Program for nth Catalan Number

Difficulty Level : Medium • Last Updated : 23 Nov, 2022

Catalan numbers are defined as a mathematical sequence that consists of positive integers, which can be used to find the number of possibilities of various combinations.

The **nth** term in the sequence denoted $\mathbf{C_n}$, is found in the following formula: $\frac{(2n)!}{(n+1)!n!)}$

> *The first few Catalan numbers for n = 0, 1, 2, 3, ... are : 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, ...*

Catalan numbers occur in many interesting counting problems like the following.

parentheses that are correctly matched. For n = 3, possible expressions are **((()))**, **()(())**, **()()()**, **(())()**, **(()())**.

2. Count the number of possible Binary Search Trees with **n** keys (See this)
3. Count the number of full binary trees (A rooted binary tree is full if every vertex has either two children or no children) with **n+1** leaves.
4. Given a number **n**, return the number of ways you can draw n chords in a circle with **2 x n** points such that no **2** chords intersect.

See this for more applications.

Examples:

Input: *n = 6*
Output: *132*

Input: *n = 8*
Output: *1430*

Recommended: Please solve it on "***PRACTICE***" first, before moving on to the solution.

Recursive Solution for Catalan number:

Catalan numbers satisfy the following recursive formula:

$C_0 = 1 \ and \ C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} \ \ for \ n \geq 0$

- Base condition for the recursive approach, when **n <= 1**, return **1**

adding the product of both into **res**.

- Return the **res**.

Following is the implementation of the above recursive formula.

C++

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

// A recursive function to find nth catalan number
unsigned long int catalan(unsigned int n)
{
    // Base case
    if (n <= 1)
        return 1;

    // catalan(n) is sum of
    // catalan(i)*catalan(n-i-1)
    unsigned long int res = 0;
    for (int i = 0; i < n; i++)
        res += catalan(i) * catalan(n - i - 1);

    return res;
}

// Driver code
int main()
{
    for (int i = 0; i < 10; i++)
        cout << catalan(i) << " ";
    return 0;
}
```

Java

```
        // A recursive function to find nth catalan number
```



```
        {
            int res = 0;

            // Base case
            if (n <= 1) {
                return 1;
            }
            for (int i = 0; i < n; i++) {
                res += catalan(i) * catalan(n - i - 1);
            }
            return res;
        }

        // Driver Code
        public static void main(String[] args)
        {
            CatalnNumber cn = new CatalnNumber();
            for (int i = 0; i < 10; i++) {
                System.out.print(cn.catalan(i) + " ");
            }
        }
    }
```

Python3

```
# A recursive function to
# find nth catalan number


def catalan(n):
    # Base Case
    if n <= 1:
        return 1

    # Catalan(n) is the sum
    # of catalan(i)*catalan(n-i-1)
    res = 0
    for i in range(n):
        res += catalan(i) * catalan(n-i-1)
```

```
# Driver Code
for i in range(10):
```



```
# Nikhil Kumar Singh (nickzuck_007)
```

C#

```
// A recursive C# program to find
// nth catalan number
using System;

class GFG {

    // A recursive function to find
    // nth catalan number
    static int catalan(int n)
    {
        int res = 0;

        // Base case
        if (n <= 1) {
            return 1;
        }
        for (int i = 0; i < n; i++) {
            res += catalan(i) * catalan(n - i - 1);
        }
        return res;
    }

    // Driver Code
    public static void Main()
    {
        for (int i = 0; i < 10; i++)
            Console.Write(catalan(i) + " ");
    }
}

// This code is contributed by
// nitin mittal.
```

```
// Catalan Number
```



```
function catalan($n)
{

    // Base case
    if ($n <= 1)
        return 1;

    // catalan(n) is sum of
    // catalan(i)*catalan(n-i-1)
    $res = 0;
    for($i = 0; $i < $n; $i++)
        $res += catalan($i) *
                catalan($n - $i - 1);

    return $res;
}

// Driver Code
for ($i = 0; $i < 10; $i++)
    echo catalan($i), " ";

