C++ Semestre 2021/2 - Avaliação

Florianópolis, dezembro de 2021.

Prof. Eduardo Augusto Bezerra, eduardo.bezerra @ ufsc.br

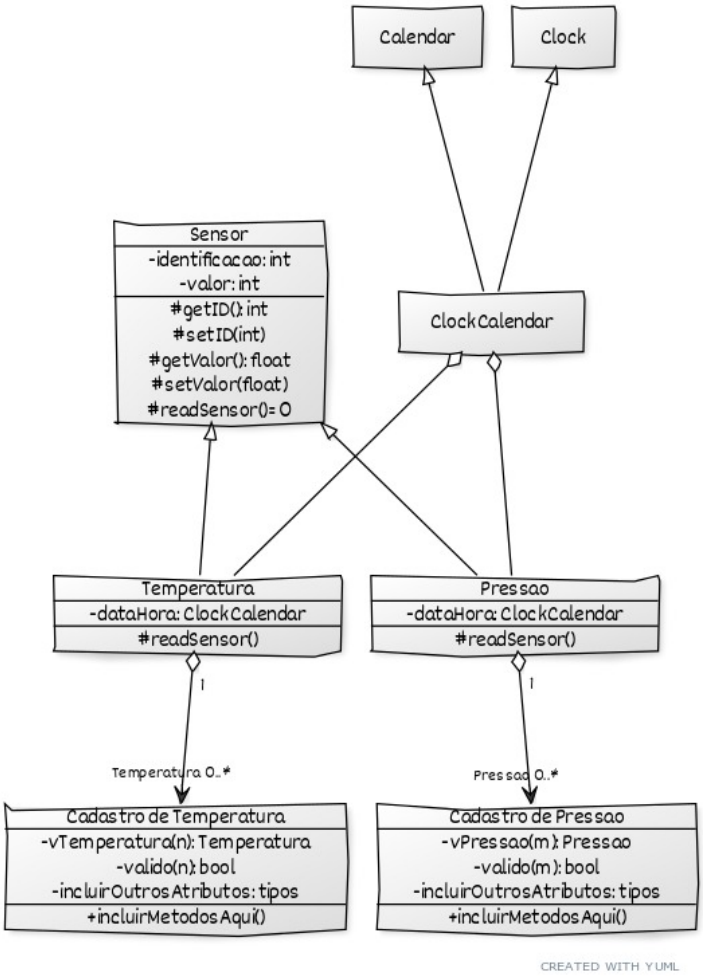
UFSC / CTC / EEL

Escrever um programa em C++ para gerência do armazenamento de leituras de sensores. O programa deverá possuir as seguintes características e funcionalidades:

- Utilizar a classe abstrata *Sensor*, que contém os atributos "identificacao" e "valor". O atributo "identificacao" deve ser utilizado para armazenar o número de série dos sensores (identificação única). O atributo "valor" armazena o valor obtido a partir da leitura de um determinado sensor. A classe possui métodos para leitura e escrita nos dois atributos. A função virtual pura "readSensor()" é utilizada para armazenar no atributo "valor" o dado lido de um determinado sensor, e a implementação dessa função irá depender da interface de comunicação disponível no sensor em questão. Conforme novos sensores forem sendo incluídos no sistema, novas implementações para essa função deverão ser fornecidas pela classe que herdar *Sensor*.
- A classe "Temperatura" deverá herdar "Sensor", e fornecer uma implementação para a função virtual pura.
- A classe "Pressao" deverá herdar "Sensor", e fornecer uma implementação para a função virtual pura.
- Em um sistema embarcado, as classes "Temperatura" e "Pressao" são utilizadas para realizar a interface lógica com os respectivos sensores, que possuem diferentes formas de leitura dos diferentes sensores, justificando assim o uso do conceito de classe abstrata nesse contexto.
- O cadastro de temperaturas (vetor) deverá possuir dois campos (em cada posição do vetor):
  - Um campo para armazenar um objeto Temperatura, que contém a temperatura lida do sensor e a respectiva data/hora da leitura (objeto *ClockCalendar*);
  - Um campo para sinalizar se existe informação válida armazenada no respectivo campo.
- O cadastro de pressões (vetor) deverá possuir dois campos (em cada posição do vetor):
  - Um campo para armazenar um objeto Pressao, que contém a pressao lida do sensor e a respectiva data/hora da leitura (objeto *ClockCalendar*);
  - Um campo para sinalizar se existe informação válida armazenada no respectivo campo.
- As classes de ambos cadastros devem possuir os métodos necessários para manipular os atributos.
- **Não é permitido usar a STL para implementar a estrutura de armazenamento de dados de pressão e temperatura.**
- O limite máximo de elementos no vetor de temperaturas (*n*), e de pressões (*m*) deve ser definido estáticamente por intermédio de constantes no corpo do programa, antes da geração do executável.
- Devem ser utilizadas as classes *Clock*, *Calendar*, e *ClockCalendar* desenvolvidas na aula sobre herança múltipla.
- O programa deverá possuir facilidades para o usuário realizar operações de entrada de dados (a identificação do sensor deve ser única), consulta ao valor de uma determinada leitura, listagem de todos as leituras mostrando todos os campos, e exclusão de leituras.
- O menu do programa deverá possuir, no mínimo, as seguintes opções:
  1. Leitura de sensor de temperatura (que inclui o armazenamento do valor lido no vetor)
  2. Excluir temperatura
  3. Consultar determinada temperatura
  4. Listar todas as temperaturas
  5. Leitura de sensor de pressao (que inclui o armazenamento do valor lido no vetor)
  6. Excluir pressao
  7. Consultar determinada pressao
  8. Listar todas as pressões
- Notar que no menu não existe uma opção para "sair" do programa, uma vez que o objetivo final é a utilização em um sistema embarcado, onde o programa deverá permanecer em laço infinito. Assumir

