TP-2 Automação residencial

AUTHOR
Versão 1.0

Sumário

Table of contents

TP-PDS2

Trabalho prático da disciplina PDS2, sistema de automação residencial

Alunos: Arthur Caetano de Oliveira - 2023039309; Breno Pereira Miranda - 2023038930; Davi Nascimento Andrade da Silva - 2023038817; Lucas Abreu Velloso - 2023038795; Lucas Monteiro Henriques- 2023038868;

O Manual de estilo ultilizado no projeto pode ser acessado pelo link: `https://google.github.io/styleguide/cppguide.html#C++_Version`

1. Apresentação do Problema:

Nos últimos anos, principalmente durante a pandemia do Covid-19, houve o surgimento e crescimento explosivo da indústria 4.0. Além disso, houve grande desenvolvimento e popularização das tecnologias da informação, que tomaram papéis importantes em todas as áreas de consumo, auxiliando as pessoas a resolverem diversos problemas e a melhorarem a qualidade de vida. Uma das novas aplicações que surgiu nesse crescimento foi a automação residencial, atra- vés de soluções de engenharia de controle e automação, que permite controlar diversos aspectos de uma casa de forma automática, como iluminação, temperatura, segurança, etc.

A falta de automação residencial pode gerer diversas problematicas como falta de conectividade e personalização , conforto e segurnaça comprometido,além de ineficiencia energetica e consequentimente mais gastos financeiros.

Tendo em vista a relevância do tópico apresentado e a proximidade com nossa área de estudo no curso, decidimos realizar esse trabalho com o objetivo de desenvolver um sistema de automação residencial, que visa controlar e automatizar vários aspectos de uma casa comum, como controlar ativação de luzes, abertura de cortinas, ar condicionado,criação de modos que empregam o conjunto dessas funcionalidades etc.

1. Visão Geral da Solução:

Nossa solução visa que o usuário consiga controlar , de forma rapida ,facil e totalmente remota, sua casa. O programa ira conseguir controlar,por meio de um menu no terminar, cada objeto separadamente ou de maneira geral tanto no cômodo quanto na casa,todas as entidades do programa possuem classes de objetos que são organizadas segundo a lógica de programação orientada a objeto.

1. Estrutura do Projeto:

O projeto será separado em classes, 7 no total, cada classe irá possuir um arquivo .h (encontrados presente na pasta "include") e um .cc (encontrados na pasta "src"),além de uma main que possui o código referente ao display do menu("encontrado na pasta src).No própio github é possível analisar a presença de pasta "testes" no qual mostra testes minimalista sobre as funcionalidades presentes nas classe ar_condicionado, cortina ,trancajanela,lampada. Além do uso de um makefile utilizado para compilar o código de maneira mais eficiente e rápida.

Todos os arquivos necessários e que serão ultilizados ficara disponível no link: `https://github.com/DaviNascimentoAndrade/TP-PDS2`

1. Instruções de Instalação:

O usuário precisará de: • Ambiente de edição • Compilador(compatível com o sistema operacional do usuário)

1. Instruções de Uso:

O usuário, através de um menu disponibilizado no terminal, terá a oportunidade de interagir com o sistema, empregando números e palavras para realizar diversas transações.

O usuário terá a capacidade de personalizar individualmente cada dispositivo, podendo alterar seu status por meio das funcionalidades específicas disponibilizadas para cada aparelho, bem como modificar seu nome. Além disso, será possível modificar o status de um grupo de dispositivos similares. Essas funcionalidades são facilmente acessíveis por meio de escolhas simples, utilizando tanto números quanto palavras.

Além de criar modos pré-determinados pelo úsuario sobre a casa também utilizado palavras e números.

1. Principais Dificuldades:

Entre as dificuldades apresentadas no Tp-2 ,a integração das classes, as descrições dos cartões CRCs, o controle de versão por meio das funcionalidades do GitHub e Git, o tratamento de exceções, e a concepção de um design abrangente na main que incorpore todas as funcionalidades do projeto representam desafios significativos para os colaboradores envolvidos.

Índice dos Componentes

Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Índice dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Classes

Referência da Classe ArCondicionado

`#include <ar_condicionado.h>`

Membros Públicos

- **ArCondicionado** ()
- void **SetNome** (string nome)
- string **Nome** ()
- void **SetIntensidade** (int in)
- int **Intensidade** ()
- void **SetLigar** (bool li)
- bool **Ligado** ()
- void **SetTemperatura** (int temp)
- int **Temperatura** ()

Construtores e Destrutores

ArCondicionado::ArCondicionado ()

```
8                                   {
9   intensidade_ = 0;
10   temperatura_ = 0;
11   ligado_ = 0;
12 }
```

Documentação das funções

int ArCondicionado::Intensidade ()

```
31                                    {
32   return intensidade_;
33 }
```

bool ArCondicionado::Ligado ()

```
39                                {
40   return ligado_;
41 }
```

string ArCondicionado::Nome ()

```
18                                {
19   return nome_;
20 }
```

void ArCondicionado::SetIntensidade (int *in*)

```
22                                                {
23   if(in >= 0 && in <= 100){
24     intensidade_ = in;
25   }
26   else{
27     cout << "Intensidade inválida, selecione um valor de 0 a 100";
28   }
29 }
```

void ArCondicionado::SetLigar (bool *li*)

```
35                                          {
36   ligado_ = li;
```

```
37 }
```

void ArCondicionado::SetNome (string *nome*)

```
14                                          {
15   nome_ = nome;
16 }
```

void ArCondicionado::SetTemperatura (int *temp*)

```
43                                              {
44   if(temp > 16 || temp < 32){
45     temperatura_ = temp;
46   }
47   else{
48     cout << "Temperatura inválida. Por favor, insira um dos valores válidos";
49   }
50 }
```

int ArCondicionado::Temperatura ()

```
52                                  {
53   return temperatura_;
54 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/**ar_condicionado.h**
- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/**ar_condicionado.cc**

Referência da Classe Casa

`#include <casa.h>`

Membros Públicos

- **Casa** ()
- void **SetNome** (string nm)
- string **Nome** ()
- void **ListarComodos** ()
- void **AtivarModo** (string nm)
- void **ListarModos** ()
- void **AdicionarComodo** (string nm)
- void **RemoverComodo** (string nm)

Atributos Públicos

- map< string, **Comodo** > **comodos_**
- map< string, **Casa** > **modos_**

Construtores e Destrutores

Casa::Casa ()

```
8           {
9
10 }
```

Documentação das funções

void Casa::AdicionarComodo (string *nm*)

