Trabalho 1 - C++

Universidade Federal de Santa Catarina (UFSC) - 2021.2

Professor: Eduardo Augusto Bezerra eduardo.bezerra@ufsc.br

Aluno: Luis Antonio Spader Simon <luisspaders@gmail.com>

Link para o relatório Online

Trabalho 1 - C++

Universidade Federal de Santa Catarina (UFSC) - 2021.2 EEL7323 - Programação C++ para Sistemas Embarcados Curso: Engenharia

https://grateful-snowshoe-7e7.notion.site/Trabalho-1-C-40faf091357445d782bdb1fd79495e6f

EEL7323 - Programação C++ para Sistemas Embarcados

Sumário (interativo = clique para ir à seção)

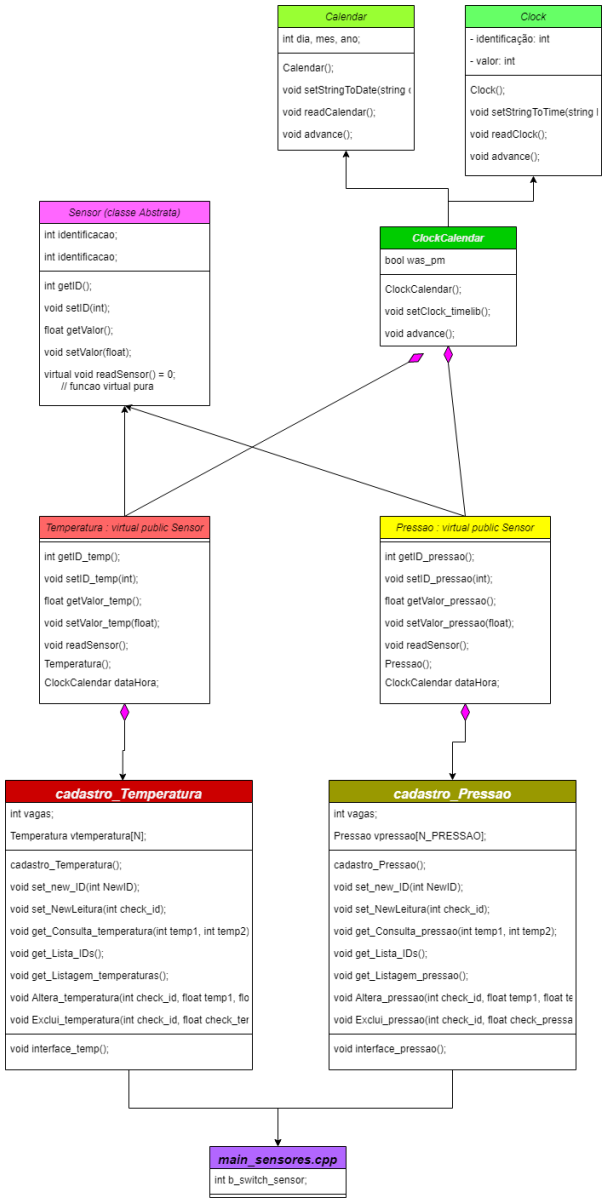
▼ Expandir Sumário

Documentação

Diagrama de Classes

Modificações

- Não foram feitas modificações consideráveis, pode se ver através do diagrama ao lado, algumas alterações no ClockCalendar, assim como nas classes Temperatura e Pressão.
- As classes de Cadastro_Temperatura e Cadastro_Pressao implementam já uma interface com o usuário, sendo que
- O arquivo `main_sensores.cpp` não possui grandes implementações de interface e pode ser visto abaixo:




```
int main()
{
  int b_switch_sensor;
  while (true)
  {
    cout << " ================== Programa
 de Sensores ================== " << endl;
    cout << " Selecione o tipo de sensor co
m que deseja trabalhar" << endl;

    cout << " 1 - Temperatura" << endl;
    cout << " 2 - Pressao" << endl;
    cin >> b_switch_sensor;

    switch (b_switch_sensor)
    {
    case 1: // TEMPERATURA
      /* code */
      interface_temp();
      break;
    case 2: // PRESSAO
      interface_pressao();
      break;

    default:
      cout << "Opcao invalida, tente novame
nte" << endl;
```

```
      break;
    }
  }
  return 0;
}
```

- Onde `interface_temp();` e `interface_pressao();` são os métodos de interface com os sensores.

