Tratamento de Exceções

1. Crie uma classe `ExcecaoSoma` que receba um valor `x` e `n` números. Depois disso o programa deve somar esses `n` números enquanto a soma não for superior a `x`. O programa deve imprimir o valor somado (antes de atingir um número maior que `x`) e deverá informar quantos números foram somados e qual a média. O seu programa deve ser implementado utilizando as seguintes regras:

   - Você deve incluir tratamentos de exceção para lidar com a entrada de dados. Exemplo: o valor informado deve ser um número, logo o programa não deve permitir a entrada de números menores ou iguais a zero. Além disso, o valor `x` lido da entrada não pode ser maior que 100;
   - Quando a soma for superior a `x`, o programa deverá gerar uma exceção (com o uso do `throws`) do tipo `ExcecaoAcimaDeX` que basicamente informa o usuário acerca da exceção. Para isso você deve explorar hierarquia de classes comuns (a partir da classe `Exception` ou derivadas).

   Você pode utilizar o código abaixo (`main.cpp`) como base para modificação e teste da sua implementação.

```
#include <iostream>

int main(){
    Excecaosoma es;
    es.somaValores();
    return 0;
}
```

2. Crie uma classe `PosicoesVetor` para preencher valores de um vetor de inteiros com `y` posições. O usuário irá informar os `y` valores a serem inseridos e suas respectivas posições no array. O programa deve tratar as exceções quando for informada uma posição inexistente do vetor e quando o valor informado não for um número.

   Você pode utilizar o código abaixo (`main.cpp`) como base para modificação e teste da sua implementação.

```
#include <iostream>

int main(){
    int y = 0;

    std::cout << "Digite o tamanho do vetor: " << std::endl;
    std::cin >> y;

    PosicoesVetor v(y);

    v.preencherValores();
    return 0;
}
```

Considerações Gerais!

- Exercício individual.
- Entrega: conforme agendado no PVANET Moodle;
- Conforme estrutura abaixo apresentada crie um projeto para resolução de cada exercício (ex.: pratica7_exercicio1.zip, pratica7_exercicio2.zip, etc). Cada projeto deve conter os arquivos `.h`, `.cpp`, e `main.cpp` criados para resolução do exercício. Envie, através do PVANet Moodle, uma pasta compactada (.rar ou .zip) contendo todos os projetos (também compactados). A pasta compactada deve conter informações do aluno (ex.: julio_reis-pratica7.zip).

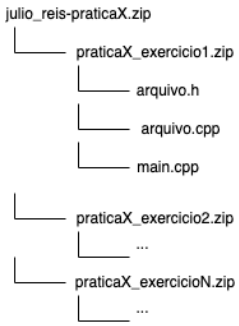

Figura 1: Estrutura de diretórios.

- O seu `main.cpp` deve conter, minimamente, instruções para criação (instanciação de objetos) e chamadas das funções implementadas (TODAS!!!). Para teste, você pode usar os exemplos fornecidos.