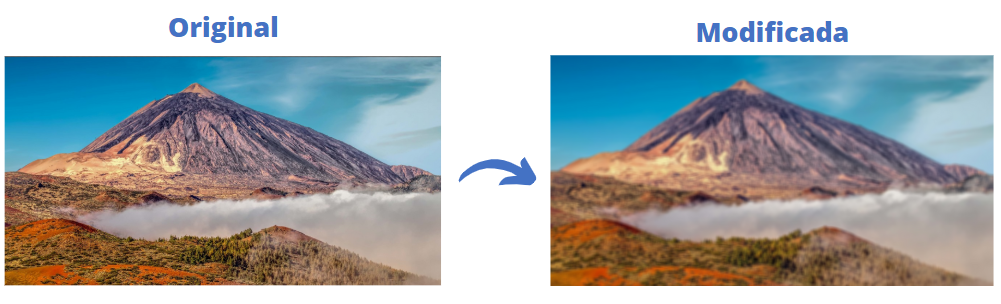


# Práctica 3

1. Elegir cualquier problema de tratamiento de imágenes e implementarlo en C/C++. Puede ser de tratamiento se vídeo.

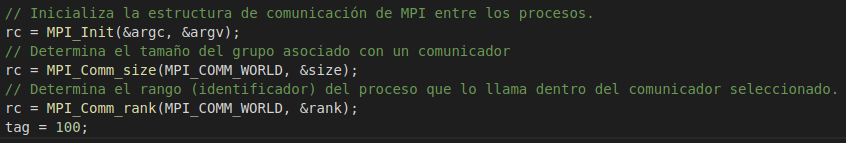
Para este apartado he tratado de buscar un algoritmo secuencial que permita el tratamiento de una imagen. Tras buscar entre varios, me he decantado por el que podemos ver a continuación “[https://gist.github.com/OmarAflak/ aca9d0dc8d583ff5a5dc16ca5cdda86a](https://gist.github.com/OmarAflak/%20aca9d0dc8d583ff5a5dc16ca5cdda86a)”.

El algoritmo comentado con anterioridad nos permite aplicar un filtro gaussiano que nos generará un desenfoque a una imagen. Esto lo podremos ver en la siguiente diapositiva:

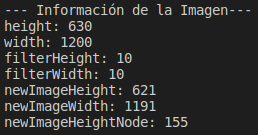
2. Implementa una versión MPI para este algoritmo.

Tras entender el código secuencial me he dispuesto a modificarlo para crear una versión usando MPI. Los pasos para lograr esto han sido:

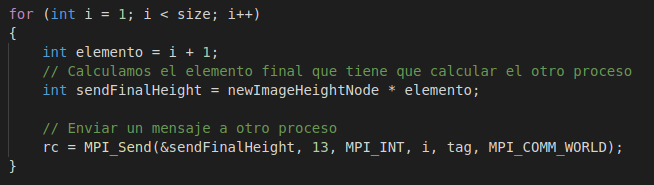
1. Inicializar la estructura de MPI.



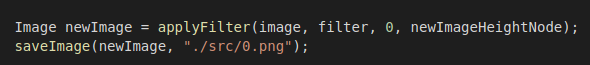
1. Dividir el programa en dos. Uno para si el proceso es el inicial y otro para el resto de los procesos.
2. Añadir Información relevante sobre la imagen por consola.

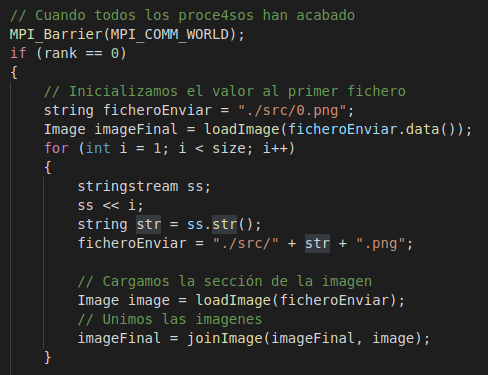


1. Dividimos la imagen por el número de proceso. Esto nos ayudará a saber que sección deberá realizar cada proceso. A su vez, se calculará y se enviará el tamaño final de cada sección para que dicho proceso sepa que sección deberá de computar.



1. Posteriormente, cada proceso (incluido el principal) tendrá que realizar el filtrado de la sección que tiene asignada y, guardar el resultado en la carpeta “src” bajo el nombre “proceso.png”.



1. Por último, tras acabar todos los procesos el nodo principal tendrá que unificar las imágenes para completar la diapositiva modificada.

Cabe resaltar que se ha modificado la función de aplicar el filtrado debido a que era necesario que solo realizara una sección de la imagen y devuelva únicamente dicha sección.

3. Desarrolla una versión OpenMP para este algoritmo.

4. Compara ambas versiones.