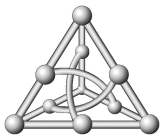

UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Avaliação de Expressões Aritméticas

Trabalho 1

Estruturas de Dados — 2025

1 Descrição do Trabalho

Uma *expressão completamente parentizada* é aquela em que cada operação binária está envolvida por um par de parênteses, indicando de forma explícita a ordem de avaliação da expressão. Por exemplo, a expressão **(3 + (2 * 5))** está completamente parentizada, pois todas as operações estão envoltas por parênteses.

Essas expressões podem ser representadas graficamente por meio de uma árvore binária conforme visto em sala, onde:

- Cada nó interno representa uma operação (como +, -, *, /),
- Cada nó folha representa um operando inteiro.

A estrutura da árvore determina a ordem em que as operações devem ser realizadas, conforme definido pela hierarquia dos parênteses.

2 Objetivo

Você deverá implementar, utilizando a linguagem C/C++, um programa que:

- Leia um conjunto de k expressões aritméticas completamente parentizadas, compostas pelas quatro operações básicas (+, -, *, /) e operandos inteiros;
- Construa a árvore binária correspondente a cada expressão;
- Imprima a representação da árvore (por exemplo, usando travessias como pré-ordem ou forma gráfica);
- Usando uma TAD pilha, avalie a expressão representada pela árvore e imprima o resultado;
- Libere toda a memória alocada para a árvore após a avaliação de cada expressão, deixando-a vazia para a próxima.

3 Requisitos da Entrega

O projeto deve estar estruturado em cinco arquivos separados:

- arvore.c – Implementação da TAD (Tipo Abstrato de Dados) Árvore Binária;
- pilha.c - Implementação da TAD (Tipo Abstrato de Dados) Pilha e suas operações;
- pilha.h - Arquivo de cabeçalho contendo a definição das estruturas e protótipos das funções da pilha;
- arvore.h – Arquivo de cabeçalho contendo a definição das estruturas e protótipos das funções da árvore;
- main.c – Arquivo principal, responsável pela leitura das expressões, chamada das funções para construção, impressão e avaliação da árvore.

Exemplo de Entrada

```
(3 + (2 * 5))
((7 - 4) * (6 + 2))
((1 + 2) * ((3 - 4) + 5))
```

Exemplo de Saída

```
Expressão: (17 - (2 * 15))
Árvore (pós-ordem): 3 2 15 * -
Resultado: 13

Expressão: ((17 - 5) * (6 + 4))
Árvore (pós-ordem): 17 5 - 6 4 + *
Resultado: 120
```

4 Data e forma de Entrega

1. O trabalho pode ser feito por grupos de **até duas pessoas**. Cada membro do grupo deve obrigatoriamente entregar uma cópia do trabalho no AVA (idêntico ao do outro membro). Após abrir uma sessão digitando seu *login* e sua senha, vá até o tópico e escolha "Entrega do T1". Você pode entregar o trabalho quantas vezes quiser até às **23 horas e 59 minutos** do dia **16 de maio de 2025**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

2. O trabalho será corrigido usando o compilador `gcc`. Caso use outro compilador, antes da entrega, certifique-se que seu código compila corretamente com o `gcc`.

3. Cada grupo deve gerar um único arquivo .tgz com o comando dado a seguir e depois submeter este arquivo no AVA:

```
$ tar czvf jose_silva.tgz main.c arvore.c pilha.c arvore.h pilha.h
```

   onde `$` é o sinal de pronto do sistema e não deve ser digitado, `tar` é um utilitário de arquivamento do Linux, `jose_silva` é o seu *login* e `.tgz` é uma extensão que identifica este arquivo.

4. Certifique-se de comentar adequadamente seu código e organizar os arquivos conforme solicitado. Cada arquivo deve conter em suas primeiras linhas um comentário com o nome do(s) membro(s) do grupo com RGA, nome da disciplina, e um comentário geral sobre o que faz o arquivo.

```
/*************************************************
 *
 * Nome dos(as) estudantes:
 * Trabalho 1
 * Disciplina:
 * Objetivo:
 */
#include <stdio.h>
```

5. Dê atenção especial ao tratamento de memória (uso de malloc/free em C).

6. **Conduta ética**

   Cada grupo tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não faça o trabalho com outros grupos e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, mas NÃO copie o programa!

7. Trabalhos considerados plagiados terão nota ZERO. Projetos que não compilarem receberão, no máximo, 40% da nota. Alguns estudantes poderão ser chamados para explicar seu código desenvolvido.