- **Arquivos do diagrama:**

  Flowchart Maker & Online Diagram Software

  diagrams.net (formerly draw.io) is free online diagram software. You can use it as a flowchart maker, network diagram software, to

  https://app.diagrams.net/#G1i1xKUMFHxw-4hV7hqrfaRK6-9ho97Xvp

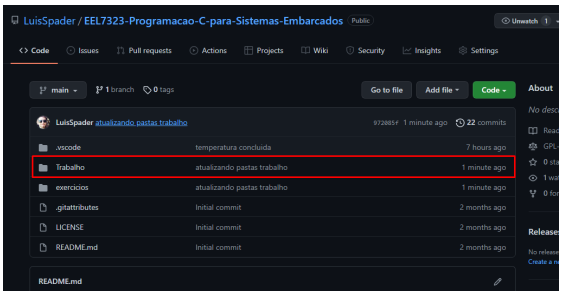
  Site do diagrama

  https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f491ddfb-9c99-4766-a690-7f9496945afe/C_Sensores.drawio.pdf

  pdf do diagrama

Repositório

Localizado na pasta 'Trabalho' conforme indicado na imagem abaixo:



Link do Repositório:

https://github.com/LuisSpader/EEL7323-Programacao-C-para-Sistemas-Embarcados

Arquivo zip do repositório (clique para download):

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cafd4719-4c1c-4984-98bd-55cd36a1a1d6/EEL7323-Programacao-C-para-Sistemas-Embarcados-main.zip

Alguns trechos de código

▼ Cadastro_pressao.h

```
/* File cadastro_Pressao.cpp

  Luis Antonio Spader Simon <luisspaders@gmail.com>
  Curso: Engenharia Eletrônica - Graduação - CTC - UFSC
  2021

  Descricao: Definicao das funcoes membro para os "headers" declarados na classe.h

*/

//---------------------------- LIBRARIES ---------------------- //
#include <string.h> // Para trabalhar com strings
#include <iostream> // Este arquivo específico inclui declarações básicas da biblioteca de E/S do C++
// #include <string.h>  // Para trabalhar com strings
#include <stdlib.h>  // This header defines several general purpose functions, including dynamic memory management, random number generation, communication with the environment, integer arithmetics, searching, sorting and converting.
using namespace std; // Esse comando é utilizado de forma a evitar a indicação std:: antes de usar o comando cout, etc...

//---------------------------- MY LIBRARIES ---------------------- //
#include "Pressao.cpp"

//---------------------------- CONSTANTS ---------------------- //
#define N_PRESSAO 2

class cadastro_Pressao
```

▼ Pressao.h

```
/*
 * file: Pressao.h
 *
 * Descricao: Classe Pressao utilizada para realizar a leitura do sensor de Pressao, e data data/hora da leitura realizada.
 *
 * Autor: Eduardo Augusto Bezerra
 * Data: 09/12/2021
 *
 * Ultima Alteracao: Eduardo Augusto Bezerra
 * Data da ultima alteracao: 09/12/2021
 *
 */

// #include "Sensor.cpp"
// #include "ClockCalendar.cpp"

class Pressao : virtual public Sensor
{

protected:
public:
  int getID_pressao();
  void setID_pressao(int);
  float getValor_pressao();
  void setValor_pressao(float);
  void readSensor();

  Pressao();
  ClockCalendar dataHora;
};
```

▼ Pressao.h

```
/*
 * file: Pressao.cpp
 *
```

```
{
private:
  // bool temp_valida;

public:
  int vagas;
  Pressao vpressao[N_PRESSAO];
  cadastro_Pressao(); // Construtor: inicia
objeto
                      // ~cadastro_Pressao
(); // Destrutor: destroi objeto

  void set_new_ID(int NewID);

  void set_NewLeitura(int check_id);

  void get_Consulta_pressao(int temp1, int
 temp2);
  void get_Lista_IDs();
  void get_Listagem_pressao();
  void Altera_pressao(int check_id, float t
emp1, float temp2); // o número de matrícul
a não pode ser alterado
  void Exclui_pressao(int check_id, float c
heck_pressao, bool apaga_todos);
};
```

