Projeto IEMANJA

O Identificador de Expressões Malformadas e Avaliador de Notações Julgadas Adequadas (IEMANJA) será o projeto final da disciplina de Estruturas de Dados Básicas I. Ele valerá 10,0 pontos, ou seja, 100% da nota da Unidade III.

1. Objetivos

O objetivo desse projeto é verificar o domínio do conteúdo de *pilhas*, *filas* e *deques* por parte dos alunos da disciplina. Para isso, essas estruturas serão utilizadas na implementação de um identificador/avaliador de expressões matemáticas simples, o qual analisará expressões passadas, indicando se elas são válidas ou não e divulgando o resultado daquelas que forem consideradas corretas.

2. Ações Realizadas

O IEMANJA verificará apenas expressões simples, pois o foco principal do projeto é o uso das estruturas estudadas em sala. Dessa forma,

As expressões de entrada passarão por um conjunto de testes para avaliar sua validade. Para cada etapa, o identificador/avaliador deverá indicar se a expressão condiz com as regras estabelecidas e, caso contrário, divulga uma mensagem informando o problema encontrado. As etapas de avaliação serão:

1) Verificação de presença de caracteres inválidos;
2) Verificação da formação dos números;
3) Verificação do balanceamento de parênteses; e
4) Verificação de validade do formato da expressão infixa.

Caso se detecte alguma falha durante algum dos processos de validação citados acima, a expressão em questão será julgada inadequada e não será processada. O programa passará então para a próxima expressão na fila de validação.

Caso uma expressão seja julgada correta, dar-se-á início ao processo de tradução da expressão, a qual deverá ser convertida para uma representação pós-fixa, também conhecida como Notação Polonesa Reversa (NPR). Após a geração da versão pós-fixada da expressão inicial, a avaliação será realizada, indicando o resultado do cálculo realizado.

3. Entrada de dados

A entrada de dados consistirá de um arquivo de texto simples, o qual conterá diversas linhas. Em cada linha deverá ser colocada uma expressão para análise. O programa continuará a ler o arquivo de entrada até que tenha exaurido todas as suas linhas.

As linhas não possuem um valor máximo de caracteres determinado. Logo, o IEMANJA deverá estar preparado para lidar com qualquer tamanho de expressão, capturando todos os caracteres disponíveis até ser encontrada a indicação de nova linha. Sobre este tópico, vale ainda salientar que existem diferenças nas leituras de arquivos em diferentes sistemas operacionais. No Windows, as linhas geralmente são terminadas com a combinação CRLF (*Carriage Return/Line Feed*), o que em C/C++ seria traduzido para "**`\r\n`**". Já no Linux, o fim das linhas é sinalizado apenas pelo LF.

4. Verificações

Antes de se analisar a expressão passada como entrada, é necessário que algumas verificações básicas sejam realizadas para se garantir de que se trata de uma expressão válida para avaliação posterior.

4.1. Verificação de presença de caracteres inválidos

As expressões avaliadas deverão conter apenas os seguintes caracteres:

- Números inteiros ou reais;
- Operadores matemáticos básicos;

    - Adição (+)
    - Subtração (-)
    - Multiplicação (*)
    - Divisão (/)
    - Potenciação (^)
    - Operadores unários (+, -)
- Parênteses; e
- Espaços em branco.

Caso a expressão apresente algum caractere diferente dos supracitados, o programa deverá apresentar o código de erro 1. Além disso, deverá apresentar o caractere inválido e o seu índice na expressão.

4.2. Verificação da formação dos números

Como o programa receberá cadeias representando expressões, será necessário extrair os números presentes em meio ao texto para que se possa convertê-los posteriormente. Levando em consideração que números podem ser representados de diversas formas distintas, a seguir definiremos o que será considerado como número para o IEMANJA:

- Números inteiros: sequência de dígitos.
- Números reais: sequência de dígitos com um ponto separador entre eles. Cada número real deverá começar e terminar com um dígito, possuindo um ponto separador em alguma posição entre eles. Ou seja, variações de representação como “1.” ou “.25” não serão consideradas válidas. Além disso, dessa definição podemos constatar que cada número real terá no mínimo 3 caracteres.

