**Manipulação de Texto, Áudio e Imagem/Vídeo**

Aluno nº mec.  
Aluno nº mec.

Cada alínea 3 tópicos: como uso, decisões, como estou a fazer a quantificação, Resultados e Análises

**Introdução:** Este projeto requer a implementação de programas em linguagem C++ e a utilização do repositório Github para gerir as várias versões das aplicações desenvolvidas, por forma a dar solução aos vários exercícios para manipulação de informação nos formatos de texto, áudio e imagem.

O projeto está estruturado em 5 partes (A, B, C, D, e E), com os seguintes tópicos:

**Parte A.** Aquisição de fundamentos na linguagem de programação C++;

**Parte B.** Criação de cópias integrais dos ficheiros de texto, áudio e imagem;

**Parte C.** Extração de informação quanto à quantificação e distribuição dos dados (entropia), recorrendo também à apresentação de histogramas;

**Parte D**. Manipulação dos dados com o objetivo da redução de bits e a análise da redução da qualidade final em relação ao original;

**Parte E.** Elaboração do presente relatório,descrevendo todo o processo de desenvolvimento do Projeto.

Iremos, portanto, detalhar o funcionamento de cada um dos programas, justificando as decisões tomadas e apresentando algumas conclusões.

O link do projeto no Github é <https://github.com/zpdomingues16/icproject1>

Para documentar a aplicação foi utilizado o *software* Doxygen, no formato html, que pode ser acedido através do *link: (link)*

# Parte B

**Exercício 2:**  Este exercício recebe o nome de dois ficheiros : *argv*[1] (nome do primeiro ficheiro) e *argv*[2] (nome do segundo ficheiro) , lendo o conteúdo do primeiro e passando-o caracter a caracter para o segundo. Se o ficheiro de destino existir, o seu conteúdo será substituído, caso tal não aconteça, será criado um novo ficheiro com o nome atribuído.

Usámos as classes *ifstream* (leitura de um ficheiro) e *ofstream* (escrita e criação de ficheiros). Depois, dentro de um ciclo, aplicámos a função *get* da classe *ifstream,* que permite obter um único caracter da *ifstream* e copia-o para o novo ficheiro. Com recurso ao código *endl* optámos por colocar cada caracter em linhas diferentes.

Abordámos o problema desta forma, porque nos pareceu ser a maneira mais intuitiva, simples e mais rápida de implementar.

**Exercício 3:** O exercício 3 tem como principal objetivo, implementar um programa que copie um ficheiro de áudio (formato wav) para um novo ficheiro de áudio.

Seguindo as indicações do professor na aula prática e tendo por base as indicações/instruções presente no *Github*, decidimos trabalhar com a *library* “*Audiofile*”.

Daí procedemos à instanciação de dois objetos do tipo *AudioFile*, um para o ficheiro original e outro para o ficheiro de destino(cópia). Tanto o nome do ficheiro original, como o da cópia são passados como argumentos do programa (em argv[1] e argv[2]). Utilizámos depois dois ciclos for (um dentro do outro) para obter o número de samples em cada um dos canais do ficheiro original e por fim atribuímos os mesmos valores aos parâmetros do segundo objeto do tipo *AudioFile* instanciado. Os ficheiros de áudio a serem trabalhados, têm de estar na mesma pasta do programa.

**Exercício 4:** O exercício 4 tinha como premissa a cópia de um ficheiro de imagem, para um novo ficheiro. Com recurso à *library* *OpenCV*, criámos um objeto do tipo *Mat* (chamado *img*) que recebe um nome de um ficheiro de imagem (passado em *argv*[1]). Criámos depois um segundo objeto do tipo *Mat* (uma matriz de pixéis todos com o valor 0) com o mesmo tamanho de *img.rows* e *img.cols*.

Esta matriz é preenchida tendo em conta o parâmetro CV\_8UC3, indicativo de um 8-bit *unsigned integer* matriz/imagem com 3 canais, neste caso BGR.

