BCC202 – Estruturas de Dados I (2020-01)

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (www.decom.ufop.br/)

Trabalho Prático 2 (TP2) - 10 pontos, peso 2.

- Data de entrega: 18/04/2021 até 23:55. O que vale é o horário do Moodle, e não do *seu*, ou do *meu*, relógio!!!

- O padrão de entrada e saída deve ser respeitado exatamente como determinado no enunciado.

- Parte da correção é automática, não respeitar as instruções enunciadas pode acarretar em perda de pontos.

- Entregar um relatório.

- Entregar um vídeo.

- Procedimento para a entrega:.

1. Submissão: via **Moodle**.
2. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
3. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
4. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos *.h* e *.c* sempre que cabível.
5. Os arquivos a serem entregues, incluindo aquele que contém *main()*, devem ser compactados (*.zip*), sendo o arquivo resultante submetido via **Moodle**.
6. Dentre os arquivos submetidos, deve existir um intitulado *compilcao.txt*, contendo os comandos especificados no *prompt/console* para compilar e executar seu programa.
7. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.

- Procedimento para o vídeo:

1. Na apresentação, mostrem as chamadas dos programas e suas execuções no vídeo.
2. Na execução do programa, utilize dados que demonstrem que o programa esta funcionando e também para casos especiais, **se houver**.
3. Ainda na apresentação, comentem seu código para facilitar o entendimento de quem esteja assistindo.
4. Nos códigos, comentem de maneira a evidenciar o que está sendo feito naquele trecho.
5. Coloque nomes de variáveis sugestivas (mesmo que grandes).
6. Tempo máximo de vídeo: 10min. Tempo extra será desconsiderado.

- Bom trabalho!

1 Objetivos

Este trabalho prático tem como objetivo principal fundamentar conceitos aprendidos em sala e nas práticas.

2 Descrição do Problema

Dois problemas serão trabalhados.

2.1 Problema 1

Implementar o algoritmo *QuickSort* utilizando pelo menos duas formas diferentes de seleção do pivô (uma sendo necessariamente aleatório) e combinado com o método de ordenação por inserção (*InsertionSort*).

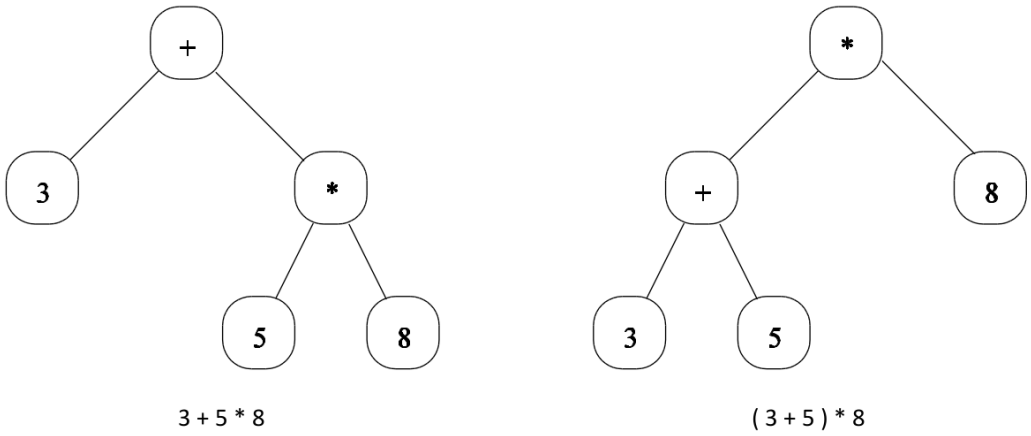
Sabe-se que o algoritmo *QuickSort* executa no pior caso em tempo $O(n \log n)$ e o algoritmo de ordenação por *InsertionSort* no pior caso em tempo $O(n^2)$. No entanto, os fatores constantes do *InsertionSort* o tornam mais rápido que o *QuickSort* para um pequeno valor de n. Assim, faz sentido usar o *InsertionSort* quando os sub-problemas tornam-se suficientemente pequenos.

Implemente o *QuickSort*, onde o caso de parada não é 0 ou 1, mas sim um tamanho k, onde o *InsertionSort* é usado para ordenar esse subproblema de tamanho k.

