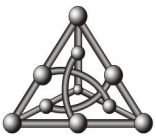


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Calculadora Polonesa
Trabalho 2

Algoritmos e Programação II

1 Descrição

A **notação polonesa** ou **notação de prefixo** é uma forma de notação para lógica, aritmética e álgebra. Não precisa de parênteses ou outros delimitadores para indicar os cálculos que devem ser realizados primeiramente, mas mesmo assim não há ambiguidade quanto à ordem de resolução. Os operadores devem **preceder** os dois valores numéricos associados a uma operação. O matemático polonês Jan Łukasiewicz criou essa notação em torno de 1920 para simplificar a lógica nas sentenças matemáticas. Não é muito usado na matemática convencional, mas muito usado na ciência da computação.

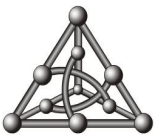
Por exemplo, considerando que queremos realizar a soma de a e b, utilizando a notação convencional escrevemos $a + b$. Já utilizando a notação polonesa, como os operadores devem preceder os valores, escrevemos $+\ a\ b$.

Já a **notação polonesa inversa**, também conhecida como **notação pós-fixada**, foi inventada pelo filósofo e cientista da computação australiano Charles Hamblin em meados dos anos 1950. Ela deriva da notação polonesa, apresentando contudo diversas vantagens em relação à esta, quer na computação automatizada, quer no cálculo manual assistido por instrumentos de cálculo. A notação polonesa inversa tem larga utilização no mundo científico pela fama de permitir uma linha de raciocínio mais direta durante a formulação e por dispensar o uso de parênteses mas mesmo assim manter a ordem de resolução. Nela, os operadores devem **suceder** os dois valores numéricos associados.

Por exemplo, considerando que queremos realizar a soma de a e b, utilizando a notação convencional escrevemos $a + b$. Já utilizando a notação polonesa inversa, como os operadores devem suceder os valores, escrevemos $a\ b\ +$. Vejamos alguns exemplos:

Operação	Notação Polonesa Inversa
$a + b$	$a\ b +$
$\frac{a+b}{c}$	$a\ b\ +\ c\ /$
$\frac{(a*b)-(c*d)}{e*f}$	$a\ b\ *\ c\ d\ *\ -\ e\ f\ *\ /$
$a + ((b + c) * d) - e$	$a\ b\ c\ +\ d\ *\ +\ e\ -$
$a + ((b + c) * d) - e$	$b\ c\ +\ d\ *\ a\ +\ e\ -$

Seu trabalho é desenvolver uma calculadora que utilize a **notação polonesa inversa**, realizando as operações que serão fornecidas pelo(a) usuário(a).



2 Entrada e saída

A entrada consiste inicialmente de um número inteiro $k > 0$, que indica o número de casos de teste a serem fornecidos como entrada. Para cada caso de teste, é fornecido em uma única linha um conjunto de operações utilizando a notação polonesa inversa. Cada conjunto de operações pode conter, separados por espaço: (i) números, (ii) operadores ou (iii) o caractere especial '@', que apenas indica o fim do caso de teste atual.

Você pode considerar ainda que os números sempre serão passados no formato `X.Y`, onde `X` é uma sequência de pelo menos 1 dígito e no máximo 10 dígitos, podendo ainda conter um sinal + ou −, e `Y` é uma sequência de 1 ou 2 dígitos. Desse modo, um número sempre contém ao menos 3 caracteres e no máximo 14 caracteres, incluindo as casas decimais, os sinais e o ponto. Por exemplo, o número `-12345678.90` possui 12 caracteres.

Para cada caso de teste, a saída consiste na impressão do resultado final das operações, ou seja, o resultado obtido após a última operação antes de '@'. Ao final de cada caso de teste imprima uma quebra de linha. Além disso, cada resultado impresso deve ter a precisão de duas casas decimais.

Não é permitido o uso de bibliotecas matemáticas, sendo assim todas as operações devem ser implementadas em funções criadas por você (p. ex. cálculo da potência). Seu programa deve suportar as seguintes operações (considere que a e b são números):

Nome	Notação convencional	Notação Polonesa Inversa
Adição	$a + b$	$a\ b +$
Subtração	$a - b$	$a\ b -$
Multiplicação	$a * b$	$a\ b *$
Divisão	$a\ /\ b$ ou $\frac{a}{b}$	$a\ b\ /$
Exponenciação inteira	$a\ \hat{}\ \lfloor b \rfloor$ ou $a^{\lfloor b \rfloor}$	$a\ b\ \hat{}$
Piso do $\log_2$*	$\lfloor \log_2 a \rfloor$	a L
Fatorial do piso*	$\lfloor a \rfloor !$	a !
Modulo do piso (resto da divisão inteira)	$\lfloor a \rfloor$ % $\lfloor b \rfloor$ ou $\lfloor a \rfloor \bmod \lfloor b \rfloor$	$a\ b$ %

