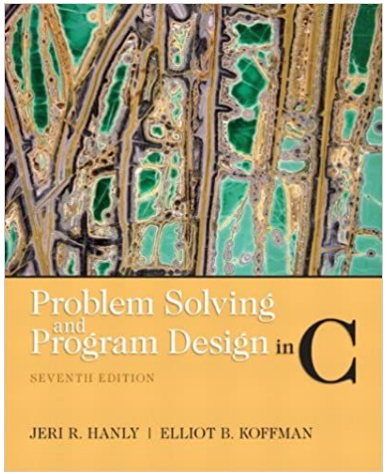
BIRZEIT UNIVERSITY



Faculty of Engineering and Technology
Department of Computer Science

Introduction to Computers and Programming (Comp 133)

References :
Book : Problem Solving and Program Design in C (7th Edition) 7th Edition
Slides : Dr. Radi Jarrar , Dr. Abdallah Karakra , Dr. Majdi Mafarja.

Recursive

Chapter 9

Recursive

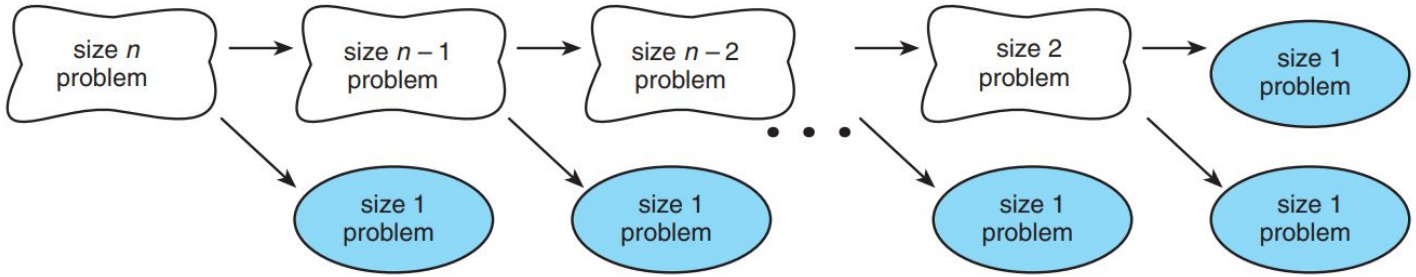
- Recursive function function that calls itself or that is part of a cycle in the sequence of function calls.
- The ability to invoke itself enables a recursive function to be repeated with different parameter values.
- You can use recursion as an alternative to iteration (looping)
- A recursive solution is less efficient than an iterative solution in terms of computer time due to the overhead for the extra function calls.
- The use of recursion enables us to specify a very natural, **simple solution to a problem** that would otherwise be **very difficult to solve**

Chapter 9

- Recursive
  - The Nature of Recursion

The Nature of Recursion

- To solving a particular problem, one of the basic design techniques is to break the task into smaller subtasks.
- The problem of size 1 can be solved easily (i.e., the simple case).
- We can **recursively split the problem** into a problem of size 1 and another problem of size n-1



The Nature of Recursion

- **Recursive Problem**

```
void message()
{
    printf("This is a recursive function.\n“);
    message();
 }
```

- The function is like **an infinite loop** because there is **no code to stop it from repeating**
- Like a loop, a recursive function must have some **criteria to control the number of times it repeats**

The Nature of Recursion

- Function must have some algorithm to control the number of times it repeats.
- Passes an integer argument, which holds the number of times the function is to call itself.

```
void message(int times)
{
  if (times > 0)
  {
     printf("This is a recursive function.\n“);
     message(times - 1);
  }
}
```

The Nature of Recursion

- Recursion Function in general :

  if (stopping case)

  solve it //base case

  else

  reduce the problem using recursion (call function itself)

- **Activation frame : representation of one call to a function**

Recursive Function

- For example, the problem of adding (or multiplying) n consecutive integers.
  - 1 + 2 + 3 +... + n = n + [1 + 2 + 3 + .. + (n-1)]
    - **sumR(n) = n + sumR(n-1)**
  - 1 * 2 * 3 *... * n = n * [1 * 2 * 3 * .. * (n-1)]
    - **timesR(n) = n * timesR(n-1)**

```
int sum(int n)
{
   int res = 0;
   for(int i = 1; i = n; i++)
      res = res + i;

   return res;
}
```

```
int sumR(int n)
        {
          if(n == 1)
             return 1;
          else
             return n + sumR(n-1);
        }
```