▼ `cadastro_Pressao.cpp`

```
/* File cadastro_Pressao.cpp

  Luis Antonio Spader Simon <luisspaders@gm
ail.com>
  Curso: Engenharia Eletrônica - Graduação
 - CTC - UFSC
  2021

  Descricao: Definicao das funcoes membro p
ara os "headers" declarados na classe.h
*/
using namespace std; // Esse comando é util
izado de forma a evitar a indicação std:: a
ntes de usar o comando cout, etc...

#include "cadastro_Pressao.h"
#include <iomanip> // para usar setfill() e
setw()

// Construtor
cadastro_Pressao::cadastro_Pressao()
{
  // n_leituras = 0;
  // temp_valida = false;
  vagas = N_PRESSAO;
}

// ============== MÉTODOS DA CLASSE =======
==== //
```

```
 * Descricao: Implementacao da Classe Press
ao utilizada para realizar a leitura do sen
sor de Pressao, e data data/hora da leitura
realizada.
 *
 * Autor: Eduardo Augusto Bezerra
 * Data: 09/12/2021
 *
 * Ultima Alteracao: Eduardo Augusto Bezerr
a
 * Data da ultima alteracao: 09/12/2021
 *
 */
#include "Pressao.h"
#include <cstdlib>   // para usar srand() e
rand()
#include <ctime>     // para usar time()
#include <stdlib.h>  // This header defines
several general purpose functions, includin
g dynamic memory management, random number
 generation, communication with the environ
ment, integer arithmetics, searching, sorti
ng and converting.
using namespace std; // Esse comando é util
izado de forma a evitar a indicação std:: a
ntes de usar o comando cout, etc...
#include <iomanip>   // para usar setfill()
e setw()

Pressao::Pressao()
{
  setValor(-300);
  setID(0);
}

// Aqui está a implementação para a Função
 Virtual Pura (função sem declaração) herda
da da classe 'Sensor'
// void Pressao::readSensor(int ID, float l
eitura)
void Pressao::readSensor()
{

  dataHora.setClock_timelib();

  // Simulacao de leitura de sensor
  setValor(static_cast<float>(rand()) / sta
tic_cast<float>(RAND_MAX));
}

// SET
void Pressao::setID_pressao(int newID)
{
  setID(newID);
}

void Pressao::setValor_pressao(float newVal
or)
{
  setValor(newValor);
}
```

```
void cadastro_Pressao::set_new_ID(int NewI
D)
{

  if (NewID <= 0) // NewID > 0
  {
    cout << "----------------------------
--------------------------------\n";
    cout << "Valor deve ser um inteiro maio
r que zero!" << endl;
    cout << "----------------------------
--------------------------------\n";
    return;
  }
  for (int i = 0; i < N_PRESSAO + 1; i++)
  {
    if (vpressao[i].getID_pressao() == NewI
D) // verifica id repetido
    {
      cout << "----------------------------
-----------------------------------\n";
      cout << "                   Este ID j
a existe!" << endl;
      cout << "----------------------------
-----------------------------------\n";
      return;
    }
    else
    {
      if (vpressao[i].getID_pressao() == 0)
// quando pressao = 0 = vazio
      {
        vpressao[i].setID_pressao(NewID);
        cout << "----------------------------
-------------------------------------\n";
        cout << "             Sensor de ID:
" << NewID << " registrado!" << endl;
        cout << "----------------------------
-------------------------------------\n";
        return;
        // break;
      }
      else if (i == N_PRESSAO)
      {
        cout << "----------------------------
-------------------------------------\n";

        cout << "           Nao ha espaco p
ara novos IDs" << endl;
        cout << "----------------------------
-------------------------------------\n";
        return;
      }
    }
  }
}

void cadastro_Pressao::set_NewLeitura(int c
heck_id)
{
  if (vagas == 0) // VETOR CHEIO
  {
```

```
// GET
int Pressao::getID_pressao()
{
  return getID();
}

float Pressao::getValor_pressao()
{
  return getValor();
}
```

```
    cout << "----------------------------
--------------------------------\n";
    cout << "Leitura nao pode ser armazenad
a. Espaco de armazenamento esta cheio!" <<
 endl;
    cout << "----------------------------
--------------------------------\n";

    // break;
    return;
  }

  // CASO O ID SEJA VÁLIDO O LOOP ANTERIOR
 NÃO DARÁ 'EXIT' na função (id válida) E O
 QUE ESTÁ ABAIXO SERÁ EXECUTADO
  for (int i = 0; i < N_PRESSAO; i++)
  {
// loop de checagem id válido (já existente
em alguma linha)
    if (vpressao[i].getID_pressao() == chec
k_id) // se achamos algum ID igual ao check
_id, entao podemos cadastrar uma nova leitu
ra
    //  porém essa nova leitura pode ser um
a linha que já tem o ID mas não tem a press
ao (= -300)
    //  ou a nova leitura sera em uma linha
onde nao temos NENHUM ID
    {
      // aí aqui fazemos o loop para procur
ar registros vazios (pressao = -300) com o
 ID passado que já foi validado no 'if' ant
erior
      for (int i = 0; i < N_PRESSAO; i++)
      {
        if ((vpressao[i].getID_pressao() ==
0 || vpressao[i].getID_pressao() == check_i
