



## Programación Avanzada – Laboratorio 1

### 1. Objetivos

Objetivo principal: Implementar algoritmos básicos de aritmética de polinomios: sumas, restas y multiplicación de dos polinomios.

Objetivos secundarios:

- Implementar algoritmos clásicos de sumas, restas y multiplicación de polinomios de forma estable.
- Analizar y aplicar un algoritmo del tipo dividir-y-conquistar, específicamente el algoritmo de Karatsuba para la multiplicación de polinomios.
- Obtener una implementación eficaz de la multiplicación de polinomios aplicable para polinomios de cualquier grados, optimizada para la multiplicación de dos polinomios del mismo grado.

El laboratorio se trabajará en grupos de 2 o 3 alumnos, entregando un resultado (informe y programas) por grupo.

### 2. Programación

La programación debe ser en C (no en C++), utilizando a lo más las librerías estándares siguientes:

- `stdio.h`
- `stdlib.h`
- `math.h`
- `time.h`

Además, para evitar cualquier problemas de compatibilidad, se recomienda utilizar solamente la función `clock()` para la medición de tiempo.

Se trabajará con polinomios con coeficientes enteros de tipo “long” (64 bits). Para evitar problemas de tamaño de los coeficientes (en el resultado de la multiplicación), se supondrá que los polinomios de entrada de la multiplicación tienen coeficientes con valor absoluto menor o igual a  $2^{20}$ , y que el grado de los polinomios es de a lo más  $2^{22}$ .



Observación: teóricamente, las funciones deberían poder tomar entradas más grandes, pero para eso se necesitaría utilizar otro tipo de datos para los coeficientes (por ejemplo: float/double).

Crear funciones para:

1. Creación de polinomios;
2. Generación de polinomios aleatorios (de grado dado);
3. Recibir un polinomio entrado manualmente [opcional];
4. Escribir/imprimir un polinomio;
5. Crear una copia de un polinomio dado;
6. Sumar dos polinomios;
7. Restar Polinomios (calcular la diferencia entre dos polinomios);
8. Multiplicar polinomios con la formula clásica, guardando la tabla completa de productos de coeficientes;
9. Multiplicar polinomios con la formula clásica, reduciendo la utilización de memoria via una estrategia reducir-y-conquistar en una de las entradas;
10. Seleccionar/copiar parte de un polinomio (desde un grado mínimo hasta un grado máximo);
11. Multiplicar un polinomio por una potencia de  $x$ , o sumar dos polinomios multiplcando una de las entradas por una potencia de  $x$  [se puede programar solamente una de las dos funciones o programar ambas];
12. Multiplicar polinomios con 1 paso de inducción para el algoritmo dividir-y-conquistar clásico;
13. Multiplicar polinomios con 1 paso de multiplicación de Karatsuba (dividir-y-conquistar);
14. Multiplicación de polinomios de cualquier grado, optimizada para la multiplicación de dos polinomios del mismo grado.

Además, se debe utilizar una de las funciones de multiplicación clásica (sin inducción) y la función de multiplicación con un paso de multiplicación de Karatsuba para determinar el grado mínimo a partir de cual es conveniente utilizar la estrategia dividir-y-conquistar de Karatsuba, lo que se utilizará en la función de multiplicación optimizada.

Generación aleatoria de polinomios: para la versión final, los coeficientes deberían ser eligidos con una distribución cerca de uniforme (por ejemplo basada en la función `rand()`) con valores entre  $-2^{20}$  y  $2^{20}$ . No olvidarse de iniciar el generador aleatorio en la versión final.



### 3. Se solicita

1. Programar los diferentes algoritmos en C, asegurando que hacen un uso apropiado de memoria (por ejemplo que se liberan correctamente todos los espacios de memoria utilizados al terminar las funciones) y liberando la memoria antes de cerrar las funciones.
2. El programa final debe permitir verificar que todos los algoritmos de multiplicación devuelven el mismo resultado (i.e. tienen diferencias iguales a 0) y evaluar los tiempos de trabajo para el algoritmo de multiplicación final. Pueden ser en un programa global o dos programas distintos (a elección del grupo).
3. Determinar los límites práctico de las diferentes funciones de multiplicación, en particular los grados máximos de entrada que las funciones pueden manipular, y si posible justificar la razón, por ejemplo si hay límites debidas a la generación de las estructuras de datos utilizadas (arreglos, etc), o por el tiempo de trabajo.
4. Si no está indicado en la programación (con el indicador *const*), cuales parámetros de las funciones no pueden ser modificados para las funciones. Además, documentar cuando algunos parámetros pueden ser repetidos en el mismo llamado de la función.
5. Analizar experimentalmente los tiempos de cálculo obtenidos, comparando:
  - La multiplicación directa (clásica) con el algoritmo utilizando Karatsuba (para grados pequeño y medianos, típicamente hasta grado  $2^{12}$  o  $2^{13}$ ).
  - El crecimiento a grande escala del algoritmo utilizando Karatsuba con su crecimiento teórico (idealmente hasta grado  $2^{22}$ ).

Observaciones: Si es conveniente, utilizar escalas logarítmicas para las gráficas. El tiempo puede ser como proporción de una operación específica (valor base) en vez de segundos. Para tiempos menores a 1 segundo, se recomienda utilizar el promedio de tiempo sobre varios cálculos similares.

6. El informe debería detallar:
  - La estructura de datos utilizada para los polinomios (con las razones generales de porque se eligió).
  - El formato utilizado en los polinomios (puede tener coeficiente principal 0 o no, etc).
  - Las estrategias utilizadas en la funciones 9, 12, 13 y 14 (multiplicación de polinomios sin producir una tabla de todos los productos de coeficientes).
  - Como se hicieron las mediciones de tiempo (como se seleccionaron de los grados medidos, sobre cuantos cálculos se evaluó, en que computador/procesador, etc).
  - Conclusiones obtenida de los análisis de tiempos.

## 4. Evaluación

La nota del laboratorio se calculará según la ponderación siguiente:

- Algoritmos [25 %]:  
El informe detalla las elecciones que se utilizaron durante la implementación y la determinación del grado mínimo para aplicar la multiplicación de Karatsuba.
- Análisis [15 %]:  
El análisis de los tiempos experimentales es correcto y completo, y justifica las conclusiones obtenidas.
- Informe [10 %]:  
El informe está escrito en lenguaje apropiado, sin faltas de ortografía o gramática..
- Implementación [30 %]  
El programa está escrito de forma que puede ser leído y/o re-utilizado fácilmente por otros programadores: la redacción es limpia (con espacios y divisiones claras) y bien documentada, las sub-funciones y las variables tienen nombres naturales (que indican a que sirven) o van acompañadas de comentarios aclarando a que sirven.  
Todas las versiones (algoritmos) de la multiplicación dan el resultado correcto.
- Eficiencia [20 %]  
Los tiempos de calculo de la multiplicación iterativa completa comparados con una implementación “básica” que satisface lo solicitado. Rangos de notas:
  - 1: no incluye una multiplicación iterativa completa.
  - 1.1–2.4: más de 20 veces más lenta
  - 2.5–3.9: 5 a 20 veces más lenta
  - 4–5.4: 2 a 5 veces más lenta
  - 5.5–6.9: a lo más 2 veces más lenta
  - 7: tiempos parecidos o mejores