Universidade de Brasília
Departamento de Ciência da Computação

7 de maio de 2018
(2018/1)

Estruturas de Dados – Turma A – Trabalho 1
Professor: Marcos F. Caetano <mfcaetano@unb.br>
Monitor: Verificar lista no Moodle da disciplina.
Resumo: Avaliação de Expressões Aritméticas (forma posfixa) e Calculadora

1 Introdução

Logo no início do ciclo escolar o estudante é apresentado ao maravilhoso mundo da matemática, assim que domina as operações básicas o aluno começa a estudar expressões numéricas. A forma mais comum de se representar expressões numéricas é a forma infixada, que é quando as operações estão entre dois operandos (Ex. 7+5), porém tal notação quando em expressões muito grandes pode ser um pouco confusa, isso se deve principalmente a dupla interpretação que algumas precedências de sinais podem possuir. A fim de evitar tais confusões usamos parênteses para indicarmos quais conjuntos de operações devem ser realizadas primeiro.

Mas imagine que existisse uma forma de representar expressões numéricas complexas sem o uso de parênteses e sem a possibilidade de haver confusão quanto a ordem das operações?

Existem duas formas de representações alternativas que solucionam este problema. Uma delas é conhecida como Notação Polonesa Reversa(RPN) também chamada de notação pósfixada, inventada pelo filósofo e cientista da computação australiano Charles Hamblin em meados dos anos 1950. Ela deriva da segunda forma de representação a notação polonesa ou também notação préfixada, introduzida em 1920 pelo matemático polonês Jan Łukasiewicz. Hamblin apresentou seu trabalho numa conferência em Junho de 1957, e o publicou em 1957 e 1962.

2 Descrição

2.1 Problema

Os prefixos “pre”, “pos” e “in” referem-se à posição relativa do operador em relação aos dois operandos da expressão. O que dificulta a avaliação de expressões na forma infixa, por um sistema computacional, é a existência de prioridades entre os operadores, como por exemplo A + B * C, e a existência de parênteses que modificam esta prioridade, como por exemplo (A + B) * C. Estes problemas não são encontrados nas outras representações (prefixa e posfixa), uma vez que a ordem de execução das operações é estabelecida apenas pela ordem dos operandos e operadores na expressão.

- Notação prefixa (notação polonesa): + A B
- Notação infixa : A + B
- Notação posfixa (notação polonesa reversa): A B +

2.2 Programa

Neste trabalho deverá ser desenvolvido um programa com dois modos de operação descritos a seguir:

- Resolução de Expressão

  - O programa receberá como entrada uma expressão na forma (usual) infixa
  - Primeiramente, o programa deverá analisar a validade da expressão infixa
  - Caso seja válida, o programa transformará a expressão da forma infixa para a forma posfixa
  - Finalmente, o programa calculará o resultado da expressão

- Calculadora
  - O programa aguarda pela entrada dos números ou operações
  - Quando um número é digitado ele é empilhado, a pilha deve ser sempre exibida em tela
  - Quando uma operação é digitada os operandos são desempilhados a operação realizada e seu resultado empilhado novamente
  - Caso a operação seja inválida ou a quantidade de operandos insuficiente uma mensagem de erro deve ser impressa

Um menu deverá ser desenvolvido apresentando as duas opções de operação ao usuário. Estruturas de repetição devem ser utilizadas afim de permitir ao usuário navegar entre os dois modos de operação sem que tenha que finalizar o programa.

2.3 Resolução de Expressão

2.3.1 Entrada e saída de dados

A expressão infixa de entrada pode ser recebida através da entrada padrão (console) através do comando *scanf* da linguagem de programação C. Por fim, o resultado final também pode ser impresso na saída padrão através do comando *printf*.

- **OBS:** As expressões poderão ser formadas por números do tipo ponto flutuante (*float*), contendo várias casas decimais antes e depois da vírgula. Serão fornecidos somente números positivos;

2.3.2 Validação da expressão

O segundo passo é analisar os delimitadores da equação (rastrear escopo).
Exemplo de expressão válida: ((A + B) * C + (D + E)/(F + G) + H) / I
Exemplo de expressão inválida: (A + B) * C + (D + E)/(F + G) + H) / I
Para isso, utilizaremos o seguinte algoritmo.
Obs.: Cuidado! A expressão deve ser válida mesmo que não tenha nenhum delimitador de equações. Por exemplo a expressão A+B*C é válida.