d) & vpressao[i].getValor_pressao() == -30
0) // QUANDO ACHARMOS LUGAR VAZIO (pressao
 = -300) => CADASTRO DE NOVA LEITURA
        {
          vpressao[i].setID_pressao(check_i
d);
          vpressao[i].readSensor();
          --vagas;

          cout << "----------------------
--------------------------------------\n";
          cout << "Registro efetuado: \n";
          cout << "    ID    |      Data
|     Hora     |   Valor" << endl;
          cout << setw(8) << setfill('0') <
< vpressao[i].getID_pressao() << "  |  ";
 // ID
          vpressao[i].dataHora.readCalendar
();
 // Data
          cout << "   |   ";
          vpressao[i].dataHora.readClock();
// Hora
          cout << "   |  ";
          cout << vpressao[i].getValor_pres
```

```
sao() << endl; // Valor
          cout << "-----------------------
--------------------------------------\n
 \n";

          // cout << "ID: " << vpressao[i].
getID_pressao() << "; Valor: " << vpressao
[i].getValor_pressao() << endl;
          return;
        }
       }
    }
  } // caso nao dado exit (return;) irá che
gar até aqui. Isso só ocorre se a nao achou
nenhuma ID compatível
  cout << "-------------------------------
-------------------------------\n";
  cout << "                   ID nao encont
rada!"
       << endl;
  cout << "-------------------------------
-------------------------------\n \n";
}

void cadastro_Pressao::get_Lista_IDs()
{
  cout << "-------------------------------
-------------------------------\n";
  for (int i = 0; i < N_PRESSAO; i++)
  {

    cout << "ID[" << i << "] - " << vpressa
o[i].getID_pressao() << endl;
    cout << "-----------------------------
---------------------------------\n";
  }
}

void cadastro_Pressao::get_Consulta_pressao
(int temp1, int temp2)
{
  bool temp_valida = false;

  cout << "-------------------------------
-------------------------------\n";
  for (int i = 0; i < N_PRESSAO; i++)
  {
    if ((vpressao[i].getValor_pressao() > t
emp1) & (vpressao[i].getValor_pressao() < t
emp2))
    {
      cout << "    ID    |      Data     |
Hora     |   Valor" << endl;
      cout << setw(8) << setfill('0') << vp
ressao[i].getID_pressao() << "  |  "; // ID
      vpressao[i].dataHora.readCalendar();
// Data
      cout << "   |   ";
      vpressao[i].dataHora.readClock(); //
 Hora
      cout << "   |  ";
      cout << vpressao[i].getValor_pressao
```

```
() << endl; // Valor
      cout << "----------------------------
-----------------------------------\n";

      temp_valida = true;
    }
  }
  if (temp_valida = true) // quando teve pe
lo menos 1 vpressao no intervalo
  {
    cout << "------------------------------
--------------------------------\n";
    // terminamos impressao
    temp_valida = false; // resetamos varia
vel
  }
  else // quando nao teve nenhuma vpressao
 no intervalo
  {
    cout << " Nao ha temperaturas registrad
as no intervalo entre " << temp1 << "ºC e "
<< temp2 << "ºC " << endl; // mensagem ao u
suario
    cout << "------------------------------
--------------------------------\n";
    // terminamos impressao
  }
}

void cadastro_Pressao::get_Listagem_pressao
()
{
  if (vagas == N_PRESSAO)
  {
    cout << "------------------------------
--------------------------------\n";
    cout << "   Nao ha leituras registrada
s" << endl; // se a turma está vazia, para
 por aqui
    cout << "------------------------------
--------------------------------\n";
    return;
  }
  else //
  {
    cout << "------------------------------
--------------------------------\n";
    cout << "    ID    |      Data     |
Hora     |   Valor" << endl;

    for (int i = 0; i < N_PRESSAO; i++)
    {
      if (vpressao[i].getID_pressao() > -30
0)
      {
        cout << setw(8) << setfill('0') <<
 vpressao[i].getID_pressao() << "  |  "; //
ID
        vpressao[i].dataHora.readCalendar
();
 // Data
        cout << "   |   ";
```

```
        vpressao[i].dataHora.readClock();
 // Hora
        cout << "   |  ";
        cout << vpressao[i].getValor_pressa
o() << endl; // Valor
      }
    }
  }
  cout << "------------------------------
------------------------------\n";
}

void cadastro_Pressao::Exclui_pressao(int c
heck_id, float check_pressao, bool apaga_to
dos)
{
  if (vagas == N_PRESSAO) // SEM REGISTROS
  {
    cout << "------------------------------
--------------------------------\n";
    cout << "            Nao ha leitura reg
istrada para excluir" << endl;
    cout << "------------------------------
--------------------------------\n";
    return;
  }
  // COM ALGUM REGISTRO

  // if (apaga_todos == true) // quando apa
gamos todos
  // {
  // for (int i = 0; i < N_PRESSAO; i++)
  // {
  //   if ((vpressao[i].getID_pressao() ==
 check_id))
  //   {
  //     cout << "------------------------
------------\n";
  //     cout << "Pressao " << check_pressa
