Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

Auxiliar 3
recursividad

Recursividad

Las funciones recursivas son aquellas que en sus instrucciones se invocan a si mismas. Es decir:

```
static public Tipo funcion(Tipo x){
    ...
    ...
    funcion(x-1); //es sólo un ejemplo
    ...
    return ...
}
```

Una función conocidísima que ocupa recursividad es la funcion factorial que se define como sigue.

$$0!=1$$

$$n!=n*(n-1)!$$

Factorial

En lenguaje JAVA sería:

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

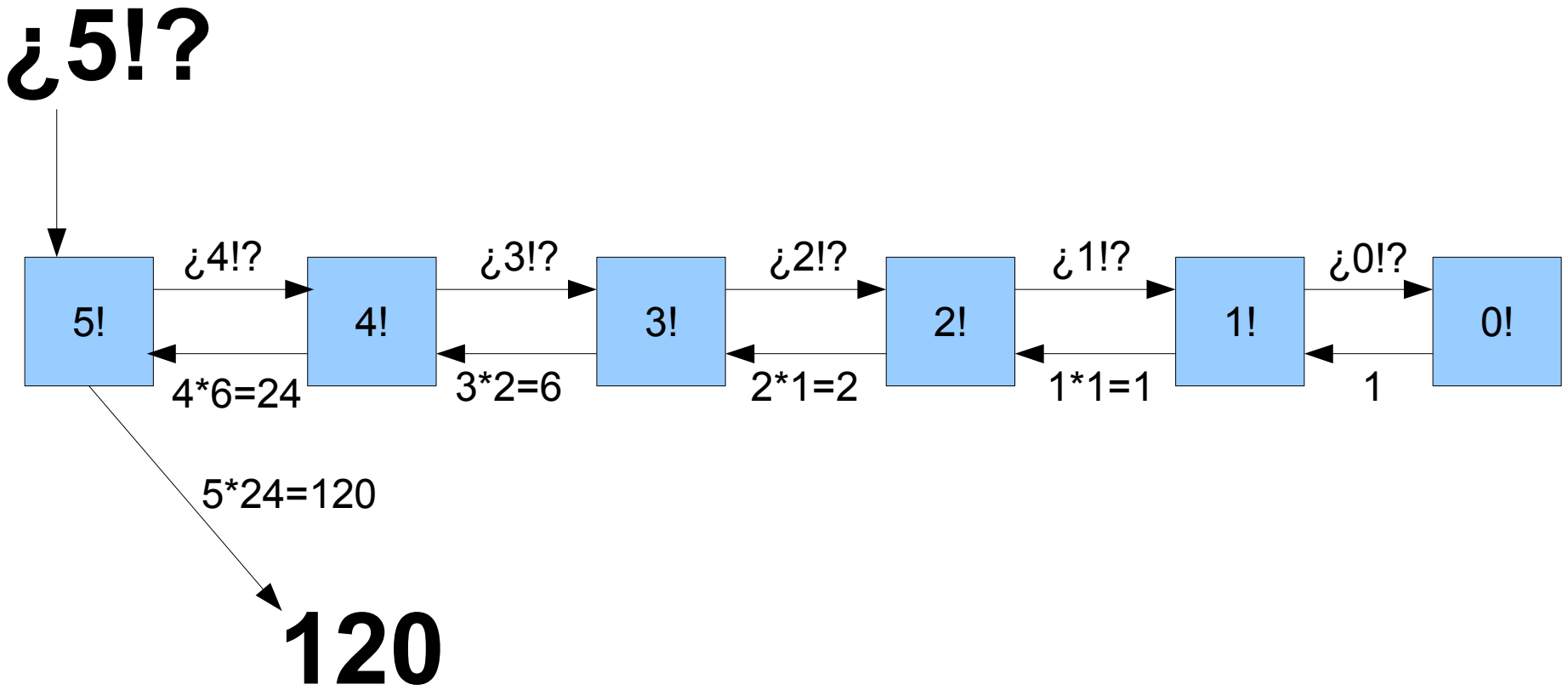
La primera linea de instrucciones es el caso base, es decir la condición para la salida a la recursión (si no existe sigue infinitamente).

