Recursion

Another Kind of Loop

Topics

Recursion

Recursion

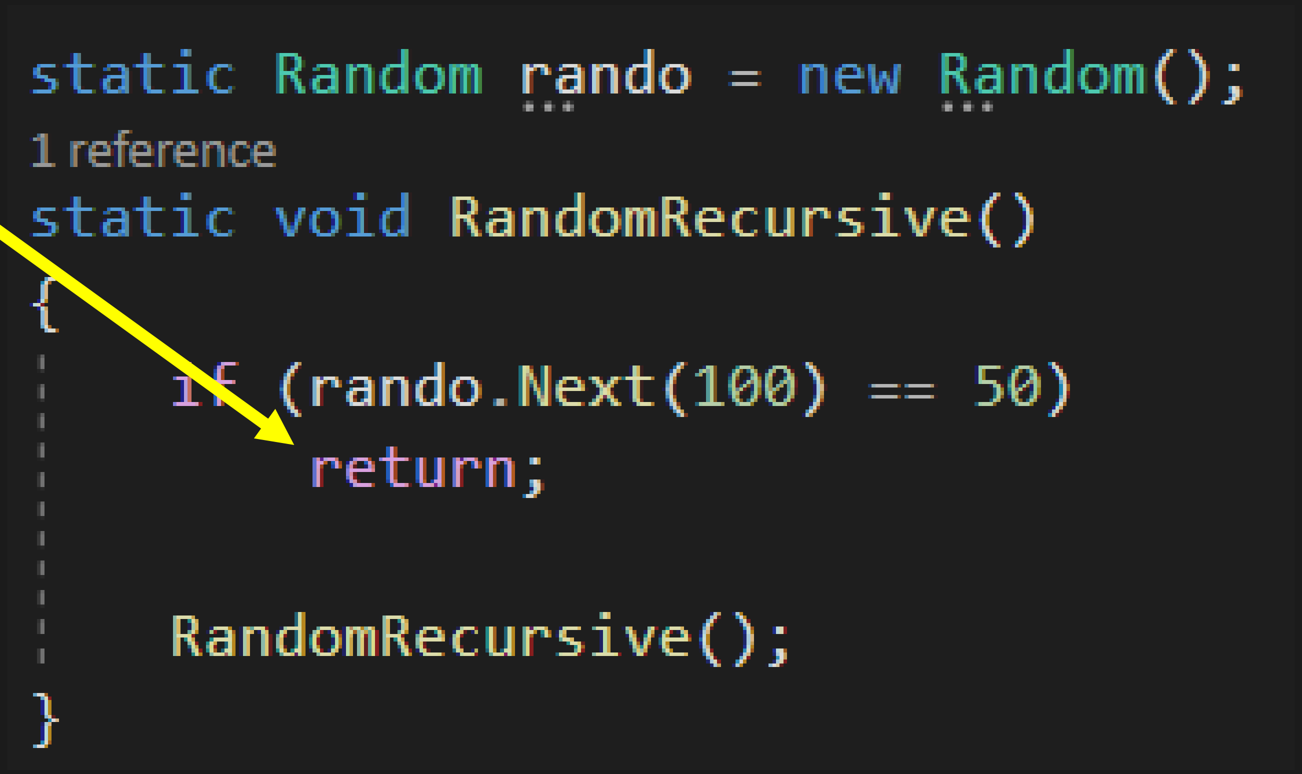
- Recursion is another kind of loop.
- Most languages support recursion by ***allowing a function to call itself***.
- This creates a loop.

```
void Method(int num)
{
        num += 1;
        Method(num);
}
```

Recursive call

Exit Conditions

- ***All*** recursive functions require an **exit condition** or else you have an infinite loop
- An exit condition is reached by completing (returning from) the method ***without*** calling itself.



```
static Random rando = new Random();
1 reference
static void RandomRecursive()
{
    if (rando.Next(100) == 50)
        return;

    RandomRecursive();
}
```