// This code is contributed aj_36
?>
```

Javascript

```
// Javascript Program for nth
// Catalan Number

// A recursive function to
// find nth catalan number
function catalan(n)
{

    // Base case
    if (n <= 1)
        return 1;
```

```
            res += catalan(i) *
                   catalan(n - i - 1);
```



```
    }

    // Driver Code
    for (let i = 0; i < 10; i++)
        document.write(catalan(i) + " ");

    // This code is contributed _saurabh_jaiswal
```

Output

```
 1 1 2 5 14 42 132 429 1430 4862
```

Time Complexity: The above implementation is equivalent to nth Catalan number.

$T(n) = \sum_{i=0}^{n-1} T(i) * T(n-i-1)\ for\ n \geq 1;$

The value of **nth** Catalan number is exponential which makes the time complexity exponential.
Auxiliary Space: O(n)

Dynamic Programming Solution for Catalan number:

> *We can observe that the above recursive implementation does a lot of repeated work. Since there are overlapping subproblems, we can use dynamic programming for this.*

Below is the implementation of the above idea:

- Create an array **catalan[]** for storing **ith** Catalan number.
- Initialize **catalan[0] and catalan[1] = 1**

catalan[i – j – 1] into **catalan[i]**.
- Finally, return **catalan[n]**

Follow the steps below to implement the above approach:

C++

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

// A dynamic programming based function to find nth
// Catalan number
unsigned long int catalanDP(unsigned int n)
{
    // Table to store results of subproblems
    unsigned long int catalan[n + 1];

    // Initialize first two values in table
    catalan[0] = catalan[1] = 1;

    // Fill entries in catalan[] using recursive formula
    for (int i = 2; i <= n; i++) {
        catalan[i] = 0;
        for (int j = 0; j < i; j++)
            catalan[i] += catalan[j] * catalan[i - j - 1];
    }

    // Return last entry
    return catalan[n];
}

// Driver code
int main()
{
    for (int i = 0; i < 10; i++)
        cout << catalanDP(i) << " ";
    return 0;
}
```

Java

```
    // Catalan number
    static int catalanDP(int n)
```



```
        int catalan[] = new int[n + 2];

        // Initialize first two values in table
        catalan[0] = 1;
        catalan[1] = 1;

        // Fill entries in catalan[]
        // using recursive formula
        for (int i = 2; i <= n; i++) {
            catalan[i] = 0;
            for (int j = 0; j < i; j++) {
                catalan[i]
                    += catalan[j] * catalan[i - j - 1];
            }
        }

        // Return last entry
        return catalan[n];
    }

    // Driver code
    public static void main(String[] args)
    {
        for (int i = 0; i < 10; i++) {
            System.out.print(catalanDP(i) + " ");
        }
    }
}
// This code contributed by Rajput-Ji
```

Python3

```
# A dynamic programming based function to find nth
# Catalan number


def catalan(n):
    if (n == 0 or n == 1):
```

```
    # Initialize first two values in table
```



```
    # Fill entries in catalan[]
    # using recursive formula
    for i in range(2, n + 1):
        for j in range(i):
            catalan[i] += catalan[j] * catalan[i-j-1]

    # Return last entry
    return catalan[n]


# Driver code
for i in range(10):
    print(catalan(i), end=" ")
# This code is contributed by Ediga_manisha
```

C#

```
using System;

class GFG {

    // A dynamic programming based
    // function to find nth
    // Catalan number
    static uint catalanDP(uint n)
    {
        // Table to store results of subproblems
        uint[] catalan = new uint[n + 2];

        // Initialize first two values in table
        catalan[0] = catalan[1] = 1;

        // Fill entries in catalan[]
        // using recursive formula
        for (uint i = 2; i <= n; i++) {
            catalan[i] = 0;
            for (uint j = 0; j < i; j++)
                catalan[i]
```

```
        return catalan[n];
    }
```



```
    static void Main()
    {
        for (uint i = 0; i < 10; i++)
            Console.Write(catalanDP(i) + " ");
    }
}

// This code is contributed by Chandan_jnu
```