La segunda linea es la invocación a si misma. Las llamadas a si mismo deben acercarso o converger al caso base (factorial de n-1).

```
factorial(5) > 5!=0 > 5*factorial(5-1)
factorial(4) > 4!=0 > 4*factorial(4-1)
...
,,,
factorial(0) > 0==0 > 1
```

Factorial

Así la función preguntará el resultado de la función evaluada en el numero menos uno hasta llegar a 0.



Contar Dígitos

Vamos a escribir ahora una función que cuente los dígitos de un entero positivo.
Primero, de forma iterativa:

```
static public int digitos(int z){
   int n=1,aux=x;
      while( aux >= 10 ){
         aux = aux/10;
         n =n +1;
      }
   return n;
}
```

Ahora de forma recursiva:

```
static public int digitos(int z){
   if( x < 10 )return 1;
   return 1 + digitos(x/10);
}
```

Bastante más corto y claro no?

Ejercicio

Escriba una función llamada potencia que calcule x elevado a y, con x un real cualquiera e y un entero cualquiera.

```
static public double potencia(double x, int y)
    ...
    ...
}
```

Solución

Solución 1: iterativa

```
static public double potencia(double x, int y){
   double aux=1;
   int i=1;
   while( i <= Math.abs(y) ){
       aux = aux * x;
      i = i + 1;
   }
   if( y >= 0 )return aux;
   return 1/aux;
}
```

Es claro, pero se puede hacer aún más corto y conciso, y por su puesto recursivo.

Solución

Solución 2: x^y=x*x^(y-1) con x^0=1

```
static public double potencia(double x, int y){
   if( y == 0 ) return 1;
   else if( y > 0 )return x * potencia(x,y-1);
   else return 1/potencia(x,-y);
}
```

Solución 3: x^y=x*x^(y-1) si y impar x^y=x^(y/2)*x^(y/2)

```
static public double potencia(double x, int y){
   if( y == 0 )return 1;
   else if( y < 0 )return 1/potencia(x,-y);
   else if( y%2 == 1 ) // y es impar?
      return x * potencia(x,y-1);
   else{
      double aux = potencia(x, y/2);
      return aux * aux;
   }
}
```

Solución

Solución 4:
x^y=x^(y/2)*x^(y/2) si y es par,
x^y=x*x^[(y-1)/2]*x^[(y-1)/2] si es impar.

```
static public double potencia(double x, int y){
   if( y == 0 )return 1;
   else if( y < 0 )return 1/potencia(x,-y);
   else{
      double aux = potencia(x, y/2);
      if( y%2 == 0 )  // si y es par ?
      return aux * aux;
      else
      return x * aux * aux;
   }
}
```

Es la solución más óptima y realiza log2(y) llamadas recursivas.

Solución

Solución 5:
No es la más óptima, pero fue la que en su mayoría pusieron los alumnos de la subsección 2.

```
static public double potencia(double x, int y){
   if( y == 0 )return 1;
   if( y < 0) return 1/x * potencia(x,y+1);
   return x*potencia(x,y-1);
}
```

Para exponente negativo, multiplica (1/x)*(1/x)*...

Ejercicios para la próxima clase

void binario(int x)
Ej: binario(13); escribe 1101
Indicación: para convertir un decimal a binario, se divide sucesivamente por 2 y se toman los restos (0 o 1)

double exp(double x)
Ejemplo: exp(1) entrega 2.71...con 5 decimales
Considere que exp(x) = $1 + x + x^2/2! + x^3/3! + \ldots$

FIN

Profesor: Andrés Muñoz

Auxiliares: Oscar Alvarez
Pedro Valencia