Recursividad

Objetivos

- Diseñar algoritmos recursivos en pseudocódigo.
- Implementar algoritmos recursivos en C++.
- Realizar la traza de las llamadas de algoritmos recursivos utilizando la implementación C++.

Actividades

Actividad 1

La obtención de la representación binaria de un número natural se puede obtener de forma recursiva. El procedimiento consiste en realizar la división entera del número entre 2, a continuación se divide el cociente obtenido entre 2 y así sucesivamente se van dividiendo entre 2 todos los cocientes hasta obtener un cociente igual a 0. Una vez llegado a este punto, la codificación en binario del número se obtiene cogiendo los restos de todas las divisiones en orden contrario a como se han obtenido.

Ejemplo: Para convertir el número 12 a binario se realizan las siguientes divisiones

- 12 / 2 → cociente 6 y resto 0.
- 6 / 2 → cociente 3 y resto 0.
- 3 / 2 → cociente 1 y resto 1.
- 1 / 2 → cociente 0 y resto 1.

Los restos obtenidos en las divisiones son 0, 0, 1 y 1. Si los cogemos en orden contrario a su obtención, el número 12 en binario es 1100.

1.1.- Realiza el análisis por casos de el algoritmo recursivo descrito anteriormente y escribe su pseudocódigo. Este algoritmo recibe como argumento el número natural a convertir (incluido el 0) e imprime su representación binaria. Nombra al algoritmo binario. Haz una traza para diversos números y comprueba que se obtiene el resultado correcto. ¿Qué tipo de recursividad tiene este algoritmo?

1.2.- Implementa en C++ una función llamada **binario** correspondiente al pseudocódigo del algoritmo del apartado anterior. Crea un programa que pida un número e imprima su valor en binario llamando a esta función. En el caso de que el número sea menor que 0 la función escribe un texto indicando el error. En la Figura 1 se muestra un ejemplo de ejecución del programa para el número 12.

```
Introduce numero natural (incluido el 0): 12

El numero en binario es:
1100
```

Figura 1: Ejemplo de ejecución de la función binario.

1.3.- Implementa una nueva función llamada **binario_traza** que haga lo mismo que la función binario y además imprima la traza de llamadas recursivas que se ejecutan. En cada llamada se indica:

- el número de llamada recursiva: 1.-, 2.-, 3.-, etc.
- el nombre de la función junto con el valor del parámetro de la función entre paréntesis.

Puedes añadir a la función los parámetros que consideres necesarios para producir el formato de salida especificado pero no los muestres en la traza.

Incluye la función en el mismo programa que la función anterior y realiza una llamada a binario_traza después de la llamada a la función binario. Verifica que se obtiene la misma representación binaria y que la traza es correcta.

En la Figura 2 se muestra un ejemplo de ejecución del programa para el número 12.

```
Introduce numero natural (incluido el 0): 12

El numero en binario es:
1100

La traza es:
1.- binario_traza(12)
2.- binario_traza(6)
3.- binario_traza(3)
4.- binario_traza(1)
1100

Presione una tecla para continuar . . .
```

Figura 2: Ejemplo de ejecución de las funciones binario y binario_traza.

Actividad 2

Las combinaciones sin repetición de n elementos cogidos de k en k, $C_{n,k}$, son el número total de grupos distintos de k elementos que se pueden formar a partir de un total de n elementos (k≤n).

Ejemplo: Si queremos distribuir n=4 personas (Antonio, José, María y Carmen) en grupos de k=3 personas, tenemos un total de $C_{4,3} = 4$ formas distintas de distribuirlas. En concreto, los grupos que se pueden formar son:

- Grupo 1: Antonio, José, María
- Grupo 2: Antonio, José, Carmen
- Grupo 3: Antonio, María, Carmen
- Grupo 4: José, María, Carmen

El valor $C_{n,k}$ se puede calcular recursivamente de la siguiente forma

$$C_{n,k} = \begin{cases} 1 & \text{si } k = 0 \text{ o } k = n \\ C_{n-1,k-1} + C_{n-1,k} & \text{en caso contrario} \end{cases}$$

donde $n, k \in N \cup \{0\}$ y $n \geq k$.

2.1.- Escribe el pseudocódigo de un algoritmo recursivo llamado **combinaciones** que obtenga el valor $C_{n,k}$, según la especificación anterior. ¿Qué tipo de recursividad es?

2.2.- Implementa en C++ una función llamada **combinaciones** correspondiente al pseudocódigo del algoritmo del apartado anterior y crea un programa que pida por teclado los valores de n y k, realice una llamada a la función e imprima el número total de combinaciones que devuelve la función. Comprueba su funcionamiento con diferentes valores de n y k.

2.3.- Crea una nueva función llamada **combinaciones_traza** que además de calcular el número de combinaciones, imprima la traza de llamadas recursivas. En cada llamada se indica:

- el número de llamada recursiva: 1.-, 2.-, 3.-, etc.
- el nombre de la función junto con el valor de los parámetros n y k,
- las llamadas recursivas que se realicen desde una función se imprimirán un nivel más adentro.

Puedes añadir a la función los parámetros que consideres necesarios para producir el formato de salida especificado pero no los muestres en la traza.

Incluye esta función en el mismo programa que la función anterior y realiza una llamada a combinaciones_traza después de la llamada a la función combinaciones. Verifica que se obtiene el mismo resultado y que la traza es correcta.

En la Figura 3 se muestra un ejemplo de ejecución del programa para n=4 y k=3. Como se puede apreciar, la función combinaciones_traza(4,3) (llamada 1) realiza dos llamadas: combinaciones_traza(3,2) y combinaciones_traza(3,3) (llamadas 2 y 7, respectivamente). Estas dos llamadas se imprimen un nivel más adentro de la llamada 1 y entre ellas al mismo nivel. La función combinaciones_traza(3,2) (llamada 2) realiza dos llamadas: combinaciones_traza(2,1) y combinaciones_traza(2,2) (llamadas 3 y 6, respectivamente). Estas dos llamadas se imprimen un nivel más adentro de la llamada 2 y entre ellas al mismo nivel.

Y así sucesivamente.

```
Introduce total de elementos (n): 4
Introduce tamaño del grupo (k):  3

El numero de grupos es: 4


La traza es:
1.- combinaciones_traza(4,3)
    2.- combinaciones_traza(3,2)
        3.- combinaciones_traza(2,1)
            4.- combinaciones_traza(1,0)
            5.- combinaciones_traza(1,1)
        6.- combinaciones_traza(2,2)
    7.- combinaciones_traza(3,3)

El numero de grupos es: 4

Presione una tecla para continuar . . .
```

Figura 3: Ejemplo de ejecución de las funciones combinaciones y combinaciones_traza.