Depois, tal como no exercício anterior, fazemos dois ciclos for (um dentro do outro) para percorrer as colunas e as linhas da imagem original e atribuímos esses valores à img2. Isto é feito com recurso à função *at(vec3b)* da classe *Mat* (vec3b, representa um vector com 3 entradas de um byte, neste caso BGR). Para terminar, os dados são escritos num novo ficheiro de imagem, através da utilização da função *imwrite()*, que neste caso terá como argumento o nome do segundo ficheiro passado em *argv[2].*

# Parte C

**Exercício 5**: Para obter o

1. Como usar o programa
2. Abordagem
3. Resultados e Análise

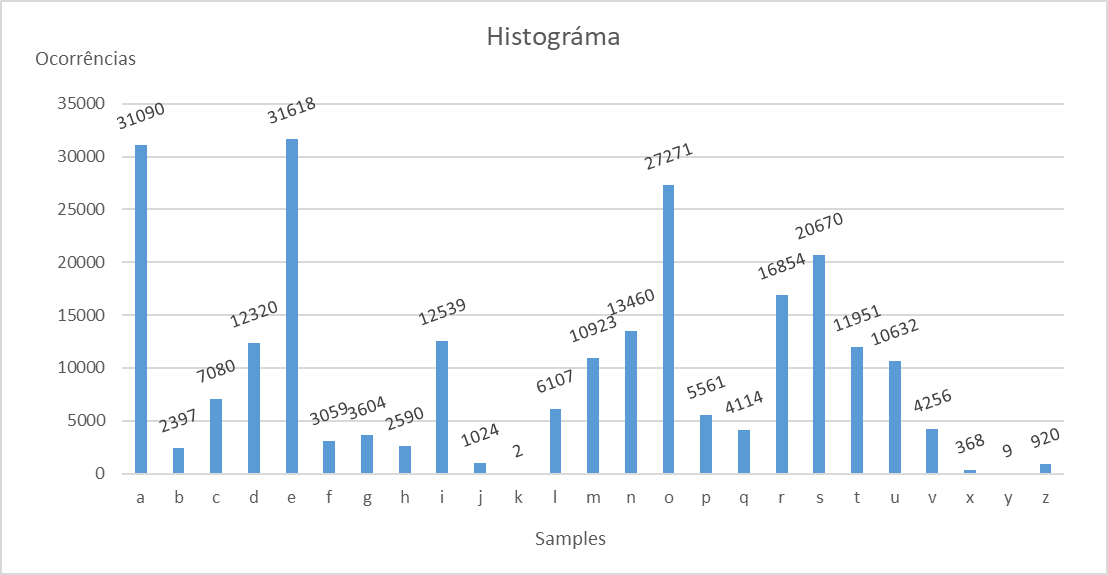


Figura 1. Histograma do alfabeto do taxto "Os Lusíadas", de Luís de Camões.

