LECTURE AND PROCERM DESTON IN C. PROPRIED TO PROCERM DESTON IN C. PROCERM DEST

FIGURE 9.1 SPLITTING A PROBLEM INTO SMALLER PROBLEMS

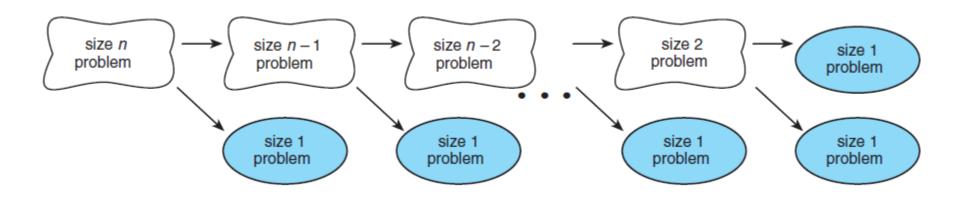


FIGURE 9.2 RECURSIVE FUNCTION MULTIPLY

```
1. /*
2. * Performs integer multiplication using + operator.
   * Pre: m and n are defined and n > 0
3.
   * Post: returns m * n
   */
6. int
multiply(int m, int n)
8. {
9.
       int ans;
10.
11. if (n == 1)
12.
             ans = m; /* simple case */
13.
     else
14.
             ans = m + multiply(m, n - 1); /* recursive step */
15.
16.
       return (ans);
17. }
```

FIGURE 9.5 TRACE OF FUNCTION MULTIPLY

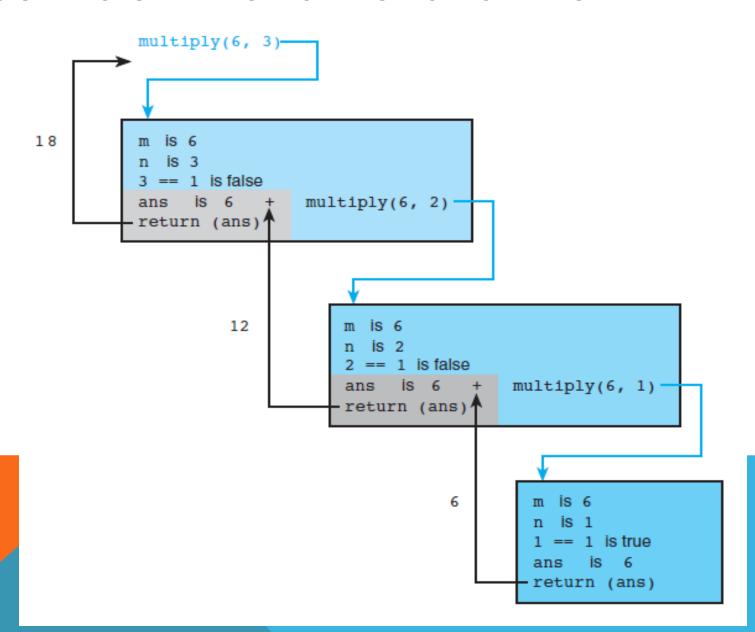


FIGURE 9.9 RECURSIVE FUNCTION MULTIPLY WITH PRINT STATEMENTS TO CREATE TRACE AND OUTPUT FROM MULTIPLY(8, 3)

```
1.
2.
   * *** Includes calls to printf to trace execution ***
   * Performs integer multiplication using + operator.
3.
    * Pre: m and n are defined and n > 0
4.
5.
    * Post: returns m * n
6.
    */
7.
   int
   multiply(int m, int n)
8.
9.
10.
          int ans;
11.
12.
      printf("Entering multiply with m = %d, n = %d\n", m, n);
13.
14.
          if (n == 1)
15.
                ans = m; /* simple case */
16.
          else
                ans = m + multiply(m, n - 1); /* recursive step */
17.
      printf("multiply(%d, %d) returning %d\n", m, n, ans);
18.
19.
20.
          return (ans);
21. }
22.
   Entering multiply with m = 8, n = 3
23.
   Entering multiply with m = 8, n = 2
24.
   Entering multiply with m = 8, n = 1
25.
   multiply(8, 1) returning 8
26.
   multiply(8, 2) returning 16
multiply(8, 3) returning 24
```

FIGURE 9.10 RECURSIVE FACTORIAL FUNCTION

```
/ ±
* Compute n! using a recursive definition
3. * Pre: n >= 0
4. */
5. int
factorial(int n)
7. {
8.
          int ans;
9.
10.
          if (n == 0)
11.
                ans = 1;
12.
          else
13.
                ans = n * factorial(n - 1);
14.
15.
          return (ans);
16.
```

FIGURE 9.11 TRACE OF FACT = FACTORIAL(3);

