UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
SCC0604 – Programação Orientada a Objetos

Prof. Robson L. F. Cordeiro

Lista 12 - C++ Threads e Exceções

1**)** Refaça o Exercício 2 da Lista 10, desta vez em C++.

a) Escreva uma classe `contaCorrente` que simule uma conta bancária de um dado cliente. Ela terá dois métodos:
- `void deposita(int valor)`: acrescenta valor ao saldo;
- `void saca(int valor)`: decrementa valor do saldo.

Instancie objetos da classe `std::thread` para criar threads que rodem simultaneamente fazendo operações sobre a mesma conta. Cada uma irá fazer 5.000 depósitos e 5.000 saques na mesma conta (concorrência). Em cada operação de saque, ou de depósito, os métodos irão seguir a sequência:
1. Lê saldo;
2. Calcula novo saldo em uma variável temporária;
3. Atribui ao saldo da conta o valor armazenado na variável temporária.

Demonstre o problema de concorrência de dados com este programa. Use valores de saldo, depósito e saque que possam ser verificados; e, se necessário, varie a quantidade de operações.

b) Resolva o problema usando bloqueios por meio de um objeto da classe `std::mutex`, e verifique que o problema de concorrência desapareceu. Assim como visto em aula, analise e teste estratégias de bloqueio que permitam maior ou menor paralelismo. Escolha e use a estratégia que você considere ser a melhor, justificando sua escolha.

2) Modifique a classe `contaCorrente` do exercício anterior de forma que os métodos `saca()` e `deposita()` lancem exceções relacionadas aos argumentos, isto é, sacar além do valor do saldo, ou depositar um valor zero ou negativo. Crie as classes `saldoInsuficiente` e `depositoInvalido` para representar esses dois tipos de exceção, respectivamente. Ambas as classes devem estender a classe de exceção `std::runtime_error`, redefinindo a descrição do erro de maneira apropriada. Em seguida, crie uma função `main()` que instancia um objeto da classe `contaCorrente`, e usa os métodos `saca()` e `deposita()` tratando as exceções propostas com os comandos `try` e `catch`. No `catch`, deve-se imprimir na tela o retorno do método `what()` do objeto de exceção recebido. Cause condições apropriadas para a execução de ambas as exceções.

3) Refaça o Exercício 5 da Lista 4, desta vez em C++.

a) Pense em uma classe `pilha`. Implemente a classe e suas duas principais operações (`void push(int valor)` e `int pop()`). `push()` é um método que deve lançar uma exceção do tipo `pilhaCheia` sempre que não couber elementos na pilha. `pop()` deve lançar uma exceção `pilhaVazia` caso a pilha não tenha mais elementos para serem retirados. Ambas as exceções são herdeiras da exceção `pilhaExcecao`. Implemente todas as 3 classes de exceção e os métodos que as lançam.

b) Implemente também uma função `main()` que mostra como criar a pilha e invocar os métodos `push()` e `pop()` com os respectivos tratamentos de exceções. Cause condições apropriadas para a execução de ambas as exceções.

Para entrega: códigos dos projetos NetBeans referentes aos exercícios acima em um arquivo zip ➔ entregar via Tidia→Atividades