Recursive Function

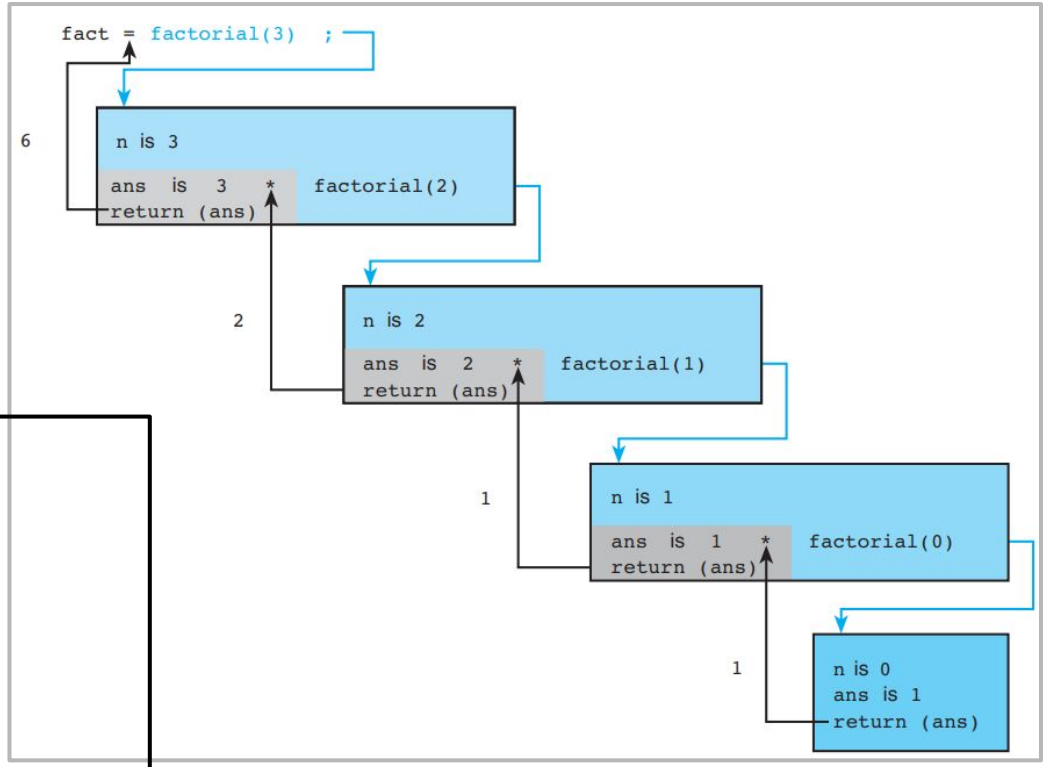
- **Base case :** To terminates recursive function **a base case** must be met.
- Let f(x)=f(x -1)+3 , f(0)=4 , find f(7)

```
int f(int x)
{
  if (x == 0)
    return 4; //base case
 else
    return f(x-1)+3;
}
```

Recursive Function Example

- **Find n!**
  - n! = n * (n-1)!
  - **n! = 1 if n = 0 //base case**
  - n! = n*(n-1)! **if n > 0**

