Nome: Hélio Henrique Strappazzon
Matricula: 22111165 Email: hélio.strappazzon@edu.pucrs.br

Professora: Eduarda Rodrigues Monteiro

RELATORIO TRABALHO 2:

CALCULADORA DE EXPRESSÕES ARITMETICAS

INTRODUÇÃO:

A proposta deste trabalho é criar uma calculadora de expressões aritméticas usando pilhas, capaz de lidar com expressões complexas que incluem os operadores básicos, além de parênteses e colchetes para definir a ordem das operações. A solução envolve a criação de uma classe chamada Calculadora, que processa expressões escritas em um arquivo de texto, verificando se estão corretas e calculando o resultado, ou apontando possíveis erros de sintaxe. Para isso, a implementação utiliza uma pilha para os números e outra para os operadores, além de uma estrutura auxiliar para assegurar que os delimitadores (parênteses, colchetes e chaves) estejam corretamente fechados. Durante o desenvolvimento do projeto, foram utilizadas as classes Pilha e LinkedOfInteger que foram apresentadas em aula e disponibilizadas na plataforma do Moodle pela professora Eduarda Rodrigues Monteiro.

DESCRIÇÃO DO ALGORITMO:

A classe Calculadora foi projetada para avaliar expressões aritméticas de maneira matematicamente correta, utilizando uma abordagem que envolve o uso de pilhas e métodos específicos para lidar com operadores e operandos. Ela trabalha com três componentes principais: uma pilha de operandos, uma pilha de operadores e uma pilha auxiliar para delimitadores. A estrutura da classe inclui métodos para avaliar expressões (avaliarExpressao), processar operadores (processarOperador), verificar operadores válidos (isOperador), determinar precedência (precedencia), conferir correspondência de delimitadores (isMatchingDelimiter), e controlar o tamanho máximo da pilha (updateMaxPilhaSize, getMaxPilhaSize, resetMaxPilhaSize). Além disso, a classe oferece métodos para processar arquivos com múltiplas expressões (processarArquivo), assegurando saídas corretas e tratamento adequado de erros de sintaxe.

MÉTODO avaliarExpressao(String expressao):

Este método avalia uma expressão aritmética fornecida em forma de string, retornando o resultado ou lançando uma exceção em caso de erro de sintaxe. Ele utiliza pilhas para operandos, operadores e delimitadores para processar a expressão de acordo com suas regras de precedência e associatividade.

MÉTODO processarOperador(Pilha operandos, char operador):

O método processa um operador específico, desempilhando os operandos do topo da pilha e realizando a operação correspondente. Em seguida, empilha o resultado de volta na pilha de operandos e atualiza o tamanho máximo da pilha se necessário.

MÉTODO isOperador(char c):

Este método verifica se um caractere é um operador aritmético válido (como +, -, *, /, ^). Retorna true se o caractere é um operador, e false caso contrário.

MÉTODO precedencia(char operador):

O método determina a precedência de um operador aritmético. Ele retorna um valor inteiro que indica a precedência, com 1 para adição e subtração, 2 para multiplicação e divisão, e 3 para exponenciação.

MÉTODO isMatchingDelimiter(char open, char close):

Este método verifica se um par de delimitadores de abertura e fechamento corresponde corretamente (por exemplo, (e), [e], { e }). Ele retorna true se os delimitadores correspondem, e false caso contrário.

MÉTODO updateMaxPilhaSize(int size):

O método atualiza o tamanho máximo da pilha (maxPilhaSize) se o tamanho atual fornecido for maior que o valor armazenado anteriormente. Isso ajuda a rastrear o tamanho máximo da pilha durante o cálculo.

MÉTODO getMaxPilhaSize():

Este método retorna o tamanho máximo atingido pela pilha durante os cálculos de expressões. Ele fornece o valor atual de maxPilhaSize.

MÉTODO resetMaxPilhaSize():

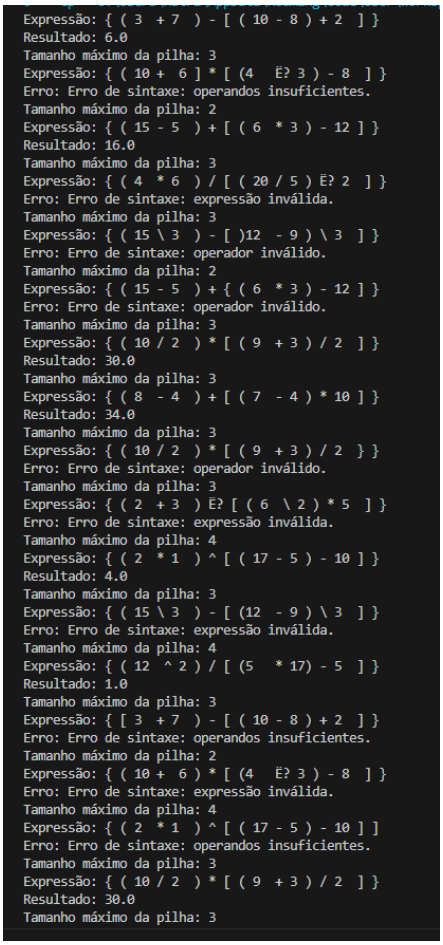
O método reseta o tamanho máximo da pilha para zero, redefinindo o valor de maxPilhaSize. Isso é útil para iniciar a medição do tamanho da pilha do zero para uma nova expressão.

MÉTODO processarArquivo(String caminhoArquivo):

Este método processa um arquivo contendo expressões aritméticas, lendo cada linha, avaliando a expressão e imprimindo os resultados ou os erros de sintaxe encontrados. Além disso, ele imprime o tamanho máximo da pilha atingido durante o cálculo de cada expressão e reseta o valor de maxPilhaSize após cada expressão ser processada.

RESULTADO:

Como podemos ver no print do resultado do terminal logo ao lado, a solução proposta foi capaz de ler o arquivo, calcular as expressões e identificar os erros nas expressões que não estavam corretamente elaboradas. Após executar, a classe App gera uma Calculadora e processa o arquivo expressões.txt, calculando todas as presentes dentro do arquivo.



CONCLUSÃO:

Apesar dos desafios enfrentados, principalmente no desenvolvimento dos métodos processarOperador() e processarArquivo(), a classe Calculadora oferece uma solução robusta e eficiente para o processamento de expressões aritméticas, proporcionando resultados precisos e gerenciando adequadamente casos de erro de sintaxe durante a avaliação. A complexidade da solução proposta, medida em notação O é executada em tempo constante O(1), graças ao uso de estruturas de pilha bem otimizadas.