**Exercício 6:**

1. Como usar o programa
2. Abordagem
3. Resultados e Análise

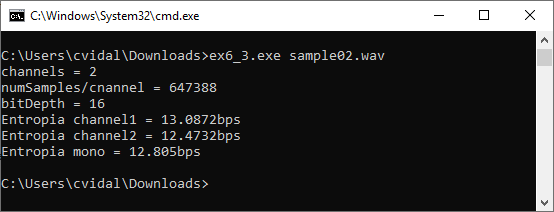


Figura 2. Prompt da execução do programa do exercício 6

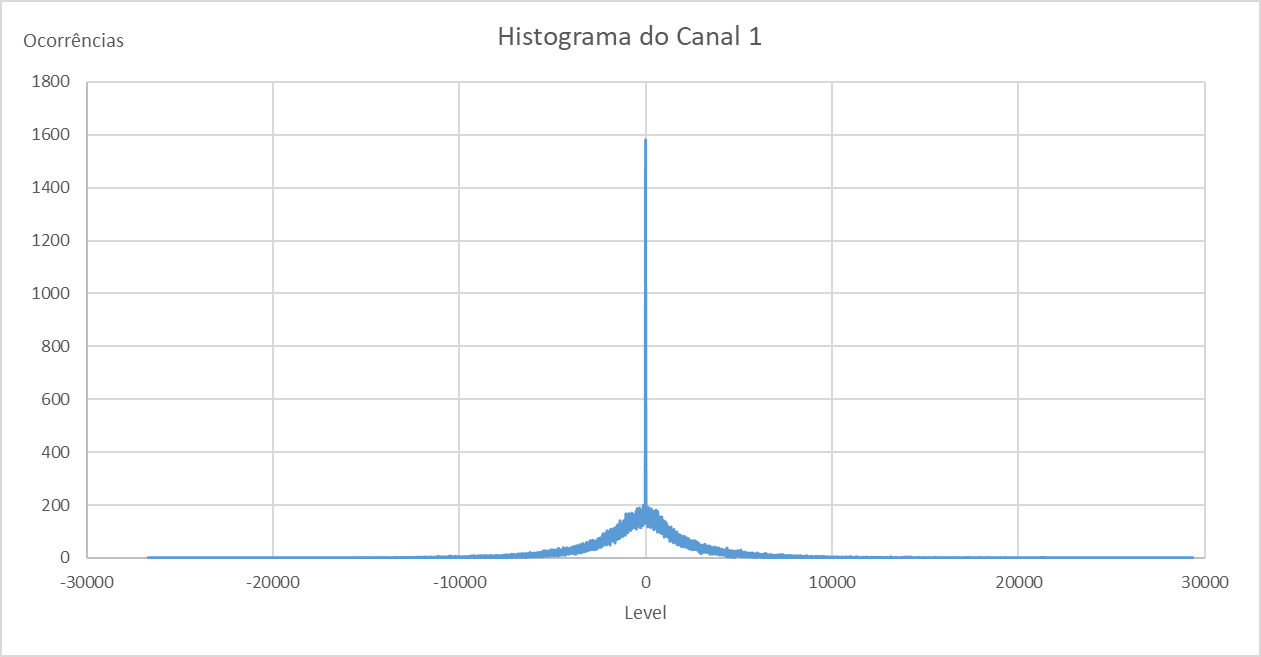


Figura 3. Histográma do canal 1.

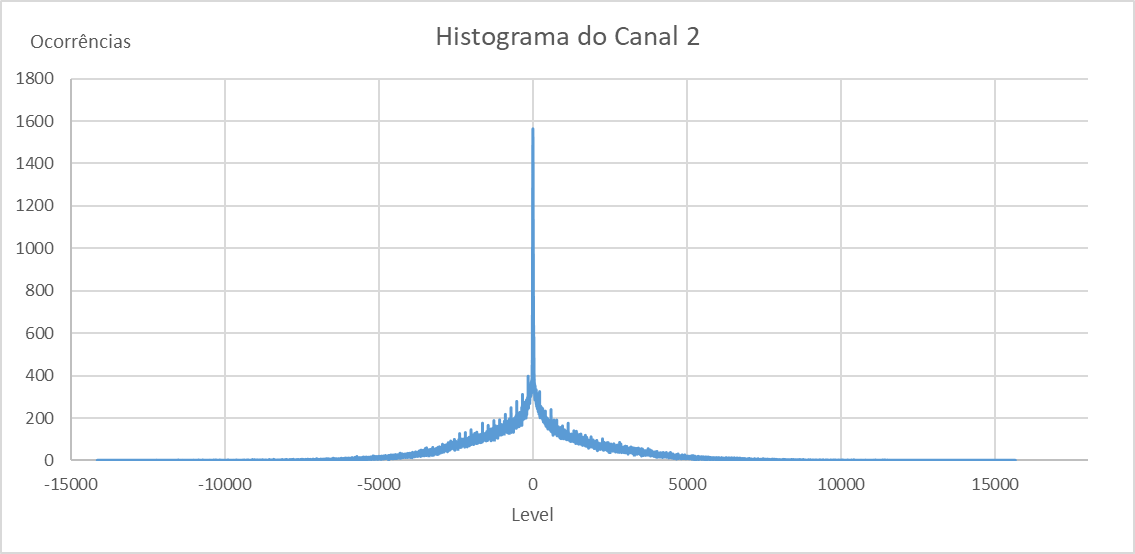


Figura 4. Histogáma do canal 2.