Caso a expressão analisada apresente algum problema na formação dos números, o programa deverá apresentar o código de erro 2. Assim como no primeiro caso, deverá apresentar o índice de início do número problemático.

4.3. Verificação do balanceamento de parênteses

Esta etapa consiste em verificar se todos os parênteses utilizados na expressão possuem um par associado. Para realizar essa ação, geralmente uma pilha é a estrutura adequada. Cada vez que se encontrar um parêntese de abertura, ele deverá ser inserido na pilha. Cada vez que se encontrar um parêntese de fechamento, um dos elementos presentes na pilha deverá ser removido.

Caso (i) durante uma operação de remoção a pilha esteja vazia; ou (ii) ao final da leitura de todos os parênteses a lista não esteja vazia, isso significa que os parênteses estão desbalanceados. Logo, o programa deverá apresentar o código de erro 3. Para este caso, não será necessário indicar um índice.

4.4. Verificação de validade do formato da expressão infixa

A última verificação a ser feita é a que analisa se a expressão passada atende a definição de uma expressão infixa, ou seja, se todos os operadores aparecem entre os operandos. Um caso especial que deverá ser trabalhado é o dos operadores unários, os quais estão relacionados apenas a um operando.

Um exemplo de expressão malformada seria a “2+5*”. Como pode ser percebido, o operador de multiplicação possui apenas um operando do lado esquerdo. Caso se encontre um problema desse tipo, o programa deverá apresentar o código de erro 4.

5. Códigos de Erro

Como visto nas seções anteriores, existem diversas formas de uma expressão ser considerada errada. Sendo assim, o programa deverá emitir códigos de erro específicos para cada situação. Além disso, uma mensagem explicativa também deve ser apresentada, de acordo com a tabela a seguir:

Código	Mensagem
Erro 1	Caractere inválido encontrado na posição **x**.
Erro 2	Número inválido encontrado a partir da posição **x**.
Erro 3	Os parênteses da expressão estão desbalanceados.
Erro 4	Expressão infixa malformada.

6. Extração dos componentes internos

Para que a expressão possa ser avaliada, seus componentes internos precisam primeiramente ser extraídos. Operadores, operandos e parênteses precisam ser separados e tratados como entidades próprias, e não como simples caracteres em um texto. Para isso, alguns passos devem ser seguidos:

1) **Extração de espaços em branco e quebras de linha**
   Embora sejam considerados válidos, esses componentes não contribuem para a avaliação da expressão. Sendo assim, precisam ser removidos

2) **Identificação de números e conseguinte extração**
   Diferentemente dos demais componentes, os números não são representados por apenas um caractere, mas sim por uma sequência. O programa deverá identificar os possíveis números e extraí-los adequadamente. Vale salientar que em se tratando de uma expressão infixa, os números deverão aparecer no início, no final ou entre operadores e parênteses.

3) **Extração de operadores e parênteses**
   Os operadores e parênteses são identificados com apenas um caractere.

4) **Inserção dos componentes internos em uma fila**
   Ao extrair cada um dos componentes anteriormente citados, eles deverão ser inseridos em uma fila, a qual será responsável por representar a expressão com seus componentes devidamente separados.

6.1. Caso especial: operadores unários

Um caso especial a ser levado em consideração é o dos operadores unários, os quais podem ser de adição ou subtração. Esses operadores serão aplicados apenas a um operando, logo geralmente não aparecem entre dois deles. Além disso, caso sejam deixados da mesma forma que estão, ocorrerão problemas durante a conversão da notação.

Caso se encontre um operador unário de adição, ele deve ser simplesmente ignorado, ou seja, não será inserido na fila que representará a expressão.

