Proyecto de programación orientada al rendimiento

J. Daniel Garcia (coordinador)
Arquitectura de Computadores
Departamento de Informática
Universidad Carlos III de Madrid

2024

1. Objetivo

Este proyecto tiene como objetivo fundamental hacer el que los estudiantes se familiaricen con la **optimización de programas secuenciales**.

En concreto, la práctica se centrará en el desarrollo de software secuencial en el lenguaje de programación C++ (incluyendo las mejoras de C++20).

2. Visión General

En este proyecto se construirá una aplicación de procesamiento de imágenes que realizará diversas transformaciones.

El formato de las imágenes será el formato PPM. Los archivos en dicho formato tendrán la extensión **ppm** y representan una imagen en color en codificación RGB.

La herramienta desarrollada permitirá realizar las siguientes operaciones:

- Obtención de metadatos de una imagen.
- Escalado del número de niveles de intensidad.
- Escalado del tamaño de la imagen mediante interpolación bilineal.
- Eliminación de los colores menos frecuentes de la imagen.
- Compresión basada en mapa de colores.

2.1. El formato PPM

El formato PPM (*Portable Pixel Map*) permite representar imágenes rectangulares en color.
Un archivo en formato PPM contiene una imagen [1] representada de la siguiente forma:

- Un número mágico que identifica el tipo de archivo. El número mágico para archivos en format PPM es la secuencia de caracteres **"P6"**.
- Uno o más espacios (carácter en blanco, tabuladores o caracteres de salto de línea).
- La anchura, formateada como caracteres de texto en decimal.

[1] Aunque el formato lo permite, en este trabajo no se considera el caso de más de una imagen en un mismo archivo.

- Uno o más espacios.
- La altura, formateada como caracteres de texto en decimal.
- Uno o más espacios.
- El valor máximo de color (intensidad), formateado como caracteres de texto decimal. Debe ser un número mayor que cero y menor que **65536**.
- Un solo carácter en blanco, que será típicamente el carácter de nueva línea (**newline**).
- Una secuencia de bytes que representa los píxeles de cada fila de la imagen (de arriba hacia abajo). Cada fila está formada por tantos píxeles como la anchura de la imagen. Hay tantas filas como la altura de la imagen.

La representación de cada píxel, depende del valor máximo de color. En aquellas imágenes en la que el valor máximo es **255** o menor, cada píxel está representado por tres bytes consecutivos que representan los valores para los colores rojo verde y azul).

Si el valor máximo está entre **256** y **65535**, cada píxel está representado por **6** bytes consecutivos, con **2** bytes para rojo, **2** para verde y **2** para azul. En este último caso, el orden de los bytes es *little-endian*.

2.2. Escalado de intensidad

La operación de escalado de intensidad consiste en representar la imagen utilizando un valor máximo distinto para cada valor RGB. Esto implica que se debe recalcular el valor de cada píxel de la imagen redondeando siempre al entero más cercano.

Como ejemplo, considera una imagen que está representada con un valor máximo de intensidad de **255**. Si en esa imagen hay un píxel con los valores [**R** = **0**, **G** = **128** y **B** = **255**], el valor del píxel resultante dependerá el nuevo valor máximo que se desee.

- Si el nuevo valor máximo es **128** se tendría:

$$R = \frac{0 \cdot 128}{255} = 0$$

$$G = \frac{128 \cdot 128}{255} = 64{,}25 \rightarrow 64$$

$$B = \frac{255 \cdot 128}{255} = 255$$

- Si el nuevo valor máximo es **65535** se tendría:

$$R = \frac{0 \cdot 65535}{255} = 0$$

$$G = \frac{128 \cdot 65535}{255} = 32896 \rightarrow 32896$$

$$B = \frac{255 \cdot 65535}{255} = 65535$$

IMPORTANTE

- Ten en cuenta que cuando se transforma una imagen con un valor de intensidad máxima menor de 256 a otro valor de intesidad máxima igual o superior a 256, se estará pasando de una representación de 3 bytes por píxel a otra representación de 6 bytes por píxel.
- Del mismo modo, cuando se pasa de un valor de intesidad máximo igual o superior a 256 a otro valor menor de 256 se estará pasando de una representación de 6 bytes por píxel a otra representación de 3 bytes por píxel.

