Tarea 1 - Estructuras de Datos (INF-134)

Publicación: Viernes 21 de Marzo
Entrega: Viernes 04 de Abril - 23:55h vía Aula

2025-1

1 Reglas

- La tarea debe realizarse en grupos de 2 personas. Cualquier situación excepcional debe ser aprobada por el ayudante coordinador (Tomás Barros).
- Debe usarse el lenguaje de programación C++. Al realizar la evaluación, la tarea será compilada usando el compilador g++ en Linux, usando el comando

  ```
  g++ tarea1.cpp -o tarea1 -Wall.
  ```

  No se aceptarán entregas realizadas con MinGW en Windows o compiladores online como Dev-C++, Replit, GDB, etc. En caso de usar Mac, se recomienda usar XCode, y de ser posible, testear la tarea también en Linux.
- No se permite usar la biblioteca STL, así como ninguno de los contenedores y algoritmos definidos en ella (e.g. vector, list, etc.). En general no se permite importar ninguna estructura de datos, salvo por string. Para la tarea 1 se permite importar la librería `stb_image`, como se comenta posteriormente.
- Una tarea que no compile tendrá nota 0. En dicha situación, el/la estudiante puede apelar con el ayudante coordinador.
- La tarea puede entregarse con máximo 1 día de atraso, en cuyo caso la nota máxima será 50.
- Solo una de las tres tareas del semestre puede tener nota inferior a 30.
- Si se detecta **COPIA** o abuso de herramientas de generación automática de código (ChatGPT), la situación será revisada por el ayudante coordinador y profesor coordinador. Si se considera que el estudiante no entiende el código que escribió, tendrá nota 0 en la tarea.

2 Problema a resolver

Las imágenes pueden ser representadas de muchas maneras en un computador. Una de ellas es el formato .png, el cual podemos leer en C++ utilizando la librería `stb_image`[1], incluida en la tarea:

```
unsigned char* data = stbi_load(filename, & width, & height, & channels, 0);
```

Si bien generalmente pensamos en una imagen como un arreglo tridimensional (ancho, alto, canales (RGB)), en esta tarea lo estamos almacenando en un arreglo unidimensional.

2.1 Preguntas Teóricas

Para cada una de estas preguntas se deben citar fuentes confiables. Se considera una fuente fidedigna a cualquier publicación, libro o documentación que tenga un moderado apoyo comunitario, fama o renombre [2]. En caso de no existir una fuente única con la información, citar aquellas fuentes que permitieron derivar la respuesta.

Las respuestas a estas preguntas deben adjuntarse en un .txt o .md adjunto en la tarea.

Pregunta 1 (5 pts)

¿Por qué si los pixeles de la imagen son números enteros $\in [0, 255]$, utilizamos el tipo de dato `unsigned char`?

Pregunta 2 (5 pts)

¿Cuales serían las diferencias entre usar `unsigned int` versus `unsigned char`? Responder cuantitativamente para una imagen RGB de dimensiones (N,N,3).

Pregunta 3 (5 pts)

Usando este esquema de almacenamiento unidimensional: ¿Cuál sería el índice del primer pixel del canal azul?

Pregunta 4 (5 pts)

¿Cómo podemos convertir una imagen RGB a escala de grises?

[1] Revise los detalles de la librería en: `https://solarianprogrammer.com/2019/06/10/c-programming-reading-writing-images-stb_image-libraries/`

[2] Use las directrices especificadas en `https://biblioteca.usm.cl/busqueda/uso_etico_informacion` para listar sus citas. Los modelos de lenguaje, como ChatGPT, no son fuentes, si bien pueden proveerlas

2.2 Problema

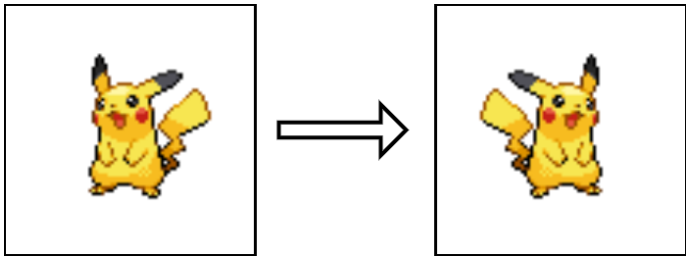
Queremos implementar algunas operaciones clásicas sobre imágenes. Incluido con este PDF se entrega un código `base.cpp`, que incluye un struct `Imagen` y un par de funciones (`save()` y `load()`) capaces de leer y almacenar en memoria un archivo .png, además de un main que ejecuta un ejemplo para facilitar la comprensión del almacenamiento de la imagen.