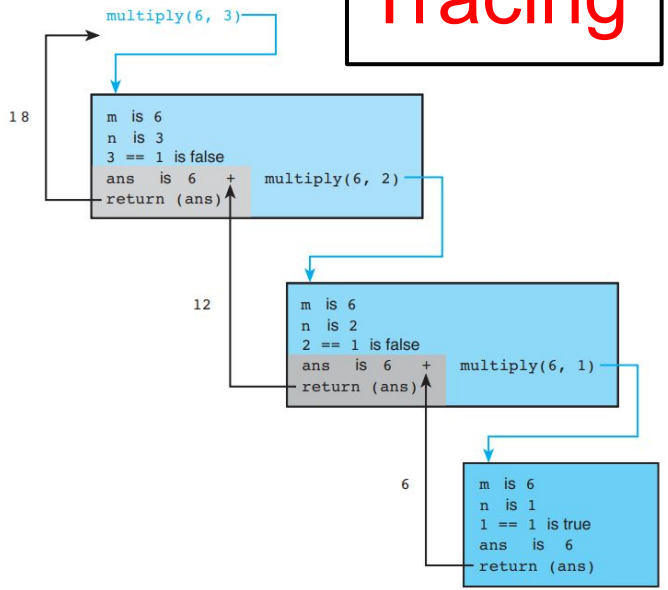
```
int factorial(int n)
{
 if (n ==0)
    return (1);
else
    return (n * factorial(n-1));
}
```



Recursive Function Example

- Performs integer multiplication using + operator.
  - Pre: m and n are defined and n > 0
  - Post: returns m * n

Tracing

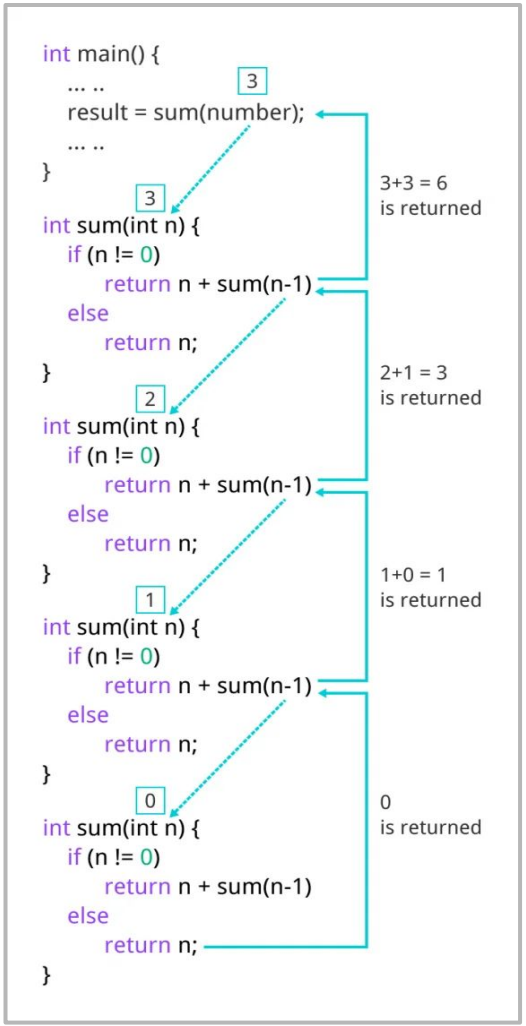



```
int multiply(int m, int n)
 {
  int ans;

  if (n == 1)
   ans = m; /* simple case */
  else
  ans = m + multiply(m, n - 1); /* recursive step */
   return (ans);
 }
```

Tracing a Recursive Function

```
// Asume int result=sum(3);
int sum(int n)
{
    if (n != 0)
      return n + sum(n-1);
    else
      return n;
}
```

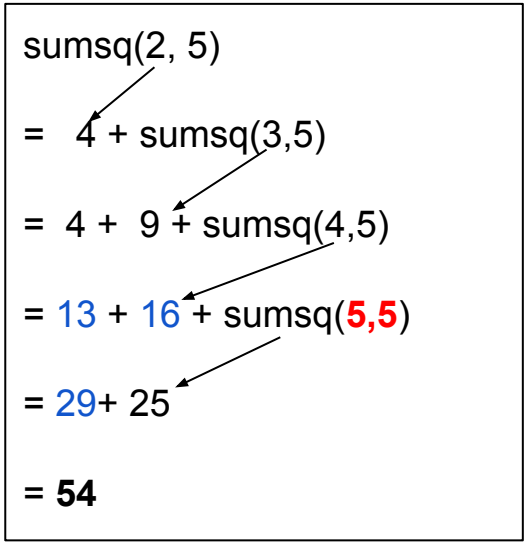


https://www.programiz.com/c-programming/c-recursion

Tracing a Recursive Function

- Find sum of squares of a series.
  - $m^2 + (m + 1)^2 + (m + 2)^2 + \ldots\ldots\ldots + (n)^2$
  - $m^2 + [(m + 1)^2 + (m + 2)^2 + \ldots\ldots\ldots + (n)^2]$
  - **sumsqr (m, n) = m^2 + sumsqr (m+1, n)**

```
int sumsqr ( int m, int n) {
   if (m ==n )
      return n *n;
  else
      return ( m * m + sumsqr(m+1, n);
}
```