o << "de check_id " << vpressao[i].getID_pr
essao() << " excluida" << endl;
  //     cout << "------------------------
------------\n";
  //     vpressao[i].setID_pressao(0);
  //     vpressao[i].setValor_pressao(0);
  //     ++vagas;
  //   }
  // }

  // else // quando vamos apagar só 1
  // {

  for (int i = 0; i < N_PRESSAO; i++)
  {
    // ENCONTRA ID E TEMPERATURA
    if ((vpressao[0].getValor_pressao() >
 (check_pressao - 0.01)) & (vpressao[0].get
Valor_pressao() < (check_pressao + 0.01)) &
(vpressao[0].getID_pressao() == check_id))
    {
      cout << "------------------------
----------------------------------\n";
```

```
      cout << "                    Registr
o excluido:" << endl;
      cout << "ID: " << vpressao[0].getID_p
ressao() << "; Pressao: " << vpressao[0].ge
tValor_pressao() << endl;
      cout << "-------------------------
-----------------------------------\n";
      vpressao[0].setID_pressao(0);
      vpressao[0].setValor_pressao(0);
      ++vagas;
      return;
    }
    cout << vpressao[0].getValor_pressao();
    cout << vpressao[0].getID_pressao();
    // return;
    // }
    // else if (i = (N_PRESSAO - 1)) // se
 chegou aqui não achou nenhuma, logo ID não
existe
    // {
    cout << "-------------------------
-----------------------------------\n";
    cout << "                Registro nao
 encontrado! " << endl;
    cout << "ID: " << check_id << "; Pressa
o: " << check_pressao << endl;
    cout << "-------------------------
-----------------------------------\n";
    //   return;
    // }
    // }
  }
}

void interface_pressao()
{

  bool continuar = true;
  int b_id, b_switch;
  float b_pressao1, b_pressao2;
  cadastro_Pressao pressao;

  while (continuar == true)
  {
    cout << "------------------------ Pres
sao ----------------------------\n";
    cout << "Capacidade: " << N_PRESSAO <<
 " registros; Leituras disponiveis para reg
istro: " << pressao.vagas << endl;

    cout << "1 - Cadastro de Sensor" << end
l;
    cout << "2 - Registro Leitura do Senso
r" << endl;
    cout << "3 - Consultar " << endl;
    cout << "4 - Lista IDs" << endl;
    cout << "5 - Listar todas as leituras"
 << endl;
    cout << "6 - Excluir " << endl;
    cout << "7 - Voltar " << endl;

    cout << "Digite a operacao desejada:";
```

```
    cin >> b_switch;

    switch (b_switch)
    {
      // --------------- Pressao ----------
-----
    case 1: // 1 - Cadastro de Sensor
    {
      cout << "Digite o numero de identific
acao (ID): ";
      cin >> b_id;
      pressao.set_new_ID(b_id);
      break;
    }

    case 2: // 2 - Registo Leitura do Senso
r
    {
      cout << "Digite o numero de identific
acao (ID): ";
      cin >> b_id;
      pressao.set_NewLeitura(b_id);
      break;
      // o valor de leitura já é realizado
 na classe 'Pressao' que fornece um rand pa
ra o método setValor da classe 'Sensor'
      // a data tbm gera automatico -> pega
do sistema operacional
    }

    case 3: // 3 - Consultar
    {
      cout << "Digite o numero de identific
acao (ID): ";
      cin >> b_id;

      cout << "Digite os limites do interva
lo de pressao (temp1 a temp2) em graus Cels
ius" << endl;
      cout << "Pressao 1: ";
      cin >> b_pressao1;
      cout << "Pressao 2: ";
      cin >> b_pressao2;

      pressao.get_Consulta_pressao(b_pressa
o1, b_pressao2);
      break;
    }

    case 4: // 4 - Listar IDs
    {
      pressao.get_Lista_IDs();
      break;
    }
    case 5: // 4 - Listar todas as leituras
de 1 ID
    {
      pressao.get_Listagem_pressao();
      break;
    }

    case 6: // Excluir
```

```
    {

      cout << "Digite o numero de identific
acao (ID): ";
      cin >> b_id;

      // cout << "Qual dos registros voce d
eseja excluir:" << endl;
      // for (int i = 0; i < N_PRESSAO; i+
+)
      // {
      //   if (pressao.vpressao[i].getValor
_pressao() > (-273.15))
      //   {
      //     cout << i << " - " << vpressao
[i].getValor_pressao() << endl;
      //   }
      // }
      cout << "Voce deseja excluir um ou to
dos os registros?\n";
      cout << "1 - Apenas um registro do se
nsor" << endl;
      cout << "2 - Todos os registros do se
nsor" << endl;
      cin >> b_pressao2;

      cout << "Digite a pressao que deseja
 excluir: ";
      cin >> b_pressao1;
      pressao.Exclui_pressao(b_id, b_pressa
o1, false);

      // if (b_pressao2 == 2)
      // {
      //   pressao.Exclui_pressao(b_pressao
1, 0, true);
      //   break;
      // }
      // else
      // {
      //   cout << "Digite a pressao que de
seja excluir: ";
      //   cin >> b_pressao1;
      //   pressao.Exclui_pressao(b_pressao
1, b_pressao1, false);
      break;
      // }
    }

    case 7: // Sai Sensor
    {
      continuar = false;
    }

    default:
    {
      cout << "Opcao invalida, tente novame
nte" << endl;
      break;
    }
    }
  }
```