O caso do operador unário de subtração necessitará de um cuidado extra. Existem diversas formas de se lidar com essa situação, mas uma sugestão é a de ao se encontrar este operador, inserir na pilha uma multiplicação de alta precedência por -1. Ou seja, adiciona-se um símbolo que simbolizará uma multiplicação que deverá ser feita antes de qualquer outra operação. Um exemplo é utilizar um “x” para indicar essa operação. Assim, ao invés de se enfileirar o operador unário “-”, são enfileirados dois elementos: um “x” e um “-1”. Durante a avaliação, a primeira ação que será realizada é a multiplicação desses valores por -1, gerando assim sua versão negativa.

Ao final do processo de extração de componentes, o resultado será uma fila contendo cada um dos componentes devidamente separados e prontos para a etapa de conversão de notação.

7. Conversão de notação

Nessa etapa, a expressão já estará devidamente analisada e separada em componentes, restando apenas ser convertida para sua representação pós-fixa antes de ser avaliada. A conversão é realizada com auxílio de 3 estruturas: (i) uma fila de entrada; (ii) uma pilha de operadores e parênteses; e (iii) uma fila de saída.

A fila de entrada é a fila produzida após a extração dos componentes. Por sua vez, a pilha de operadores e parênteses será a responsável por armazenar provisoriamente esses componentes específicos durante a conversão. Por fim, a fila de saída armazenará a representação pós-fixa da expressão.

Uma outra informação que precisa ser levada em consideração é a prioridade dos operadores trabalhados, a qual pode ser relembrada na tabela a seguir:

Prioridade	Operações
1	( )
2	^
3	* /
4	+ -

A maior prioridade (1) é dada aos parênteses. Eles não são considerados uma operação propriamente dita, mas em uma notação infixa são responsáveis por indicarem quais trechos devem ser analisados em conjunto. Logo, possuem a mais alta precedência dentre as operações. A potenciação vem logo em seguida. Em terceiro lugar, temos a multiplicação e a divisão, as quais compartilham a prioridade. Por fim, a soma e a subtração aparecem como as operações de menor prioridade. Vale salientar que as operações de mesma prioridade são realizadas na ordem em que aparecem na expressão.

Para realizar a conversão, seguimos os seguintes passos:

1) Obtenha um componente da fila de entrada.
2) Se o componente for um operando, insira-o na fila de saída.
3) Se o componente for um parêntese de abertura, empilhe-o na pilha.
4) Se o componente for um parêntese de fechamento, desempilhe todos os operadores até encontrar um parêntese de abertura. Os operadores desempilhados devem ser inseridos na fila de saída no momento em que são retirados da pilha.
5) Se o componente for um operador, desempilhe todos os operadores cujas precedências sejam maiores ou iguais e os coloque na fila de saída. Em seguida, empilhe o operador em questão.
6) Após ter processado todos os componentes da fila de entrada, caso haja operadores restantes na pilha, eles deverão ser desempilhados e inseridos na fila de saída.

7.1. Exemplo de conversão

Tomemos como exemplo a expressão $1 + 8.2 * (15 - 5/2.5)$. Seguindo os procedimentos, teremos:

Passo 1: Análise do primeiro componente da fila de entrada. Por ser um operando, será inserido diretamente na fila de saída.

Fila de entrada:	**1**	+	8.2	*	(	15	-	5	/	2.5	)
Pilha de operadores:											
Fila de saída:											

Passo 2: Análise do segundo componente da fila de entrada. Como é um operador, verificamos a pilha de operadores em busca de operadores de maior ou igual precedência. Como a lista está vazia, então simplesmente empilhamos o operador.

Fila de entrada:		**+**	8.2	*	(	15	-	5	/	2.5	)
Pilha de operadores:											
Fila de saída:	1										

Passo 3: Análise do terceiro componente da fila de entrada. Por ser um operando, será inserido diretamente na fila de saída.