PHP

```
<?php
// PHP program for nth Catalan Number

// A dynamic programming based function
// to find nth Catalan number
function catalanDP( $n)
{

    // Table to store results
    // of subproblems
    $catalan= array();

    // Initialize first two
    // values in table
    $catalan[0] = $catalan[1] = 1;

    // Fill entries in catalan[]
    // using recursive formula
    for ($i = 2; $i <= $n; $i++)
    {
        $catalan[$i] = 0;
        for ( $j = 0; $j < $i; $j++)
            $catalan[$i] += $catalan[$j] *
                    $catalan[$i - $j - 1];
    }

    // Return last entry
    return $catalan[$n];
```

```
        echo catalanDP($i) , " ";
```


Javascript

```
// Javascript program for nth Catalan Number

// A dynamic programming based function
// to find nth Catalan number
function catalanDP(n)
{

    // Table to store results
    // of subproblems
    let catalan= [];

    // Initialize first two
    // values in table
    catalan[0] = catalan[1] = 1;

    // Fill entries in catalan[]
    // using recursive formula
    for (let i = 2; i <= n; i++)
    {
        catalan[i] = 0;
        for (let j = 0; j < i; j++)
            catalan[i] += catalan[j] *
                   catalan[i - j - 1];
    }

    // Return last entry
    return catalan[n];
}

    // Driver Code
    for (let i = 0; i < 10; i++)
        document.write(catalanDP(i) + " ");

// This code is contributed _saurabh_jaiswal
```

```
1 1 2 5 14 42 132 429 1430 4862
```


Auxiliary Space: O(n)

Binomial Coefficient Solution for Catalan number:

*We can also use the below formula to find **nth** Catalan number in **O(n)** time.*

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Below are the steps for calculating $\mathbf{nC_r}$.

- Create a variable to store the answer and change **r** to **n – r** if **r** is greater than **n – r** because we know that **C(n, r) = C(n, n-r)** if r > n – r
- Run a loop from **0** to **r-1**
  - In every iteration update ans as **(ans*(n-i))/(i+1)**, where i is the loop counter.
- So the answer will be equal to **((n/1)*((n-1)/2)*...*((n-r+1)/r)**, which is equal to $\mathbf{nC_r}$.

Below are steps to calculate Catalan numbers using the formula: $2nC_n/(n+1)$

- Calculate $\mathbf{2nC_n}$ using the similar steps that we use to calculate $\mathbf{nC_r}$
- Return the value $2nC_n/ (n + 1)$\

Below is the implementation of the above approach:

C++

```
// Returns value of Binomial Coefficient C(n, k)
```



```
{
    unsigned long int res = 1;

    // Since C(n, k) = C(n, n-k)
    if (k > n - k)
        k = n - k;

    // Calculate value of [n*(n-1)*---*(n-k+1)] /
    // [k*(k-1)*---*1]
    for (int i = 0; i < k; ++i) {
        res *= (n - i);
        res /= (i + 1);
    }

    return res;
}

// A Binomial coefficient based function to find nth catalan
// number in O(n) time
unsigned long int catalan(unsigned int n)
{
    // Calculate value of 2nCn
    unsigned long int c = binomialCoeff(2 * n, n);

    // return 2nCn/(n+1)
    return c / (n + 1);
}

// Driver code
int main()
{
    for (int i = 0; i < 10; i++)
        cout << catalan(i) << " ";
    return 0;
}
```

Java

```
    static long binomialCoeff(int n, int k)
    {
```



```
        // Since C(n, k) = C(n, n-k)
        if (k > n - k) {
            k = n - k;
        }

        // Calculate value of [n*(n-1)*---*(n-k+1)] /
        // [k*(k-1)*---*1]
        for (int i = 0; i < k; ++i) {
            res *= (n - i);
            res /= (i + 1);
        }

        return res;
    }

    // A Binomial coefficient based function
    //  to find nth catalan number in O(n) time
    static long catalan(int n)
    {
        // Calculate value of 2nCn
        long c = binomialCoeff(2 * n, n);

        // return 2nCn/(n+1)
        return c / (n + 1);
    }

    // Driver code
    public static void main(String[] args)
    {
        for (int i = 0; i < 10; i++) {
            System.out.print(catalan(i) + " ");
        }
    }
}
```

