CS1102 – PROGRAMACIÓN ORIENTADA A OBJETOS 1
CICLO 2019-1

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CIENCIA DE LA COMPUTACIÓN

Unidad 2: Funciones y Recursividad

http://bit.ly/2HRBWgq

Profesores:

Ernesto Cuadros- Vargas, PhD. *ecuadros@utec.edu.pe*

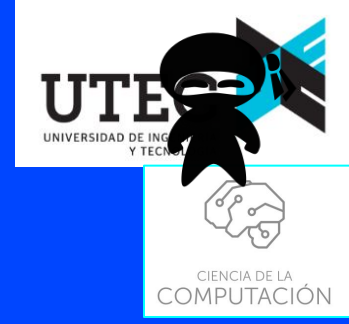
María Hilda Bermejo, M. Sc. *mbermejo@utec.edu.pe*

Telegram:

1. Configurar tu cuenta

2. http://bit.ly/2TJnwBq

CS1102 – PROGRAMACIÓN ORIENTADA A OBJETOS 1
CICLO 2019-1



Encuesta:

link → http://bit.ly/2Zqjxxx

Profesor:
Ernesto Cuadros-Vargas

CIENCIA DE LA COMPUTACIÓN

Logro de la sesión:

Al finalizar la sesión, los alumnos desarrollan sus programas utilizando recursividad.

Recursividad

La recursividad ocurre cuando ….

Para hallar el resultado una función se invoca a si misma.

El algoritmo para hallar el factorial de un número es un ejemplo clásico del uso de recursividad.

fact(7) = 7*6*5*4*3*2*1 ó **fact(7) = 7 * fact(6)**

fact(9) = 9*8*7*6*5*4*3*2*1 ó **fact(8) = 8 * fact(7)**

fact(14) = 14 * fact(13)

Generalizando:

```
fact(n) = n * fact(n-1)
```

Veamos cómo funciona la recursividad, hallando el factorial de 5

```
fact(5) = 5 * fact(4)
```

Recursión:

```
fact(5) = 5 * fact(4)
          4 * fact(3)
```

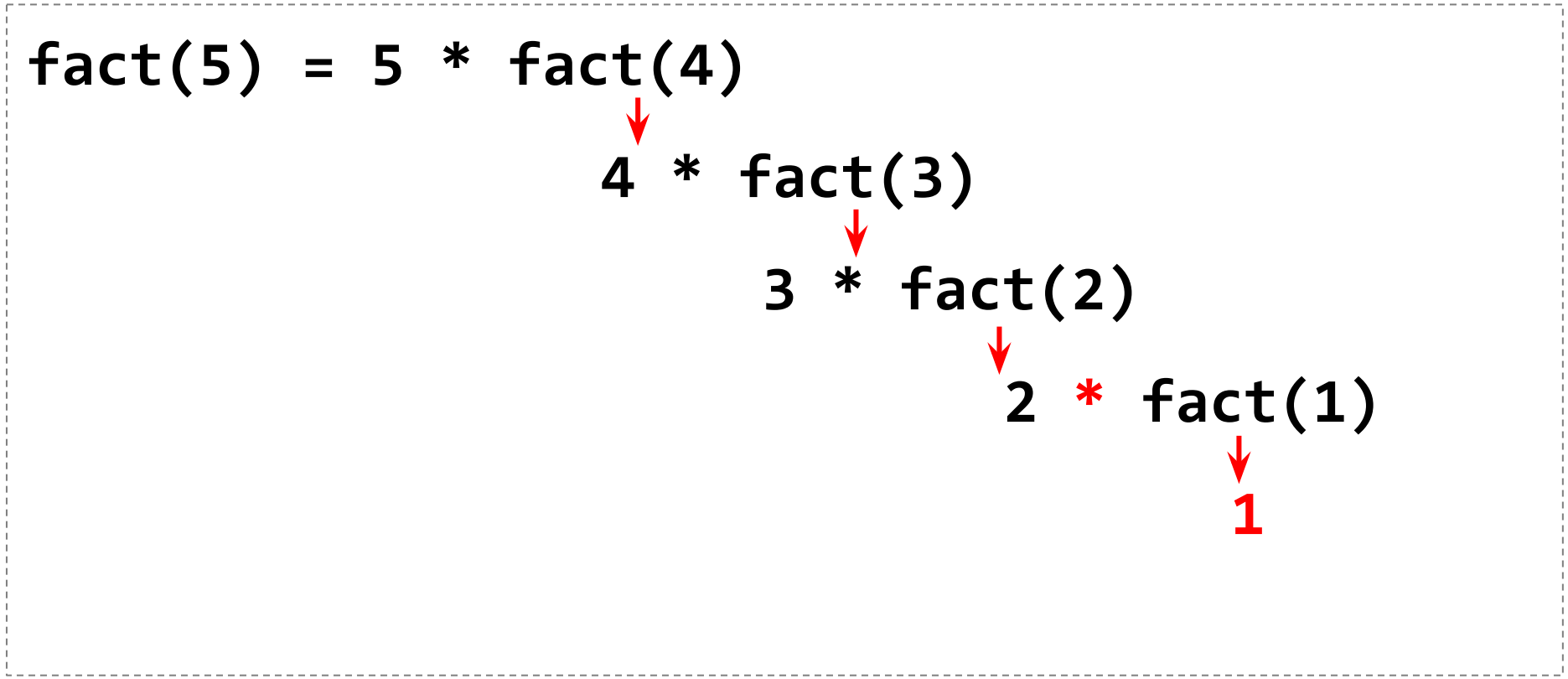
Recursión:

```
fact(5) = 5 * fact(4)
          4 * fact(3)
              3 * fact(2)
```