```
57                                      {
58   bool existe = false;               //testa se o nome já é usado no map
59   for(auto it: comodos_){
60     if(it.first == nm){
61       existe = true;
62       cout << "Esse cômodo já existe" <<endl;
63       return;
64     }
65   }
66   if(!existe){
67     comodos_[nm];  //o map cria um valor apenas com uma chave, desde que ela não
exista
68   }
69 }
```

void Casa::AtivarModo (string *nm*)

```
31                                  {
32     bool existe = false;              //testa se o modo existe
33     for(auto it: modos ){
34       if(it.first == nm){
35         existe = true;
36         break;
37       }
38     }
39   if(!existe){
40     cout << "Esse modo não existe!" << endl;
41     return;
42     }
43
44     comodos_ = modos_[nm].comodos_;
45     cout << "Modo "<< nm << " ativado!" << endl;
```

```
46
47 }
```

void Casa::ListarComodos ()

```
23                          {
24   cout << "Comôdos: "<<endl;
25   for(auto it : comodos_){
26     cout << it.first << endl;
27   }
28 }
```

void Casa::ListarModos ()

```
50                      {
51   for(auto it = modos_.begin(); it != modos_.end(); it++){
52     cout << it->first << endl;
53   }
54 }
```

string Casa::Nome ()

```
18                  {
19   return nome_;
20 }
```

void Casa::RemoverComodo (string *nm*)

```
72                                  {
73   bool existe = false;             //testa se existe
74   for(auto it: comodos_){
75     if(it.first == nm){
76       existe = true;
77       comodos_.erase(nm);
78       return;
79     }
80   }
81   if(!existe){
82     cout << "Esse cômodo não existe";  //o map cria um valor apenas com uma chave,
desde que ela não exista
83   }
84 }
```

void Casa::SetNome (string *nm*)

```
13                              {
14   nome_ = nm;
15 }
```

Atributos

map<string, Comodo> Casa::comodos_

map<string, Casa> Casa::modos_

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/**casa.h**
- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/**casa.cc**

Referência da Classe Comodo

`#include <comodo.h>`

Membros Públicos

- **Comodo** ()
- void **AdicionarDispositivo** (int tipo, string nome)
- void **ListarDispositivos** (int tipo)
- void **RemoverDispositivo** (int tipo, string nome)
- void **ConfigurarTodos** (int tipo)
- void **SetNome** (string nome)
- string **Nome** ()

Atributos Públicos

- map< string, **Lampada** > **lampadas_**
- map< string, **Cortina** > **cortinas_**
- map< string, **ArCondicionado** > **ares_condicionados_**
- map< string, **Tranca** > **trancas_**
- map< string, **Janela** > **janelas_**

Construtores e Destrutores

Comodo::Comodo ()

```
13              {
14
15 }
```

Documentação das funções

void Comodo::AdicionarDispositivo (int *tipo*, string *nome*)

```
17                                                      {
18   switch (tipo)
19   {
20     // Case = 1, adiciona Lampada
21     case 1:
22       lampadas_[nome] = Lampada();
23       break;
24     // Case = 2, adiciona Cortina
25     case 2:
26       cortinas_[nome] = Cortina();
27       break;
28     case 3:
29     // Case = 3, adiciona Ar condicionado
30       ares_condicionados_[nome] = ArCondicionado();
31       break;
32     case 4:
33     // Case = 4, adiciona Tranca
34       trancas_[nome] = Tranca();
35       break;
36     case 5:
37     // Case = 5, adiciona Janela
38       janelas_[nome] = Janela();
39       break;
40     default:
41       // Tratamento de exceção do tipo
42       cout<<"Tipo invalido"<<endl;
43       break;
44   }
45 }
```

void Comodo::ConfigurarTodos (int *tipo*)

```
148                                     {
149   switch (tipo)
150   {
151     //Configura todos os dispositivos do tipo Lampada
152     case 1:{
153       int x;
154       string y;
155       cout<<"Digite a intensidade da Lampada de 0 a 100: ";
156       cin>>x;
157       cout<<"Digite a cor da Lampada (Amarelo, Vermelho, Azul, Branco, Laranja,
Verde ou Roxo): ";
158       cin>>y;
159
160       for(auto it : lampadas_){
161         lampadas_[it.first].SetIntensidade(x);
162         lampadas_[it.first].SetCor(y);
163       }
164       break;
165     }
166     //Configura todos os dispositivos do tipo Cortina
167     case 2:{
168       int x;
169       cout<<"Digite a intensidade da Cortina: ";
170       cin>>x;
171       for(auto it : cortinas_){
172         cortinas_[it.first].SetIntensidade(x);
173       }
174       break;
175     }
176     //Configura todos os dispositivos do tipo Ar condicionado
177     case 3:{
178       bool l;
179       cout<<"Digite true para ligar o ar condicionado ou false para desligar: ";
180       cin>>l;
181       int i;
182       cout<<"Digite a intensidade do ar condicionado: ";
183       cin>>i;
184       int x;
185       cout<<"Digite a temperatura do ar condicionado: ";
186       cin>>x;
187       for(auto it : ares_condicionados_){
188         if(it.second.Ligado() == true){
189           ares_condicionados_[it.first].SetLigar(true);
190           ares_condicionados_[it.first].SetIntensidade(i);
191           ares_condicionados_[it.first].SetTemperatura(x);
192         }
193       }
194       break;
195     }
196     //Configura todos os dispositivos do tipo Ar condicionado
197     case 4:{
198       cout<<"Digite o modo da tranca : ";
199       bool x;
200       for(auto it : trancas_){
201         trancas_[it.first].SetAtiva(x);
202       }
203       break;
204     }
205     //Configura todos os dispositivos do tipo janela
206     case 5:{
207       int x;
208       bool y;
209       cout<<"Digite a config. da tranca (1 fechado, 0 aberto): ";
210       cin>>y;
211       cout<<"Digite a intensidade da Janela: ";
212       cin>>x;
213       for(auto it : janelas_){
214         janelas_[it.first].SetIntensidade(x);
215         janelas_[it.first].SetTranca(y);
216       }
217       break;
218     }
219     // Tratamento de exceção do tipo
220     default:
221       cout<<"Tipo invalido"<<endl;
```

```
222       break;
223     }
224   }
```

void Comodo::ListarDispositivos (int *tipo*)