2.3. Escalado de tamaño

Esta operación consiste en el cambio de tamaño de una imagen usando de la interpolación bilineal. El proceso consiste en determinar el valor de un píxel de la nueva imagen a partir de los píxeles de la imagen original. Para determinar este valor se debe determinar cuáles son los píxeles más próximos en la imagen original.

Si la imagen original tiene una anchura en píxeles w y una altura h, las coordenadas en dicha imagen van desde $(0, 0)$ hasta $(w - 1, h - 1)$. Así mismo, si la nueva imagen tiene una anchura en píxeles w' y una altura h', las coordenadas en dicha imagen van desde $(0, 0)$ hasta $(w' - 1, h' - 1)$.

Las coordenadas del píxel original (x, y) expresadas con números reales se correspondera con:

$$x = \frac{x' \cdot w}{w'}$$

$$y = \frac{y' \cdot h}{h'}$$

Para cada píxel en la nueva imagen con coordenadas (x', y'), se pueden determinar cuales son los 4 píxeles más próximos en la imagen original, determinando los valores x_l, x_h, y_l e y_h.

$$x_l = \lfloor x \rfloor$$

$$x_h = \lceil x \rceil$$

$$y_l = \lfloor y \rfloor$$

$$y_h = \lceil y \rceil$$

Una vez determinadas estas coordenadas, se calcularan mediante interpolación lineal los colores c_1 (interpolando entre los puntos (x_l, y_l) y (x_h, y_l)) y c_2 (interpolando entre los puntos (x_l, y_h) y (x_h, y_h)).

Por último, se realiza una interpolación en el eje y entre los dos valores previamente obtenidos.

2.4. Eliminación de colores poco frecuentes

Esta operación consiste en elminar los n colores menos frecuentes sustituyéndolos por los colores que queden en la imagen.

La operación tiene las siguientes etapas:

1. Deteminación de la frecuencia absoluta de cada color. Es decir, el número de apariciones de cada tupla (r, g, b).

2. Identificación de los n colores menos frecuentes. A igualdad de frecuencia se preferirá eliminar primero aquellos con un mayor valor en la componente b, luego en la componente g y finalmente en la componente r.

3. Para cada color a eliminar, se determinará cuál es el color que no se vaya a eliminar que se encuentra a menor distancia del color que se va a eliminar. Para determinar la distancia se utilizará la distancia euclídea.

4. Para todos los píxeles de la imagen original se sustituye cada ocurrencia del color original por el color determinado en el paso anterior.

2.5. Compresión de imágenes

En esta operación se genera una representación de la imagen en un formato ficticio CPPM, que realiza una compresión basada en tabla de colores.

El formato del archivo en este caso es el siguiente:

- Un número mágico que identifica el tipo de archivo. El número mágico para archivos en format CPPM es la secuencia de caracteres `"C6"`.
- Un único espacio en blanco.
- La anchura, formateada como caracteres de texto en decimal.
- Un único espacio en blanco.
- La altura, formateada como caracteres de texto en decimal.
- Un único espacio en blanco.
- El valor máximo de color (intensidad), formateado como caracteres de texto decimal. Debe ser un número mayor que cero y menor que **65536**.
- Un único espacio en blanco.
- El número de entradas de la tabla de colores, formateado como caracteres de texto en decimal.
- Un carácter de salto de línea.
- Una secuencia de colores expresados como valores RGB. El número de colores será exactamente el indicado en el campo anterior como número de entradas de la tabla de colores. Si el valor máximo de intensidad de la tabla de colores es **255** o menos, cada color se representa como 3 bytes. En otro caso cada color se representa como 6 bytes.
- Una secuencia de valores de píxeles. Donde cada píxel se representa como un único número representado en binario (*little endian*). El número se corresponderá con el índice del color en la tabla de colores y ocupará:
  - 1 byte si la tabla de colores tiene hasta 2^8 colores.
  - 2 bytes si la tabla de colores tiene hasta 2^{16} colores.
  - 4 bytes si la tabla de colores tiene hasta 2^{32} colores.
  - No se soportan tablas de colores mayores.