Algoritmo. Percorrer a expressão da esquerda para direita da seguinte forma:

- Inicie uma pilha vazia
- Se um inicializador de escopo for encontrado, o mesmo é empilhado
- Se um finalizar de escopo for encontrado, a pilha é verificada
  - Se estiver vazia, então a equação não é válida
  - Se não, desempilhar e comparar com o finalizador
- Ao final, a pilha deve estar vazia
- Imprima em tela se a expressão é válida ou não
- Se expressão for inválida finalize a execução e volte ao menu. Se for válida prossiga para próximo passo

2.3.3 Transformação da forma infixa para posfixa

O passo seguinte é transformar a expressão infixa em posfixa.
Exemplo de expressão infixa: A * B + C - (D/E + F)
Exemplo de expressão posfixa: A B * C + D E / F + -
Para esta transformação, utilizaremos o seguinte algoritmo.

Algoritmo. O Processo de conversão infixa para posfixa pode ser efetuado da seguinte forma:

- Inicie uma pilha vazia
- Realize uma varredura na expressão infixa, copiando todos os operandos encontrados diretamente para a expressão de saída
  - Ao encontrar um operador:
    - Enquanto a pilha não estiver vazia e houver no seu topo um operador com prioridade maior ou igual ao encontrado (trabalharemos apenas com os operadores +, -, /, *), desempilhe o operador e copie-o na saída
    - Empilhe o operador encontrado
  - Ao encontrar um parêntese de abertura, empilhe-o
  - 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
- Ao final da varredura, esvazie a pilha, movendo os símbolos desempilhados para a saída
- Imprima em tela o resultado final da transformação

Veja o funcionamento da versão final do algoritmo na conversão da expressão A*(B+C)/D.

2.3.4 Avaliação da expressão

Finalmente, a última etapa consiste na avaliação da expressão posfixa obtida anteriormente. Para isso, usaremos o seguinte algoritmo:

Algoritmo. Os componentes da expressão são processados da esquerda para a direita como a seguir:

- Inicie uma pilha vazia
- Se o próximo componente da expressão é um operando, o valor do componente é colocado na pilha
- Se o próximo componente da expressão é um operador, então os seus operandos estão na pilha. O número requerido de operandos é retirado da pilha, a operação específica é realizada, e o resultado é armazenado de volta na pilha.
- Ao final, a pilha conterá um único dado que é o valor final da expressão

Simule o funcionamento deste algoritmo com a expressão posfixa obtida da expressão infixa A*(B+C)/D.

DICA: Teste a implementação desenvolvida com a seguinte expressão: 3,25-1*2+3,25-1. O resultado correto é 3,50.