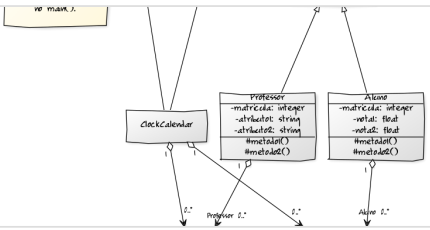
```
  // return 0;
}
```

Referências

Herança em C++

Florianópolis, setembro de 2019. Prof. Eduardo Augusto Bezerra, eduardo.bezerra @ ufsc.br UFSC / CTC / EEL Em uma das aulas anteriores, foi solicitada a implementação de um

https://gse.ufsc.br/bezerra/disciplinas/cpp/material/exercicio_heranca.html



https://s3-us-west-2.amazonaws.com/secure.notion-static.com/31cb838b-b2a6-4420-846a-77fddf57bdb0/notas_aula.pdf

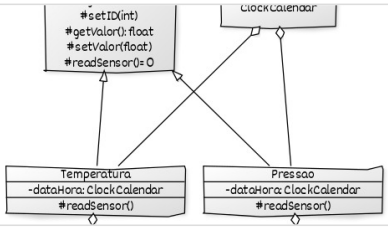
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f12d3691-95a7-4942-bde0-cd7d3628841e/Notas_de_aula_do_curso_de_Linguagem_de_ProgramaoC.pdf

Descrição das atividades

Avaliação 1 2021/2

Florianópolis, dezembro de 2021. Prof. Eduardo Augusto Bezerra, eduardo.bezerra @ ufsc.br UFSC / CTC / EEL Utilizar a classe abstrata Sensor, que contém os atributos

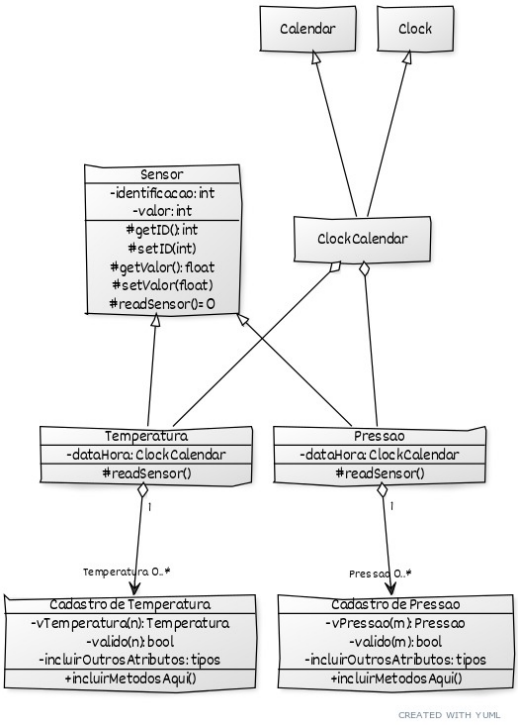
https://gse.ufsc.br/bezerra/disciplinas/cpp/2021.2/P1/Enunciado/avaliacao.html



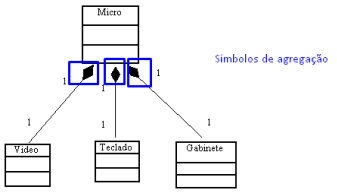
Escrever um programa em C++ para gerência do armazenamento de leituras de sensores. O programa deverá possuir as seguintes características e funcionalidades:

- [ ] A entrega das soluções deverá ser realizada até o dia 16/12, utilizando o link disponível no Moodle.

