UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

# LABORATORIO ELECTRÓNICA DIGITAL III
# DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
# PRÁCTICA 1 – PERMUTACIONES GRÁCILES

## OBJETIVOS

- Recordar los conceptos fundamentales de la programación en lenguaje C
- Desarrollar habilidades en la estructuración de programas en lenguaje C
- Programar de forma eficiente algoritmos de complejidad matemática considerable

## DESCRIPCIÓN

Una **permutación** es un arreglo ordenado de todos los elementos de un conjunto, donde importa el orden en que aparecen estos elementos. Por ejemplo, si tienes el conjunto $\{1,2,3\}$, algunas permutaciones posibles son $(1,2,3)$, $(1,3,2)$, $(2,1,3)$, etc. En general, un conjunto de n elementos tiene exactamente n! permutaciones.

Las **permutaciones gráciles** (Graceful Permutations) son un tipo especial de permutaciones con ciertas propiedades numéricas particulares. Se dice que una permutación de números enteros positivos es **grácil** si al restar cada par de elementos consecutivos, los resultados son números enteros positivos diferentes entre sí, formando exactamente el conjunto de números del 1 hasta $n-1$.

Formalmente, una permutación $a_1, a_2, \dots, a_n$ es grácil si:

- Las diferencias absolutas $|a_{i+1} - a_i|$ para $i = 1,2, \dots, n-1$ son todas distintas.
- Estas diferencias cubren exactamente el conjunto de números $\{1, 2, \dots, n-1\}$.

**Ejemplo:** Para $n = 4$, la permutación $(1,4,2,3)$ es grácil, porque:

- $|4-1| = 3$, $|2-4| = 2$, $|3-2| = 1$.
- Las diferencias obtenidas son $\{3,2,1\}$, exactamente los números del 1 al 3.

**Importancia de las Permutaciones Gráciles:**

Las permutaciones gráciles están relacionadas con la teoría de grafos, especialmente con el etiquetado grácil de grafos (**Graceful Labeling**), un problema clásico y abierto en matemáticas discretas. El etiquetado grácil es fundamental en el estudio de grafos para aplicaciones como la comunicación, distribución de frecuencias en telecomunicaciones, y algoritmos eficientes de asignación. Investigar sobre permutaciones gráciles permite avanzar en áreas como criptografía, diseño de redes, algoritmos de optimización y otras aplicaciones prácticas de la teoría combinatoria y la matemática discreta.

Calcular el número de permutaciones gráciles es un problema intrínsecamente difícil debido a su crecimiento factorial, la ausencia de algoritmos eficientes conocidos y las estrictas restricciones combinatorias involucradas. El desafío computacional es significativo y representa un área activa de investigación en matemáticas discretas, teoría combinatoria y ciencia computacional.

En esta práctica debes desarrollar un programa en C que recibe como parámetros en consola un número entero N (1<N<50), a partir del cual el programa debe computar el número de permutaciones gráciles en un conjunto de números enteros consecutivos entre 1 y N. La salida del programa será el número de permutaciones gráciles encontradas y el tiempo en segundos que tardo en hacerlo. Adicional a N el programa también recibe como entrada el tiempo máximo en minutos M que el programa dispone para encontrar la respuesta. En el caso en que el tiempo M se agote antes de completar la tarea, el programa terminará indicando que no pudo completar el trabajo y el número de permutaciones gráciles que alcanzo a encontrar en el tiempo proporcionado.

La interacción entre el usuario y el programa es a través de consola. Los parámetros N y M se reciben en la línea de comandos al momento de llamar el ejecutable y no por medio de preguntas ya que esto afectaría la medición del tiempo de ejecución.

## PROCEDIMIENTO

1. Elegir compañero de laboratorio y reportarlo tanto al profesor de la teoría como al profesor del laboratorio vía correo electrónico.
2. Analizar el problema que se debe resolver y plantear una estrategia para dar solución al mismo. Dibuje un diagrama de flujo con la estrategia planteada.
3. Defina claramente las estructuras de datos que necesita en su estrategia de solución.
4. Estructure su programa en C por medio de funciones.
5. Codifique la estrategia de solución en lenguaje C.
6. Documente su código en Doxygen.
7. Evalué la eficiencia de su solución midiendo el tiempo de ejecución para diferentes valores de N. Comience con el valor N=2 e incremente en 1 sucesivamente hasta que su programa requiera más de 60 minutos para entregar la respuesta. Grafique el tiempo de ejecución en función de N. Verifique también que el número de permutaciones grácile sea el correcto. En el siguiente enlace puede encontrar la respuesta correcta para Ns entre 0 y 41.

   https://oeis.org/A006967/b006967.txt

8. Escriba un reporte en formato IEEE describiendo el trabajo realizado y los resultados alcanzados.
9. Suba el reporte y el código C a la plataforma Moodle en un solo archivo comprimido antes de la media noche del lunes 30 de marzo. Marque el archivo comprimido con el 1er apellido de cada miembro del grupo.

## EVALUACIÓN

- 30% Eficiencia
  - Mas 15 minutos para N=12 15%
  - Mas de 30 minutos para N=13 20%
  - Mas de 60 minutos para N=14 25%
  - Menos de 60 minuto para N=14 30%
- 30% Sustentación
- 20% Estructura del código y documentación
- 20% Reporte