Fila de entrada:			**8.2**	*	(	15	-	5	/	2.5	)
Pilha de operadores:	+										
Fila de saída:	1										

Passo 4: Análise do quarto componente da fila de entrada. Como é um operador, verificamos a pilha de operadores em busca de operadores de maior ou igual precedência. Como o único operador da pilha é de precedência inferior, o componente é adicionado à pilha sem que outras ações sejam realizadas.

Fila de entrada:				*****	(	15	-	5	/	2.5	)
Pilha de operadores:	+										
Fila de saída:	1	8.2									

Passo 5: Análise do quinto componente da fila de entrada. Por ser um parêntese de abertura, ele é simplesmente adicionado à pilha de operadores.

Fila de entrada:					**(**	15	-	5	/	2.5	)
Pilha de operadores:	+	*									
Fila de saída:	1	8.2									

Passo 6: Análise do sexto componente da fila de entrada. Por ser um operando, será inserido diretamente na fila de saída.

Fila de entrada:						**15**	-	5	/	2.5	)
Pilha de operadores:	+	*	(								

Fila de saída:	1	8.2									

Passo 7: Análise do sétimo componente da fila de entrada. Como é um operador, verificamos a pilha de operadores em busca de operadores de maior ou igual precedência. O que encontramos é um parêntese de abertura, o qual só pode ser extraído ao se encontrar um parêntese de fechamento. Logo, iremos apenas empilhar o operador.

Fila de entrada:								-	5	/	2.5	)
Pilha de operadores:	+	*	(									
Fila de saída:	1	8.2	15									

Passo 8: Análise do oitavo componente da fila de entrada. Por ser um operando, será inserido diretamente na fila de saída.

Fila de entrada:								**5**	/	2.5	)
Pilha de operadores:	+	*	(	-							
Fila de saída:	1	8.2	15								

Passo 9: Análise do nono componente da fila de entrada. Como é um operador, verificamos a pilha de operadores em busca de operadores de maior ou igual precedência. Como o operador no topo da pilha é de precedência inferior, o componente é adicionado a ela sem que outras ações sejam realizadas.

Fila de entrada:									**/**	2.5	)
Pilha de operadores:	+	*	(	-							
Fila de saída:	1	8.2	15	5							

Passo 10: Análise do décimo componente da fila de entrada. Por ser um operando, será inserido diretamente na fila de saída.

Fila de entrada:										**2.5**	)
Pilha de operadores:	+	*	(	-	/						
Fila de saída:	1	8.2	15	5							

Passo 11: Análise do décimo primeiro componente da fila de entrada. Por ser um parêntese de fechamento, iremos desempilhar todos os operadores da pilha até encontrar um parêntese de abertura. Ao desempilhamos, os operadores serão inseridos na fila de saída.

Fila de entrada:											**)**
Pilha de operadores:	+	*	(	-	/						
Fila de saída:	1	8.2	15	5	2.5						

Passo 12: Fila de entrada está vazia. Todos os operadores restantes na pilha deverão ser inseridos na fila de saída.

Fila de entrada:											
Pilha de operadores:	+	*									
Fila de saída:	1	8.2	15	5	2.5	/	-				

Resultado da conversão: $1\ 8.2\ 15\ 5\ 2.5\ /\ -\ *\ +$.

8. Avaliação de notação pós-fixa

Após a conversão da notação infixa para a pós-fixa, a avaliação da expressão enfim poderá ser realizada. Para isso, utilizaremos a fila que contém a expressão pós-fixa e uma pilha de operandos. Essa pilha será a responsável por receber todos os operandos existentes na fila que contém a expressão pós-fixa, assim como armazenar os valores parciais dos cálculos realizados. Através da combinação dessas estruturas e dos passos a seguir, conseguiremos calcular o resultado da expressão e estar aptos a divulga-lo ao final.

