|  |  |  |  |
| --- | --- | --- | --- |
| Number | Factorial of number | Execution time for factorial by iteration function | Execution time for factorial by recursion function |
| 2 | 2 | 0.0002053s | 0.000275s |
| 4 | 24 | 0.0002750s | 0.0002795s |
| 8 | 40320 | 0.0002836s | 0.0002951s |
| 16 | 20922789888000 | 0.0003052s | 0.0003210s |
| 20 | 2432902008176640000 | 0.0003449s | 0.0003534s |

As observed from the table and illustrated in the graph below, itirative functions consume less time than recursive functions, even though the time difference is small in this program. This is attributed to the high-speed and large storage capacity of contemporary coomputer devices. The reason for this difference lies in the operational mechanism of each of these functions, itirative functions operate by repeating the same programming command with a variation in one of the variables within the function itself, without the need for recursive function calls. On the contrary, recursive functions rely on calling the function again with changes in variables, leading to an increase in the size of the stacks used in code compiler.

When calculating a large number like 1000 using the iterative function, the results showed a zero value for the factorial, The reason for this is that the factorial value for such a number exceeds the storage capacity allotted to each memory location reserved for this variable. As a result, the value appears as zero with longer execution time, presumably due to the increased iteration value for the programming command.

As for calculating the large number using the recursive function, the program stops, presenting a stack overflow problem.

Through research I found this little definition of this problem:- a stack overflow occurs if the call stack pointer exceeds the stack bound. The call stack may consist of a limited amount of address space, often determined at the start of the program. The size of the call stack depends on many factors, including the programming language, machine architecture, multi-threading, and amount of available memory. When a program attempts to use more space than is available on the call stack (that is, when it attempts to access memory beyond the call stack's bounds, which is essentially a buffer overflow), the stack is said to overflow, typically resulting in a program crash.

In conclusion, I found from this experiment that using iterative functions is better than using recursive functions in terms of execution time and potential issues especially If a problem arises that requires a solution within a loop.

I apologize for the extended report beyond one page.