Python3

```
# Python program for nth Catalan Number
```

```
    # since C(n, k) = C(n, n - k)
```



```
    # initialize result
    res = 1

    # Calculate value of [n * (n-1) *---* (n-k + 1)]
    # / [k * (k-1) *----* 1]
    for i in range(k):
        res = res * (n - i)
        res = res / (i + 1)
    return res

# A Binomial coefficient based function to
# find nth catalan number in O(n) time


def catalan(n):
    c = binomialCoefficient(2*n, n)
    return c/(n + 1)


# Driver Code
for i in range(10):
    print(catalan(i), end=" ")

# This code is contributed by Aditi Sharma
```

C#

```
// C# program for nth Catalan Number
using System;
class GFG {

    // Returns value of Binomial Coefficient C(n, k)
    static long binomialCoeff(int n, int k)
    {
        long res = 1;

        // Since C(n, k) = C(n, n-k)
        if (k > n - k) {
```

```
        // [k*(k-1)*---*1]
        for (int i = 0; i < k; ++i) {
```



```
        }

        return res;
    }

    // A Binomial coefficient based function to find nth
    // catalan number in O(n) time
    static long catalan(int n)
    {
        // Calculate value of 2nCn
        long c = binomialCoeff(2 * n, n);

        // return 2nCn/(n+1)
        return c / (n + 1);
    }

    // Driver code
    public static void Main()
    {
        for (int i = 0; i < 10; i++) {
            Console.Write(catalan(i) + " ");
        }
    }
}

// This code is contributed
// by Akanksha Rai
```

PHP

```
<?php
// PHP program for nth Catalan Number

// Returns value of Binomial
// Coefficient C(n, k)
function binomialCoeff($n, $k)
{
    $res = 1;
```

```
        // Calculate value of [n*(n-1)*---*(n-k+1)] /
        //                    [k*(k-1)*---*1]
```



```
            $res *= ($n - $i);
            $res = floor($res / ($i + 1));
        }

        return $res;
    }

    // A Binomial coefficient based function
    // to find nth catalan number in O(n) time
    function catalan($n)
    {
        // Calculate value of 2nCn
        $c = binomialCoeff(2 * ($n), $n);

        // return 2nCn/(n+1)
        return floor($c / ($n + 1));
    }

    // Driver code
    for ($i = 0; $i < 10; $i++)
    echo catalan($i), " " ;

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

Javascript

```
    // Javascript program for nth Catalan Number

    // Returns value of Binomial
    // Coefficient C(n, k)
    function binomialCoeff(n, k)
    {
        let res = 1;

        // Since C(n, k) = C(n, n-k)
        if (k > n - k)
            k = n - k;
```

```
            res *= (n - i);
            res = Math.floor(res / (i + 1));
```



```
        return res;
    }

    // A Binomial coefficient based function
    // to find nth catalan number in O(n) time
    function catalan(n)
    {

        // Calculate value of 2nCn
        c = binomialCoeff(2 * (n), n);

        // return 2nCn/(n+1)
        return Math.floor(c / (n + 1));
    }

    // Driver code
    for (let i = 0; i < 10; i++)
    document.write(catalan(i) + " " );

    // This code is contributed by _saurabh_jaiswal
```

Output

```
 1 1 2 5 14 42 132 429 1430 4862
```

Time Complexity: O(n).

Auxiliary Space: O(1)

We can also use the below formulas to find **nth** Catalan number in O(n) time.

$$C_n = \frac{(2n)!}{(n+1)!n!} = \prod_{k=2}^{n} \frac{n+k}{k} \quad for\ n \geq 0$$

$$C_n = \frac{2(2n-1)}{n+1} * C_{n-1} \quad | \quad n > 0$$

Catalan number using the multi-Precision library:

to calculate Catalan numbers.