*considere que a será sempre positivo

3 Exemplo de entrada

```
5
1.0 2.0 + 2.5 - 2.5 * @
5.0 +2.0 / 2.5 ^ 2.0 + L @
+6.0 4.00 % -1.25 - ! -1.0 * @
5.0 1.0 2.0 + 4.0 * + 3.0 - @
1.0 2.0 + 4.0 * 5.0 + 3.0 - @
```

4 Exemplo de saída

```
1.25
3.00
-6.00
14.00
14.00
```

5 Exigências

Você DEVE usar a seguinte estrutura de dados em seu trabalho:

```
/* Enumeração que define se um dado é um operando ou o tipo da
   operação caso o dado seja um operador */
typedef enum {
   NUM,
   ADD = '+',
   SUB = '-',
   MUL = '*',
   DIV = '/',
   EXP = '^',
   LOG = 'L',
   FAT = '!',
   MOD = '%',
   FIM = '@'
} tipo_op;

/* Define um registro do tipo celula que pode ser usado em pilhas
   ou listas. Contém o tipo do dado que representa e permite o
   armazenamento de um valor, caso se aplique */
typedef struct cel {
   double valor;
   tipo_op tipo;
   struct cel *prox;
} celula;
```

Note que caso uma célula represente uma operação, o conteúdo armazenado no campo *valor* é irrelevante.

Além disso, você DEVE utilizar uma **fila** para armazenar os dados lidos, ou seja, ler e armazenar na fila todos **operadores e operandos** antes de começar a realizar qualquer operação.

Por fim, você DEVE utilizar uma **pilha** para armazenar **operandos** retirados da fila ou obtidos a partir de operações. Assim, ao realizar operações você irá obter da pilha os valores numéricos (operandos) a serem operados. Faça alguns testes e perceberá que a pilha é necessária para resolver o problema.

6 Entrega

Instruções para entrega do seu trabalho:

1. **Cabeçalho**

   Seu trabalho deve ter um cabeçalho com o seguinte formato:

```
/*************************************************
 *
 * Nome do(a) estudante
 * Trabalho 2
 * Professor(a): Nome do(a) professor(a)
 *
 */
```

2. **Compilador**

   Os(as) professores(as) usam o compilador da linguagem C da coleção de compiladores GNU `gcc`, com as opções de compilação `-Wall -std=c99 -pedantic` para corrigir os programas. Se você usar algum outro compilador para desenvolver seu programa, antes de entregá-lo verifique se o seu programa tem extensão `.c`, compila sem mensagens de alerta e executa corretamente.

3. **Forma de entrega**

   A entrega será realizada diretamente na página da disciplina no AVA/UFMS. Um fórum de discussão deste trabalho já se encontra aberto. Após abrir uma sessão digitando seu *login* e sua senha, vá até o tópico "Trabalhos", e escolha "T2 - Entrega". Você pode fazer o upload de vários rascunhos, mas **o envio definitivo deve ser feito até a data indicada no AVA**. Apenas com o envio definitivo seu trabalho será corrigido. Encerrado o prazo, não serão mais aceitos trabalhos.

4. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

   Trabalhos com erros de compilação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

6. **O que entregar?**

   Você deve entregar um único arquivo contendo APENAS o seu programa fonte com o mesmo nome de seu login no passaporte UFMS, como por exemplo, `fulano.silva.c`. NÃO entregue qualquer outro arquivo, tal como o programa executável, já compilado.

7. **Verificação dos dados de entrada**

   Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se, por exemplo, o seu programa pede um número entre 1 e 10 e o usuário digita um número negativo, uma letra, um cifrão, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

8. **Arquivo com o programa fonte**

   Seu arquivo contendo o programa fonte na linguagem C deve estar bem organizado. Um programa na linguagem C tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Não esqueça da documentação de seu programa e de suas funções.

   Dê o nome do seu usuário do Passaporte UFMS para seu programa e adicione a extensão `.c` a este arquivo. Por exemplo, `fulano.silva.c` é um nome válido.

9. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não faça o trabalho em grupo e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!

   Trabalhos envolvidos em plágio, mesmo que parcial, terão nota ZERO.