```
48                                          {
49   switch (tipo)
50   {
51     // Case = 1, imprime todas as Lampadas
52     case 1:
53       for(auto it : lampadas_){
54         cout <<it.first;
55         cout <<" [intensidade: "<<lampadas_[it.first].Intensidade();
56         cout <<", cor: "<<lampadas_[it.first].Cor()<<"]"<<endl;
57       }
58       break;
59     // Case = 2, imprime todas as Cortinas
60     case 2:
61       for(auto it : cortinas_){
62         cout <<it.first;
63         cout <<" [intensidade: "<<cortinas_[it.first].Intensidade();
64         cout <<"]"<<endl;
65       }
66       break;
67     case 3:
68     // Case = 3, imprime todos os Ar condicionados
69       for(auto it : ares_condicionados_){
70         cout <<it.first;
71         cout <<" [intensidade: "<<ares_condicionados_[it.first].Intensidade();
72         cout <<", temperatura: "<<ares_condicionados_[it.first].Temperatura();
73         cout <<", status: "<<ares_condicionados_[it.first].Ligado()<<"]"<<endl;
74       }
75       break;
76     case 4:
77     // Case = 4, imprime todas as Trancas
78       for(auto it : trancas_){
79         cout <<it.first;
80         cout <<" [status: "<<trancas_[it.first].Ativa();
81         cout <<"]"<<endl;
82       }
83       break;
84     case 5:
85       // Case = 5, imprime todas as janelas
86       for(auto it : janelas_){
87         cout <<it.first;
88         cout <<" [intensidade: "<<janelas_[it.first].Intensidade();
89         cout <<", status da tranca: "<<janelas_[it.first].Tranca()<<"]"<<endl;
90       }
91       break;
92     default:
93       // Tratamento de exceção do tipo
94       cout<<"Tipo invalido"<<endl;
95       break;
96   }
97 }
```

string Comodo::Nome ()

```
230                     {
231   return nome_;
232 }
```

void Comodo::RemoverDispositivo (int *tipo*, string *nome*)

```
100                                                 {
101   switch (tipo)
102   {
103     case 1:
104       //Remove um dispositivo, do tipo Lampada, caso haja o mesmo nome
105       for(auto it : lampadas ){
106         if(it.first == nome){
107           lampadas_.erase(nome);
108           break;
109         }
110       }
111
```

```
112     //Remove um dispositivo, do tipo Cortina, caso haja o mesmo nome
113     case 2:
114       for(auto it : cortinas_){
115         if(it.first == nome){
116           cortinas_.erase(nome);
117           break;
118         }
119       }
120     //Remove um dispositivo, do tipo Ar condicionado, caso haja o mesmo nome
121     case 3:
122       for(auto it : ares_condicionados_){
123         if(it.first == nome){
124           ares_condicionados_.erase(nome);
125           break;
126         }
127       }
128     //Remove um dispositivo, do tipo Tranca, caso haja o mesmo nome
129     case 4:
130       for(auto it : trancas_){
131         if(it.first == nome){
132           trancas_.erase(nome);
133           break;
134         }
135       }
136     //Remove um dispositivo, do tipo Janela, caso haja o mesmo nome
137     case 5:
138       for(auto it : janelas_){
139         if(it.first == nome){
140           janelas_.erase(nome);
141           break;
142         }
143       }
144   }
145 }
```

void Comodo::SetNome (string *nome*)

```
226                                 {
227   nome_ = nome;
228 }
```

Atributos

map<string,ArCondicionado> Comodo::ares_condicionados_

map<string,Cortina> Comodo::cortinas_

map<string, Janela> Comodo::janelas_

map<string,Lampada> Comodo::lampadas_

map<string,Tranca> Comodo::trancas_

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/**comodo.h**
- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/**comodo.cc**

Referência da Classe Cortina

`#include <cortina.h>`

Membros Públicos

- **Cortina** ()
- void **SetIntensidade** (int novaIntensidade)
- int **Intensidade** ()
- string **Nome** ()
- void **SetNome** (string novoNome)

Construtores e Destrutores

Cortina::Cortina ()

```
7                 {
8   intensidade_cortina_ = 0;
9 }
```

Documentação das funções

int Cortina::Intensidade ()

```
20                          {
21   return intensidade_cortina_;
22 }
```

string Cortina::Nome ()

```
24                      {
25   return nome_;
26 }
```

void Cortina::SetIntensidade (int *novaIntensidade*)

```
11                                                  {
12   if (novaIntensidade >= 0 && novaIntensidade <= 100) {
13     intensidade_cortina_ = novaIntensidade;
14   }
15   else {
16     cout << "A intensidade deve estar entre 0 e 100." <<endl;
17   }
18 }
```

void Cortina::SetNome (string *novoNome*)

```
28                                      {
29   nome_ = novoNome;
30 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/**cortina.h**
- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/**cortina.cc**

Referência da Classe Janela

`#include <janela.h>`

Membros Públicos

- **Janela** ()
- void **SetNome** (string nm)
- string **Nome** ()
- void **SetIntensidade** (int in)
- int **Intensidade** ()
- void **SetTranca** (bool tr)
- bool **Tranca** ()

Construtores e Destrutores

Janela::Janela ()

```
7               {
8   tranca_ = 0;
9   intensidade_ = 0;
10 }
```

Documentação das funções

int Janela::Intensidade ()

```
33                          {
34     return intensidade_;
35 }
```

string Janela::Nome ()

```
18                      {
19   return nome_;
20 }
```

void Janela::SetIntensidade (int *in*)

```
23                                  {
24   if(in < 0 || in > 100){
25     cout<< "A intensidade deve estar entre 0 e 100"<<endl;
26   }
27   else{
28     intensidade_ = in;
29   }
30 }
```

void Janela::SetNome (string *nm*)

```
13                                {
14   nome_ = nm;
15 }
```

void Janela::SetTranca (bool *tr*)

```
38                                {
39     tranca_ = tr;
40 }
```

bool Janela::Tranca ()

```
43                      {
44     return tranca_;
45 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/**janela.h**
- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/**janela.cc**

Referência da Classe Lampada

`#include <lampada.h>`

Membros Públicos

- **Lampada** ()
- void **SetIntensidade** (int i)
- int **Intensidade** ()
- void **SetCor** (string c)
- string **Cor** ()
- void **ListarCores** ()
- void **SetNome** (string name)
- string **Nome** ()

Construtores e Destrutores

Lampada::Lampada ()

```
8                 {
9   intensidade_ = 0;
10 }
```

Documentação das funções

string Lampada::Cor ()

```
38                    {
39   return cor_;
40 }
```

int Lampada::Intensidade ()

```
21                        {
22   return intensidade_;
23 }
```

void Lampada::ListarCores ()