- [x] ~~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*.~~

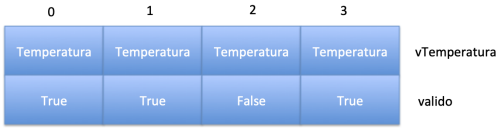


Exemplo 1 – Um microcomputador com vídeo, teclado e gabinete



Classe Sensor

- [x] ~~Classe sensor~~
  - [x] ~~identificação (id único): numero de série do sensor~~
    - [x] ~~leitura~~
    - [x] ~~escrita~~
  - [x] ~~valor: valor de leitura → Vetor~~
    - [x] ~~Ambas abaixo com Data e Hora (ClockCalendar)~~
    - [x] ~~leitura~~
    - [x] ~~escrita~~




```
If (vTemperatura[i].getValor() == 20)
        valido[i] = false;

for (int i = 0; i < MAX_TEMP; i++)
      If (valido[i])
            cout << vTemperatura[i].getValor() ;
```

Menu

- [x] ~~O menu do programa deverá possuir, no mínimo, as seguintes opções:~~

- [x] ~~readSensor() – **Função Virtual pura**, usada para armazenar o valor~~
  - [x] ~~implementação da função dependerá da interface de cada sensor (creio que cada sensor terá uma classe de leitura para passar para esta classe readSensor)~~
  - [x] ~~Conforme novos sensores forem sendo incluídos no sistema, novas implementações para essa função deverão ser fornecidas pela classe que herdar Sensor.~~
- [x] ~~Interface lógica com os diferentes sensores (diferentes formas de leitura) pode ser feito com:~~
  - Sobrecarga de funções
  - Classe abstrata → **Função Virtual pura**
    - Classe Abstrata (classe Pai), possui pelo menos 1 função virtual pura (sem declaração):
      - `virtual void rotate (int) = 0; // virtual pura`
    - Classe filha (possui a declaração da função):
      - `void rotate (int) {...}` //aqui está a implementação da função virtual pura 'rotate', que herdamos da Classe Abstrata 'Shape'
  - Polimorfismo:
    - Classe pai: `virtual void setJ(int newj);`
- [x] ~~Leitura de sensor de temperatura (que inclui o armazenamento do valor lido no vetor)~~
- [x] ~~Excluir temperatura~~
- [x] ~~Consultar determinada temperatura~~
- [x] ~~Listar todas as temperaturas~~
- [x] ~~Leitura de sensor de pressao (que inclui o armazenamento do valor lido no vetor)~~
- [x] ~~Excluir pressao~~
- [x] ~~Consultar determinada pressao~~
- [x] ~~Listar todas as pressões~~
- [x] ~~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.~~

Diagrama

- [x] ~~No diagrama, está indicado o uso de herança múltipla na classe ClockCalendar, que herda Clock e Calendar. Essa herança múltipla é obrigatória.~~

Classe filha: `void setJ(int newj);`

Classe Temperatura

- [x] A classe "Temperatura" deverá
  - [x] herdar "Sensor",
  - [x] e fornecer uma implementação para a função virtual pura.
- [x] O cadastro de temperaturas (vetor) deverá possuir dois campos (em cada posição do vetor):
  - [x] 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.
  - [x] 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.
  - [x] As classes de ambos cadastros devem possuir os métodos necessários para manipular os atributos.

Classe Pressão

- [x] A classe "Pressao" deverá
  - [x] herdar "Sensor",

- [x] As relações de agregação indicadas no diagrama, entre as classes Temperatura, Pressao, e ClockCalendar, podem ser substituídas por herança, caso necessário.
- [x] 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.

