail Recursion

class GFG {

    // A tail recursive function
    // to calculate factorial
    static int factTR(int n, int a)
    {
        if (n == 0)
            return a;

        return factTR(n - 1, n * a);
    }

    // A wrapper over factTR
    static int fact(int n)
    {
        return factTR(n, 1);
    }

    // Driver code
    static public void main (String[] args)
    {
        System.out.println(fact(5));
    }
}

// This code is contributed by Smitha.
```

Python3

```
# A tail recursive function
# to calculate factorial
def fact(n, a = 1):

    if (n == 0):
        return a
```

```
# Driver program to test
#  above function
print(fact(5))

# This code is contributed
# by Smitha
# "improved by Ujwal"
```

C#

```
// C# Code for Tail Recursion
using System;

class GFG {

    // A tail recursive function
    // to calculate factorial
    static int factTR(int n, int a)
    {
        if (n == 0)
            return a;

        return factTR(n - 1, n * a);
    }

    // A wrapper over factTR
    static int fact(int n)
    {
        return factTR(n, 1);
    }

    // Driver code
    static public void Main ()
    {
        Console.WriteLine(fact(5));
    }
}

// This code is contributed by Ajit.
```

PHP

```
<?php
```

```
    if ($n == 0) return $a;

    return factTR($n - 1, $n * $a);
}

// A wrapper over factTR
function fact($n)
{
    return factTR($n, 1);
}

// Driver program to test
// above function
echo fact(5);

// This code is contributed
// by Smitha
?>
```

Javascript

```
<script>

// Javascript Code for Tail Recursion

// A tail recursive function
// to calculate factorial
function factTR(n, a)
{
    if (n == 0)
        return a;

    return factTR(n - 1, n * a);
}

// A wrapper over factTR
function fact(n)
{
    return factTR(n, 1);
}

// Driver code
document.write(fact(5));

// This code is contributed by rameshtravel07
```

Output :

```
120
```

Next articles on this topic:

Tail Call Elimination

QuickSort Tail Call Optimization (Reducing worst case space to Log n)

References:

http://en.wikipedia.org/wiki/Tail_call

http://c2.com/cgi/wiki?TailRecursion

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Previous Next

RECOMMENDED ARTICLES Page : 1 2 3

01 Tail Recursion for Fibonacci
15, May 17

05 Practice questions for Linked List and Recursion
22, Apr 19

03 **Tail Call Elimination**
29, Mar 16

04 **QuickSort Tail Call Optimization (Reducing worst case space to Log n)**
30, Mar 16

06 **Practice Questions for Recursion | Set 1**
12, May 10

08 **Practice Questions for Recursion | Set 3**
18, Jun 10

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : Smitha Dinesh Semwal, jit_t, ujwalkundur, divyeshrabadiya07, divyesh072019, pratham76, mukesh07, rameshtravel07

Article Tags : Quick Sort, tail-recursion, Analysis, Recursion

Practice Tags : Recursion

Improve Article | Report Issue

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org