3. Tareas

3.1. Aplicación a desarrollar

Se desarrollarán dos aplicaciones con distintas estrategias de implementación. La aplicación **imtool-soa** utilizará la estrategia de implementación SOA (*structure of arrays* o estructuras de arrays) y la aplicación **imtool-aos** utilizará la estrategia de implementación AOS (*arrays of structures* o arrays de estructuras). Ambas aplicaciones tendrán una interfaz de línea de mandatos.

IMPORTANTE

Por simplicidad, en el resto de este documento se utiliza el nombre **imtool** para referirse indistintamente a **imtool-soa** e **imtool-aos**.

3.1.1. Parámetros de la aplicación

La aplicación tomará los siguientes parámetros:

- Ruta del archivo de entrada.
- Ruta del archivo de salida.
- Operación a ejecutar.
- Posibles parámetros adicionales dependiendo de la operación.

La operación a ejecutar será una de las siguientes:

- **info**: Presenta por la salida estándar información de los metadatos de la imagen.
- **maxlevel**: Realiza un escalado de intensidad a un nuevo valor máximo. En este caso, el nuevo valor máximo se suministra como un parámetro adicional.
- **resize**: Realiza un escalado del tamaño de la imagen. En este caso, la nueva anchura y la nueva altura se suministran como parámetros adicionales.
- **cutfreq**: Elimina los colores menos frecuentes. En este caso, el número de valores adicionales se suministran como parámetros adicionales.
- **compress**: Realiza una compresión de la imagen al formato **cppm**. En este caso, no hay parámetros adicionales.

3.1.2. Análisis de los argumentos de la aplicación

Si el número de argumentos recibidos por la aplicación es menor que tres, se generará un mensaje de error y el programa terminará con el código de error **-1**.

```
$ imtool
Error: Invalid number of arguments: 0
$ imtool photo.ppm
Error: Invalid number of arguments: 1
$ imtool photo.ppm out.ppm
Error: Invalid number of arguments: 2
```

Si el número de argumentos es igual o superior a tres, el tercer argumento deberá ser una de las siguientes cadenas: **info**, **maxlevel**, **resize**, **cutfreq**, **compress**. Cualquier otro valor como tercer parámetro hará que se imprima un mensaje de error y se genere el código de error **-1**.

```
$ imtool photo.ppm out.ppm copy
Error: Invalid option: copy
```

Si la opción es **info**, el número de argumentos debe ser exactamente tres. En otro caso, se genera un mensaje de error y si genera el código de error **-1**.

```
$ imtool photo.ppm out.ppm info 100
Error: Invalid extra arguments for info: 100
```

Si la opción es **maxlevel**, el número de argumentos debe ser exactamente cuatro. El cuarto argumento debe ser un número entero entre los valores **0** y **65535**. En otro caso, se genera un mensaje de error y si genera el código de error **-1**.

```
$ imtool photo.ppm out.ppm maxlevel
Error: Invalid number of extra arguments for maxlevel: 0
$ imtool photo.ppm out.ppm maxlevel 100 100
Error: Invalid number of extra arguments for maxlevel: 2
$ imtool photo.ppm out.ppm maxlevel -1
Error: Invalid maxlevel: -1
$ imtool photo.ppm out.ppm maxlevel 70000
Error: Invalid maxlevel: 70000
$ imtool photo.ppm out.ppm maxlevel copy
Error: Invalid maxlevel: copy
```

Si la opción es **resize**, el número de argumentos debe ser exactamente cinco. El cuarto argumento debe ser un número entero positivo que indica el nuevo ancho de la imagen. El quinto argumento debe ser un número entero positivo que indica el nuevo alto de la imagen. En otro caso, se genera un mensaje de error y si genera el código de error **-1**.

```
$ imtool photo.ppm out.ppm resize
Error: Invalid number of extra arguments for resize: 0
$ imtool photo.ppm out.ppm resize 100
Error: Invalid number of extra arguments for resize: 1
$ imtool photo.ppm out.ppm resize 100 200 300
Error: Invalid number of extra arguments for resize: 3
$ imtool photo.ppm out.ppm resize -100 200
Error: Invalid resize width: -100
$ imtool photo.ppm out.ppm resize 100 -200
Error: Invalid resize height: -200
$ imtool photo.ppm out.ppm resize 100 max
Error: Invalid resize height: max
```