Ejemplo del struct Imagen:

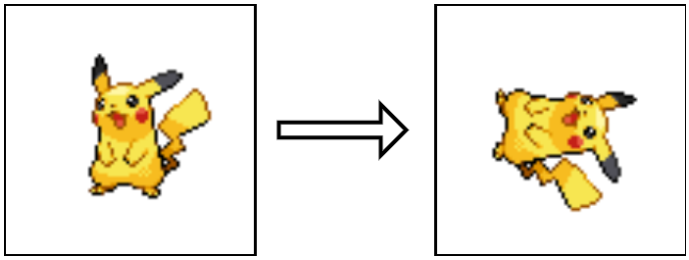
```
struct Imagen{
    unsigned char* data;
    int width, height, channels;
};
```

Pregunta 5 (40 pts)

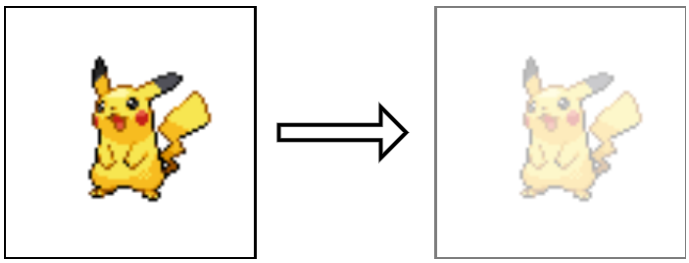
Se pide implementar una función para cada una de las siguientes operaciones:



Operación 1: No recibe parámetros

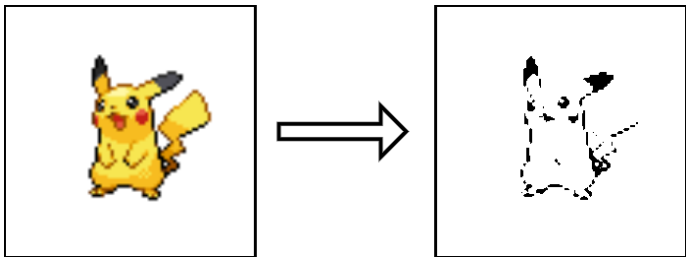


Operación 2: No recibe parámetros



Operación 3: Recibe un parámetro *float* indicando grado de *atenuación*; Se multiplican los valores de cada pixel por un valor $\in [0, 1]$

.

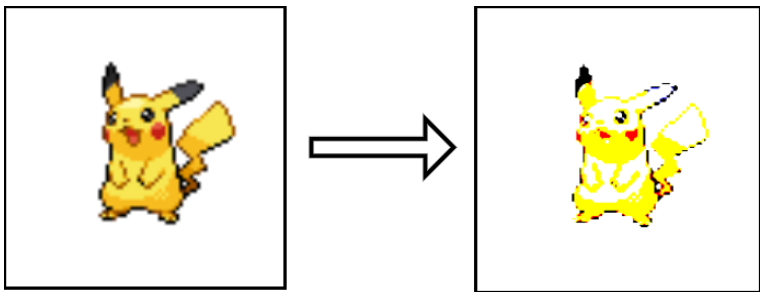


Operación 4: Recibe un parámetro *int* indicando límite sobre el cual un pixel se vuelve blanco, y bajo el cual se vuelve negro

Las funciones no deben retornar una nueva imagen; modifican la imagen actual.

Pregunta 6 (10 pts)

La siguiente implementación de la Operación 4 produce un resultado incorrecto. ¿Cuál es el error, y como podría resolverse a nivel de código? La respuesta puede justificarse teóricamente o con código.



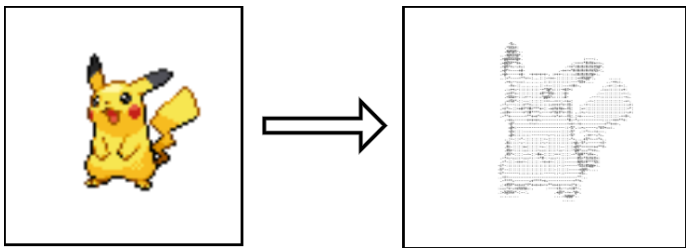
Implementación incorrecta de la operación 4

Pregunta 7 (30 pts)

ASCII Art es la práctica de recrear imágenes y videos tradicionales usando exclusivamente caracteres de texto plano (ASCII). Queremos programar un conversor de .png a ASCII Art.