Recursión:

```
fact(5) = 5 * fact(4)
                  4 * fact(3)
                        3 * fact(2)
                              2 * fact(1)
```

Recursión:



Recursión:

```
fact(5) = 5 * fact(4)
                ↓
              4 * fact(3)
                    ↓
                  3 * fact(2)
                        ↓
                      2 * fact(1)
                            ↑
                            1
```

Recursión:

```
fact(5) = 5 * fact(4)
          4 * fact(3)
              3 * fact(2)
                  2 * 1
```

Recursión:

```
fact(5) = 5 * fact(4)
                ↓
          4 * fact(3)
                ↑
            3 * 2
```

Recursión:

```
fact(5) = 5 * fact(4)
```

```
          4 * 6
```

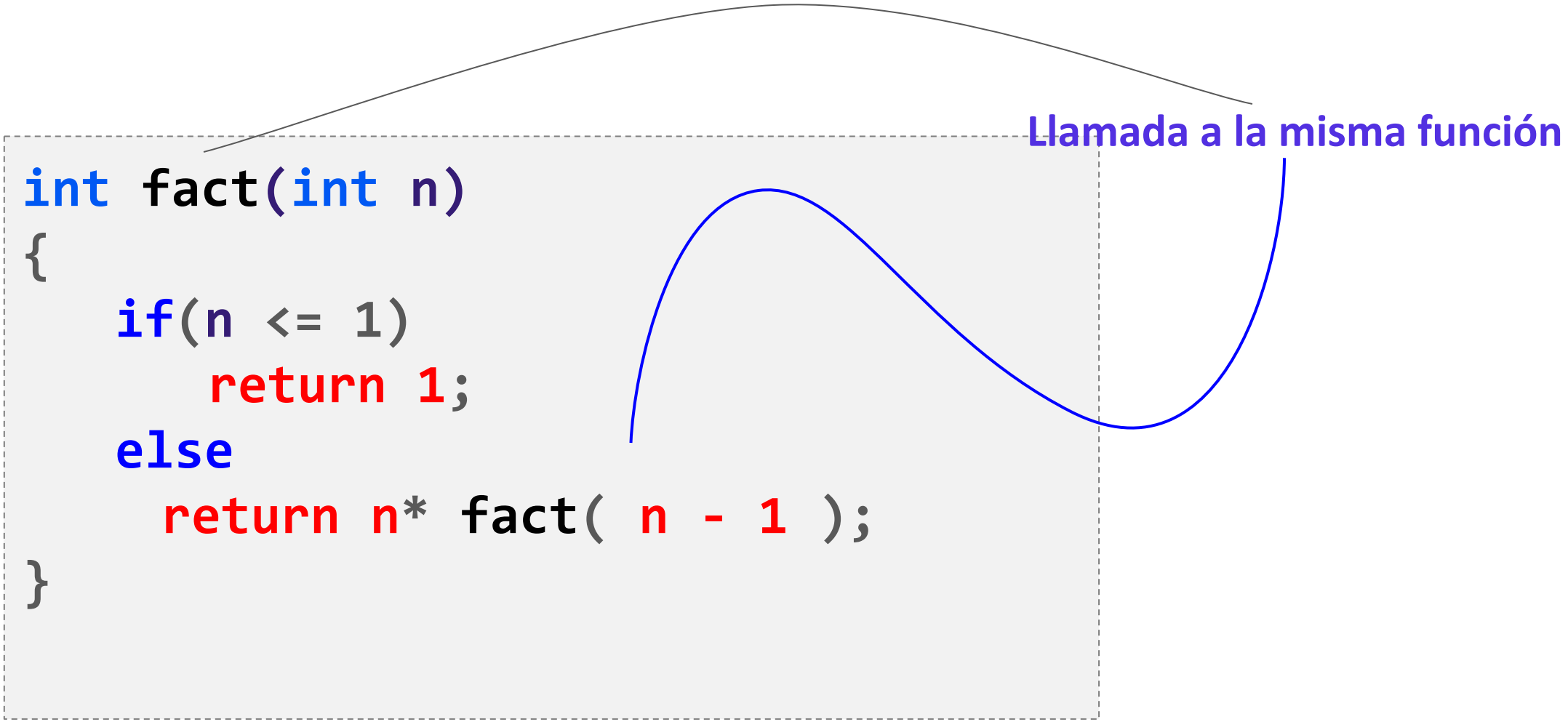
Recursión:

```
fact(5) = 5 * 24
```

Recursión:

```
fact(5) = 120
```

Recursión:


Llamada a la misma función
int fact(int n)
{
if(n <= 1)
return 1;
else
return n* fact(n - 1);
