Recursion vs Iteration

Learn how to calculate the factorial of a number iteratively.

We'll cover the following

- Recursive solution
- Differences
- Why use recursion?

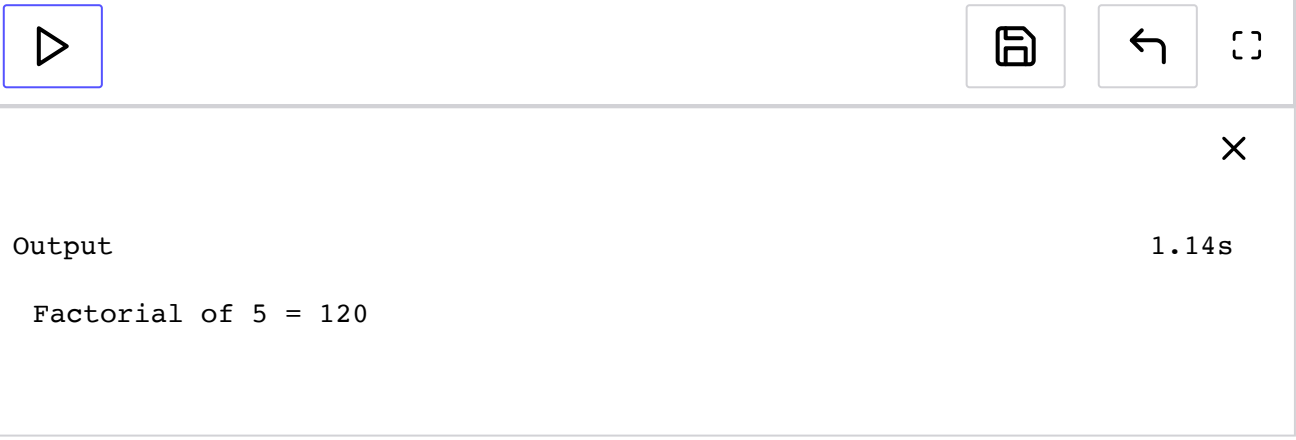
Recursive solution#

We can implement a recursive solution iteratively. Let's write a program to calculate the factorial of a number using a loop. In the iterative solution, we are not calling the same problem with a simpler version; instead, we are using the `counter` variable and we keep incrementing it until the given condition is true.

```
 2
 3  using namespace std;
 4
 5  // Iterative factorial function
 6  int factorial(int n) {
 7    int fact = 1;
 8
 9    if (n == 0) {
10      fact = 1;
11    }
12
13    for (int counter = 1; counter <= n; counter++) {
14      fact = fact * counter;
15
```

```
  }
  return fact;
}

// main function
int main() {
  int n = 5;
  int result;
  // Call factorial function in main and store the returned value in result
  result = factorial(n);
  // Prints value of result
  cout << "Factorial of " << n << " = " << result;
  return 0;
}
```



Differences#

The following are the differences between recursion and iteration:

- In the computer language, **iteration** allows you to repeat a particular set of instructions until the specified condition is met. The **recursive function** allows you to keep calling itself in the function body until some condition is met.

- The sole purpose of iteration and recursion is to achieve repetition. Loops achieve repetition through the repetitive structure, whereas recursion achieves repetition through repetitive function calls.

- Iteration terminates when loop condition fails. On the other hand, recursion terminates when the base condition evaluates to true.

- Iteration happens inside the same function, which is why it takes less memory. In the recursive function, there is the overhead of function calls that makes our program slow and consumes more memory since each function call calls another copy of the function.
- In iteration, our code size is very large. Meanwhile, recursion helps to write shorter code.
- Iterative code is faster than recursive code.
- Infinite loops will stop further execution of the program but do not lead to system crash. Infinite recursive calls, on the other hand, will result in a CPU crash because of memory overflow.

Why use recursion?#

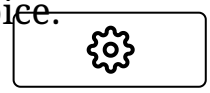
Using depends upon the requirements of your problem. The problems that can be defined in terms of itself are the best candidates for recursion.

Use recursion when a problem can be divided into simpler versions of itself. Consider the example of a file searching system. In a file searching system, you start with the main folder and then go through each subfolder to find a particular file.

For such a type of problem, use recursion. The recursive solution helps you write shorter and more understandable code that makes debugging easier.

However, if performance is your main concern, write the iterative solution since it takes less time and memory for its execution. Recursive calls take more time and extra memory.

That said, whichever you want to use is totally your choice.



Quiz

Q Which of the following statements are true?

(You can select multiple correct options)

Selected Option

- [x] **A)** Iterative code takes less memory than recursive code.
- [ ] **B)** Iterative code takes more memory than recursive code.

Selected Option

- [x] **C)** Iterative code is faster than recursive code.
- [ ] **D)** Iterative code is slower than recursive code.

Submit Answer

Reset Quiz

Let’s solve some challenges related to recursion in the upcoming lessons.