```
42                          {
43   for(auto it = cores_.begin(); it != cores_.end(); it++){
44     std::cout << *it << endl;
45 }
46 }
```

string Lampada::Nome ()

```
52                    {
53   return nome_;
54 }
```

void Lampada::SetCor (string *c*)

```
25                          {
26   bool valido = false;
27   for(auto it = cores_.begin(); it != cores_.end(); it++){
28     if(c == *it){
29       cor_ = *it;
30       valido = true;
31     }
32   }
33   if(!(valido)){
34     cout<< "Cor inválida, consulte as cores disponíveis"<<endl;
35   }
36 }
```

void Lampada::SetIntensidade (int *i*)

```
12                                      {
13   if(i < 0 || i > 100){
14     cout<< "A intensidade deve estar entre 0 e 100"<<endl;
15     }
16   else{
17     intensidade_ = i;
18     }
19 }
```

void Lampada::SetNome (string *name*)

```
48                                      {
49   nome_ = name;
50 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/**lampada.h**
- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/**lampada.cc**

Referência da Classe Tranca

`#include <tranca.h>`

Membros Públicos

- **Tranca** ()
- void **SetNome** (string name)
- string **Nome** ()
- void **SetAtiva** (bool ativo)
- bool **Ativa** ()

Construtores e Destrutores

Tranca::Tranca ()

```
8               {
9   ativa_=0;
10 }
```

Documentação das funções

bool Tranca::Ativa ()

```
24                  {
25   return ativa_;
26 }
```

string Tranca::Nome ()

```
16                   {
17   return nome_;
18 }
```

void Tranca::SetAtiva (bool *ativo*)

```
20                            {
21   ativa_=at;
22 }
```

void Tranca::SetNome (string *name*)

```
12                                {
13   nome_=name;
14 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/**tranca.h**
- D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/**tranca.cc**

Arquivos

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/ar_condicionado.h

```
#include <iostream>
#include <string>
```

Componentes

class **ArCondicionado**

ar_condicionado.h

Ir para a documentação desse arquivo.

```
1 #ifndef ARCONDICIONADO_H_
2 #define ARCONDICIONADO_H_
3
4 #include<iostream>
5 #include <string>
6
7 using namespace std;
8
9 class ArCondicionado {
10
11   public:
12
13     // cria um novo arcondicionado sem nome
14     ArCondicionado();
15
16     // coloca um nome para o ar condicionado
17     void SetNome(string nome);
18
19     // retorna o nome do ar condicionado
20     string Nome();
21
22     // seta a intensidade
23     void SetIntensidade(int in);
24
25     // retorna a intensidade
26     int Intensidade();
27
28     // liga ou desliga o ar condicionado(0 desligado)
29     void SetLigar(bool li);
30
31     // retorna status da tranca da janela
32     bool Ligado();
33
34     // configura temperatura do ar-condicionado.
35     //PRECONDIÇÃO:a temperatura deve ter valor entre 16 e 30
36     void SetTemperatura(int temp);
37
38     //retorna temperatura do ar condicionado
39     int Temperatura ();
40
41   private:
42     int temperatura_;
43     bool ligado_;
44     int intensidade_;
45     string nome_;
46 };
47
48 #endif // ARCONDICIONADO_H_
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/casa.h

```
#include <iostream>
#include <map>
#include "comodo.h"
#include "lampada.h"
#include "ar_condicionado.h"
#include "tranca.h"
#include "janela.h"
#include "cortina.h"
#include <string>
```

Componentes

class **Casa**

casa.h

Ir para a documentação desse arquivo.

```
1 #ifndef CASA_H_
2 #define CASA_H_
3
4 #include<iostream>
5 #include<map>
6 #include "comodo.h"
7 #include "lampada.h"
8 #include "ar_condicionado.h"
9 #include "tranca.h"
10 #include "janela.h"
11 #include "cortina.h"
12 #include<string>
13
14 using namespace std;
15
16 class Casa {
17
18   public:
19
20     // cria uma novo casasem nome
21     Casa();
22
23     // coloca um nome para a casa
24     void SetNome(string nm);
25
26     // retorna o nome da casa
27     string Nome();
28
29     //Lista todos os cômodos da casa;
30     void ListarComodos();
31
32     //Aplica as conficurações de um modo salvo;
33     void AtivarModo (string nm);
34
35     //Lista os modos da casa;
36     void ListarModos();
37
38     //Adiciona um cômodo para que o usuário possa edita-lo;
39     void AdicionarComodo(string nm);
40
41     //Apaga um cômodo e todos os dispositivos nele;
42     //Precondiçao: deve haver no minimo um comodo
43     void RemoverComodo(string nm);
44
45     map<string, Comodo> comodos_;
46     map<string, Casa> modos_;
47
48   private:
49     string nome_;
50
51 };
52
53 #endif // CASA_H_
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/comodo.h

```
#include <map>
#include <string>
#include "lampada.h"
#include "ar_condicionado.h"
#include "tranca.h"
#include "janela.h"
#include "cortina.h"
```

Componentes

class Comodo Definições e Macros

- #define **COMODO_H**

Definições e macros

#define COMODO_H

comodo.h

Ir para a documentação desse arquivo.

```
1 #ifndef COMODO_H_
2 #define COMODO_H
3
4 #include <map>
5 #include <string>
6 #include "lampada.h"
7 #include "ar_condicionado.h"
8 #include "tranca.h"
9 #include "janela.h"
10 #include "cortina.h"
11
12 class Comodo{
13
14   public:
15
16     //Cria um comodo vazio
17     Comodo();
18
19     //Adiciona um novo dispositivo de algum tipo
20     void AdicionarDispositivo(int tipo, string nome);
21
22     //Lista todos os dispositivos de um tipo, sendo
Lampada,Cortina,Arcondicionado,Tranca e Janela, 1,2,3,4 e 5 respectivamente
23     void ListarDispositivos(int tipo);
24
25     //Remove um dispositivo, de um tipo, do comodo, sendo
Lampada,Cortina,Arcondicionado,Tranca e Janela, 1,2,3,4 e 5 respectivamente
26     //Precondição: o dispositivo deve existir para que ele possa ser removido
27     void RemoverDispositivo(int tipo, string nome);
28
29     //Configura todos os dispositivos de um mesmo tipo no comodo, sendo
Lampada,Cortina,Arcondicionado,Tranca e Janela, 1,2,3,4 e 5 respectivamente
30     void ConfigurarTodos(int tipo);
31
32     //Configura o nome do comodo;
33     void SetNome(string nome);
34
35     //Retorna o nome do Comodo
36     string Nome();
37
38     map<string,Lampada> lampadas_;
39     map<string,Cortina> cortinas_;
40     map<string,ArCondicionado> ares_condicionados_;
41     map<string,Tranca> trancas_;
42     map<string, Janela> janelas_;
43
44   private:
45
46     string nome_;
47 };
48
49
50 #endif // COMODO_H_
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/cortina.h

```
#include <string>
```