}

Recursión:

```
unsigned long fact(unsigned int n)
{
    if(n <= 1)
        return 1L;
    return n* fact( n - 1 );
}
```

Llamada a la misma función

Factorial - Iterativo

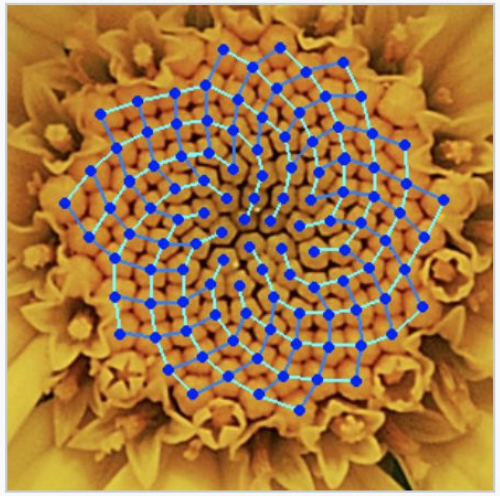
```
#include <iostream>
using namespace std;

unsigned  long factorial(unsigned int  n)
{unsigned long f;

 f=1;
 for(int i=2;i<=n; i++)
   f*=i;
 return f;
}

int main()
{
 unsigned int n;
 cout << "Numero: ";
 cin >> n;
 cout << "Factorial ("<< n <<") = ";
 cout << factorial(n);
 return 0;
}
```

Factorial - Recursivo

```
#include <iostream>
using namespace std;

unsigned long factorial(unsigned int  n)
{
 if (n<=1)
   return 1L;
 return (n * factorial(n-1));
}

int main()
{
 unsigned int n;
 cout << "Numero: ";
 cin >> n;
 cout << "Factorial ("<< n <<") = ";
 cout << factorial(n);
 return 0;
}
```

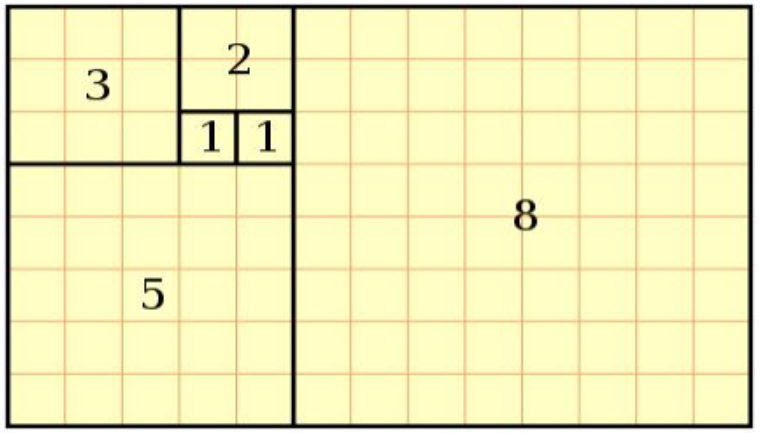
Serie de Fibonacci:

0 1 1 2 3 5 8 13 21 34 55 89 144 233 ...



