Recursion

ESC108A Elements of Computer Science and Engineering

B. Tech. 2017

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

© Ramaiah University of Applied Sciences

Objectives

- At the end of this lecture, student will be able to
  - apply the concept of recursion
  - identify recursion in a C programming language
  - use macro programming constructs in the C programming language

© Ramaiah University of Applied Sciences

Contents

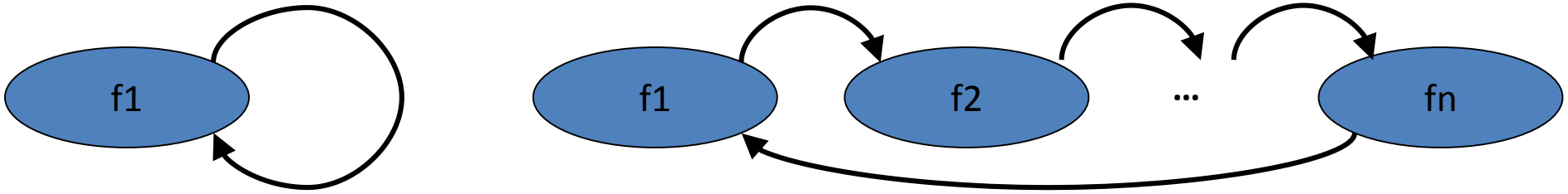
- Recursion
- C preprocessor

© Ramaiah University of Applied Sciences

Recursion

- C functions can be used recursively
  - A function may call itself either directly or indirectly

- When a function calls itself recursively, each invocation gets a fresh set of all the automatic variables, independent of the previous step



© Ramaiah University of Applied Sciences

Factorial: In Mathematics

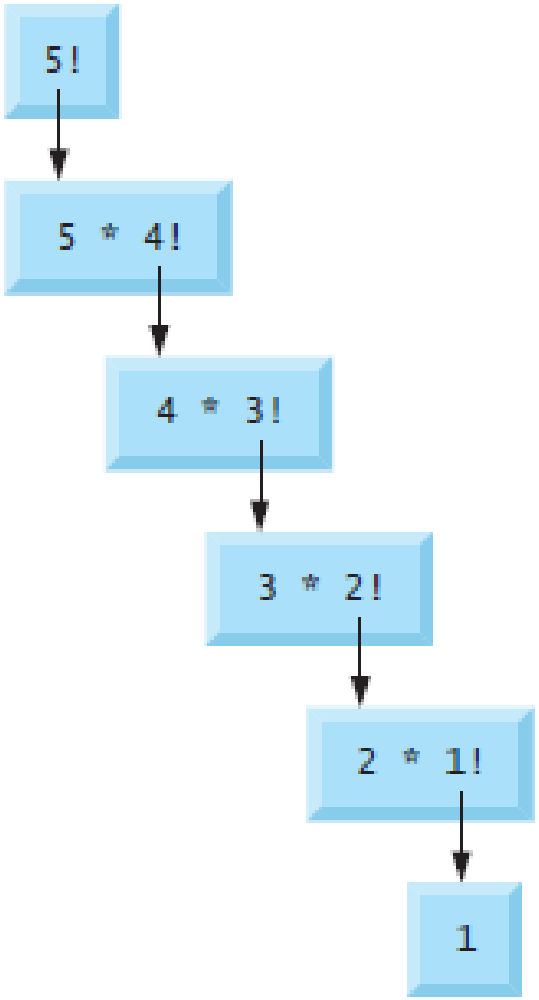
$$N! = \begin{cases} 1 & \text{if } N = 0 \\ N*(N-1)! & \text{if } N > 0 \end{cases}$$

Example using Definition

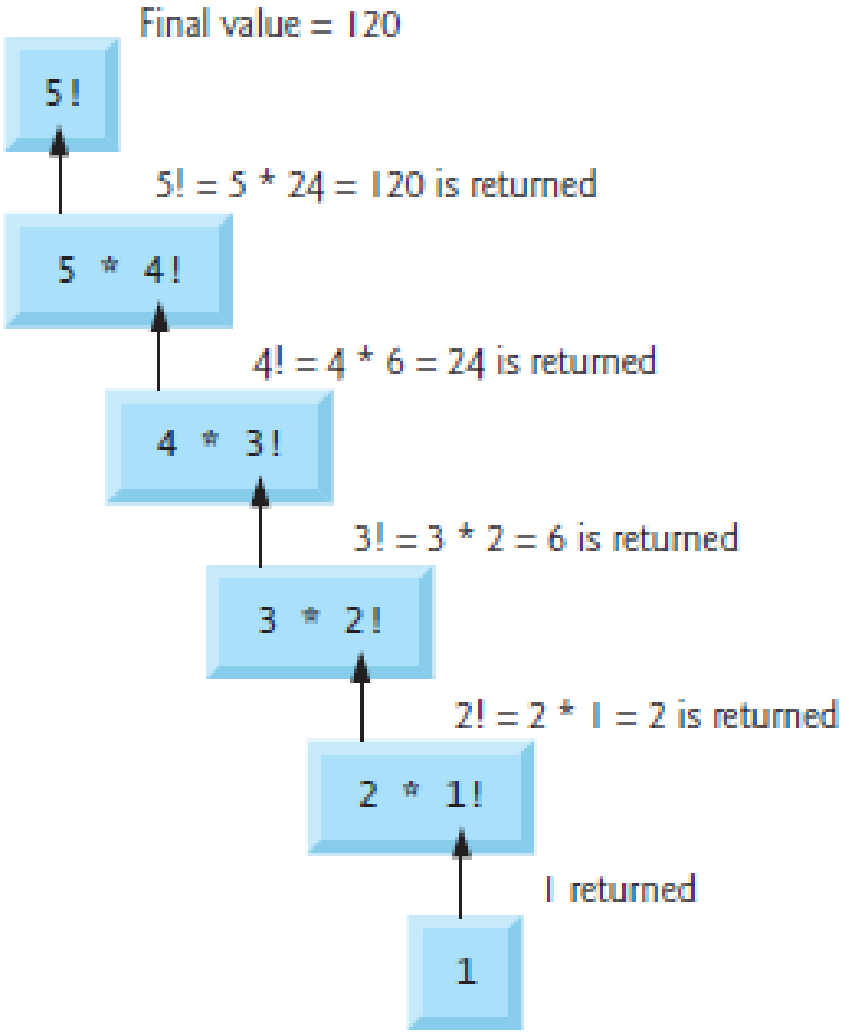
4! = 4 * 3!
= 4 * 3 * 2!
= 4 * 3 * 2 * 1!
= 4 * 3 * 2 * 1 * 0!
= 4 * 3 * 2 * 1 * 1