2.3.5 Fluxograma

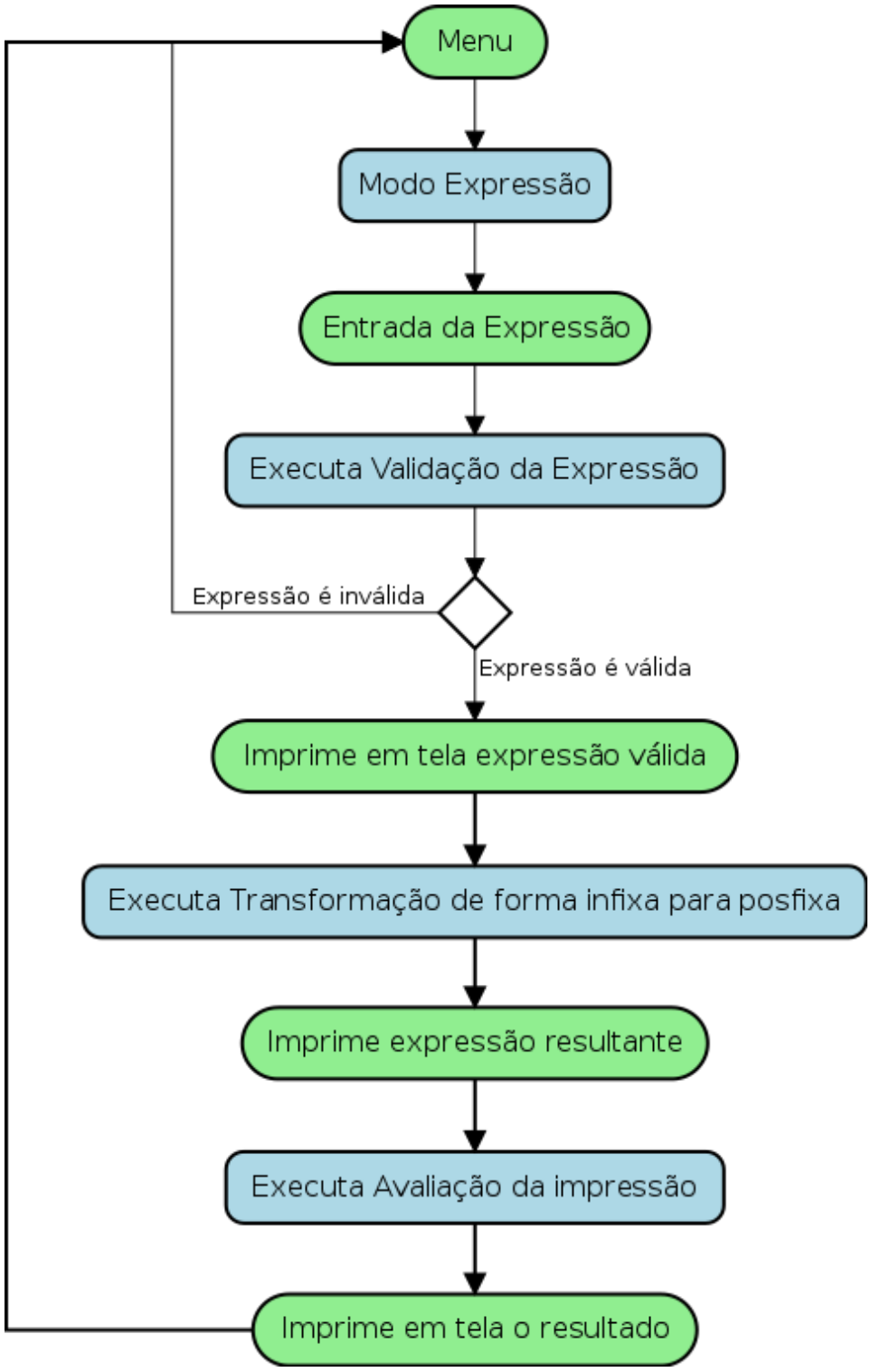

Figura 1: Fluxograma

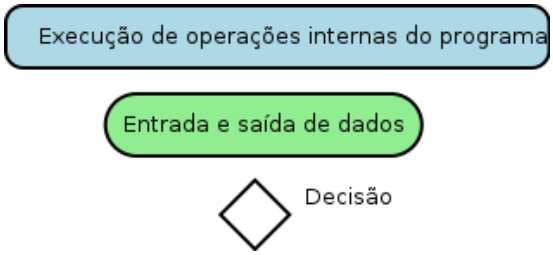

Figura 2: Legenda

2.4 Calculadora

2.4.1 Funcionamento

- Inicie uma pilha vazia e aguarde pela entrada dos dados.

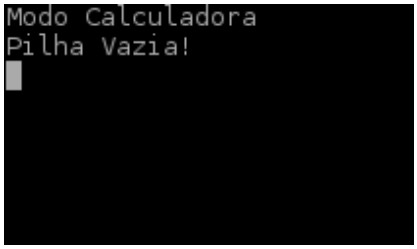

Figura 3: Pilha vazia

- Assim que um número for digitado empilhe, lembrando que a pilha deve ser toda exibida em tela.

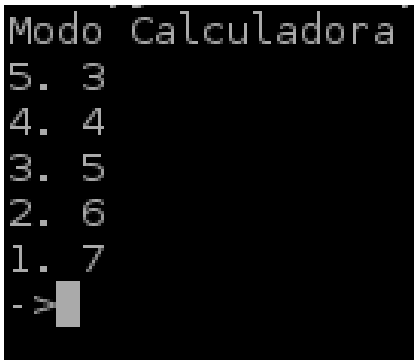

Figura 4: Pilha exemplo

- Assim que uma operação é executada os operandos devem ser desempilhados, em seguida avaliada a operação e o resultado empilhado novamente. No exemplo a operação de adição foi efetuada na pilha que está mostrada na figura 4.

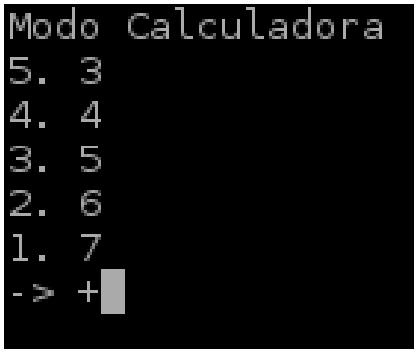

Figura 5: Exemplo operação

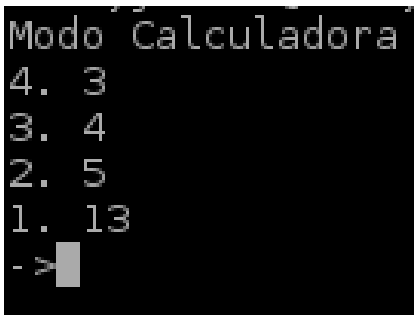

Figura 6: Exemplo operação

- Quando a quantidade de operandos for insuficiente uma mensagem de erro deve ser exibida na tela.