Botón de Camomila amarilla mostrando la ordenación en espirales de módulos 21 (color azul) y 13 (color cian). Este tipo de arrollamientos utilizando números consecutivos de Fibonacci aparecen en una gran variedad de plantas.

Espiral de Fibonacci en la sección de la concha de un nautilus.



Al construir bloques cuya longitud de lado sean números de Fibonacci se obtiene un dibujo que se asemeja al rectángulo áureo (véase Número áureo).

Serie de Fibonacci:

Los números de Fibonacci quedan definidos por la ecuación:

$$f_n = f_{n-1} + f_{n-2}$$

partiendo de dos primeros valores predeterminados:

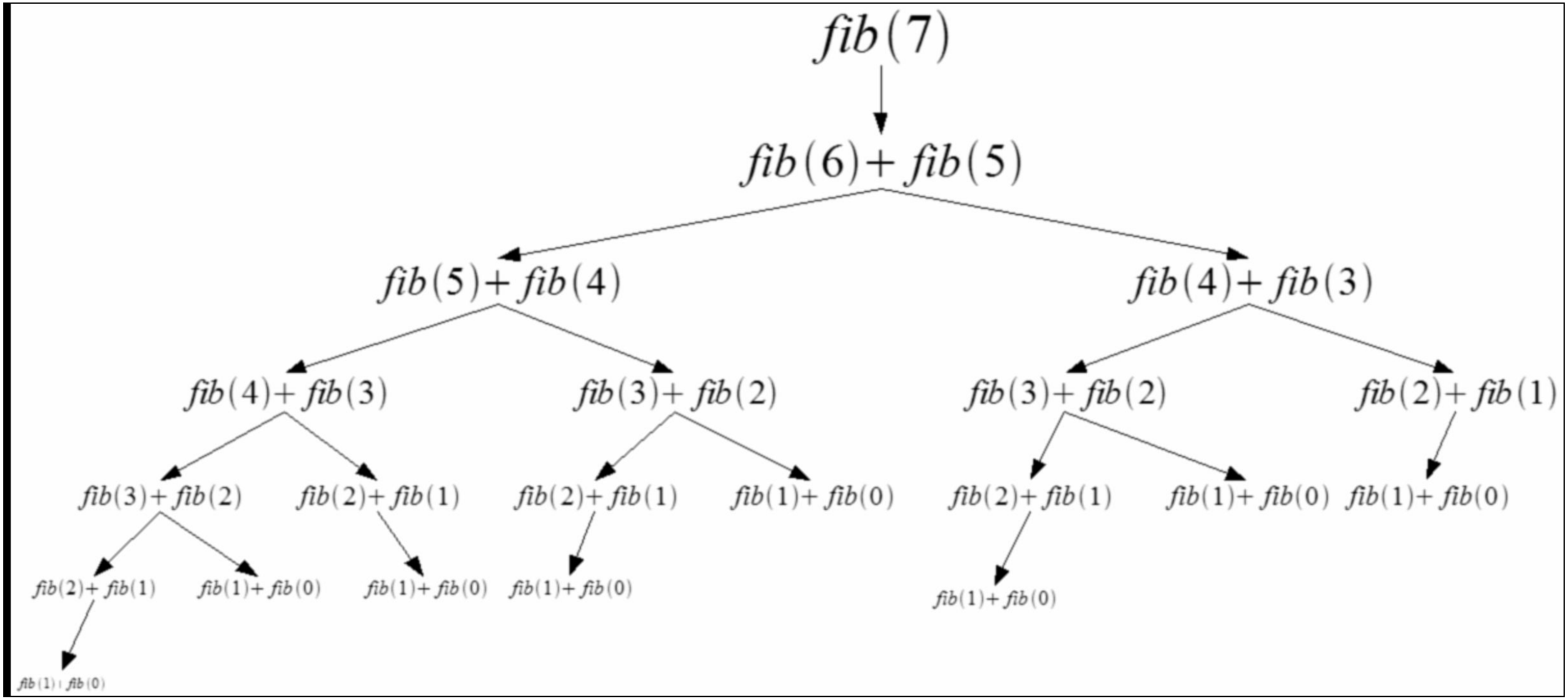
$$f_0 = 0$$
$$f_1 = 1$$

se obtienen los siguientes números:

- $f_2 = 1$
- $f_3 = 2$
- $f_4 = 3$
- $f_5 = 5$
- $f_6 = 8$
- $f_7 = 13$
- $f_8 = 21$

para $n = 2, 3, 4, 5, \ldots$

```
f(n) = f(n-1) + f(n-2)


Partiendo de:
f(0) = 0
f(1) = 1
```