© Ramaiah University of Applied Sciences

Recursion - Example



(a) Sequence of recursive calls.



(b) Values returned from each recursive call.

© Ramaiah University of Applied Sciences

Recursive Function

- Determine the base case(s)
  - the one for which you know the answer
  - There must be a terminating condition

- Determine the general case(s)
  - the one where the problem is expressed as a smaller version of itself

© Ramaiah University of Applied Sciences

Factorial Using Recursion

```
Algorithm fact(x :Integer):Integer
var xFactorial :Integer; {The factorial}
begin
        {assert x=>0}
        If (x <= 1) then
        begin
                xFactorial := 1;
        end
        else     {if it is greater than 1}
        begin
             xFactorial  := x * fact ( x-1 );
        end
end
```

© Ramaiah University of Applied Sciences

Equivalence with Iteration

- All recursive functions have iterative equivalents
- Example:

```
Algorithm fact(x :integer):Integer
Var iLoop, xFactorial :Integer; {The factorial}
begin
        {assert x=>0}
        xFactorial := 1;
        for iLoop in 1 to n, step 1 do
        begin
            xFactorial  := xFactorial * iLoop;
        end
end
```

© Ramaiah University of Applied Sciences

Recursion vs. Iteration

Both Recursion and Iteration is

- Based on a control structure
  - Iteration uses a repetition structure
  - recursion uses a selection structure
- Involve repetition
  - Iteration explicitly uses a repetition structure
  - recursion achieves repetition through repeated function calls
- Involve a termination test
  - Iteration terminates when the loop-continuation condition fails
  - Recursion terminates when a base case is recognized

© Ramaiah University of Applied Sciences

Recursion vs. Iteration contd.

Both iteration and recursion can occur infinitely

- An infinite loop occurs with iteration if the loop-continuation test never becomes false
- infinite recursion occurs if the recursion step does not reduce the problem each time in a manner that converges on the base case

© Ramaiah University of Applied Sciences

Why Avoid Recursion?

- What happens when the value of n increases?
- How many variables will be there in memory?

© Ramaiah University of Applied Sciences

Merits and Demerits of Recursion

- Merits
  - Reduce unnecessary calling of function
  - Easier to write and understand than the non-recursive equivalent

- Demerits
  - Take more time compared to iterative methods
  - A stack of the values being processed must be maintained
  - Consume additional memory
  - Recursive solution is always logical and it is very difficult to trace

© Ramaiah University of Applied Sciences

C Preprocessor

- C provides certain language facilities by means of a preprocessor
  - A macroprocessor program that processes the source program before it is compiled

  Source Program -> C Preprocessor -> Compiler -> Object Prorgam

- Before the source code passes through the compiler, it is examined by the preprocessor for any preprocessor directives

© Ramaiah University of Applied Sciences

Preprocessor Directives

- Preprocessor directives are placed in the source program before the main line
- Statements which begin with # symbol
  - #include
  - #define

- Three types
  1. Macro substitution directives
  2. File inclusion directives
  3. Conditional compilation directives

© Ramaiah University of Applied Sciences

Macro Substitution

- To replace a token by an arbitrary sequence of characters
- A definition has the form

  # define name replacement_text

- Possible to define macro with arguments
- E.g., define a macro called max

  #define max(A,B) ((A)>(B)? (A) : (B))

  The line

  X=max(p+q, r+s); will be replaced by the line

  X=((p+q)>(r+s) ? (p+q) : (r+s));

© Ramaiah University of Applied Sciences

File Inclusion

- To include the contents of the file during compilation
- Avoids rewriting those functions or macro definitions

#include “filename”

– Searching of the file typically begins where the source program was found

#include <filename>

– Searching follows an implementation-defined rule to find the rule

© Ramaiah University of Applied Sciences

Summary

- Functions can call themselves - recursion
- Recursive programs need memory and hence are discouraged
- All recursive programs have iterative equivalents
- Preprocessor directives are placed in the source program before the main line

© Ramaiah University of Applied Sciences

Further Reading

Kernighan, B. W. and Richie, D. (1992) *The C Programming Language.* 2nd ed., New Delhi:PHI.

© Ramaiah University of Applied Sciences