

## 1 Sobre a entrega do trabalho

São requisitos para atribuição de notas a este trabalho:

- Uso de um arquivo Makefile para facilitar a compilação. Os professores rodarão “make” e deverão obter o arquivo executável funcional com a sua solução. Este executável deverá estar no subdiretório calculadora;
- Opções de compilação: deve incluir `-Wall` e `-std=c90`. Haverá desconto na nota se compilador mostrar algum “warning”;
- Os professores também irão rodar com o `valgrind`, cada erro também vai gerar desconto na nota;
- Arquivo de entrega:
  - deve estar no formato tar comprimido (`.tar.gz`);
  - O `tar.gz` deve ser criado considerando-se que existe um diretório com o nome do trabalho. Por exemplo, este trabalho é o calculadora;
  - Então seu `tar.gz` deve ser criado no diretório pai do subdiretório calculadora, o qual deve conter todos os arquivos que serão entregues (`tar czvf calculadora.tar.gz calculadora`), de maneira que os professores, ao abrir o `tar.gz` com o comando “`tar xzvf calculadora.tar.gz`” obterão um diretório calculadora com as suas respostas;
  - Os professores testarão seus programas em uma máquina do departamento de informática (por exemplo, `cpu1`), por isso, antes de entregar seu trabalho faça um teste em máquinas do `dinf` para garantir que tudo funcione bem.

## 2 O trabalho

Você vai baixar o `calculadora.tar.gz` anexo a este enunciado e vai abri-lo para fazer o trabalho, você vai precisar de todos os arquivos. Ele contém os seguintes arquivos sob um diretório de nome calculadora:

**libpilha.h:** arquivo (read only) de *header* para o TAD PILHA;

**calculadora.c:** arquivo com a implementação da calculadora básica;

**makefile:** sugestão de um Makefile que você pode usar (ou adaptar, se quiser).

Os arquivos `.h` não podem ser alterados em nenhuma hipótese. Na correção, os professores usarão os arquivos originais.

- Use boas práticas de programação, como indentação, bons nomes para variáveis, comentários no código, bibliotecas, ... Um trabalho que não tenha sido implementado com boas práticas vale zero.
- Quaisquer dúvidas com relação a este enunciado devem ser solicitadas via sistema moodle sendo que os professores devem receber o questionamento. Na dúvida não tomem decisões sobre a especificação, perguntem!
- Pode ser durante as aulas também.

### 3 O problema

A calculadora fornecida no arquivo `calculadora.c` avalia expressões aritméticas lidas da entrada padrão, em notação infixa, que usam:

- operadores binários `+`, `-`, `*` e `/`, com as suas respectivas precedências;
- valores numéricos de ponto flutuante;
- subexpressões entre parênteses.

A expressão fornecida na entrada padrão (`stdin`) é lida até que um `\n` seja encontrado. Após a leitura, a expressão é avaliada e o valor resultante é mostrado na saída padrão (`stdout`). Caso algum erro ocorra durante a leitura ou a avaliação da expressão uma mensagem de erro é mostrada na saída padrão de erro (`stderr`).

Você deve implementar o TAD pilha e alterar a implementação fornecida para que a calculadora suporte 3 novos recursos:

1. Incluir o operador de exponenciação, com a precedência adequada, no conjunto de operadores suportados. O caracter `^` deve ser usado para representar o operador;

2. A calculadora deve suportar múltiplas linhas na entrada. Cada linha conterá uma expressão diferente e terminará com um caractere de final de linha (`\n`). A cada linha lida, a calculadora deve avaliar e mostrar o valor resultante antes de ler a próxima linha. Caso a linha contenha o caracter `q` a calculadora deve encerrar sua execução. Caso a leitura ou a avaliação de uma linha resulte em um erro, a mensagem de erro correspondente deve ser mostrada na saída padrão de erro (`stderr`) e uma nova linha deve ser lida;
3. Além dos operandos numéricos, a calculadora deve suportar nas expressões o caracter `m`, que representa uma memória local que guarda o valor da última expressão avaliada com sucesso. No início da execução o valor dessa memória é zero.

## 4 Exemplo de entrada

A calculadora deve receber a sequência das expressões pela entrada padrão e a cada expressão avaliada deve mostrar o resultado na saída padrão. Como exemplo do funcionamento considere a seguinte sequência de entrada e suas saídas correspondentes:

```
entrada: m \n
saída: 0
entrada: 1 + 2 \n
saída: 3
entrada: (4 / 2) ^ 2 + m * 5 \n
saída: 19
entrada: (6 - m) / 2 \n
saída: -6.5
entrada: a * b \n
saída: ERRO: caracter desconhecido (coluna 1)
entrada: m \n
saída: -6.5
entrada: q \n
fim da execução
```

## 5 Como entregar o trabalho?

Entregue um único arquivo de nome `calculadora.tar.gz` no sistema moodle contendo pelo menos estes arquivos:

- `libpilha.h`, mesmo que você não possa modificá-los;
- `libpilha.c`;
- `calculadora.c`, contendo a sua implementação (este arquivo contém o seu `main()`);
- `makefile`. Os professores rodarão “make” para compilar e gerar o executável;
- Se você usou ou implementou mais arquivos que são importantes para o funcionamento do programa, entregue-os no mesmo pacote.

## 6 Considerações finais

Boa parte do desenvolvimento e evolução do software livre depende de entender, criticar e alterar código escrito por outras pessoas. Esse trabalho vai oferecer a oportunidade de amadurecimento na Linguagem C e em especial no uso de tipos de dados abstratos.

A implementação é relativamente simples, mas exige bastante cuidado. Vocês podem transformar grande parte do código da função `main` fornecida em uma função que avalia uma expressão lida e retorna o seu resultado. Não hesitem em reescrever código, quase sempre é menos trabalhoso do que conviver com o custo das decisões equivocadas passadas.

Bom trabalho!