Overhead

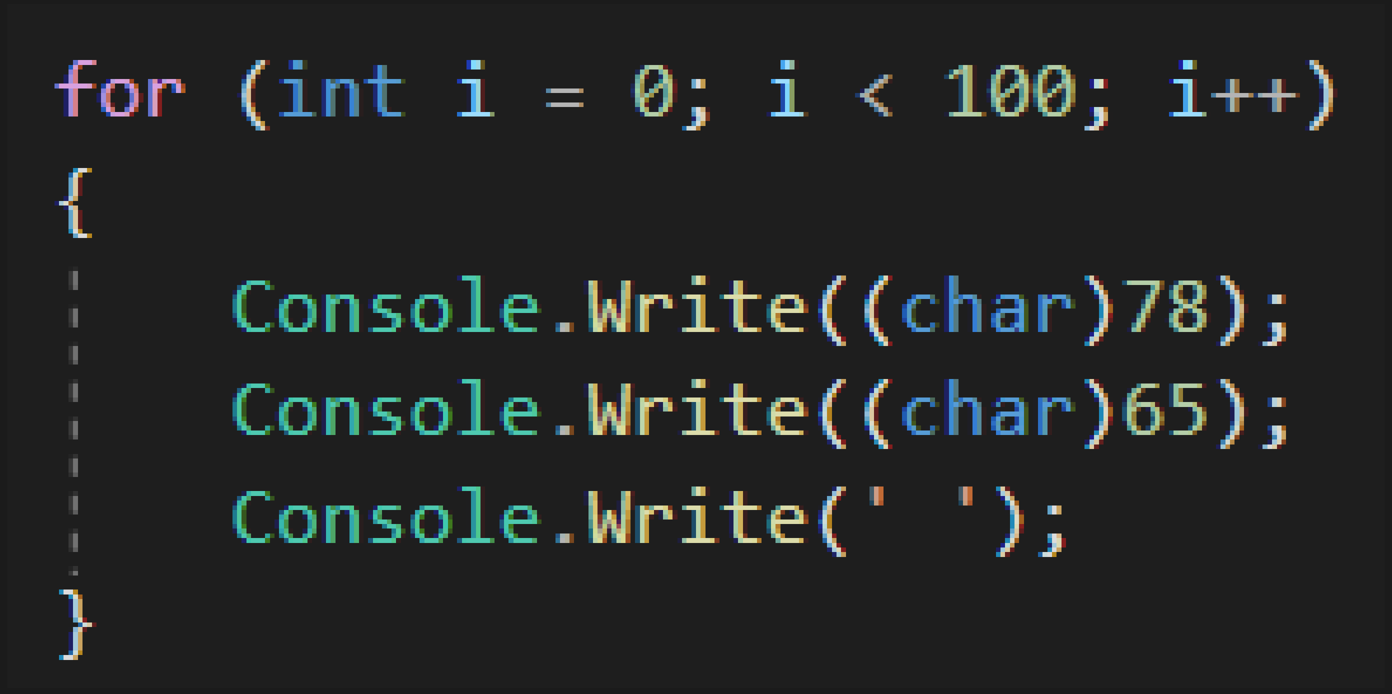
- ***Do not*** use recursion when a simple loop would suffice.
- Calling a method recursively has some overhead.
  - Methods create entries on the stack so if you call it too many times you will run out of stack space and crash your app.
- The variables it creates on the stack however can be used for maintaining state.

When?

- So when do we use recursion?
- Recursion is a way to solve problems by solving smaller versions of the same problem. *Huh?*
  - In other words, break the larger problem into smaller sub-problems of the same type and then combine those results into the final solution.

Recursive Challenge

- Turn this loop into a recursive method called Bats.



```
for (int i = 0; i < 100; i++)
{
    Console.Write((char)78);
    Console.Write((char)65);
    Console.Write(' ');
}
```

- Call Bats from Main.

