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

Trabalho Prático 1

Resolvedor de Expressão Numérica

Valor: 15 pontos
Data de entrega: 02/05/2023

Neste trabalho, seu objetivo será implementar um resolvedor de expressões númericas. O sistema terá suporte a modificação da representação da expressão numérica, assim como a possibilidade de solução automática da expressão.

1. Descrição

Você está ensinando matemática em uma escola de ensino fundamental e ao chegar ao tópico de expressões numéricas você decidi criar um resolvedor automático para não precisa resolver manualmente os exercícios. Sabendo que seus alunos são exigentes e muitos são interessados em programação, seu programa também deve ter a opção de ler a expressão numérica e converter da forma de representação usual(chamada de infixa) para a notação polonêsa inversa (também chamada de posfixa).

O sistema deve suportar expressões com as 4 operações básicas: multiplicação (X); divisão (/); adição (+) e subtração (-). Os números podem ser inteiros ou fracionários (parte inteira separada da fracionária por “,”) e a divisão de inteiros pode gerar um número fracionário. O seu sistema deve também suportar o uso de parênteses na notação usual para a mudança da ordem de precedência dos operadores. Sua responsabilidade é fazer uma implementação eficiente e, munido do conhecimento de tipos abstratos de dados e alocação dinâmica de memória, fazer este trabalho será moleza!

A primeira tarefa é você definir uma "representação interna" para as expressões numéricas. Essa representação deve suportar a implementação das seguintes operações:

- **Ler Expressão Numérica:** esta operação deve ler uma expressão numérica passada como argumento e armazenar essa expressão no programa. A expressão pode estar em notação infixa ou posfixa e o programa deve verificar se a expressão numérica é válida.

Por exemplo, 3 + 4 é uma expressão válida, mas 2 + - x não é. E só deve armazenar a expressão se a mesma for válida

- **Converter para Notação Posfixa:** esta operação deve converter a expressão armazenada no programa para a notação posfixa. Mais informações sobre a notação posfixa podem ser encontradas em https://pt.wikipedia.org/wiki/Notacão_polonesa_inversa
- **Converter para Notação Infixa:** esta operação deve converter a expressão armazenada no programa para a notação infixa com parênteses. Mais informações obre essa notação podem ser encontradas em https://pt.wikipedia.org/wiki/Notação_infixa
- **Resolver Expressão:** esta operação deve resolver a expressão armazenada e apresentar o valor numérico final correspondente a computação da expressão.

2. O que deve ser feito?

Você deverá implementar o sistema descrito acima. O resolvedor deverá suportar todas as quatro operações descritas. Para cada operação processada, será gerada uma saída. Cada operação pode gerar um conjunto definido de saídas, especificado abaixo. O formato da saída deve corresponder **exatamente** ao formato especificado, pois será através dela que a correção do resolvedor será feita.

A entrada do programa será um arquivo de texto contendo, em cada linha, uma operação simulando um conjunto de operações sobre o seu programa. A simulação deve terminar quando o final do arquivo for alcançado. Para cada linha do arquivo de entrada, ou seja, para cada operação, deverá ser impressa uma linha na saída padrão (na tela), conforme especificado abaixo. A seguir, estão especificadas as operações, conforme presentes no arquivo de entrada:

- **LER TIPOEXP EXP**
  - Operação de ler uma expressão. O parâmetro TIPOEXP pode ser INFIXA ou POSFIXA, dependendo da natureza da expressão. O parâmetro EXP é uma string com no máximo 1000 caracteres, onde os números e operadores estão separados por espaço.
  - O programa deve armazenar essa expressão utilizando uma estrutura de dados apropriada, cuja definição faz parte do trabalho. Ao se ler uma nova expressão a expressão anterior deve ser apagada. O programa deve verificar se a expressão é válida antes de armazenar a mesma e só deve armazenar a expressão se esta for válida (seja em notação infixa ou posfixa).
  - Saída esperada (*em todas elas, EXP deve ser substituído pelo valor correspondente*):
    - EXPRESSAO OK: EXP - caso a expressão seja lida e armazenada com sucesso;
    - ERRO: EXP NAO VALIDA- caso a expressão não veja válida e não armazenada.
  - Exemplo: LER INFIXA ((5 + 3) * 4)

- **INFIXA**
    - Operação de converter a expressão armazenada no programa para notação infixa (notação usual). Esta operação deve acrescentar os parênteses necessários.
    - Saída esperada (*em todas elas, EXP deve ser substituído pelo valor correspondente*):
        - INFIXA: EXP - caso a conversão seja realizada com sucesso;
        - ERRO: EXP NAO EXISTE - caso não haja expressão armazenada.
    - Exemplo: INFIXA: (5 + (3 * 2))

- **POSFIXA**
    - Operação de converter a expressão armazenada no programa para notação posfixa (notação polonesa).
    - Saída esperada (*em todas elas, EXP deve ser substituído pelo valor correspondente*):
        - POSFIXA: EXP - caso a conversão seja realizada com sucesso;
        - ERRO: EXP NAO EXISTE - caso não haja expressão armazenada.
    - Exemplo: POSFIXA: 5 3 2 * +

- **RESOLVE**
    - Operação de resolver a expressão armazenada no programa. Esta operação deve computar o valor representado pela expressão armazenada no programa.
    - Saída esperada (*em todas elas, VAL deve ser substituído pelo valor correspondente*):
        - POSFIXA: VAL - caso a conversão seja realizada com sucesso;
        - ERRO: EXP NAO EXISTE - caso não haja expressão armazenada.
    - Exemplo: VAL : 11

3. O que implementar:

Você deverá implementar pelo menos um Tipo Abstrato de Dados (TADs)/ Classe para representar a expressão numérica. Você precisará ainda de outros auxiliares para as representações das notações infixas e posfixas.

DICA 1: A notação polonesa permite o uso de um algoritmo simples para a solução do expressão numérica, se o TAD apropriado for escolhido. Que estrutura seria interessante para armazenar essa representação, tendo em vista a resolução da expressão?

DICA 2: A representação da prioridade indicada pelos parênteses na notação infixa representa uma ordenação das operações e uma hierarquia das mesmas. Que estrutura de dados é a mais apropriada para a representação dessa hierarquia?

4. 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.

5. 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, 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.

6. 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.

7. Avaliação

O trabalho será avaliado de acordo com:

- Definição e implementação das estruturas de dados e funções - (30% da nota total)
- Corretude na execução dos casos de teste - (20% 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.

8. 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.