que o hardware alvo foi concebido especialmente para execução desse programa, e não faz sentido "encerrar" o programa, uma vez que o sistema embarcado em questão não possui outras funcionalidades.

- Deverá ser fornecida uma implementação em C++ para o diagrama de classes apresentado na figura a seguir.



- Todas as soluções deverão utilizar, obrigatoriamente, a Classe Sensor fornecida a seguir (arquivos .h e .cpp). Essa classe não poderá ser alterada. No momento da correção das soluções fornecidas, serão removidos os arquivos referentes à classe Sensor, e serão utilizados os arquivos disponíveis nessa especificação.
- São fornecidos também templates para as classes Temperatura e Pressão. Essas classes poderão ser modificadsa livremente. Nos templates fornecidos, existe uma sugestão de geração de leituras pseudo-aleatórias de pressão e temperatura, simulando leituras de sensores reais.
- No diagrama, está indicado o uso de herança múltipla na classe ClockCalendar, que herda Clock e Calendar. Essa herança múltipla é obrigatória.
- As relações de agregação indicadas no diagrama, entre as classes Temperatura, Pressao, e ClockCalendar, podem ser substituídas por herança, caso necessário.
- As relações de agregação indicadas no diagrama, entre as classes Cadastro de Temperatura, Cadastro de Pressao, Temperatura e Pressao, podem ser substituídas por herança, caso necessário.
- Essas alterações no tipo de relação entre as classes não deve alterar o nível hierárquico apresentado no Diagrama de Classes da figura. Por exemplo, é permitido alterar o tipo de relacionamento entre a Classe Cadastro de Temperatura e a Classe Temperatura para herança, mas a hierarquia apresentada no diagrama precisa ser mantida, ou seja, Cadastro de Temperatura deverá se tornar uma classe filha de Temperatura, que por sua vez é uma classe filha de Sensor.
- Deverá ser fornecido um novo Diagrama de Classes, apresentando a solução final implementada na solução. O diagrama deverá listar todos os atributos e métodos implementados na solução. Notar que o Diagrama de Classes da figura acima está incompleto.
- As soluções devem ser implementadas de forma o mais genérica possível, para possibilitar a compilação em diversos sistemas operacionais. No momento da correção, os programas serão

compilados utilizando o g++ na linha de comando, sem o uso de ferramentas de desenvolvimento ou interfaces gráficas. Uma sugestão é construir um Makefile para facilitar a compilação na linha de comando.
- A entrega das soluções deverá ser realizada até o dia 16/12, utilizando o link disponível no Moodle.

Lista de arquivos a serem utilizados na solução, e templates:

Arquivo	Descrição
Sensor.h	Interface para classe Sensor. **Este arquivo não pode ser alterado!**
Sensor.cpp	Implementação da classe Sensor. **Este arquivo não pode ser alterado!**
Temperatura.h	Interface para classe Temperatura. Para implementar a solução esperada, é possível utilizar essa classe sem nenhuma alteração. Porém, se necessário, podem ser realizadas alterações.
Temperatura.cpp	Implementação da classe Temperatura. Implementação incompleta, mas com dica de como gerar dados simulados de leituras de temperaturas.
Pressao.h	O template para a Classe Pressao não é fornecido, pois essa classe é praticamente identica a Classe Temperatura.