Factorial of a Number in C

Factorial of a number is the product of all positive integers less than or equal to the number itself. For example, the factorial of 5 (represented as 5!) is 1 * 2 * 3 * 4 * 5 = 120. In this article, we will explore different methods to find the factorial of a number in C using various approaches, including iterative, recursive, and the use of functions.

Factorial Program using Iterative Solution

Iterative solutions are those that employ loops for solving problems. Let's understand how to calculate factorials using iterative solutions.

Using For Loop

The 'for' loop is one of the most common iterative solutions in C programming. It iterates over a block of code a certain number of times. Here is how we can find the factorial of a number in C using a 'for' loop.

```
#include<stdio.h>

int main()
    int                         1

    printf "Enter a number: "
    scanf "%d"

    for    1


    printf "Factorial of %d is: %d"
    return 0
```

In the above code, we initialize factorial to 1 and iterate from 1 to number. For each iteration, we multiply the current factorial with the iterator i and update the factorial.

Using While Loop

Another popular loop in C programming is the 'while' loop. Let's see how we can leverage the 'while' loop to find the factorial of a number.

```
#include<stdio.h>

int main()
   int                            1

   printf "Enter a number: "
   scanf "%d"

   while               1




   printf "Factorial is: %d"
   return 0

```

In the 'while' loop, we decrement the number in every iteration until it is greater than 1. Meanwhile, we multiply factorial with a number.

Factorial Program using Recursive Solution

In recursive solutions, a function calls itself until a base condition is met. Recursive methods can be handy for solving problems such as finding the factorial of a number.

Using Recursion

Here is how we can find the factorial of a number in C using recursion.

```
#include<stdio.h>

long int factorial(int number)
  if                1
     return                                   -1
  else
     return 1


int main()
   int
```

```
    printf "Enter a number: "
    scanf "%d"

    printf "Factorial of %d is: %ld"
    return 0
}
```

In the above code, the factorial function calls itself with number-1 until the number is greater or equal to 1. For each recursive call, the function returns the product of the number and factorial(number-1). If the number is less than 1, it returns 1, which is the base case for factorial calculation.

Using Ternary Operator

A ternary operator can be used to find the factorial of a number using a recursive method. It works as a short form for an 'if-else' statement.

```
#include<stdio.h>

long int factorial(int number)
{
    return        1                    -1    1
}

int main()
{
    int

    printf "Enter a number: "
    scanf "%d"

    printf "Factorial of %d is: %ld"
    return 0
}
```

In this code, the ternary operator checks if the number is greater or equal to 1, if true, it returns the product of the number and factorial(number-1); otherwise, it returns 1.

Algorithm of Factorial Program in C

The algorithm for a factorial program in C typically follows these steps:

1. Start the program.
2. Declare variables to store the input number and the factorial result.
3. Prompt the user to enter a number.
4. Read the input number from the user.
5. Set the factorial result variable to 1.
6. Use a loop (e.g., for loop or while loop) to iterate from 1 to the input number.
7. Inside the loop, multiply the factorial result by the current loop counter.
8. After the loop, the factorial result will hold the factorial of the input number.
9. Print the factorial result.
10. End the program.

Factorial Program using tgamma() Method

In C, the tgamma function provides the Gamma function, which is essentially (number-1)!. We can use this function to easily calculate factorials.

```
#include <stdio.h>
#include <math.h>

int
    int number;

            "Enter a number: " ;
            "%d"     number ;

            "Factorial of %d is: %.0f"  number                number    1 ;
    return 0;
```

In the code above, we prompt the user to enter a number, read it using scanf(), and then calculate the factorial using tgamma(number + 1). The tgamma() function is applied to (number + 1) to calculate the factorial of a number. The result is then printed using printf().

Factorial Program using Function

We can define a separate function for calculating factorial and call this function whenever we need to find the factorial of a number in C using function.

```
#include<stdio.h>

long int factorial(int number)
```

```
    long int          1

    for int     1
                      
    

    return
    

int main()
    int

    printf "Enter a number: "
    scanf "%d"

    printf "Factorial of %d is: %ld"
    return 0
```

In this code, we have a separate function factorial, which calculates and returns the factorial of the given number.

Factorial Program using Pointers

Pointers in C can also be used to calculate the factorial of a number. Here's how you can do it:

```
#include<stdio.h>

void factorial(int number, long int *fact)
          1

    for int     1
                      
    
    

int main()
    int
    long int

    printf "Enter a number: "
```

```
    scanf "%d"


    printf "Factorial of %d is: %ld"
    return 0
```

In this code, we pass a pointer fact to the factorial function. Inside the function, we calculate the factorial and update the value pointed by the fact pointer.

Factorial Program to Find Factorial Series

A factorial series is a sequence of factorial values. Here's how we can print a factorial series using C.

```
#include<stdio.h>

long int factorial(int number)
    long int          1

    for int     1


    return

int main()
    int

    printf "Enter a number: "
    scanf "%d"

    for int     0
        printf "Factorial of %d is: %ld\n"


    return 0
```

In this program, we calculate and print the factorial of all numbers from 0 to the given number.

Conclusion

There are multiple methods to calculate the factorial of a number in C. Whether it's using an iterative solution, recursion, the tgamma() method, or pointers, the most suitable method will depend on the specific needs of your program. It's a good idea to be familiar with each approach as they all reinforce key concepts in C programming.

To further enhance your programming and coding skills and build a successful career in the field, consider exploring upGrad's Data Science and Analytics Bootcamp. They have comprehensive courses that are ideal for every skill level, ensuring that you get a grasp of all the concepts hands-on.

FAQs

1. What is the time complexity of the factorial function using recursion?
The time complexity of the factorial function implemented using recursion is linear, denoted as O(n). This is because the function makes n recursive calls, each of which involves a constant amount of work.

2. Why is the initial value of 'factorial' set to 1 in the for loop and while loop examples?
The initial value of 'factorial' is set to 1 because the factorial of 0 is 1, and multiplication with 0 would result in the factorial being 0 for all numbers.

3. What is the purpose of the 'tgamma()' function in C?
The 'tgamma()' function in C returns the gamma function of a number. For positive integers, the gamma function of a number is equal to the factorial of (number-1). We add 1 to our number before passing it to 'tgamma()' to find the factorial.

4. Is there a difference in performance between the iterative and recursive solutions for finding the factorial of a number in C?
Yes, the recursive solution can be slower and more memory-intensive than the iterative one, especially for large input values. This is because each recursive call adds a new layer to the system's call stack. In comparison, the iterative solution only requires a single loop and does not add to the call stack.

5. Can the factorial of a negative number be calculated?
The factorial function is only defined for non-negative integers. So, technically, you cannot find the factorial of a negative number. If you try to calculate it, you should return an error or a special value to indicate that the input is invalid.