Si la opción es **cutfreq**, el número de argumentos debe ser exactamente cuatro. El cuarto argumento debe ser un número entero positivo. En otro caso, se genera un mensaje de error y si genera el código de error **-1**.

```
$ imtool photo.ppm out.ppm cutfreq
Error: Invalid number of extra arguments for cutfreq: 0
$ imtool photo.ppm out.ppm cutfreq 100 100
Error: Invalid number of extra arguments for cutfreq: 2
$ imtool photo.ppm out.ppm cutfreq -1
Error: Invalid cutfreq: -1
$ imtool photo.ppm out.ppm cutfreq copy
Error: Invalid cutfreq: copy
```

Si la opción es **compress** el número de argumentos debe ser exactamente tres. En otro caso, se genera un mensaje de error y si genera el código de error **-1**.

```
$ imtool photo.ppm out.ppm compress 100
Error: Invalid extra arguments for info: 100
```

3.2. Desarrollo de software

3.2.1. Programa a desarrollar

Esta tarea consiste en el desarrollo de una versión secuencial de la aplicación descrita utilizando C++20. Por favor, ten en cuenta que no se permite utilización de varios hilos de ejecución (*multithreading*) ni de paralelismo de ninguna clase.

Se desarrollarán dos programas ejecutables independientes: **imtool-soa** e **imtool-aos** que implementarán respectivamente las estrategias **SOA** y **AOS**.

- **SOA – Structure of Arrays**: Se representarán los píxeles de una imagen como tres secuencias independientes. Cada una de las secuencias contendrá elementos que podrán estar en el rango de **0** a **255** si el nivel máximo de intensidad es menor o igual a **255**, o bien en el rango de **0** a **65535** en otro caso.
- **AOS – Arrays of Structures**: Se representarán los píxeles de una imagen como una única secuencia de valores. Cada valor de la secuencia estará formada por tres campos que podrán estar en el rango de **0** a **255** o en el rango de **0** a **65535** en otro caso.

Recomendaciones

- Evalúe los distintos contenedores de seucuencia que ofrece la biblioteca estándar. Puede encontrar una lista de los mismos en `https://en.cppreference.com/w/cpp/container`.
- Identifique los tipos enteros más apropiados para cada contexto ofrecidos por el lenguaje (`https://en.cppreference.com/w/cpp/language/types#Integral_types`) y por la biblioteca estándar (`https://en.cppreference.com/w/cpp/types/integer`).

3.2.2. Componentes

Se desarrollarán los siguientes componentes

- **common**: Biblioteca con los archivos fuente comunes a las dos versiones del programa (**soa** y **aos**).
- **imgsoa**: Biblioteca con todos los componentes de procesamiento de imágenes con la estrategia **soa** usados desde el programa principal.
- **imgaos**: Biblioteca con todos los componentes de procesamiento de imágenes con la estrategia **aos** usandos desde el programa principal.
- **utest-common**: Pruebas unitarias de todos los componentes de la biblioteca **common**.
- **utest-imgsoa**: Pruebas unitarias de todos los componentes de la biblioteca **imgsoa**.
- **utest-imgaos**: Pruebas unitarias de todos los componentes de la biblioteca **imgaos**.
- **ftest-soa**: Pruebas funcionles de la aplicación **imtool-soa**.
- **ftest-aos**: Pruebas funcionles de la aplicación **imtool-aos**.
- **imtool-soa**: Programa ejecutable con la aplicación usando la estrategia **SOA**.
- **imtool-aos**: Programa ejecutable con la aplicación usando la estrategia **AOS**.