Tracing a Recursive Function

- What is returned value for the following function

```
int mystery(int x, int y) {
   if (x < y)
      return x;
  else
      return mystery(x - y, y);
}
```

```
mystery(6, 13)      // 6
mystery(14, 10)     // 4
mystery(37, 10)     // 7
mystery(8, 2)       // 0
mystery(50, 7)      // 1
```

Tracing a Recursive Function

- Trace the following recursive function for **N= 7**.

```
int speed (int N)
{
  if (N == 2)
     return 5;
  if (N % 2 == 0)
    return (1 + speed(N/2));
 else
    return (2+speed(3 + N));
}
```

Speed(7)

= 2 + speed(10)

= 2 + 1 + speed(5)

= 3 + 2 + speed(8)

= 5 + 1 + speed (4)

= 6 + 1 + speed (2)

= 7 + 5

= 12

Tracing a Recursive Function

- Trace the following recursive function.

```
int value(int a, int b) {
 if (a <= 0)
    return 1;
 else
    return (b*value(a-1,b+1));
}
```

value(1, 5)

= 5 * value(0, 6)

= 5 * 1

= 5

value(3,3)

= 3 * value(2, 4)

= 3 * 4 * value(1, 5)

= 3 * 4 * 5 * value(0, 6)

= 3 * 4 * 5 * 1

= 60

Recursive Function

- Print a user-entered **n characters** in reverse order.

```
void print_reverse (int n)
{
  char next;
  if (n == 1)
    {                  /* stopping case */
      scanf (" %c", &next);
      printf (" %c", next);
    }
  else
    {
      scanf (" %c", &next);
      print_reverse (n - 1);
      printf (" %c", next);
    }
  return;
}
```

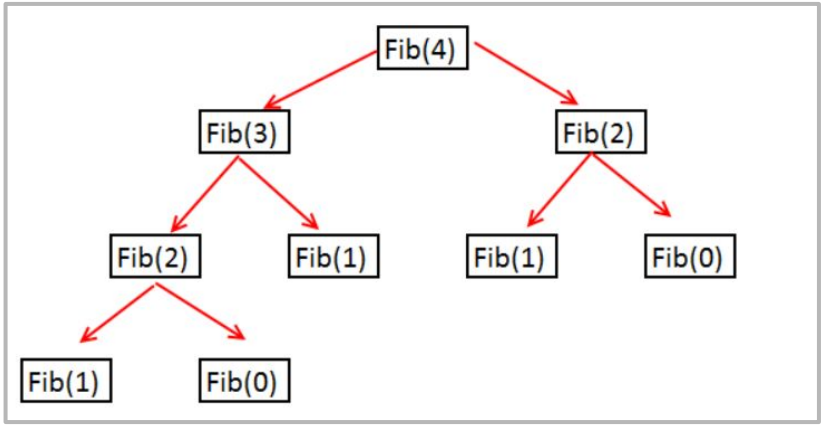
Recursive Function

- Write recursive function to calculate $\mathbf{x^y}$.

```c
#include<stdio.h>
int power(int,int);
int main()
{
    int x,y;
    printf("Enter x and y ");
    scanf("%d%d",&x,&y);
    printf("power=%d",power(x,y));
    return 0;
}
int power(int x,int y)
{
    if(y>0)
     return x*power(x,y-1);
    else
     return 1;
}
```

Recursive Function

- Write C program to print **Fibonacci** series program using recursive methods
  - Fibonacci number is defined as the sum of the two preceding numbers:
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …




```
int fib(int i){
    if(i==0) return 0;
    else if(i==1) return 1;
    else return (fib(i-1)+fib(i-2));
}
```

Thank You.

جامعة بيرزيت
BIRZEIT UNIVERSITY