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

- [x] ~~e fornecer uma implementação para a função virtual pura.~~
- [x] ~~O cadastro de pressões (vetor) deverá possuir dois campos (em cada posição do vetor):~~
  - [x] ~~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.
- [x] ~~As classes de ambos cadastros devem possuir os métodos necessários para manipular os atributos.~~

Observações

Geral

- [x] ~~**Não é permitido usar a STL para implementar a estrutura de armazenamento de dados de pressão e temperatura.**~~
- [x] ~~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.~~
- [x] ~~Devem ser utilizadas as classes *Clock*, *Calendar*, e *ClockCalendar* desenvolvidas na aula sobre herança múltipla.~~
- [x] ~~Notar que no menu não existe uma opção para "sair" do programa, uma vez que o objetivo final é a~~

Arquivo	Descrição	Aa
Sensor.h	Interface para classe Sensor. **Este arquivo não pode ser alterado!**	Unt
Sensor.cpp	Implementação da classe Sensor. **Este arquivo não pode ser alterado!**	Unt
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.	Unt
Temperatura.cpp	Implementação da classe Temperatura. Implementação incompleta, mas com dica de como gerar dados simulados de leituras de temperaturas.	Unt

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

Arquivo	Descrição	Aa
Pressao.h	O template para a Classe Pressao não é fornecido, pois essa classe é praticamente identica a Classe Temperatura.	Unt

Compilação

- [x] ~~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.~~

Funcionalidades

- [x] ~~O programa deverá:~~

- [x] ~~possuir facilidades para o usuário realizar operações de entrada de dados (a identificação do sensor deve ser única)~~
- [x] ~~consulta ao valor de uma determinada leitura~~
- [x] ~~listagem de todos as leituras mostrando todos os campos~~
- [x] ~~exclusão de leituras.~~

Arquivos

▼ `Sensor.h`

```
/*
 * file: Sensor.h
 *
 * Descricao: Classe base Sensor a ser utilizada n
a P1.
 *
 * Autor: Eduardo Augusto Bezerra
 * Data: 09/12/2021
 *
 * Ultima Alteracao: Eduardo Augusto Bezerra
 * Data da ultima alteracao: 09/12/2021
 *
 */


//////////////////////////////////////////////////////

// Class Sensor - e' uma classe abstrata, pois pos
sui
//                  pelo menos uma funcao virtual
 pura.
//

class Sensor {
      int identificacao;
      float valor;
  protected:
    int getID();
    void setID(int);
    float getValor();
    void setValor(float);
    virtual void readSensor() = 0;    // funcao vi
rtual pura
};
```

▼ `Sensor.h`

```
/*
 * file: Sensor.cpp
 *
 * Descricao: Classe base Sensor a ser utilizada n
a P1.
 *
 * Autor: Eduardo Augusto Bezerra
 * Data: 09/12/2021
 *
 * Ultima Alteracao: Eduardo Augusto Bezerra
 * Data da ultima alteracao: 09/12/2021
 *
 */

#include "Sensor.h"

int Sensor::getID(){
  return identificacao;
}

void Sensor::setID(int newID){
  identificacao = newID;
}

float Sensor::getValor(){
  return valor;
}

void Sensor::setValor(float newValor){
  valor = newValor;
}
```

▼ `Temperatura.h`

```
/*
 * file: Temperatura.h
 *
 * Descricao: Classe Temperatura utilizada para re
alizar a leitura do sensor de temperatura, e data
 data/hora da leitura realizada.
 *
 * Autor: Eduardo Augusto Bezerra
 * Data: 09/12/2021
 *
 * Ultima Alteracao: Eduardo Augusto Bezerra
 * Data da ultima alteracao: 09/12/2021
 *
 */

#include "Sensor.cpp"
#include "ClockCalendar.cpp"

class Temperatura : public Sensor {
      ClockCalendar dataHora;
  protected:
    void readSensor();
};
```

▼ `Temperatura.cpp`

```
/*
 * file: Temperatura.cpp
 *
 * Descricao: Implementacao da Classe Temperatura
 utilizada para realizar a leitura do sensor de te
mperatura, e data data/hora da leitura realizada.
 *
 * Autor: Eduardo Augusto Bezerra
 * Data: 09/12/2021
 *
 * Ultima Alteracao: Eduardo Augusto Bezerra
 * Data da ultima alteracao: 09/12/2021
 *
 */

#include "Temperatura.h"
#include <cstdlib>   // para usar srand() e rand()
#include <ctime>     // para usar time()

void Temperatura::readSensor(){
  // Data/hora da leitura do sensor - substituir o
s parametros do construtor do ClockCalendar por ti
me()
  dataHora = ClockCalendar (2021, 6, 30, 11, 59, 5
5, true);
  // Simulacao de leitura de sensor
  srand (static_cast <unsigned> (time(0)));
  setValor(static_cast <float> (rand()) / static_c
ast <float> (RAND_MAX));
}
```