A continuación, se describen algunos componentes con más detalle:

- **common**: Esta biblioteca contendrá los siguientes componentes:
  - **progargs.hpp**/**progargs.cpp**: Tratamiento de los argumentos del programa recibidos por **main()**.
  - **binaryio.hpp**/**binario.cpp**: Soporte para entrada salida binaria.
  - Otros componentes comunes a las versiones **soa** y **aos**.
- **imgaos**: Esta biblioteca contendrá los siguientes componentes:
  - **imageaos.hpp**/**imageaos.cpp**: Soporte a imágenes con represetnación **aos**.
  - Otros componentes específicos para la solución **aos**.
- **imgsoa**: Esta biblioteca contendrá los siguientes componentes:
  - **imagesoa.hpp**/**imagesoa.cpp**: Soporte a imágenes con represetnación **soa**.
  - Otros componentes específicos para la solución **soa**.
- **utest-common**: Este ejecutable integrará todas las pruebas unitarias para la biblioteca **common**.
  - Contendrá por cada componente de la biblioteca **common** con sus correspondientes pruebas unitarias.
- **utest-imgaos**: Este ejecutable integrará todas las pruebas unitarias para la biblioteca **imgaos**.
  - Contendrá por cada componente de la biblioteca **imgaos** con sus correspondientes pruebas unitarias.
- **utest-img-soa**: Este ejecutable integrará todas las pruebas unitarias para la biblioteca **imgsoa**.
  - Contendrá por cada componente de la biblioteca **imgsoa** con sus correspondientes pruebas unitarias.
- **imtool-aos**: Programa ejecutable para la versión **aos**. Contendrá solamente el siguiente archivo fuente:
  - **main.cpp**: Contendrá exclusivamente la función **main()** no permitiéndose ninguna función auxiliar.
- **imtool-soa**: Programa ejecutable para la versión **soa**. Contendrá solamente el siguiente archivo fuente:
  - **main.cpp**: Contendrá exclusivamente la función **main()** no permitiéndose ninguna función auxiliar.
- **ftest-aos**: Contendrá todas las pruebas funcionales para el ejecutable **imtool-aos**.
- **ftest-soa**: Contendrá todas las pruebas funcionales para el ejecutable **imtool-soa**.

3.2.3. Compilación del proyecto

Todos los archivos fuente deben compilar sin problemas y no emitirán ninguna advertencia del compilador. Se incluirá un archivo de configuración de CMake con las siguientes opciones:

CMakeLists.txt principal

```
cmake_minimum_required(VERSION 3.26)
project(imtool LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_EXTENSIONS OFF)

# Set compiler options
add_compile_options(-Wall -Wextra -Werror -pedantic -pedantic-errors -Wconversion -Wsign-conversion)
set(CMAKE_CXX_FLAGS_RELEASE "${CMAKE_CXX_FLAGS_RELEASE} -march=native")

# Enable GoogleTest Library
include(FetchContent)
FetchContent_Declare(
        googletest
        GIT_REPOSITORY https://github.com/google/googletest.git
        GIT_TAG v1.15.2
)
FetchContent_MakeAvailable(googletest)

# Enable GSL Library
FetchContent_Declare(GSL
        GIT_REPOSITORY "https://github.com/microsoft/GSL"
        GIT_TAG v4.0.0
        GIT_SHALLOW ON
)
FetchContent_MakeAvailable(GSL)

# Run clang-tidy on the whole source tree
# Note this will slow down compilation.
# You may temporarily disable but do not forget to enable again.
set(CMAKE_CXX_CLANG_TIDY clang-tidy;
        -format-style=file;
        -header-filter=.;)


# All includes relative to source tree root.
include_directories(PUBLIC .)

# Process cmake from sim and fluid directories
add_subdirectory(common)
add_subdirectory(imgaos)
add_subdirectory(imgsoa)
add_subdirectory(imtool-aos)
add_subdirectory(imtool-soa)

# Unit tests and functional tests
enable_testing()
add_subdirectory(utest-common)
add_subdirectory(utest-imgaos)
add_subdirectory(utest-imgsoa)
add_subdirectory(ftest-aos)
add_subdirectory(ftest-soa)
```

El archivo de configuración de CMake para el directorio **common** incluirá las siguientes opciones:

CMakeLists.txt para la biblioteca common

```
# Add to this list all files related to common library
add_library(common
        progargs.cpp
        binaryio.cpp
        other2.cpp
)

# Use this line only if you have dependencies from this library to GSL
target_link_libraries(common PRIVATE Microsoft.GSL::GSL)
```