Procedimentos para avaliação da expressão pós-fixa:

1) Obtenha um componente da fila que contém a expressão pós-fixa.
2) Se o componente for um operando, insira-o na pilha de operandos.
3) Se o componente for um operador, ele precisará de dois operandos. Desempilhe dois operandos presentes na pilha e realize a operação associada. Em seguida, empilhe o resultado na pilha.
4) Ao terminar de ler a expressão pós-fixa, o resultado deverá ser o único valor presente na pilha de operandos.

Para divulgar o resultado, bastará desempilhar o valor presente na lista de operandos. Esse será o resultado da expressão passada para o IEMANJA. $1\ 8.2\ 15\ 5\ 2.5\ /\ -\ *\ +$

8.1. Exemplo de avaliação

Passo 1 a 5: Análise do primeiro ao quinto componente da fila da expressão. Por serem todos operandos, serão inseridos diretamente na pilha.

Fila da expressão:	**1**	**8.2**	**15**	**5**	**2.5**	/	-	*	+
Pilha de operandos:									

Passo 6: Análise do sexto componente da fila da expressão. Como é um operador, teremos que desempilhar os dois elementos no topo da pilha, ou seja, "2.5" e "5". Note que eles serão desempilhados na ordem inversa. Isso significa que o cálculo a ser realizado será $5\ /\ 2.5$, não o contrário. O resultado dessa operação será 2, valor que será então empilhado.

Fila da expressão:						**/**	-	*	+
Pilha de operandos:	1	8.2	15	5	2.5				

Passo 7: Análise do sétimo componente da fila da expressão. Como é um operador, teremos que desempilhar os dois elementos no topo da pilha, ou seja, "2" e "15". Note que eles serão desempilhados na ordem inversa. Isso significa que o cálculo a ser realizado será $15 - 2$, não o contrário. O resultado dessa operação será 13, valor que será então empilhado.

Fila da expressão:							**-**	*	+
Pilha de operandos:	1	8.2	15	2					

Passo 8: Análise do oitavo componente da fila da expressão. Como é um operador, teremos que desempilhar os dois elementos no topo da pilha, ou seja, "13" e "8.2". Note que eles serão desempilhados na ordem inversa. Isso significa que o cálculo a ser realizado será $8.2 * 13$, embora nessa situação em específico a ordem dos operandos não influencie. O resultado dessa operação será 106.6, valor que será então empilhado.

Fila da expressão:								*****	+
Pilha de operandos:	1	8.2	13						

Passo 9: Análise do nono componente da fila da expressão. Como é um operador, teremos que desempilhar os dois elementos no topo da pilha, ou seja, "106.6" e "1". Note que eles serão desempilhados na ordem inversa. Isso significa que o cálculo a ser realizado será $106.6 + 1$, embora nessa situação em específico a ordem dos operandos não influencie. O resultado dessa operação será 107.6, valor que será então empilhado.

Fila da expressão:									**+**
Pilha de operandos:	1	106.6							

O resultado da expressão (107.6) poderá então ser desempilhado para ser apresentado ao final da avaliação da expressão.

9. Estruturas utilizadas

As seções anteriores descreveram todo o processo desde a análise da expressão inicial até o cálculo do valor que ela representa. Como pode ser visto, todas as operações podem ser realizadas utilizando as estruturas de dados *pilha* e *fila*.

Como o objetivo deste projeto é avaliar todas as estruturas utilizadas e suas diversas formas de implementação, também será possível realizar o mesmo trabalho utilizando uma *deque*. Essa estrutura consegue realizar as mesmas ações que as outras duas, podendo assim ser realizada como substituta das demais.

