Universidade Federal de Mato Grosso do Sul
Campus de Três Lagoas
Bacharelado em Sistemas de Informação
Algoritmos e Programação II

Atividade Prática I

Notação Infixa - Posfixa

1 Descrição

As expressões aritméticas, usualmente, são escritas na notação *infixa*, na qual os operadores são escritos entre os operandos. Outra notação frequentemente usada é a notação *posfixa*, na qual os operadores são escritos depois dos operandos. Essa notação é conhecida também por **notação polonesa**. A notação *posfixa* é mais econômica, pois dispensa o uso de parênteses para indicar a prioridade das operações. A seguir são apresentados alguns exemplos de expressões nas duas notações e o respectivos resultados.

Infixa	Posfixa	Resultado
(2+3*4)	2 3 4 * +	14
(2 *(3+(4*(5+(6*(7+8))))))	2 3 4 5 6 7 8+*+*+*	766
30 + 40 + 10 *3 - 15 * 200 * 4	30 40 + 10 3 * + 15 200 * 4 * -	-11900

Observe que os operandos (números da expressão) aparecem na mesma ordem na expressão *infixa* e na correspondente expressão *posfixa*.

Dados n expressões aritméticas, $1 \leq n \leq 100$, em notação *infixa*, o objetivo deste trabalho é desenvolver um programa, utilizando a linguagem de programação C, para converter essas expressões em notação *posfixa* e calcular o valor resultante da expressão.

Os operadores que serão utilizados na expressão são apresentados na Tabela 1, com suas respectivas prioridades de cálculo. As operações utilizadas nesse trabalho são operações sobre números inteiros. Devido a isso, os operandos presentes nas expressões serão números inteiros positivos, representados por uma cadeia de caracteres, com um ou mais dígitos.

Operador	Prioridade	Operação
$\wedge$	1a.	exponenciação
$*$, $/$	2a.	multiplicação, divisão
$+$, $-$	3a.	adição, subtração

Tabela 1: Operadores e prioridades de cálculo.

2 Funcionamento

Os dados de entrada para o programa estarão em um arquivo texto com a extensão "`.IN`", que contém na primeira linha um número inteiro n, $1 \leq n \leq 100$, e na sequência, n expressões na notação *infixa*, uma em cada linha do arquivo. Considere que todas as expressões do arquivo de entrada estão corretas. O formato genérico do arquivo de entrada é apresentado na Figura 1a e em seguida um exemplo com valores reais na Figura 1b.

O seu programa deve gerar um arquivo de saída com o mesmo nome do arquivo de entrada, porém com a extensão "`.OUT`". O arquivo de saída deve conter na primeira linha o número de expressões. Nas n-ésimas linhas seguintes, serão escritas as expressões na notação *posfixa*, uma em cada linha, de acordo com a ordem que foram fornecidas no arquivo de entrada. Além disso, cada expressão na notação *posfixa* deverá ser seguida pelo caractere ';', um espaço em branco, e o resultado calculado para a expressão.

```
N
expressão1
expressão2
expressão3
...
expressãoN
```

(a) Formato genérico

```
7
(2+3*4)
(2*(3+4)/5-6)
(2 *( 3+(4*(5+(6*(7+8))))))
30 + 40 + 10 *3 - 15 * 200 * 4
7-3 + 2048/2^10
(9 + 4) * (6 /2)
2^5-50+4
```

(b) Exemplo com valores reais

Figura 1: Arquivo de Entrada

Uma expressão na notação *infixa* é uma cadeia de caracteres com, no máximo, 300 caracteres. Cada expressão na notação *infixa* pode conter os operadores citados na Tabela 1, parênteses, números formados por um ou mais dígitos e espaços em branco. Neste trabalho, considere que todas as expressões do arquivo de entrada estão bem formadas. No arquivo de saída, todos os operandos e operadores devem estar separados por um espaço em branco. As Figuras 2a e 2b apresentam, respectivamente, o formato genérico e um exemplo com valores reais para o arquivo de saída. O arquivo de saída da Figura 2b foi gerado para o arquivo de entrada da Figura 1b.

```
N
expressão1_POSFIXA; resultado1
expressão2_POSFIXA; resultado2
expressão3_POSFIXA; resultado3
...
expressãoN_POSFIXA; resultadoN
```

(a) Formato genérico

```
7
2 3 4 * +; 14
2 3 4 + * 5 / 6 -; -4
2 3 4 5 6 7 8 + * + * + *; 766
30 40 + 10 3 * + 15 200 * 4 *; -11900
7 3 - 2048 2 10^/+; 6
9 4 + 6 2 /*; 39
2 5 ^ 50 - 4 +; -14
```

(b) Exemplo com valores reais.

Figura 2: Arquivo de Saída

Os formatos dos dados no arquivo são muito importantes. Os trabalhos que não respeitarem os padrões definidos receberão nota ZERO.

3 Requisitos

3.1 Codificação e Execução

O código-fonte final entregue, será compilado com os seguintes parâmetros:

```
gcc trab1.c -o trab1.exe -ansi -pedantic -Wall
```

Caso o código-fonte não compile com este comando, o trabalho não será considerado e receberá nota ZERO. É permitido o uso das seguintes bibliotecas:

- `stdio.h`
- `stdlib.h`
- `string.h`
- `ctype.h`

É proibido o uso de qualquer outra biblioteca. Caso utilize qualquer outra biblioteca, além das citadas, o trabalho não será considerado e receberá nota ZERO.

3.2 Estruturas de dados

O programa deve utilizar `pilhas` e não pode conter variáveis globais. Além disso, o programa deve ser bem comentado e deve utilizar a modularização de modo eficiente.

4 Formação dos grupos e datas

O trabalho pode ser realizado em grupos com no máximo dois integrantes (duplas). Cada grupo tem a responsabilidade sobre as cópias do trabalho, mesmo que parciais. É saudável discutir e consultar colegas de outros grupos, mas sem o compartilhamento do código fonte. Os trabalhos serão submetidos a plataformas de detecção de plágio e, os trabalhos considerados plagiados, mesmo que parcialmente, receberão nota ZERO. Ética e respeito ao direito autoral são valores cultivados em nossa profissão.

O trabalho deverá ser entregue no Ambiente Virtual de Aprendizagem da UFMS (AVA) **até as 23:59 do dia 18/10/2020**.

5 Critérios de Avaliação

Os critérios de avaliação do trabalho serão divididos entre corretude, eficiência, documentação e modularização do programa.