Relatório: Trabalho 2 - Algoritmos e Estruturas de Dados I

- Introdução:

O trabalho tem objetivo de implementar uma calculadora usando uma pilha com estrutura encadeada, que calcule o resultado de expressões com chaves, colchetes e parênteses. A solução do trabalho consiste na implementação de um TAD pilha com encadeamento simples e uma classe Calculadora com 3 métodos.

Ela contém o método “verificaExpressao”, que analisa a sintaxe da expressão, considerando apenas as chaves, colchetes e parênteses. Caso a expressão esteja errada, retorna o erro e sua causa, se estiver certa retorna uma mensagem de confirmação. A Calculadora também possui o método “calcula” que calcula o resultado final da expressão caso ela não tenha erro de sintaxe. E por fim o método “operacao” que identifica e realiza as operações (soma, subtração, multiplicação, divisão e potência) entre dois operandos, sendo utilizado no “calcula”.

- Descrição do algoritmo:

método *verificaExpressao*
//verifica a sintaxe da expressão
//recebe uma lista de String por parâmetro
//retorna uma String indicando o erro ou expressão correta
limpar pilha
para cada string do array **fazer**
    se string for igual a “{” ou “[” ou “(“ **então**
        adicionar string na pilha
    senão
        se string for igual a “}” ou “]” ou “)” **então**
            se pilha vazia
                retorna “Pilha vazia”
        retira string da pilha e armazena em aux
        se aux for igual a “{“ e a string não for igual a ”}”
            retorna “Erro de sintaxe: string no lugar de }”
        se aux for igual a “[“ e a string não for igual a ”]”
            retorna “Erro de sintaxe: string no lugar de]”
        se aux for igual a “(“ e a string não for igual a ”)”
            retorna “Erro de sintaxe: string no lugar de)”
se pilha vazia
    retorna “Sintaxe correta”
senão
    retorna “Erro na expressão”

método *calcula*
//calcula o resultado da expressão
//recebe uma lista de string por parâmetro
//retorna o resultado da expressão em Double
se mensagem for igual “Sintaxe correta”
    limpa a pilha
    para cada string do array **fazer**

se posição i do array não for igual a “}” e “]” e “)”
adiciona posição i na pilha
incrementa o count
se count for maior que o countMax
countMax recebe count
senão
retira string da pilha e armazena em aux
decrementa o count
op2 recebe aux convertido para double
se aux não for numérico
gera mensagem de erro
retorna null
retira string da pilha e armazena em operador
decrementa o count
se operador não for igual a “+” e “-” e “*” e “/” e “^”
gera mensagem de erro
retorna null
retira string da pilha e armazena em aux
decrementa o count
op1 recebe aux convertido para double
se aux não for numérico
gera mensagem de erro
retorna null
retira string da pilha e armazena em aux
decrementa o count
resultado recebe operacao passando op1, op2 e operador
result recebe resultado convertido para string
adiciona result na pilha
incrementa o count
retorna resultado
senão
retorna null

método *operacao*
//recebe por parâmetro op1(double), op2(double) e operador(string)
//retorna a operação entre os dois operandos (double)
se operador for igual a “+”
retorna op1 + op2
se operador for igual a “-”
retorna op1 - op2
se operador for igual a “*”
retorna op1 * op2
se operador for igual a “/”
retorna op1 / op2
senão

intOp1 recebe op1 convertido para inteiro
intOp2 recebe op2 convertido para inteiro
retorna intOp1 ^ intOp2

- Conclusão:

As principais dificuldades encontradas no desenvolvimento do trabalho foi identificar os erros de sintaxe detalhados, acredito que os erros encontrados estão corretos mas não especificam exatamente o que está acontecendo na expressão. Houve dificuldade também para gerar as exceções no método *calcula* quando faltava um operando na expressão por exemplo precisava verificar se a string era um numérico para poder transformar em double. Outra dificuldade encontrada foi fazer as conversões de string para double, de double para string e principalmente de double para int. A complexidade da solução proposta em notação O é O(n^2), visto que o método *verificaExpressao* é O(n^2), o *calcula* O(n^2) e o *operacao* O(1).