Componentes

class **Cortina**

cortina.h

Ir para a documentação desse arquivo.

```
1 #ifndef CORTINA_H_
2 #define CORTINA_H_
3
4 #include <string>
5
6 using namespace std;
7 class Cortina {
8   public:
9     // Construtor da classe Cortina
10     Cortina(); // Construtor
11
12     // Configura a quantidade que a cortina está fechada ou aberta.
13     // @param novaIntensidade: A nova intensidade da cortina (0-100).
14     void SetIntensidade(int novaIntensidade);
15
16     // Retorna o status de iluminação da cortina.
17     int Intensidade();
18
19     // Retorna o nome da cortina.
20     string Nome();
21
22     // Coloca um nome na cortina.
23     void SetNome(string novoNome);
24
25   private:
26     int intensidade_cortina_;
27     string nome_;
28 };
29
30 #endif // CORTINA_H_
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/janela.h

```
#include <iostream>
```

Componentes

class **Janela**

janela.h

Ir para a documentação desse arquivo.

```
1 #ifndef JANELA_H_
2 #define JANELA_H_
3
4 #include<iostream>
5
6 using namespace std;
7
8 class Janela {
9
10   public:
11
12     // cria uma janela sem nome
13     Janela();
14
15     // muda o nome da janela
16     void SetNome(string nm);
17
18     // retorna o nome da janela
19     string Nome();
20
21     // muda intensidade da abertura da janela
22     void SetIntensidade(int in);
23
24     // retorna intensidade da janela
25     int Intensidade();
26
27     // muda a tranca da janela (0 aberto)
28     void SetTranca(bool tr);
29
30     // retorna status da tranca da janela
31     bool Tranca();
32
33   private:
34
35     string nome_;
36     int intensidade_;
37     bool tranca_;
38 };
39
40 #endif // JANELA_H_
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/lampada.h

```
#include <iostream>
#include <string>
#include <vector>
```

Componentes

class **Lampada**

lampada.h

Ir para a documentação desse arquivo.

```
1 #ifndef LAMPADA_H_
2 #define LAMPADA_H_
3
4 #include <iostream>
5 #include <string>
6 #include <vector>
7 using namespace std;
8
9 class Lampada{
10 public:
11
12 // Cria uma lâmpada sem nome com intensidade 0;
13 Lampada();
14
15 //Configura a intensidade da lâmpada;
16 //PRECONDIÇÃO: A intensidade deve ser de 0 a 100;
17 void SetIntensidade(int i);
18
19 //Retorna intensidade da lâmpada;
20 int Intensidade ();
21
22 //PRECONDIÇÃO: Devem ser inseridas somente cores válidas, como amarelo,vermelho,
azul,branco,laranja,verde ou roxo
23 //Seta a cor da lâmpada
24 void SetCor(string c);
25
26 //Retorna a cor atual da lâmpada;
27 string Cor();
28
29 //Lista as cores que a lâmpada pode mostrar;
30 void ListarCores();
31
32 //Muda o nome do dispositivo;
33 void SetNome(string name);
34
35 //Retorna o nome do dispositivo.
36 string Nome();
37
38
39 private:
40 string nome_;
41 string cor_;
42 int intensidade_;
43 vector<string> cores_ ={"Amarelo",
"Vermelho","Azul","Verde","Roxo","Branco","Laranja","amarelo"
44 ,"vermelho","azul","verde","roxo","branco","laranja"};
45 };
46 #endif // LAMPADA_H_
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/include/tranca.h

```
#include <iostream>
```

Componentes

class **Tranca**

tranca.h

Ir para a documentação desse arquivo.

```
1 #ifndef TRANCA_H_
2 #define TRANCA_H_
3
4 #include<iostream>
5
6 using namespace std;
7
8 class Tranca {
9
10   public:
11
12     // Cria uma traca sem nome
13     Tranca();
14
15     // Muda o nome da tranca
16     void SetNome(string name);
17
18     // Retorna o nome da tranca
19     string Nome();
20
21     // Muda estado de ativação da tranca
22     void SetAtiva(bool ativo);
23
24     // Retorna estado de ativação da tranca
25     bool Ativa();
26
27   private:
28     string nome_;
29     bool ativa_;
30
31 };
32
33 #endif // TRANCA_H_
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/README.md

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/ar_condicionado.cc

```
#include "../include/ar_condicionado.h"
#include <string>
#include <iostream>
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/casa.cc

```
#include "../include/casa.h"
#include <iostream>
#include <string>
#include <map>
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/comodo.cc

```
#include "../include/comodo.h"
#include "../include/lampada.h"
#include "../include/ar_condicionado.h"
#include "../include/tranca.h"
#include "../include/janela.h"
#include "../include/cortina.h"
#include <iostream>
#include <string>
#include <map>
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/cortina.cc

```
#include "../include/cortina.h"
#include <iostream>
#include <string>
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/janela.cc

```
#include "../include/janela.h"
#include <iostream>
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/lampada.cc

```
#include "../include/lampada.h"
#include <string>
#include <iostream>
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/main.cc

```
#include "../include/casa.h"
#include <iostream>
#include <string>
#include <map>
```

Funções

- void **MenuComodos** (**Casa** &house)
- void **MenuModos** (**Casa** &house)
- void **AdicionarComodo** (**Casa** &house, string nome)
- void **RemoverComodo** (**Casa** &house, string nome)
- void **MenuComodo** (**Casa** &house, string comodo)
- void **MenuDispositivo** (**Casa** &house, string comodo)
- void **MenuLampada** (**Comodo** &comodo)
- void **MenuCortina** (**Comodo** &comodo)
- void **MenuArCondicionado** (**Comodo** &comodo)
- void **MenuTrancas** (**Comodo** &comodo)
- void **MenuJanelas** (**Comodo** &comodo)
- void **AdicionarModo** (**Casa** &house, string nome)
- void **RemoverModo** (**Casa** &house, string nome)
- void **Titulo** (string titulo)
- void **Wait1** ()
- int **main** ()