Para ello, el procedimiento es el siguiente:

1. Definir un arreglo de caracteres ASCII de largo N, ordenados por *luminosidad aparente*: Caracteres como '|' y '.' son más luminosos que caracteres como '#' y 'X'.
2. Dividir el rango $[0, 255]$ en N intervalos, asignando un caracter a cada intervalo.
3. Sustituir cada pixel de la imagen de entrada (en escala de grises) por su caracter correspondiente.
4. Escribir los resultado a un archivo .txt



Transformación a formato ASCII Art

Pregunta 7.1

Se debe implementar una función `to_ascii(...)` que tome como entrada una imagen y un arreglo de caracteres ASCII y retorne un arreglo bidimensional de caracteres.

Pregunta 7.2

Adicionalmente, se debe implementar una función `save_ascii(...)` que tome dicho arreglo y genere un archivo de texto con el resultado.

3 Notas adicionales

- Para cada función de las preguntas 5 y 6, la firma debe ser de la forma:

```
void operacion_1(...)
```

- En caso de utilizar alguna estructura o variable cuyo tamaño en memoria es definido en el momento de ejecución, ésta debe ser definida utilizando memoria dinámica (`new`), y debe ser liberada correctamente al finalizar el programa. Notar que las funciones de la librería `stb_image` lo hacen por defecto.[3]

[3]Se recomienda utilizar `valgrind` para revisar el uso correcto de memoria

- El *main* de su programa debe generar las cuatro imágenes .png de la pregunta 5, como también la imagen ASCII. Su programa debe funcionar para cualquier imagen .png.
- Sus imagenes .png no deben tener transparencia. Para quitarle transparencia a una imagen, se puede abrir en Paint y sobre-escribirla.
- Al usar modo oscuro, la luminosidad de los caracteres se invierte. No es problema que los caracteres queden invertidos.
- Los 20 puntos de las preguntas teóricas solo se otorgarán en caso de entregar una tarea con código que intente resolver los problemas planteados en la tarea.

4 Formato de Entrega

Se debe entregar un archivo llamado `tarea1-apellido1-apellido2.zip`[4] que contenga:

1. `tarea1.cpp` : El archivo que contiene su código
2. Los archivos `stbi_image.h` y `stbi_image_write.h` necesarios para su funcionamiento
3. Una imagen `.png` a su elección. Se recuerda tener prudencia y escoger imágenes apropiadas para el contexto académico.
4. `README.txt` con los nombres y roles de los integrantes, además de un apartado con instrucciones de compilación y ejecución en caso de ser distintas a las esperadas.

5 Directrices de programación

Las directrices corresponden a buenas prácticas de programación, las cuales serán tomadas en consideración para la evaluación de la tarea. Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota.

Cada función de código cuya función no sea inmediatamente aparente, debe tener un comentario de la siguiente forma:

```
/*****
*   TipoFunción NombreFunción
******
*   Resumen Función
******
*   Input:
*       tipoParámetro NombreParámetro : Descripción Parámetro
*        .......
******
*   Returns:
*       TipoRetorno, Descripción retorno
*****/
```

[4] apellido1 y apellido2 corresponden al primer apellido de cada integrante.

Además,

- Se deben utilizar nombres de variables descriptivos.
- Se deben seguir buenas prácticas respecto al uso de funciones. Es decir, se deben utilizar funciones para evitar redundancia y para modularizar las componentes de su código.
- Se debe considerar la indentación correcta del código.

6 Contacto

Profesores

- Elizabeth Montero (**Coordinadora**) (elizabeth.montero@usm.cl)
- Alondra Rojas (alondra.rojas@usm.cl)
- José Miguel Cazorla (jcazorla@usm.com)
- Ricardo Salas (ricardo.salas@usm)
- Juan Pablo Castillo (juan.castillog@sansano.usm.cl)
- Roberto Díaz (roberto.diazu@usm.cl)

Ayudantes (CC)

- Tomás Barros (**Coordinador**) (tomas.barros@sansano.usm.cl)
- Jaime Inostroza (jinostroza@usm.cl)
- Facundo Riquelme Lucero (friquelmel@usm.cl)
- Franco Cerca (franco.cerda@usm.cl)
- Bastián Jimenez (bjimenez@usm.cl)

Ayudantes (SJ)

- Juan Alegría (juan.alegria@usm.cl)
- Damián Rojas (damian.rojas@usm.cl)
- Ignacia Nettle Oliveri (inettle@usm.cl)
- Alvaro Aceituno Alarcon (alvaro.aceitunoa@usm.cl)
- Lucas Morrison (lucas.morrison@sansano.usm.cl)
- Javier Matamala Gonzalez (javier.matamalag@usm.cl)