UDF Centro Universitário	**Disciplina: Estrutura de Dados**	**Turno: Noturno**
	Projeto 2 **Pilhas**	**Turma: N1**

Avaliação de Expressões

Estamos tão acostumados a digitar expressões e receber resultados que raramente paramos para pensar sobre o que deve acontecer dentro do computador a partir do momento em que fornecemos uma expressão até o momento em que recebemos a resposta.

As operações são efetuadas de acordo com a ordem determinada pelas regras usuais da matemática (prioridades), ou seja, em primeiro lugar as multiplicações e divisões e, posteriormente, as somas e subtrações, sendo que os parênteses podem alterar a ordem de prioridade entre as operações.

Existem duas dificuldades ao avaliar uma expressão:

1 - A existência de prioridades diferentes para os operadores, que não nos permite efetuá-los na ordem em que encontramos na expressão;

2 - A existência de parênteses, que alteram a prioridade das operações.

Felizmente, um lógico polonês chamado Jan Lukasiewicz conseguiu criar uma representação para expressões em que não existem prioridades e nem a necessidade do uso dos parênteses.

Habitualmente, nós usamos a representação na qual o operador fica no meio dos operandos, mas existem outras possibilidades.

- **Infixa:** operador aparece entre os operandos (A + B)
- **Pré-Fixa:** operador aparece antes dos operandos (+AB)
- **Pós-Fixa:** operador aparece após os operandos (AB +)

A forma pré-fixa é chamada Notação Polonesa e a pós-fixa, Notação Polonesa Reversa (NPR). A Notação Polonesa Reversa tem se mostrado mais eficiente no uso para a avaliação de expressões e estudaremos mais profundamente esse caso.

A vantagem do uso dessa notação é que, como o operador aparece imediatamente após os operandos que deve operar, a ordem em que aparecem é a ordem em que deve ser efetuada a operação, sendo desnecessário o uso de parênteses ou regras de prioridade. Veja, na Tabela 1, alguns exemplos de conversão da notação infixa para notação polonesa reversa.

Tabela 1 – Exemplos de Conversão para NPR	
Notação Infixa	**NPR**
A + B * C	A B C * +
A * (B + C)	A B C + *
(A + B) / (C - D)	A B + C D - /
(A + B) / (C - D) * E	A B + C D - / E *

Para converter uma expressão infixa para NPR, usamos o algoritmo abaixo:

1. Parentetizar completamente a expressão (definir a ordem de avaliação).
2. Varrer a expressão da esquerda para a direita e, para cada símbolo:
3. Se for parêntese de abertura, ignorar;
4. Se for operando, copiar direto para a saída;
5. Se for operador, empilhá-lo;
6. Se for parêntese de fechamento, copiar para a saída o último operador empilhado.

Para exemplificar a aplicação do algoritmo, veja, na Tabela 2, a seguir, como usamos uma pilha para converter a expressão A + B * C - D em NPR. Note que o primeiro passo do algoritmo é parentetizar a expressão, ou seja, colocar parênteses para definir as prioridades. Portanto a expressão a ser analisada a partir do passo 2 é ((A + (B * C)) - D)

Tabela 2 – Aplicação do Algoritmo para conversão em NPR		
Simbolo	**Pilha**	**Saída**
(	P: [ ]	
(	P: [ ]	
A	P: [ ]	A
+	P: [+]	A
(	P: [+]	A

B	P: [+]	A B
*	P: [+, *]	A B
C	P: [+, *]	A B C
)	P: [+]	A B C *
)	P: [ ]	A B C * +
-	P: [-]	A B C * +
D	P: [-]	A B C * + D
)	P: [ ]	A B C * + D -

O problema do algoritmo apresentado é que precisamos parentetizar a expressão antes de colocar no programa, porém, na prática, o usuário não utiliza dessa forma a expressão. A prioridade deve ser dada pelo usuário, para que o programa saiba por onde começar. Para resolver esse problema, usaremos uma função chamada "prio", que dá a prioridade dos operadores da expressão. A função é apresentada na Figura 1 a seguir:

Figura 1 – Função prio para definir prioridades das operações

```
public static int prio(char op) {
    int resp = 0;
    switch (op) {
        case '(' : resp = 1; break;
        case '+' : resp = 2; break;
        case '-' : resp = 2; break;
        case '*' : resp = 3; break;
        case '/' : resp = 3; break;
    }
    return resp;
}
```

Portanto, o novo algoritmo pode ser descrito nos seguintes passos:

1. Inicie com uma pilha vazia.
2. Realize uma varredura na expressão infixa, copiando todos os identificadores encontrados diretamente para a saída.
   a. Ao encontrar um operador:
      I. enquanto a pilha não estiver vazia e houver, no seu topo, um operador com prioridade maior ou igual ao encontrado, desempilhe o operador e coloque-o na saída.
      II. empilhe o operador encontrado.
   b. Ao encontrar um parêntese de abertura, empilhe-o.
   c. Ao encontrar um parêntese de fechamento, remova um símbolo da pilha e copie-o na saída até que seja desempilhado o parêntese de abertura correspondente.
3. Ao final da varredura, esvazie a pilha e copie para a saída os símbolos removidos.

Percorrendo qualquer expressão em notação polonesa reversa, da esquerda para a direita, ao encontrarmos um operador, sabemos que deve operar os dois últimos valores pelos quais passamos. Percebemos, novamente, a ideia de que "os últimos serão os primeiros processados" e, novamente, a aplicação de pilhas.

Vejamos um exemplo na expressão em NPR:

AB+CD-/E*

Vamos atribuir valores numéricos às variáveis da expressão a ser avaliada:

A= 7; **B=** 3; **C=** 6; **D=** 4; **E=** 9.

Agora, seguiremos o algoritmo a seguir:

1. Iniciamos com uma pilha vazia.
2. Varremos a expressão da esquerda para a direita e para cada elemento encontrado:
3. Se for operando, empilhar;
4. Se for operador, desempilhar os dois últimos valores, efetuar a operação com eles e empilhar de volta o resultado obtido.

No final do processo, o resultado da avaliação estará no topo da pilha. Veja a aplicação do Algoritmo na Tabela 3 a seguir:

Tabela 3 – Aplicação do Algoritmo para Avaliar a Expressão

Elemento	Ação	Pilha
A	Empilha valor de A	P:[7] B
	Empilha valor de B	P:[7,3]
+	Desempilha um y=3 Desempilha outro x=7 Empilha resposta x+y=10	P:[10]
C	Empilha valor de C	P:[10,6]
D	Empilha valor de D	P:[10,6,4]
-	Desempilha um y=4 Desempilha outro x=6 Empilha resposta x-y=2	P:[10,2]
/	Desempilha um y=2 Desempilha outro x=10 Empilha resposta x/y=5	P:[5]
E	Empilha valor de E	P:[5,9]
*	Desempilha um y=9 Desempilha outro x=5 Empilha resposta x*y=45	P:[45]

Uma alternativa para dar valores aos operandos é criar um vetor de 26 posições (máximo de letras que pode ser utilizado) e, para cada letra encontrada na expressão, perguntar o seu valor.

I) Considerando o contexto descrito acima desenvolva uma aplicação capaz de processar e avaliar expressões aritméticas aplicando a estrutura de dados Pilha apresentada nas aulas anteriores e a NPR

Os códigos desenvolvidos deverão ser encaminhados por e-mail para antonio.lima@udf.edu.br (até 02/10/2019)