LINKS

[Recursive Function]

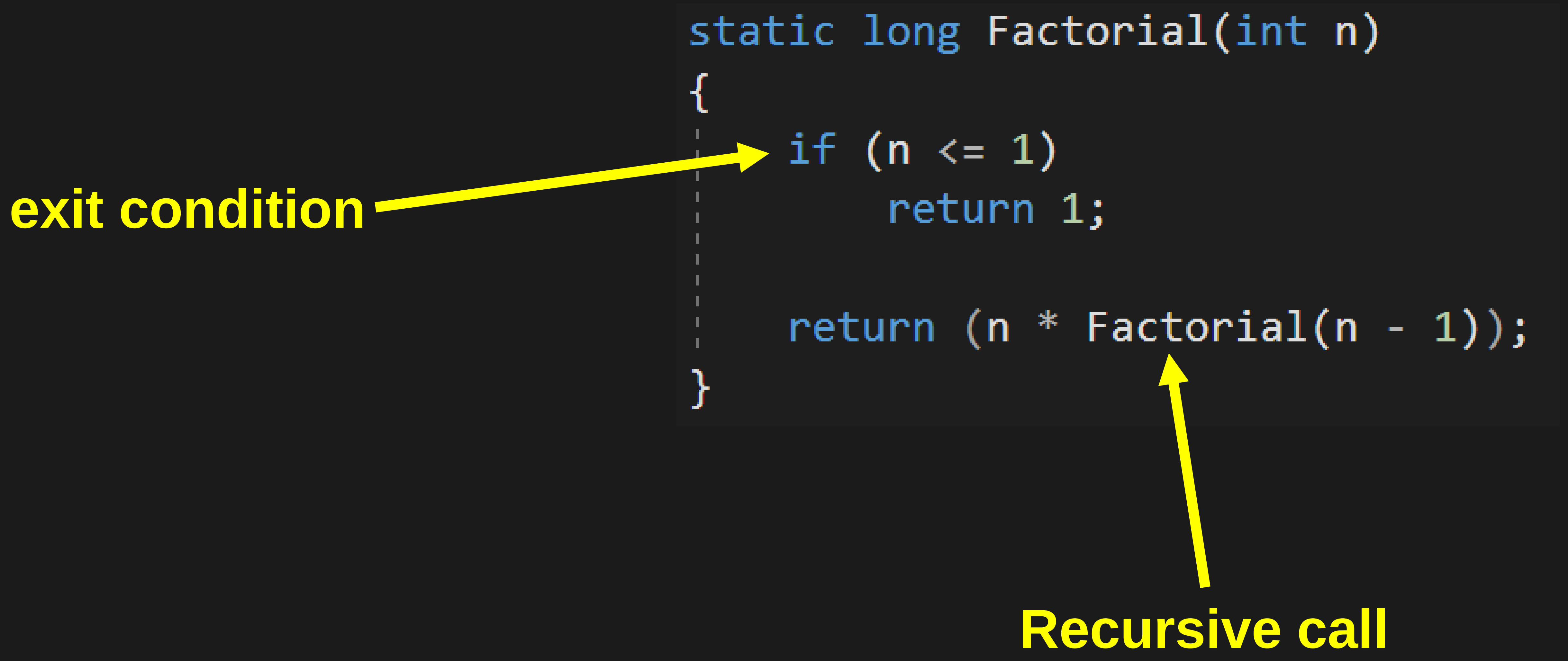
VIDEOS

Factorial

Factorial

- Factorial is a common problem that lends itself well to recursion.
- Factorial (n!) is the product of all the integers <= n.
  - 5! = 5 * 4 * 3 * 2 * 1
  - N! = N * (N-1) * (N-2)…* 1

Factorial



```
static long Factorial(int n)
{
    if (n <= 1)
        return 1;

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

Fibonacci Challenge

- Write a recursive method called Fibonacci to calculate the Fibonacci value for a given number.
- Call this from Main 40 times. N from 0 to 40
- The Formula:
  Fibonacci(N) = Fibonacci(N-1) + Fibonacci(N-2)
- Special Cases: Fibonacci(0) = 0, Fibonacci(1) = 1

LINKS

Recursive Function

VIDEOS