El archivo de configuración de CMake para el directorio **imgaos** [2] incluirá las siguientes opciones:

CMakeLists.txt para la biblioteca imgaos

```
# Add to this list all files related to imgaos library
add_library(imgaos
        image.cpp
        other1.cpp
        other2.cpp
)

# Use this line only if you have dependencies from this library to GSL
target_link_libraries(imgaos PRIVATE common Microsoft.GSL::GSL)
```

El archivo de configuración de CMake para el directorio **imtool-aos**[3] incluirá las siguientes opciones:

CMakeLists.txt para programa imtool

```
add_executable(imtool-aos main.cpp)
target_link_libraries(imtool-aos imgaos common)
```

Ten en cuenta que estas reglas son solamente ejemplos. Puedes adaptar estos archivos de configuración siempre que sigas las reglas generales establecidas en este enunciado.

Recuerda que todas las evaluaciones se realizarán con las optimizaciones del compilador activadas con la opción de CMake **-DCMAKE_BUILD_TYPE=Release**.

Importante

La única biblioteca permitida es la biblioteca estándar de C++. No se permite bibliotecas externas.

Excepción

Se permite el uso de la biblioteca de soporte a las *C++ Core Guidelines*. La última versión se puede obtener en: `https://github.com/microsoft/GSL`.

3.2.4. Reglas de calidad de código

El código fuente debe estar bien estructurado y organizado, así como apropiadamente documentado. Se recomienda (aunque no se requiere) seguir las **C++ Core Guidelines** (`http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines`).

[2] El necesario para **imgsoa** será similar.
[3] El necesario para **imtool-soa** será similar.

No obstante, si deben seguirse todas las reglas especificadas en el documento **Reglas de codificación para el lenguaje C++** que se publica separadamente.

Para facilitar, el trabajo de los equipos, se suministrará también un archivo de configuración para la herramienta **clang-tidy**.

3.2.5. Pruebas unitarias

Se definirá un conjunto de pruebas unitarias que también se entregarán. Se recomienda, el uso de GoogleTest (`https://github.com/google/googletest`). Si se desea utilizar otro marco de trabajo para pruebas unitarias se deberá obtener la autorización del coodinador de la asignatura.

En cualquier caso, se entregarán evidencias de que hay pruebas unitarias suficientes.

Si utilizas GoogleTest como marco de trabajo para pruebas unitarias, a continuación puedes ver ejemplos de archivos CMake que pueden resultarte útil.

CMakeLists.txt para pruebas unitarias de common

```
# Executable for all unit tests with list of sources
# For example, you may have one *_test.cpp for each *.cpp in common
add_executable(utest-common
        one_test.cpp
        other1_test.cpp
        other2_test.cpp)

# Library dependencies
target_link_libraries(utest-common
        PRIVATE
        common
        GTest::gtest_main
        Microsoft.GSL::GSL)
```

CMakeLists.txt para pruebas unitarias de img-aos

```
# Executable for all unit tests with list of sources
# For example, you may have one *_test.cpp for each *.cpp in img-aos
add_executable(utest-img-aos
        one_test.cpp
        other1_test.cpp
        other2_test.cpp)

# Library dependencies
target_link_libraries(utest-img-aos
        PRIVATE
        img-aos
        GTest::gtest_main
        Microsoft.GSL::GSL)
```

3.2.6. Uso de herramientas de IA

El uso de herramientas basadas en IA durante el proyecto está permitido. No obstante, se debe tener en cuenta lo siguiente:

- Si usas una herramienta basada en IA, se deben declarar los usos en la sección de diseño de la memoria del proyecto. No se realizará ninguna penalización en la declaración siempre que se incluya una declaración de uso.
- No se dará soporte al uso de dichas herramientas. En particular, si tienes preguntas sobre tu código, deberás ser capaz de explicar dicho código.

- Ten en cuenta que algunas herramientas de IA pueden generar código inseguro o código con bajo rendimiento.
- Cualquiera de tus profesores podrá requerirte una explicación de tu propio código.