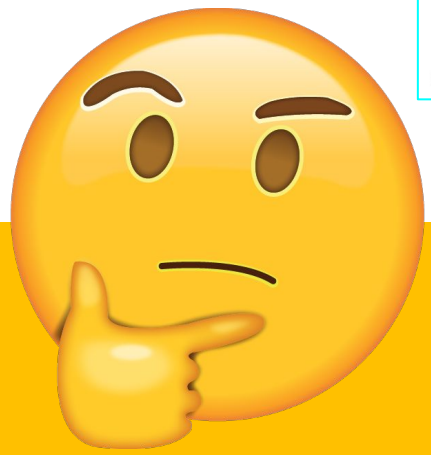

fib(7)
fib(6)+ fib(5)
fib(5)+ fib(4)
fib(4)+ fib(3)
fib(4)+ fib(3)
fib(3)+ fib(2)
fib(3)+ fib(2)
fib(2)+ fib(1)
fib(3)+ fib(2)
fib(2)+ fib(1)
fib(2)+ fib(1)
fib(1)+ fib(0)
fib(2)+ fib(1)
fib(1)+ fib(0)
fib(1)+ fib(0)
fib(2)+ fib(1)
fib(1)+ fib(0)
fib(1)+ fib(0)
fib(1)+ fib(0)
fib(1)+ fib(0)

Fibonacci - Iterativo

```
#include <iostream>
using namespace std;

long int fibonacci (long int  n)
{
 if(n<2)
   return n;
 long int a=0,b=1, c;
 for(int i=2;i<=n;i++)
 {
   c = a+b;
   a = b;
   b = c;
 }
 return c;
}

int main()
{ long int n;
 cout <<"Termino : ";
 cin >> n;
 cout << "Fibonacci ("<< n <<") = ";
 cout << fibonacci(n);
 return 0;
}
```

Fibonacci - Recursivo

```
#include <iostream>
using namespace std;

long int fibonacci (long int  n)
{
 if(n<2)
   return n;
 return( fibonacci(n-1) + fibonacci(n-2));
}

int main()
{ long int n;
 cout <<"Termino : ";
 cin >> n;
 cout << "Fibonacci ("<< n <<") =";
 cout << fibonacci(n);
 return 0;
}
```



1. **¿Qué es una función?**
2. **¿Cuándo es conveniente crear funciones?**
3. **¿Qué tipos de parámetros puede tener una función?**
4. **¿Cómo indicamos el valor que queremos que retorne la función?**
5. **¿Todas las funciones devuelven un valor?**
6. **¿Qué tipo de función es más simple de implementar: Recursivo o Iterativo?**
7. **¿Cuál es más óptimo en cuanto al consumo de los recursos (memoria, procesador, energía) del computador?**

Funciones Lambda

The following example uses a lambda function to increment all of the elements of a vector and then uses an overloaded `operator()` in a functor to compute their sum. Note that to compute the sum, it is recommended to use the dedicated algorithm `std::accumulate`.

Run this code

```
#include <vector>
#include <algorithm>
#include <iostream>

struct Sum
{
    Sum(): sum{0} { }
    void operator()(int n) { sum += n; }
    int sum;
};

int main()
{
    std::vector<int> nums{3, 4, 2, 8, 15, 267};

    auto print = [](const int& n) { std::cout << " " << n; };

    std::cout << "before:";
    std::for_each(nums.begin(), nums.end(), print);
    std::cout << '\n';

    std::for_each(nums.begin(), nums.end(), [](int &n){ n++; });

    // calls Sum::operator() for each number
    Sum s = std::for_each(nums.begin(), nums.end(), Sum());

    std::cout << "after: ";
    std::for_each(nums.begin(), nums.end(), print);
    std::cout << '\n';
    std::cout << "sum: " << s.sum << '\n';
}
```

Output:

```
before: 3 4 2 8 15 267
after:  4 5 3 9 16 268
sum: 305
```

CS1102 – PROGRAMACIÓN ORIENTADA A OBJETOS 1
CICLO 2019-1

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CIENCIA DE LA COMPUTACIÓN

Unidad 2: Funciones y recursividad

http://bit.ly/2HRBWgq

Profesores:

Ernesto Cuadros- Vargas, PhD. *ecuadros@utec.edu.pe*

María Hilda Bermejo, M. Sc. *mbermejo@utec.edu.pe*