Universidade Federal de Minas Gerais **Departamento de Ciência da Computação**	**Estrutura de Dados** **Profs Gisele Pappa e Wagner Meira Jr**

Trabalho Prático 0

Valor: 5 pontos extra
Entrega: 08/09/2022

1. Introdução

Este trabalho prático tem por objetivo recordar alguns conceitos fundamentais do desenvolvimento de programas em linguagem C/C++, além de introduzir outros conceitos em termos de tipos abstratos de dados, em particular a sua abstração, desempenho e robustez.

Você implementará um programa para conversão de uma imagem colorida para tons de cinza. Uma imagem digital em tons de cinza é uma imagem na qual o valor de cada pixel é uma amostra de um espaço de cores. Imagens desse tipo são tipicamente compostas por diversos tons de cinza, variando entre o preto como a menor intensidade e o branco como maior intensidade. Imagens em tons de cinza são geralmente resultado de um cálculo da intensidade da luz em cada pixel em cada faixa do espectro electromagnético (como, por exemplo, o espectro visível).

Para converter um pixel colorido em seu nível aproximado de cinza, deve-se primeiro obter suas primitivas vermelho, verde, e azul (da escala RGB). Embora existam diferentes estratégias para realizar essa conversão, para este trabalho, a seguinte estratégia deverá ser usada:

$$Y = \frac{49}{255}(0.30\mathrm{R} + 0.59\mathrm{G} + 0.11\mathrm{B})$$

onde R, G, e B correspondem aos valores de vermelho, verde, e azul do pixel original, respectivamente, e Y corresponde ao tom de cinza mapeado.

2. Especificações

A imagem de entrada deverá estar no formato .ppm (ASCII, P3) com representação de 8 bits por componente de cor (R, G, e B) de cada pixel (valores podem ir de 0 a 255). A imagem de saída em tons de cinza deverá estar no formato .pgm (ASCII, P2) com no máximo 50 tons de cinza (valores podem ir de 0 a 49). Em breve novas imagens serão disponibilizadas para uso na avaliação experimental. Mais detalhes sobre os formatos .ppm e .pgm podem ser encontrados no endereço http://en.wikipedia.org/wiki/Netpbm_format.

O seu programa deverá cumprir os seguintes requisitos:

1. Criar estruturas de dados para armazenar imagens nos formatos PPM e PGM
2. Fazer a alocação dos dados das imagens dinamicamente
3. Implementar uma função para leitura de uma imagem PPM
4. Implementar uma função para conversão de uma imagem PPM em uma imagem PGM
5. Implementar uma função para escrita de uma imagem PGM

A avaliação de desempenho deve ser realizada utilizando a biblioteca ***memlog***. Para cada operação deve ser avaliado o seu custo computacional à medida que variamos a resolução das imagens, ou seja, que variamos as dimensões de uma matriz que representará cada imagen, e também o seu padrão de acesso à memória e localidade de referência. É necessário ainda uma outra opção de linha de comando, **-p**, que indica o arquivo onde será feito o registro de acesso e uma opção **-l**, que indica se devem ser registrados os acessos à memória, ou apenas o tempo de execução. Uma observação pertinente é que o custo de registro de acesso à memória pode ser significativo e sugerimos que sejam feitas duas baterias de testes, uma para desempenho computacional e outra para o padrão de acesso à memória.

A robustez do código deve ser implementada utilizando as macros definidas em **msgassert.h** e devem tratar todos os tipos de erros associados ao tipo abstrato de dados definido.

Em suma a mensagem de uso do seu programa pode ser:

programa

-i "nome do arquivo de entrada" (entrada em formato .ppm)

-o "nome do arquivo de saída" (saída em formato .pgm)

-p log.out (registro de desempenho)

-l (padrão de acesso e localidade)

Está disponível no Moodle um manual do memlog e um exemplo de código de um trabalho prática que realiza operações com matrizes estáticas, ou seja, sem a utilização de ponteiros. Um primeiro passo para entender o memlog é entender esse código.

3. Entregáveis

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é **terminantemente vetado**.

Você **DEVE utilizar** a estrutura de projeto abaixo junto ao *Makefile* disponibilizado no **Moodle**:

- TP

    |- src

    |- bin

    |- obj

    |- include

    Makefile

A pasta **TP** é a raiz do projeto; a pasta **bin** deve estar vazia; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (*headers*) do projeto, com extensão ***.h**, por fim a pasta **obj** deve estar vazia. O **Makefile** disponibilizado no *Moodle* deve estar na **raiz do projeto**. A execução do **Makefile** deve gerar os códigos objeto ***.o** no diretório **obj** e o executável do TP no diretório **bin**.

Existe no Moodle um vídeo onde um dos monitores do semestre passado discute sobre a função e uso de makefiles.

4. Documentação

A documentação do trabalho deve ser entregue em formato **pdf** e também **DEVE** seguir o modelo de relatório que será postado no Moodle. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

Título, nome, e matrícula.
Introdução: Contém a apresentação do contexto, problema, e qual solução será empregada.
Método: Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
Análise de Complexidade: Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
Estratégias de Robustez: Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
Análise Experimental: Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.

Conclusões: A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
Bibliografia: Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
Instruções para compilação e execução: Esta seção deve ser colocada em um apêndice ao fim do documento e em uma página exclusiva (não divide página com outras seções).
***Número máximo de páginas:** 10

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.
Evite a descrição literal do código-fonte na documentação do trabalho.
Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

5. Submissão

Todos os arquivos relacionados ao trabalho devem ser submetidos na atividade designada para tal no Moodle. A entrega deve ser feita **em um único arquivo** com extensão **.zip**, com nomenclatura nome_sobrenome_matricula.zip, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais. O arquivo **.zip** deve conter a sua documentação no formato **.pdf** e a estrutura de projeto descrita no início da Seção 4.

6. Avaliação

O trabalho será avaliado de acordo com:

- Corretude na execução dos casos de teste - (50% da nota total)
- Apresentação da análise de complexidade das implementações - (15% da nota total)
- Estrutura e conteúdo exigidos para a documentação - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Cumprimento total da especificação - (5% da nota total)

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0. Trabalhos entregues com atrasos sofrerão penalização de 2^{d-1} pontos, com d = dias de atraso.

[1] Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

7. Considerações Finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é CRIME. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na sessão de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos**.

FaQ (Frequently asked Questions)

1. **Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO**
2. Posso utilizar o tipo String? SIM.
3. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO
4. Posso utilizar alguma biblioteca para tratar exceções? SIM.
5. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
6. As análises e apresentação dos resultados são importantes na documentação? SIM.
7. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO
8. Posso fazer o trabalho em dupla ou em grupo? NÃO
9. Posso trocar informações com os colegas sobre a teoria? SIM.
10. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM.
11. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.