```
a) initially set cat_=1 and print it
b) run a for loop i=1 to i<=n
            cat_ *= (4*i-2)
            cat_ /= (i+1)
            print cat_
c) end loop and exit
```

Below is the implementation using the muli-precision library:

C++

```
#include <bits/stdc++.h>
#include <boost/multiprecision/cpp_int.hpp>
using boost::multiprecision::cpp_int;
using namespace std;

// Function to print the number
void catalan(int n)
{
    cpp_int cat_ = 1;

    // For the first number
    cout << cat_ << " "; // C(0)

    // Iterate till N
    for (cpp_int i = 1; i <= n; i++) {
        // Calculate the number
        // and print it
        cat_ *= (4 * i - 2);
        cat_ /= (i + 1);
        cout << cat_ << " ";
    }
}

// Driver code
int main()
{
```

```
    return 0;
}
```


Java

```
import java.util.*;
class GFG {

    // Function to print the number
    static void catalan(int n)
    {
        int cat_ = 1;

        // For the first number
        System.out.print(cat_ + " "); // C(0)

        // Iterate till N
        for (int i = 1; i < n; i++) {
            // Calculate the number
            // and print it
            cat_ *= (4 * i - 2);
            cat_ /= (i + 1);
            System.out.print(cat_ + " ");
        }
    }

    // Driver code
    public static void main(String args[])
    {
        int n = 5;

        // Function call
        catalan(n);
    }
}

// This code is contributed by Debojyoti Mandal
```

Python3

```
        # For the first number
        print(cat_, " ", end='')  # C(0)
```



```
        for i in range(1, n):

            # Calculate the number
            # and print it
            cat_ *= (4 * i - 2)
            cat_ //= (i + 1)
            print(cat_, " ", end='')


# Driver code
n = 5

# Function call
catalan(n)

# This code is contributed by rohan07
```

C#

```
using System;

public class GFG {

    // Function to print the number
    static void catalan(int n)
    {
        int cat_ = 1;

        // For the first number
        Console.Write(cat_ + " "); // C(0)

        // Iterate till N
        for (int i = 1; i < n; i++) {
            // Calculate the number
            // and print it
            cat_ *= (4 * i - 2);
            cat_ /= (i + 1);
            Console.Write(cat_ + " ");
```

```
    public static void Main(String[] args)
    {
        // Function call
        catalan(n);
    }
}

// This code is contributed by Rajput-Ji
```


Javascript

```
// Function to print the number
function catalan(n)
{
    let cat_ = 1;

    // For the first number
    document.write(cat_ + " "); // C(0)

    // Iterate till N
    for (let i = 1; i < n; i++)
    {
        // Calculate the number
        // and print it
        cat_ *= (4 * i - 2);
        cat_ /= (i + 1);
        document.write(cat_ + " ");
    }
}

// Driver code
    let n = 5;

    // Function call
    catalan(n);

//This code is contributed by Mayank Tyagi
```

Time Complexity: O(n)

Catalan number using BigInteger in java:

Finding values of Catalan numbers for **N>80** is not possible even by using **long** in java, so we use **BigInteger**.

Follow the steps below for the implementation:

- Create a BigInteger variable **b** and initialize it to **1**.
- Calculate **n!** and store it into **b**.
- Calculate **n! * n!** and store into **b**.
- Create another BigInteger variable **d** and initialize it to **1**.
- Calculate **2n!** and store into **d**.
- Calculate **(2n)! / (n! * n!)** into **ans**
- Calculate **ans / (n + 1)** and return **ans**.

Below is the implementation of the above approach:

C++

```
#include <bits/stdc++.h>
using namespace std;

#define bigint long long int

bigint findCatalan(int n)
{
    bigint b = 1;

    // calculating n!
    for (int i = 1; i <= n; i++) {
        b = b * i;
    }

    // calculating n! * n!
    b = b * b;
```

```
            d = d * i;
        }
```



```
        bigint ans = d / b;

        // calculating (2n)! / ((n! * n!) * (n+1))
        ans = ans / (n + 1);
        return ans;
    }

    // Driver Code
    int main() {
        int n = 5;
        cout << findCatalan(n);
    }

