EA 04 - Callback e ponteiro de função

Programação II - DI/UFES

Prof. Vinícius F. S. Mota

Instruções

Entregar arquivo zipado contendo todos os códigos necessários e um Makefile que permita compilar o código corretamente.

Objetivo

Neste exercício será criado uma biblioteca para manipulação de vetores. Para testar, o programa principal recebe 2 argumentos:

- Tamanho do vetor [2, 1.000.000]
- Semente número aleatório que irá preencher o vetor.

Crie um TAD vetores que possa operar sobre números *inteiros*, *float* ou *double*. Para isso,a estrutura `Vetor` deve armazenar:

1. um ponteiro de `void`, que armazena os elementos propriamente ditos;
2. a quantidade de elementos do vetor;
3. e o tipo de vetor, isto é, use uma constante para definir se o vetor é `int`, `float` ou `double`. O tipo será definido na inicialização do vetor.

O TAD `Vetor` deve ser opaco, isto é, outros arquivos não devem conseguir acessar os membros internos do TAD. Desta forma, todas as operações a seguir devem ser implementadas no arquivo que define o TAD.

1. `Vetor *criar(int tam, int tipo)`

Criar um ponteiro de `Vetor` alocando memória para *N* elementos do tipo definido pelo usuário;

2. `void preencher(Vetor *v);`

Preencher um `Vetor` com *N* elementos aleatórios.

> Dicas: Como o vetor é *void*, é necessário fazer o *cast* corretamente para preenchê-lo. Por exemplo, para preencher um vetor de *int* e de *double*, supondo que o membro interno do *struct* vetor se chame `arr`:

> inteiros
>
> ```
> ((int*)v->arr)[i] = (int)rand( )%MAX;
> ```

> double
>
> ```
> ((double*)v->arr)[i] = (double)rand( )/RAND_MAX*MAX;
> ```

3. Imprimir todos os elementos do `Vetor`. Imprimir 3 casas decimais caso o vetor seja de números reais;

4. Liberar todos os elementos do vetor e o TAD `Vetor` da memória.

> **Atenção**: Penalidade proporcional ao vazamento de memória.

Para os exercícios 5 - 7 devem ser implementadas funções auxiliares para realizar o cálculo para cada tipo de dados permitido para o vetor. O programa principal deve acessar apenas as funções que recebem o TAD Vetor como argumento e retornar um *double* com o resultado. Isto é:

> Para implementar a função `double media(Vetor *v)` pode-se criar uma função média para cada tipo que o vetor pode armazenar, por exemplo: `media_int`, `media_double` e `media_float` e a função `media` faz a chamada apropriada baseada no tipo do vetor.

5. Determinar a média aritmética dos valores de um vetor. $\overline{x} = \frac{\Sigma x_i}{n}$
6. Determinar a variância dos valores de um vetor. $s^2 = \frac{\Sigma (x_i - \overline{x})^2}{n-1}$
7. Determinar o desvio padrão dos valores de um vetor. (Raiz quadrada da variância)

Callback

8. Implemente uma função `double operacao(Vetor* v, ponteiroFuncao f)`. As funções implementadas nas questões 5 a 7 devem estar de tal forma que possam ser passadas como argumento `f` da função operacao. Em seguida, crie uma nova função `double calculaOp(Vetor *v, char o)`, que recebe o `Vetor v` com os elementos e um `char o` que indica o cálculo desejado. Use:

- 'm' : média
- 'v' : variancia
- 'd' : desvio Padrão

Por exemplo, supondo que se deseja calcular a média de um vetor de inteiros, será então chamado:

```
double d = calculaOp(v, 'm')
```

Por sua vez, `calculaOp` deverá chamar `operacao(v, media_int)`

> Dica: Você pode definir constantes para os tipos de cálculos.