Funções

void AdicionarComodo (Casa & *house*, string *nome*)

void AdicionarModo (Casa & *house*, string *nome*)

int main ()

```
97          {
98
99   Casa casa;
100
101   // MENU PRINCIPAL
102   int esc = 0;
103   while (esc != 4){
104
105     esc = 0;
106     system("clear");
107     Titulo("MENU PRINCIPAL - "+ casa.Nome());
108     cout<<"  1 - Menu de comodos\n";
109     cout<<"  2 - Menu de modos\n";
110     cout<<"  3 - Mudar nome da casa\n";
111     cout<<"  4 - Sair\n";
112
113     if(casa.Nome().size() == 0){
114       cout<<"(Dica: dê um nome para sua casa com a opção n° 3)"<<endl;
115     }
116
117     esc = 0;
118     while(esc < 1 || esc > 4){
119       cout<<"Digite sua escolha: ";
120       cin>>esc;
121       if(esc < 1 || esc > 4){
122         cout<<"Escolha invalida!"<<endl;
123         Wait1();
```

```
124       }
125     }
126
127     // MENU DE COMODOS
128     if(esc == 1){
129       MenuComodos(casa);
130     }
131
132     // MENU DE MODOS
133     else if(esc == 2){
134       MenuModos(casa);
135     }
136
137     // MUDAR NOME
138     else if(esc == 3){
139       string nm = "";
140       cout<<"Digite o novo nome: ";
141       cin.ignore();
142       getline(cin, nm);
143       casa.SetNome(nm);
144     }
145   }
146   return 0;
147 }
```

void MenuArCondicionado (Comodo & *comodo*)

```
648                                       {
649
650   int esc = 0;
651   while(esc != 6){
652
653     system("clear");
654     Titulo("ARES CONDICIONADOS DE " + comodo.Nome());
655     cout<<"  1 - Listar Ares Condicionados\n";
656     cout<<"  2 - Configurar Ar Condicionado\n";
657     cout<<"  3 - Adicionar Ar Condicionado\n";
658     cout<<"  4 - Remover Ares Condicionados\n";
659     cout<<"  5 - Configurar todas os Ares Condicionados\n";
660     cout<<"  6 - Voltar\n";
661
662     esc = 0;
663     while(esc < 1 || esc > 6){
664       cout<<"Digite sua escolha: ";
665       cin>>esc;
666       if(esc < 1 || esc > 6){cout<<"Escolha invalida!"<<endl;}
667     }
668
669     if((esc == 2 || esc == 4 || esc == 5) && comodo.ares_condicionados_.size() ==
0){
670       cout<<"Não há ares! Escolha inválida!"<<endl;
671       esc = 0;
672       Wait1();
673     }
674
675     // listar a ar condicionado
676     if(esc == 1){
677
678       cout << "Ares: " << endl;
679       comodo.ListarDispositivos(3);
680       Wait1();
681     }
682
683     //Configurar um ar condicionado
684     if(esc == 2){
685
686       cout<<"Ares para configurar: \n";
687       comodo.ListarDispositivos(3);
688       string ArCondicionado = "";
689
690       // USUÁRIO ESCOLHE A LAMAPADA QUE QUER ALTERAR
691       bool ex = 0;
692       while(!ex){
693
694         cout<<"Digite o nome do ar condicionado que deseja configurar: ";
695         cin.ignore();
696         getline(cin, ArCondicionado);
```

```
697
698         for(auto it :comodo.ares_condicionados_){
699           if(it.first == ArCondicionado){
700             ex = 1;
701           }
702         }
703         if(!ex){cout<<"Esse ar condicionado não existe!\n";}
704       }
705
706       int intensidade = 0;
707       bool ligado = false;
708       string resposta = "";
709       int temperatura = 0;
710       //setando a intensidade
711       cout<< "Digite a intensidade da cortina de 0 a 100: ";
712       cin>>intensidade;
713       comodo.ares_condicionados_[ArCondicionado].SetIntensidade(intensidade);
714
715       cout<< "Digite se o ar esta ligado ou não (sim ou não) ";
716       cin>>resposta;
717       if(resposta == "SIM" || resposta == "Sim" || resposta == "sim"){
718         ligado = true;
719       }
720       else{ ligado = false; }
721       comodo.ares_condicionados_[ArCondicionado].SetLigar(ligado);
722       cout<< "Digite a temperatura do Ar Condicionado(de 16 a 32 graus): ";
723       cin>>temperatura;
724       comodo.ares_condicionados_[ArCondicionado].SetTemperatura(temperatura);
725       cout << "ArCondicionado " << ArCondicionado << " configurado!";
726       Wait1();
727     }
728
729     //ADICIONAR AR CONDIONADO
730     if(esc == 3){
731       string nome = "";
732       cout<< "Digite o nome do novo ar condicionado: ";
733       cin.ignore();
734       getline(cin, nome);
735       comodo.AdicionarDispositivo(3, nome);
736       cout << "ArCondicionado " << nome << " salvo!";
737       Wait1();
738     }
739
740     //REMOVER AR CONDICIONADO
741     if(esc == 4){
742
743       cout << "Ares:" << endl;
744       comodo.ListarDispositivos(3);
745
746       // USUÁRIO ESCOLHE O AR CONDICIONADO QUE QUER REMOVER
747       bool ex = 0;
748       string ArCondicionado = "";
749       while(!ex){
750
751         cout<<"Digite o nome do ar condicionado que deseja remover: ";
752         cin.ignore();
753         getline(cin, ArCondicionado);
754
755           for(auto it : comodo.ares_condicionados_){
756             if(it.first == ArCondicionado){
757              ex = 1;
758             }
759           }
760           if(!ex){cout<<"Esse ar condicionado não existe!\n";}
761       }
762
763       comodo.RemoverDispositivo(2,ArCondicionado);
764       cout << "ArCondicionado " << ArCondicionado << " removido!";
765       Wait1();
766     }
767
768
769     //CONFIGURAR TODAS OS ARES CONDICIONADOS
770     if(esc == 5){
771       comodo.ConfigurarTodos(3);
772       cout << "Ares Condicionados configurados!";
773       Wait1();
```

```
774     }
775   }
776 }
```