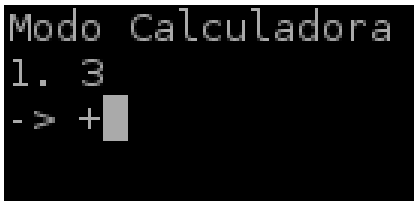

Figura 7: Exemplo erro operação

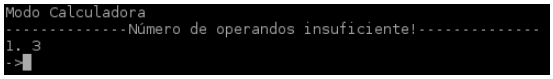

Figura 8: Exemplo erro

2.4.2 Operações

- **Básicas**
  - Adição(+)
  - Subtração(-)
  - Multiplicação(*)
  - Divisão(/)
- **Especiais**

– Repetição de operação(!)

O operador **!** somente será usado associado a algum outro operador básico, ele deve repetir a operação básica associada até que a pilha esteja vazia. Exemplo:

Pilha Antes	Operadores	Pilha Depois
1,25 2,25 3,25 4,25	+!	11,00
1,00 2,00 3,00 4,00	*!	24,00
1,00 2,00 3,00 4,00	-!	-2,00

– Cópia de elemento (c) Quando utilizado o operador desempilha um número (N) da pilha, este número informa o número de cópias. Em seguida a pilha é novamente desempilhada este segundo número (K) é o dado que será copiado N vezes na pilha. Exemplo:

Pilha Antes	Operadores	Pilha Depois
 1,00 3,00	 c	1,00 1,00 1,00
  2,00 4,00	 c	2,00 2,00 2,00 2,00

Obs.: Lembrando que a representação da pilha está invertida nos exemplos acima, sendo assim o topo da pilha é o número que aparece mais abaixo e a base o número que aparece mais acima.

- **OBS:** Os valores aceitos pela calculadora poderão ser formados por números do tipo ponto flutuante (*float*), contendo várias casas decimais antes e depois da vírgula;

3 Relatório

Um relatório final do projeto deve ser apresentado. Este relatório deve conter:

- Breve descrição do funcionamento da Notação Polonesa Reversa;
- Documento apresentando a arquitetura do sistema desenvolvido;
- Doxygen de todo o código produzido;
- *Screenshots* e explicação do funcionamento de todas as funcionalidades implementadas.
- Smoke Test - Entrega do documento descrevendo como executar os testes ou com instruções para execução dos teste automatizados. Ver documento auxiliar postado no moodle.

4 Avaliação

O código e o relatório deverão ser submetidos no Moodle até o dia **27/05 as 23:55h**; Não serão aceitos trabalho enviados fora do prazo;

A avaliação consiste em 2 etapas:

- Código e funcionamento do projeto (80%).
- Nota do Relatório e Smoke Test (20%).

A descrição detalhada da correção e nota de cada um dos pontos estará a disposição dos alunos na divulgação dos resultados.

5 Observações

As seguintes observações deverão ser consideradas pelos alunos que forem fazer o trabalho:

- O trabalho deve possuir arquivo Leia-me com instruções de compilação, execução, informação sobre ambiente de programação(versão do compilador, SO utilizado, versão da linguagem, bibliotecas utilizadas, etc);
- O trabalho deve compilar no GCC e rodar **OBRIGATORIAMENTE** na plataforma GNU/Linux;
- O programa deve ser programado na linguagem C (C99);
- Não é permitido o uso de funções para a conversão entre tipos (`atoll()`, por exemplo);
- É responsabilidade do aluno verificar se o código desenvolvido compila e executa corretamente na plataforma GNU/Linux;
- código que não compila é ZERO!
- É responsabilidade do aluno se certificar que o arquivo, no formato ZIP, contendo todo projeto (código e relatório) foi corretamente armazenado no Moodle. Ou seja, após a submissão, faça o download do seu projeto e descompacte o arquivo para ter certeza que ele está correto;
- O relatório deverá ser entregue **OBRIGATORIAMENTE** no formato de arquivo PDF;
- Códigos copiados (entre alunos ou retirados da Internet) serão considerados "cola"e todos os alunos envolvidos ganharão nota zero;
- Dúvidas sobre o trabalho deverão ser tiradas **NO FÓRUM DE DÚVIDAS DO AMBIENTE APRENDER**. Desta forma, dúvidas comuns e esclarecimentos poderão ser respondidos uma única vez para toda a turma.
- O prazo definido para submissão do trabalho é FIRME e já contempla possíveis indisponibilidades do Moodle. Não deixem para submeter o seu projeto na última hora. O sistema está configurado para vocês poderem fazer múltiplos uploads.
  - E-mails contendo o projeto enviados para o professor ou os monitores serão automaticamente descartados.