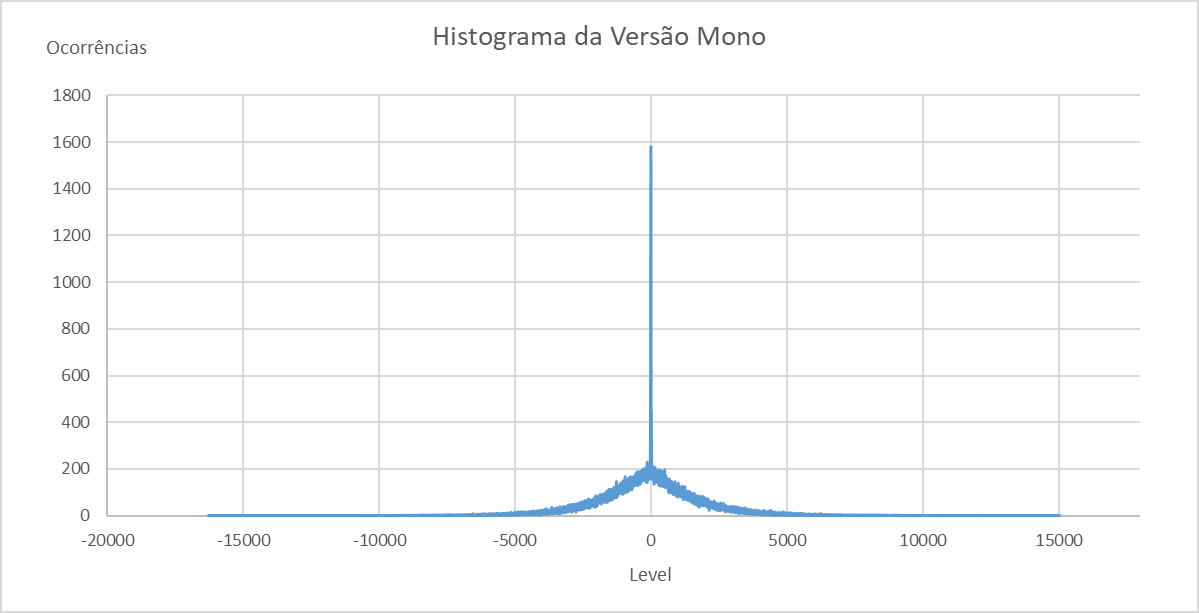


Figura 5. Histográma da versão Mono.

**Exercício 7:**

1. Como usar o programa
2. Abordagem
3. Resultados e Análise

# Parte D

**Exercício 8:**

1. Como usar o programa

Por defeito, a redução da profundidade de bits por amostra (sem a indicação de argumentos na linha de comando), é de 50%, com o mínimo de 8 bits, podendo ser introduzido um argumento “–b” para indicar a profundidade de bits desejada

1. Abordagem

A profundidade de bits por amostra (*sample*), que corresponde diretamente à resolução de cada amostra, standard para CD áudio é de 16 bits, sendo típico o uso de 24 bits para áudio de alta resolução, e 8 bits o recomendado pelo ITU-T - International Telecommunication Union Telecommunication Standardization Sector, para telecomunicações.

Neste contexto, decidimos aplicar, por defeito, uma redução por quantificação de 50%, com o mínimo de 8 bits, permitindo outras reduções, mas com uma normalização *raw* de 8, 16 ou 24bits, ou seja, uma redução para, por exemplo, de 16 bits para 12bits, será normalizada posteriormente para 16bits e uma redução para 20bits será normalizada posteriormente para 24bits, resultando num ficheiro *raw* com o mesmo tamanho, mas que permitirá uma codificação/compactação *lossless* posterior superior, pela redução do número de samples indistinguíveis.

Criamos o mesmo ficheiro de áudio *sample02.wav,* com redução de 1 (um) bit, no bit menos significativo, e atribuímos nome *red\_sample02.wav*

1. Resultados e Análise

Corremos também o programa do exercício 6 sobre o áudio reduzido *red\_sample02.wav,* para verificar as diferenças quer nos histogramas, quer das entropias, relativamente ao áudio original.

Uma imagem com texto

Descrição gerada automaticamente

Figura 6. Entropias do ficheiro de áudio *red\_sample02.wav*

Sendo a diferença mínima, podemos concluir que não houve perda de informação significativa.

**Exercício 9:**

1. Como usar o programa
2. Abordagem
3. Resultados e Análise

**Exercício 10:**

1. Como usar o programa
2. Abordagem
3. Resultados e Análise

**Exercício 11:**

1. Como usar o programa
2. Abordagem
3. Resultados e Análise

# Conclusões