void MenuComodo (Casa & *house*, string *comodo*)

```
313                                             {
314 
315   int esc = 0;
316   while(esc != 3){
317 
318     system("clear");
319     Titulo("COMODO "+ comodo +" - "+ house.Nome());
320     cout<<"  1 - Acessar Dispositivos\n";
321     cout<<"  2 - Mudar nome\n";
322     cout<<"  3 - Voltar\n";
323 
324     esc = 0;
325     while(esc < 1 || esc > 3){
326       cout<<"Digite sua escolha: ";
327       cin>>esc;
328       if(esc < 1 || esc > 3){cout<<"Escolha invalida!"<<endl;}
329     }
330 
331 
332     // Acessar dispositivos
333     if(esc == 1){
334       MenuDispositivo(house, comodo);
335     }
336 
337     // Mudar nome
338     else if(esc == 2){
339       string novo_nome = "";
340       cout<<"Digite o novo nome do comodo: ";
341       cin.ignore();
342       getline(cin, novo_nome);
343       //basicamente cria um novo valor para o map, substitui oq tinha no antigo
344       //para o novo e apaga o velho
345       house.comodos_[comodo].SetNome(novo_nome);
346       house.comodos_[novo_nome] = move(house.comodos_[comodo]);
347       house.comodos_.erase(comodo);
348       comodo = novo_nome;
349 
350       cout << "Cômodo " << comodo << " renomeado!";
351       Wait1();
352     }
353   }
354 }
```

void MenuComodos (Casa & *house*)

```
171                               {
172 
173   int esc = 0;
174   while(esc != 5){
175 
176     esc = 0;
177     system("clear");
178     Titulo("MENU DE COMODOS - "+ house.Nome());
179     cout<<"  1 - Acessar comodo\n";
180     cout<<"  2 - Adicionar comodo\n";
181     cout<<"  3 - Remover comodo\n";
182     cout<<"  4 - Status geral\n";
183     cout<<"  5 - Voltar\n";
184 
185     esc = 0;
186     while(esc < 1 || esc > 5){
187       cout<<"Digite sua escolha: ";
188       cin>>esc;
189       if(esc < 1 || esc > 5){
190         cout<<"Escolha invalida!"<<endl;
191         Wait1();
192       }
193     }
194 
195     // ACESSAR COMODO
196     if(esc == 1){
```

```
197
198       if(house.comodos_.size() < 1){
199
200         cout<<"Nao existem comodos, adicione um comodo para usar essta
opcao!"<<endl;
201         Wait1();
202       }else{
203
204         // LISTAR COMODOS DISPONÍVEIS
205         house.ListarComodos();
206         string comodo = "";
207
208         // USUÁRIO ESCOLHE O CÔMODO QUE QUER CONFIGURAR
209         bool ex = 0;
210         while(!ex){
211
212           cout<<"Digite o nome do comodo que deseja acessar: ";
213           cin.ignore();
214           getline(cin, comodo);
215
216           for(auto it : house.comodos_){
217
218             if(it.first == comodo){
219              ex = 1;
220             }
221           }
222
223           if(!ex){
224             cout<<"Escolha invalida!\n";
225           }
226         }
227
228         MenuComodo(house, comodo);
229       }
230     }
231
232     // ADICIONAR COMODO
233     else if(esc == 2){
234
235       string comodo = "";
236       cout <<"Digite o nome do cômodo que deseja adicionar: ";
237       cin.ignore();
238       getline(cin, comodo);
239       house.AdicionarComodo(comodo);
240       house.comodos_[comodo].SetNome(comodo);
241       cout << "Comôdo " << comodo << " salvo!";
242       Wait1();
243     }
244
245     // REMOVER COMODO
246     else if(esc == 3){
247       if(house.comodos_.size() < 1){
248         cout<<"Nao existem comodos para serem removidos!"<<endl;
249         Wait1();
250       }
251       else{
252         // LISTAR COMODOS DISPONÍVEIS PARA REMOVER
253         house.ListarComodos();
254         string comodo = "";
255
256         // USUÁRIO ESCOLHE O CÔMODO QUE QUER REMOVER
257         bool ex = 0;
258         while(!ex){
259
260           cout<<"Digite o nome do comodo que deseja remover: ";
261           cin.ignore();
262           getline(cin, comodo);
263
264           for(auto it : house.comodos_){
265             if(it.first == comodo){
266              ex = 1;
267             }
268           }
269           if(!ex){
270             cout<<"Esse cômodo não existe!\n";
271           }
272         }
```

```
273
274         house.RemoverComodo(comodo);
275         cout << "Cômodo " << comodo << " removido!";
276         Wait1();
277       }
278     }
279
280     // STATUS GERAL
281     else if(esc == 4){
282
283       if(house.comodos_.size() < 1){
284         cout << "Não há nenhum comodo!\n";
285         cout << "Para verificar os status, adicione um comodo com a opção 2!"<<
endl;
286         Wait1();
287       }
288       else{
289
290         cout << "Status geral (Todos os dispositivos em cada comodo):" << endl;
291         for(auto it : house.comodos_){
292
293           cout<<endl;
294           cout << it.first << ":" << endl;
295           cout << "Lâmpadas:" << endl;
296           it.second.ListarDispositivos(1);
297           cout << "Cortinas:" << endl;
298           it.second.ListarDispositivos(2);
299           cout << "Ares:" << endl;
300           it.second.ListarDispositivos(3);
301           cout << "Trancas:" << endl;
302           it.second.ListarDispositivos(4);
303           cout << "Janelas:" << endl;
304           it.second.ListarDispositivos(5);
305         }
306         Wait1();
307       }
308     }
309   }
310 }
```

