Recursividad y su transformación a iterativo

Objetivos

- Implementar algoritmos recursivos.
- Realizar la transformación de algoritmos recursivos a iterativos utilizando el esquema general de transformación de algoritmos.
- Implementar algoritmos en C++.
- Gestionar pilas en un programa utilizando la librería stack.
- Trabajar en grupo.

Actividades (Equipos de 2 o 3 alumnos)

Actividad 1:

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

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

Implementar en C++ una función llamada **combinaciones_traza** correspondiente al pseudocódigo anterior que imprima por pantalla una traza similar a la de la segunda columna de la tabla anterior. Crear 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.

Actividad 2:

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 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.

Implementa en C++ una función llamada **binario** correspondiente a un algoritmo recursivo que calcule el valor según la especificación que se indica. El resultado se deberá almacenar en una pila (consultar el Anexo). Crea un programa que pida un número e imprima su valor en binario llamando a esta función. En el caso de que haya algún error se imprimirá el valor -1.

Actividad 3:

Implementa la versión iterativa del siguiente algoritmo. La transformación se debe realizar aplicando el esquema general de transformación apropiado. Realizar un programa que pida por teclado dos números y muestre el resultado de aplicar las dos versiones del algoritmo implementado. En el caso de que `n` sea un número negativo el programa imprimirá el texto `Error`.

```
función p43(x:real, n:natural U {0}):real
   r:real
   si n=0
      r ← 1
   si no
      si impar(n)
         r ← x*p43(x,n-1)
      si no
         r ← p43(x,n/2)
         r ← r*r
      fsi
   fsi
   devolver r
ffunción
```

Actividad 4:

Implementar la versión iterativa del siguiente algoritmo. La transformación se debe realizar aplicando el esquema general de transformación apropiado. Realizar un programa que pida por teclado dos números y muestre el resultado de aplicar las dos versiones del algoritmo implementado.

```
función p44(x:entero, y:entero):entero
   si x ≤ 4
      devolver x+y
   si no
      x ← x - 4
      y ← y / 3
      devolver p44(x, y) + x*y
   fsi
ffunción
```

Modo de entrega

La práctica se realizará en equipos de **dos o tres alumnos** y se entregarán los siguientes ficheros con los nombres que se indican.

Archivo comprimido:	practica4.zip
Contenido del archivo:	p41.cpp, p42.cpp, p43.cpp, p44.cpp Cada fichero contiene el código fuente de las actividades correspondientes y los nombres de los miembros del equipo.

Todos los componentes del equipo entregarán el archivo practica4.zip en la tarea llamada “Práctica 4: Recursividad y su transformación a iterativo”, dentro del campus virtual.

Fecha fin de entrega: Domingo, 19 de marzo de 2023 a las 23:59.

Evaluación

La calificación total de la actividad es 0,2 puntos (0,05 A1, 0,05 A2, 0,05 A3 y 0,05 A4).

A continuación, se indica el sistema de evaluación:

- **Opción A: La práctica se entrega durante la sesión de prácticas.**

Cada alumno/-a del equipo entregará la práctica (practica4.zip) en la tarea de la web de la asignatura. Antes de subir la tarea a la web se debe recibir el visto bueno del profesorado y todos los miembros del equipo deben estar presentes y explicar cualquier aspecto que se solicite. Se evaluará cogiendo al azar la práctica de uno de los miembros del equipo, de forma que dicha práctica será la que se corrija.

Si hay algún miembro del equipo que no entrega la práctica en la tarea, no responde correctamente a las preguntas realizadas por el profesorado o no está presente…, no se le calificará según esta opción y puede optar a la calificación según la opción B.

- **Opción B: La práctica se entrega en horario posterior a la sesión de prácticas.**

A esta opción optarán los equipos y miembros de equipos que no cumplen los requisitos para ser evaluados según la opción A. Cada miembro del equipo debe entregar tanto la práctica en la tarea de la web de la asignatura como el programa que funcione correctamente. Se calificará cogiendo al azar la práctica de uno de los miembros del equipo, de forma que dicha práctica será la que se corrija. Los programas deben seguir las especificaciones que se dan.

La calificación máxima que se puede obtener bajo esta Opción B es un 0,1 (0,025 A1, 0,025 A2, 0,025 A3 y 0,025 A4).

- **Opción C: 0 puntos**

- El alumno/a:
  - No entrega la práctica en la tarea de la web de la asignatura.
  - No asiste a la sesión de prácticas cumpliendo con las condiciones establecidas.
- El programa no funciona correctamente y no realiza lo que se pide.
- Se detecta copia con otras prácticas. La nota será un 0 en esta práctica para todas las prácticas implicadas, aun cuando la práctica haya sido valorada previamente de forma positiva por parte del profesorado.

Anexo: La clase pila en C++

En C++ existe una librería llamada stack que tiene implementada la clase pila y funciones para trabajar con ellas. Las funciones que incorpora esta librería son las siguientes:

Función	Descripción
push()	Introduce un elemento en la cima de la pila.
pop()	Elimina el elemento de la cima de la pila.
top()	Devuelve el elemento que se encuentra en la cima de la pila.
empty()	Verdadero si la pila está vacía. En caso contrario devuelve falso.
size()	Número de elementos de la pila.

Para declarar una variable del tipo pila se utiliza la instrucción

```
stack <tipodatos> nombrepila;
```

siendo tipodatos el tipo de datos de los datos que se guardan en la pila: int, float…

En el siguiente ejemplo se muestra cómo utilizar una variable del tipo stack y la llamada a las funciones de la librería. Este programa declara una variable llamada pila del tipo stack y almacena en ella 5 números enteros que se introducen por teclado. A continuación, muestra el tamaño de la pila y finalmente imprime todos los elementos que se han apilado hasta dejar la pila vacía.

```
#include <iostream>
#include <stack>
#include <stdlib.h>

using namespace std;

int main()
{
  stack <int> pila;
  int dato,i;


  cout << "Apilando datos" << endl;
  cout << "--------------" << endl;

  for (i=1; i<=5; i++){
    cout << "Introduce dato: ";
    cin >> dato;
    pila.push(dato);
  }

  cout << endl;
  cout << "Num. elementos de la pila: ";
  cout << pila.size() << endl << endl;

  cout << "Desapilando datos" << endl;
  cout << "-----------------" << endl;
  while (! pila.empty() )
  {
    dato = pila.top();
    cout << dato << endl;
    pila.pop();
  }

  cout << endl;
  system("pause");
  return 0;
}
```