Tratamento de Exceções e Arquivos

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.

```cpp
#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.

```cpp
#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;
}
```

3. Crie uma classe `ArquivoIP` que leia um arquivo texto (`"entrada.txt"`) contendo uma lista de endereços IP e gere um outro arquivo (`"ips.txt"`), contendo um relatório dos endereços IP válidos e inválidos. Lembrando que um endereço IP possui o formato x.y.z.w, onde x, y, z e w são números no intervalo [0, 255]. O seu programa deve prover tratamento de exceções para falhas na abertura/leitura e fechamento do arquivo (`std::ios::exceptions`).

Exemplo do arquivo de entrada:

```
200.135.80.9
192.168.1.1
8.35.67.74
257.32.4.5
85.345.1.2
1.2.3.4
9.8.234.5
192.168.0.256
```

Exemplo do arquivo gerado como saída:

```
[Endereços válidos:]
200.135.80.9
192.168.1.1
8.35.67.74
1.2.3.4

[Endereços inválidos:]
257.32.4.5
85.345.1.2
9.8.234.500
192.168.0.256
```

4. Uma empresa está tendo problema com o controle de despesas de viagens de seus funcionários. Para tentar resolver tal situação, ela contratou você para desenvolver um programa que indique: o total de gastos com despesas de viagens de funcionários durante o último mês. Além disso, o funcionário com maior gasto, e o gasto médio e proporcional de cada funcionário. Para isso ela irá te fornecer o arquivo abaixo, chamado `"funcionarios.txt"`:

Exemplo do arquivo de entrada:

```
julio R$3003.72
miguel R$1245.98
lidiana R$1234.64
maria R$912.57
caio R$9874.58
paula R$789.45
```

Este arquivo contém informações relativas aos gastos com viagens do último mês, organizado da seguinte forma: cada linha contém informações de um funcionário, isto é, seu identificador (nome) e seu gasto mensal separado por espaço. A partir deste arquivo, você deve criar um programa (classe `Funcionario`) que gere um relatório, chamado `"relatorio.txt"`, no seguinte formato:

Exemplo do arquivo gerado como saída:

```
Nro. Funcionario Gasto %
1 julio R$3003.72 18%
2 miguel R$1245.98 7%
3 lidiana R$1234.64 7%
```

```
4 maria R$912.57 5%
5 caio R$9874.58 58%
6 paula R$789.45 5%
Gasto mensal total: R$17060.94
Gasto médio por funcionário: R$2843.49
```

O seu programa deve prover tratamento de exceções para falhas na abertura/leitura e fechamento do arquivo.

Considerações Gerais!

- Exercício individual.
- Entrega: conforme agendado no PVANET Moodle;
- Conforme estrutura abaixo apresentada crie um projeto para resolução de cada exercício (ex.: pratica9_exercicio1.zip, pratica9_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-pratica9.zip).

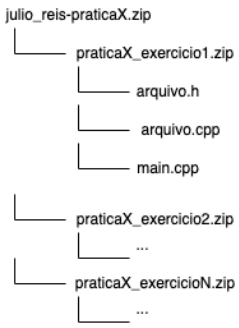

Figura 1: Estrutura de diretórios.

- 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.