Entretanto, não caberá ao grupo decidir qual estrutura de dados utilizará ou como ela será implementada. Para isso, será realizado um sorteio em sala no qual será solicitado ao grupo a implementação de uma das oito seguintes configurações:

Estruturas Utilizadas	Estrutura Fundamental
Pilha e Fila	Lista Simplesmente Encadeada
	Lista Duplamente Encadeada
	Lista Circular Simplesmente Encadeada
	Lista Circular Duplamente Encadeada
Deque	Lista Simplesmente Encadeada
	Lista Duplamente Encadeada
	Lista Circular Simplesmente Encadeada
	Lista Circular Duplamente Encadeada

10. Avaliação do projeto

O projeto será avaliado de acordo com a tabela de notas a seguir. Utilize-a como base para definição de prioridades de desenvolvimento e definição de tarefas do grupo.

Tópico	Valor
Validação de expressão Recebimento das expressões em arquivo e, em caso de serem válidas, apresentar seu resultado. Caso sejam inválidas, a mensagem de erro correspondente deverá ser apresentada.	1,0
Separação da expressão em componentes Extração adequada dos componentes das expressões infixas passadas e criação da fila de entrada.	1,0
Conversão de expressão de infixa para pós-fixa Criação de nova fila, dessa vez contendo a representação da expressão em notação pós-fixa.	2,0
Cálculo do resultado Apresentação do resultado referente à expressão passada.	1,5
Implementação adequada da estrutura sorteada Implementar todos os métodos referentes à estrutura que foi sorteada (inserção, remoção, visualização de extremidades, verificação de vacuidade e limpeza). Todos deveram ser implementados, mesmo que não utilizados durante o projeto.	1,5
Organização do código Códigos devidamente indentados e comentados. Nomes de métodos e atributos legíveis e facilmente identificáveis. Criação de métodos com funções bem definidas.	1,0
Solução estruturada Demonstração de domínio sobre a linguagem, blocos e demais estruturas utilizadas. Solução claramente dividida de forma lógica.	1,0
Efetividade do projeto Entrega de todos os tópicos solicitados no projeto, além de seguir adequadamente todas as convenções debatidas durante as aulas para desenvolvimento de código (uso de notação específica, separação de códigos-fonte por finalidade, etc.).	1,0
Tratamento de operadores unários Ponto extra dado ao tratamento adequado de operadores unários.	+1,0
Total:	**10,0** **+1,0**

O projeto entregue será avaliado de acordo com os tópicos listados acima. Entretanto, para se verificar se realmente todos os integrantes do grupo participaram do desenvolvimento, uma entrevista será realizada com o professor. Nessa entrevista, poderá ser aplicada uma penalidade que poderá variar de **0%** a **50%** da nota obtida pelo grupo. Ou seja, a nota final da unidade será dada pela seguinte expressão:

$$N_{Final} = N_{Projeto} * (1 - P)$$

Onde P é o percentual de penalidade. Esse percentual será dado de forma individual, isto é, o desempenho de um componente não prejudicará a nota dos demais.

11. Disposições finais

O projeto deverá ser desenvolvido em C++11. Para sua avaliação será usado o **g++**, logo o grupo deverá garantir que o código esteja adaptado para ser compilado adequadamente por essa ferramenta. Além dos arquivos de código-fonte, deverá ser enviado também um pequeno arquivo de texto contendo as instruções para compilação, como parâmetros para o compilador e arquivos a serem passados. Uma outra possibilidade é o envio de um Makefile contendo essas instruções, embora não seja obrigatório.

Todos os arquivos deverão ser compactados e enviados ao SIGAA na tarefa designada ao projeto até às **23:59 dia 25 de junho de 2018**. No dia seguinte, **26 de junho**, será realizada a entrevista com os grupos no horário normal de aula.

Em caso de dúvidas sobre qualquer um dos pontos descritos no documento, procure o professor para que sejam devidamente esclarecidos. Caso seja necessário, também poderá ser agendado um horário específico para atendimento presencial, mas deverá ser combinado com no mínimo 24 horas de antecedência.

Boa sorte!

```
/****************************** Fim da especificação ******************************/
```