3.3. Evaluación del rendimiento y la energía

Esta tarea consiste en la realización de una evaluación comparativa del rendimiento y el consumo energético. Para llevar a cabo la evaluación del rendimiento, se medirá el tiempo total de ejecución con la herramienta **perf**. Además, también se medirá la energía y se derivará la potencia.

Todas las evaluaciones del rendimiento se realizarán en un nodo del clúster **avignon**.

Representa en una gráfica los tiempos totales de ejecución, uso de energía y potencia para distintas imágenes. Realiza una análisis de escalabilidad para aquellas opciones de la aplicación para las que tenga sentido.

La memoria del proyecto incluirá conclusiones derivadas de los resultados. Por favor, no te limites a una mera descripción de los datos. Trata de encontrar explicaciones convincentes de estos resultados.

4. Calificación

Las calificaciones finales para este proyecto se obtienen de la siguiente forma:

- Rendimiento: 20 %.
- Energía usada: 20 %.
- Pruebas unitarias: 7 %.
- Pruebas funcionales: 3 %.
- Calidad del diseño: 5 %.
- Calidad del código: 5 %.
- Evaluación de rendimiento y energía en la memoria: 15 %.
- Contribuciones de cada miembro: 20 %.
- Conclusiones: 5 %.

ADVERTENCIAS IMPORTANTES:

- Si el código entregado no compila, la nota final de la práctica será de 0.
- Si se ignora injustificadamente alguna norma de calidad de código, la nota final de la práctica será de **0**.
- Si el tiempo de ejecución se considera desmesuradamente alto, se calificará la práctica con un **0**.
- En caso de copia todos los grupos implicados obtendrán una nota de 0. Además, se notificará a la dirección de la EPS para la correspondiente apertura de expediente disciplinario.

5. Procedimiento de entrega

La entrega del proyecto se realizará a través de Aula Global.
Para ello se habilitarán 2 entregadores separados:

- **Entregador de memoria**. Contendrá la memoria del proyecto, que será un archivo en formato pdf con el nombre **report.pdf**.
- **Entregador de código**: Contendrá todo el código fuente necesario para compilar la aplicación.
  - Debe ser un archivo comprimido (formato zip) con el nombre **imtool.zip**.

La memoria no deberá exceder de 15 páginas con una fuente mínima de 10 puntos incluyendo la portada y todas las secciones. no se tendrá en cuenta en la corrección los contenidos a partir de la página 16 si fuese el caso. Deberá contener, al menos, las siguientes secciones:

- **Página de título**: contendrá los siguientes datos:
  - Nombre de la práctica.
  - Nombre del grupo reducido en el que están matriculados los estudiantes.
  - Número de equipo asignado.
  - Nombre y nia de todos los autores.
- **Diseño**. Debe incluir el diseño de cada uno de los componentes. En esta sección se deben explicar y justificar cada una de las principales decisiones de diseño tomadas.
- **Optimización**. Debe contener una discusión de las optimizaciones aplicadas y su impacto. En particular, deberán indicarse optimizaciones realizadas sobre el código fuente original, así como optimizaciones activadas con flags de compilación adicionales que se estime oportuno.
- **Pruebas realizadas**: Descripción del plan de pruebas realizadas para asegurar la correcta ejecución. Debe incluir pruebas unitarias, así como pruebas funcionales de sistema.
- **Evaluación de rendimiento y energía**: Deberá incluir las evaluaciones de rendimiento y energía llevadas a cabo.
- **Organización del trabajo**: Deberá describir la organización del trabajo entre los miembros del equipo haciendo explícita las tareas llevadas a cabo por cada persona.
  - Debe contener una división del proyecto en tareas.
  - Las tareas deben ser suficientemente pequeñas como para que se pueda asignar una única persona a una tarea.
  - Todos los integrantes del equipo deberán realizar contribuciones relevantes en el diseño y construcción de componentes software.
  - No se podrá asignar más de una persona a una tarea. En tal caso, la tarea debe subdividirse en subtareas.
  - Se debe indicar el tiempo dedicado por cada persona a cada tarea.
- **Conclusiones**. Se valorará especialmente las derivadas de los resultados de la evaluación del rendimiento, así como las que relacionen el trabajo realizado con el contenido de la asignatura.