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

Java

```
import java.io.*;
import java.math.*;
import java.util.*;

class GFG {
    public static BigInteger findCatalan(int n)
    {
        // using BigInteger to calculate large factorials
        BigInteger b = new BigInteger("1");

        // calculating n!
        for (int i = 1; i <= n; i++) {
            b = b.multiply(BigInteger.valueOf(i));
        }

        // calculating n! * n!
        b = b.multiply(b);

        BigInteger d = new BigInteger("1");

        // calculating (2n)!
```

```
        // calculating (2n)! / (n! * n!)
        BigInteger ans = d.divide(b);
```



```
        ans = ans.divide(BigInteger.valueOf(n + 1));
        return ans;
    }

    // Driver Code
    public static void main(String[] args)
    {
        int n = 5;
        System.out.println(findCatalan(n));
    }
}
// Contributed by Rohit Oberoi
```

Python3

```
def findCatalan(n):
    b = 1

    # calculating n!
    for i in range(1, n + 1, 1):
        b = b * i

    # calculating n! * n!
    b = b * b
    d = 1

    # calculating (2n)!
    for i in range(1, 2 * n + 1, 1):
        d = d * i

    # calculating (2n)! / (n! * n!)
    ans = d / b

    # calculating (2n)! / ((n! * n!) * (n+1))
    ans = ans / (n + 1)

    return ans

# Driver Code
```

```
// C# code to implement the approach
using System;
using System.Numerics;

class GFG {
    public static BigInteger findCatalan(int n)
    {
        // using BigInteger to calculate large factorials
        BigInteger b = new BigInteger(1);

        // calculating n!
        for (int i = 1; i <= n; i++) {
            b = BigInteger.Multiply(b, new BigInteger(i));
        }

        // calculating n! * n!
        b = BigInteger.Multiply(b, b);

        BigInteger d = new BigInteger(1);

        // calculating (2n)!
        for (int i = 1; i <= 2 * n; i++) {
            d = BigInteger.Multiply(d, new BigInteger(i));
        }

        // calculating (2n)! / (n! * n!)
        BigInteger ans = BigInteger.Divide(d, b);

        // calculating (2n)! / ((n! * n!) * (n+1))
        ans = BigInteger.Divide(ans, new BigInteger(n + 1));
        return ans;
    }

    // Driver Code
    public static void Main(string[] args)
    {
        int n = 5;
        Console.WriteLine(findCatalan(n));
    }
}
```

Javascript

Read Discuss Courses Practice Video

```
function findCatalan(n){
    let b = 1

    // calculating n!
    for(let i = 1; i <= n; i++){
        b = b * i;
    }

    // calculating n! * n!
    b = b * b;
    let d = 1;

    // calculating (2n)!
    for(let i = 1; i <= 2 * n; i++){
        d = d * i
    }

    // calculating (2n)! / (n! * n!)
    let ans = d/b;

    // calculating (2n)! / ((n! * n!) * (n+1))
    ans = ans / (n + 1);

    return ans;
}

let n = 5;
console.log(findCatalan(n));

// This code is contributed by lokeshmvs21.
```

Output

```
42
```

Time Complexity: O(n)

Previous

Pascal's Triangle

Next

Find x and y satisfying ax + by = n

Related Articles

1. Program for nth Fuss–Catalan Number
2. Total number of possible Binary Search Trees using Catalan Number
3. Minimum changes required to make a Catalan Sequence
4. Applications of Catalan Numbers
5. Program to find Nth odd Fibonacci Number
6. Program to find the Nth natural number with exactly two bits set | Set 2

9. Program to delete Nth digit of a Number

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : icr0, jit_t, nitin mittal, vt_m, soumya7, 29AjayKumar, Harshit Singhal 2, ankthon, Akanksha_Rai, Chandan_Kumar, Rajput-Ji, anantiitrk, ankit_deshmukh, madhav_mohan, RohitOberoi, manishaediga23, jyoti369, _saurabh_jaiswal, mayanktyagi1709, rohan07, bunnyram19, manishku12j, amartyaghoshgfg, prasanna1995, 07prashantsgh, phasing17, rishav1329, harendrakumar123, kothavvsaakash, ajaymakvana, lokeshmvs21

Article Tags : Amazon, catalan, series, Dynamic Programming, Mathematical, Misc

Practice Tags : Amazon, Dynamic Programming, Mathematical, Misc, series