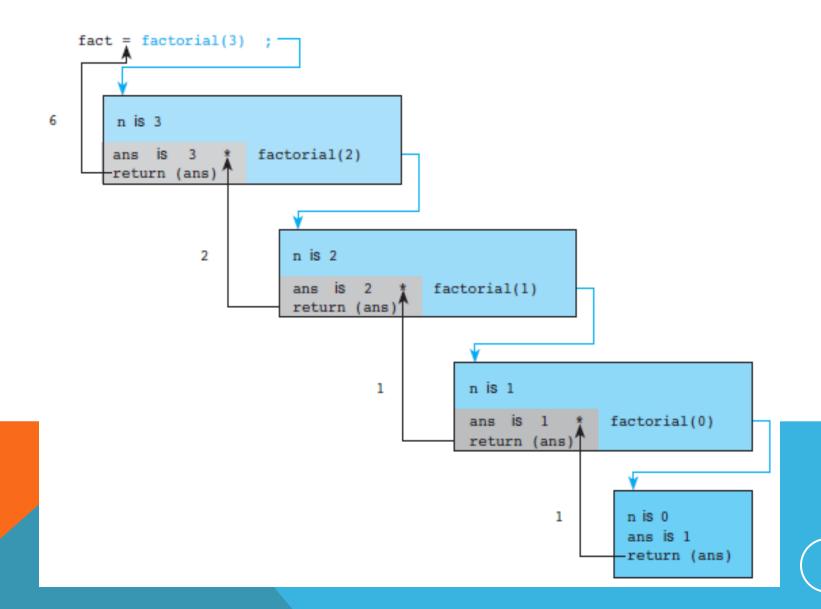


FIGURE 9.12 ITERATIVE FUNCTION FACTORIAL

```
/*
   * Computes n!
   * Pre: n is greater than or equal to zero
5. int
   factorial(int n)
7.
                        /* local variables */
8.
       int i,
9.
            product = 1;
10.
11.
      /* Compute the product n x (n-1) x (n-2) x . . . x 2 x 1 */
12.
       for (i = n; i > 1; --i) {
13.
          product = product * i;
14.
15.
16.
        /* Return function result */
17.
        return (product);
18. }
```

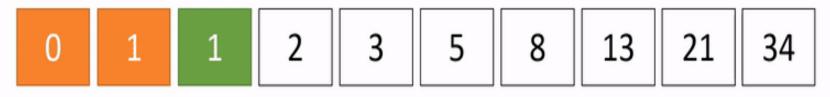
FIGURE 9.13 RECURSIVE FUNCTION FIBONACCI

```
/*
* Computes the nth Fibonacci number
3. * Pre: n > 0
   * /
5. int.
fibonacci(int n)
7. {
8.
         int ans;
         if (n == 1 || n == 2)
10.
11.
          ans = 1;
12.
         else
13.
               ans = fibonacci(n - 2) + fibonacci(n - 1);
14.
15.
        return (ans);
16. }
```

RECURSIVE FUNCTION FIBONACCI

Fibonacci Series

A series of numbers in which each number (Fibonacci number) is the sum of the two preceding numbers.





© STUDYGYAAN.com

FIBONACCI FOR DYNAMIC PROGRAMMING

Running time?

```
FIBONACCI-DP(n)

1 fib[1] \leftarrow 1

2 fib[2] \leftarrow 1

3 for i \leftarrow 3 to n

4 fib[i] \leftarrow fib[i-1] + fib[i-2]

5 return fib[n]
```

 $\Theta(n)$

```
def fib(n):
               if n == 0:
                           return 0
               elif n == 1:
                           return 1
               else:
                           return(fib(n-1)) + (fib(n-2))
def fib(n):
                                                           def fib(n):
      if n -- 0:
                                                                 11 n -- 00:
            return 0.
                                                                       return 0
      elif n -- 1:
                                                                 elif n -- 1:
            return 1
                                                                      return 1
      elset
                                                                 elser
            return (fib(n-1)) + (fib(n-2))
                                                                       return (fib(n-1)) + (fib(n-2))
der til transfer (v.)
                                                          AMERICAN PROPERTY.
                                                                                      destination (project
  AND PROPERTY LINE
                              受賞 🌞 😑 (4 m)
                                                             付付。** ** ** ** (1)
                                                                                         12 mm - 12.
     Description - N
                                 permiss di
                                                               BENEDOCK 4
                                                                                           personne 4:
                                                             ALTERNATION CO.
  WEST A SECTION
                              ACCOMPANY OF THE
                                                                                         all the form the same
     Detroited 1.
                                 RESIDENCE SHOW
                                                               territores as
                                                                                            Bests Bills 11.
                                                                                         self-read to
                                 DECEMBER OF LANCE OF STREET
                                                               mentalized the re-till office
     Detroite Bill brown
                                                                                            measure Milking
```