Faça avaliações e análises relacionadas ao tamanho de k e a escolha do pivô. Você deve avaliar também a questão da ordem do arquivo (aleatório, crescente e decrescente) e com tamanho de arquivos a serem ordenados de 100 e 500 inteiros.

2.2 Problema 2

Considere as árvores binárias para representar expressões aritméticas (operandos inteiros, operadores +, -, *, / e parênteses) como as apresentadas abaixo.



Dada as árvores apresentadas, pede-se:

a) Escreva um algoritmo que receba uma expressão matemática na forma de string e retorne uma árvore binária representando essa expressão respeitando a precedência dos operadores matemáticos.

- Menor precedência: soma (+) e subtração (-).
- Maior precedência: multiplicação (*) e divisão (/).
- O parêntese se sobrepõe a todos os outros.

b) Escreva um algoritmo que receba a árvore binária da expressão e calcula seu valor. (Dica: Caminhamento Pós-Ordem).

3 Imposições e comentários gerais

Neste trabalho, as seguintes regras devem ser seguidas:

- Seu programa não pode ter *memory leaks*, ou seja, toda memória alocada pelo seu código deve ser corretamente liberada antes do final da execução. (Dica: utilize a ferramenta *valgrind* para se certificar de que seu código libera toda a memória alocada)
- Um grande número de *Warnings* também ocasionará a redução na nota final.

3.1 Comentários Gerais:

- Clareza, identação e comentários no código também vão valer pontos. Por isso, escolha cuidadosamente o nome das variáveis e torne o código o mais legível possível.
- Trabalhos copiados (e FONTE) terão nota zero, além de os alunos envolvidos no plágio perderem toda a nota atribuída a participação e pontos extras, entre outros...
- O trabalho pode ser feito em dupla, porém, caso seja necessário, o professor pode escolher aleatoriamente um dos alunos para uma entrevista. Assim, é necessário que ambos tenham conhecimento do trabalho por completo.

3.2 O que deve ser entregue

- Código fonte do programa em C (bem identado e comentado).
- Documentação do trabalho (relatório). A documentação deve conter:

  1. **Implementação:** descrição sobre a implementação do programa. Não faça "print screens" de telas. Ao contrário, procure resumir ao máximo a documentação, fazendo referência ao que julgar mais relevante. É importante, no entanto, que seja descrito o funcionamento das principais funções e procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Muito importante: os códigos utilizados na implementação devem ser inseridos na documentação.
  2. **Impressões gerais**: descreva o seu processo de implementação deste trabalho. Aponte coisas que gostou bem como aquelas que o desagradou. Avalie o que o motivou, conhecimentos que adquiriu, entre outros.
  3. **Análise**: deve ser feita uma análise dos resultados obtidos com este trabalho.
  4. **Conclusão**: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  5. **Formato:** PDF.

3.3 Como deve ser feita a entrega

Verifique se seu programa compila e executa na linha de comando antes de efetuar a entrega. Quando o resultado for correto, entregue via *Moodle* até a 18/04/2021 até 23:55 um arquivos **.ZIP** com o nome e sobrenome dos dois alunos da dupla. Esse arquivo deve conter: (i) os arquivos *.c* e *.h* utilizados na implementação, (ii) instruções de como compilar e executar o programa, e (iii) o relatório em **PDF**.

Exemplo de um nome do arquivo a ser entregue: *pedro-silva_guilherme-silva.zip*.

4 PONTOS EXTRAS

Será concedido até 5,0 pontos extras para a dupla que desenvolver uma **interface gráfica** de qualidade para o sistema com Qt, por exemplo (ou outra biblioteca). Para valer nota extra, a interface gráfica deve implementar os seguintes recursos:

- Leitura do arquivo: uma interface para escolher o arquivo texto a ser processado.
- Visualização das árvores: alguma forma de visualização das árvores. Este recurso é especialmente interessante para o aluno debugar visualmente a árvore construída.

Vale ressaltar que ainda há a necessidade de rodar ambos os problemas propostos via terminal. Além disso, você deve especificar como compilar e rodar o seu código com a interface gráfica proposta .