Universidad
Carlos III de Madrid

PROGRAMACIÓN DE ALTAS PRESTACIONES

LAB 2 - TIPOS GENÉRICOS

Asier Sampietro y Andoni Alcelay

MÁSTER UNIVERSITARIO EN TECNOLOGÍAS DEL SECTOR FINANCIERO: FINTECH

Introducción

En las siguientes páginas se hará un resumen de lo conseguido en en laboratorio 2. En este ejercicio, partiendo de la clase matriz del laboratorio anterior, se ha creado la clase genérica Matriz, y se ha acompañado con otra clase genérica, Racional. Esta clase se utilizará para evitar el uso de la coma flotante, y ver así si tiene impacto a la hora de ejecutar los cálculos de la clase Matriz.Se analizará la operació A*B+C siendo cada una de ellas una matriz independiente y la suma de todos los valores de la diagonal principal del resultado. Esta operación se ha probado con matrices NxN donde N coge los valores desde 10 hasta 1000.

Objetivo

El objetivo del laboratorio 1 es practicar las siguientes competencias:

- Programación genérica

Descripción del código fuente

En lo siguientes puntos se analizará brevemente la utilidad e implementación de cada uno de los tres ficheros del código fuente.

main.cpp

En este fichero se sitúa la función main o principal del programa. Este main recibe varios parámetros. El primero debe ser un número natural, que determina la dimensión que tendrá la matriz, siendo 10^2 por defecto. Añadiendo un “-p” en segundo lugar y mostrarán en pantalla todas las matrices, teniendo esto un gran impacto negativo. Lo último puede omitirse para una ejecución más rápida. Estos parámetros han de meterse en orden, y la entrada no está comprobada.

matriz.h

En esta cabecera se define la clase Matriz. Al contrario que la anterior práctica, se incluyen en este fichero todas las variables y métodos que se van a ejecutar en modo genérico, ya que dicho añadido así lo requiere.
Cabe destacar que la clase Matriz se ha definido dos veces, una vez de manera genérica para cualquier tipo de elemento, y después con una especificación de tipo Racional para dicha clase. Esto se debe a las diferencias a la hora de inicializarse y coger valores de la clase Racional, que no funciona de forma adecuada con los métodos que utilizan en la clase Matriz.
Esta clase dispone de tres miembros dato: dos enteros que guardan el número de filas y columnas, y un puntero a un array genérico con la dimensión de filas por columnas. En la clase también se definen constructores, destructores y demás utilidades necesarias para el planteamiento del problema, aparte de sobrecargas para facilitar el acceso o impresión. Por último, también incluye varias sobrecargas de operadores fuera de la clase.

racional.h

La clase Racional es la representación de un número con coma flotante. Se ha definido para usarla junto a la clase Matriz y determinar si merece la pena sustituir el cómputo con el ALU específico para la coma flotante por accesos a memoria.
En esta clase se encuentran dos datamiembros, el numerador y el denominador, que cumpliran las funciones que indica el nombre. A parte de esto, tenemos constructores, un destructor y operadores de copia y movimiento que van acompañados de un cálculo del racional más simple que se pueda usar.
Fuera de la clase, también se han definido una serie de operadores que ayudarán a ejecutarlo con la clase Matriz.

Resultados

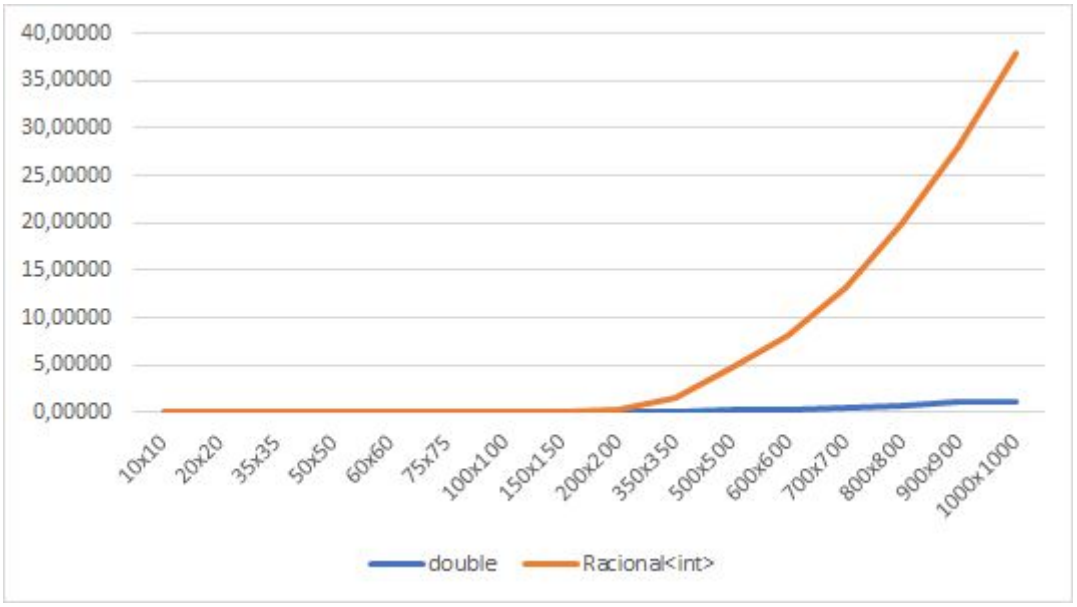
Para saber si es de utilidad la clase Racional frente a las alternativas que ofrece el lenguaje, se comparará el cálculo antes mencionado con dos tipos diferentes:

- Un “double”, que será la representación del racional de la forma que nos permite el lenguaje.
- Un “Racional<int>”, que será la representación de la implementación, usando dos valores del tipo primario “int”.

Para determinar su utilidad, se ha realizado la medición de los tiempos de cálculo de matrices de varias dimensiones, desde 10^2 hasta 1000^2. Los resultados han sido los siguientes:

Tamaño	**double**	**Racional<int>**
10x10	0,00161	0,00166
20x20	0,00151	0,00150
35x35	0,00231	0,00350
50x50	0,00260	0,00770
60x60	0,00320	0,01100
75x75	0,00465	0,02230
100x100	0,00710	0,04190
150x150	0,00923	0,12580
200x200	0,01770	0,29980
350x350	0,07614	1,61360
500x500	0,20230	4,66920
600x600	0,34250	8,17500
700x700	0,51327	13,19960
800x800	0,76320	19,84470
900x900	1,05809	27,91960
1000x1000	1,18210	37,90170

Si hacemos una representación gráfica de esta tabla, se observa que la clase Racional aumenta sus tiempos exponencialmente mientras que lo mismo con doubles lo hace en mucho menos tiempo.



Conclusiones

Pese a ser una mayor carga para la unidad lógica aritmética del procesador, el tipo primario double esta mejor optimizado para esta serie de operaciones. La clase Racional, ya sea por los requisitos que ha de tener y los múltiples accesos a memoria que realiza para dichos cálculos, ha demostrado ser visiblemente más lenta que su adversario.
Se observa un claro cuello de botella determinado por el simple hecho de tener que acceder a memoria 2 veces para representar a un racional, y triplicando estos para multiplicar o sumarlos.