void MenuCortina (Comodo & *comodo*)

```
531                                 {
532
533   int esc = 0;
534   while(esc != 6){
535
536     system("clear");
537     Titulo("CORTINAS DE " + comodo.Nome());
538     cout<<"  1 - Listar Cortinas\n";
539     cout<<"  2 - Configurar Cortina\n";
540     cout<<"  3 - Adicionar Cortinas\n";
541     cout<<"  4 - Remover Cortinas\n";
542     cout<<"  5 - Configurar todas as Cortinas\n";
543     cout<<"  6 - Voltar\n";
544
545     esc = 0;
546     while(esc < 1 || esc > 6){
547       cout<<"Digite sua escolha: ";
548       cin>>esc;
549       if(esc < 1 || esc > 6){cout<<"Escolha invalida!"<<endl;}
550     }
551
552     if((esc == 2 || esc == 4 || esc == 5) && comodo.cortinas_.size() == 0){
553       cout<<"Não há cortinas! Escolha inválida!"<<endl;
554       esc = 0;
555       Wait1();
556     }
557
558     // listar as cortinas
559     if(esc == 1){
560
561       cout <<"Cortinas:"<< endl;
562       comodo.ListarDispositivos(2);
563       Wait1();
564     }
565
```

```
566     //Configurar uma cortina
567     if(esc == 2){
568
569       cout<<"Cortinas para configurar: \n";
570       comodo.ListarDispositivos(2);
571       string cortina = "";
572
573       // USUÁRIO ESCOLHE A CORTINA QUE QUER ALTERAR
574       bool ex = 0;
575       while(!ex){
576
577         cout<<"Digite o nome da cortina que deseja configurar: ";
578         cin.ignore();
579         getline(cin, cortina);
580
581           for(auto it :comodo.cortinas_){
582             if(it.first == cortina){
583              ex = 1;
584             }
585           }
586           if(!ex){cout<<"Essa cortina não existe!\n";}
587       }
588
589       int intensidade = 0;
590       //setando a intensidade
591       cout<< "Digite a intensidade da cortina de 0 a 100: ";
592       cin>>intensidade;
593       comodo.cortinas_[cortina].SetIntensidade(intensidade);
594       cout << "Cortina " << cortina << " configurada!";
595       Wait1();
596     }
597
598     //ADICIONAR CORTINA
599     if(esc == 3){
600
601       string nome = "";
602       cout<< "Digite o nome da nova cortina: ";
603       cin.ignore();
604       getline(cin, nome);
605       comodo.AdicionarDispositivo(2, nome);
606       cout << "Cortina " << nome << " salva!";
607       Wait1();
608     }
609
610     //REMOVER CORTINA
611     if(esc == 4){
612
613       cout <<"Cortinas:"<< endl;
614       comodo.ListarDispositivos(2);
615       // USUÁRIO ESCOLHE A CORTINA QUE QUER REMOVER
616       bool ex = 0;
617       string cortina = "";
618       while(!ex){
619
620         cout<<"Digite o nome da cortina que deseja remover: ";
621         cin.ignore();
622         getline(cin, cortina);
623
624         for(auto it : comodo.cortinas_){
625           if(it.first == cortina){
626             ex = 1;
627           }
628         }
629         if(!ex){cout<<"Essa cortina não existe!\n";}
630       }
631
632       comodo.RemoverDispositivo(2, cortina);
633       cout << "Cortina " << cortina << " removida!";
634       Wait1();
635     }
636
637
638     //CONFIGURAR TODAS AS CORTINAS
639     if(esc == 5){
640
641       comodo.ConfigurarTodos(2);
642       cout << "Cortinas configuradas!";
```

```
643       Wait1();
644     }
645   }
646 }
```

void MenuDispositivo (Casa & *house*, string *comodo*)

```
357                                                 {
358
359   int esc = 0;
360   while(esc != 6){
361
362     system("clear");
363     Titulo("DISPOSITIVOS DE "+ comodo +" - "+ house.Nome());
364     cout<<"  1 - Lampadas\n";
365     cout<<"  2 - Cortinas\n";
366     cout<<"  3 - Ares condicionados\n";
367     cout<<"  4 - Trancas\n";
368     cout<<"  5 - Janelas\n";
369     cout<<"  6 - Voltar\n";
370
371     esc = 0;
372     while(esc < 1 || esc > 6){
373       cout<<"Digite sua escolha: ";
374       cin>>esc;
375       if(esc < 1 || esc > 6){cout<<"Escolha invalida!"<<endl;}
376     }
377
378     // Acessar a lampada
379     if(esc == 1){
380       MenuLampada(house.comodos_[comodo]);
381     }
382
383     // Acessar o Cortina
384     else if(esc == 2){
385      MenuCortina(house.comodos_[comodo]);
386     }
387
388     //Acessar as Ar condicionado
389     else if(esc == 3){
390      MenuArCondicionado(house.comodos_[comodo]);
391     }
392
393     //Acessar as tranca
394     else if(esc == 4){
395      MenuTrancas(house.comodos_[comodo]);
396     }
397
398     //Acessar as janela
399     else if(esc == 5){
400      MenuJanelas(house.comodos_[comodo]);
401     }
402   }
403 }
```

void MenuJanelas (Comodo & *comodo*)

```
900                                 {
901
902   int esc = 0;
903   while(esc != 6){
904
905     system("clear");
906     Titulo("JANELAS DE " + comodo.Nome());
907     cout<<"  1 - Listar Janelas\n";
908     cout<<"  2 - Configurar Janelas\n";
909     cout<<"  3 - Adicionar Janelas\n";
910     cout<<"  4 - Remover Janelas\n";
911     cout<<"  5 - Configurar todas as Janelas\n";
912     cout<<"  6 - Voltar\n";
913
914     esc = 0;
915     while(esc < 1 || esc > 6){
916       cout<<"Digite sua escolha: ";
917       cin>>esc;
918       if(esc < 1 || esc > 6){
919         cout<<"Escolha invalida!"<<endl;
```

```
920         Wait1();
921       }
922     }
923 
924     if((esc == 2 || esc == 4 || esc == 5) && comodo.janelas_.size() == 0){
925       cout<<"Não há janelas! Escolha inválida!"<<endl;
926       esc = 0;
927       Wait1();
928     }
929 
930     // listar as janelas
931     if(esc == 1){
932 
933       cout << "Janelas:" << endl;
934       comodo.ListarDispositivos(5);
935       Wait1();
936     }
937 
938     //Configurar uma janela
939     if(esc == 2){
940 
941       cout<<"Janelas para configurar: \n";
942       comodo.ListarDispositivos(5);
943       string janela= "";
944 
945       // USUÁRIO ESCOLHE A JANELA QUE QUER ALTERAR
946       bool ex = 0;
947       while(!ex){
948 
949         cout<<"Digite o nome da janela que deseja configurar: ";
950         cin.ignore();
951         getline(cin, janela);
952 
953         for(auto it :comodo.janelas_){
954           if(it.first == janela){
955             ex = 1;
956           }
957         }
958         if(!ex){
959           cout<<"Essa janela não existe!\n";
960         }
961       }
962 
963       int intensidade = 0;
964       bool trancada = false;
965       string resposta ="";
966 
967       //abrindo ou fechando
968       cout<< "Digite se quer abrir ou fechar a janela: ";
969       cin>>resposta;
970 
971       if(resposta =="FECHAR" || resposta == "Fechar" || resposta == "fechar" ){
972         trancada = true;
973       }
974       else { trancada = false; }
975 
976       comodo.janelas_[janela].SetTranca(trancada);
977 
978       cout<< "Digite a intensidade da janela: ";
979       cin>>intensidade;
980       comodo.janelas_[janela].SetIntensidade(intensidade);
981       cout << "Janela " << janela << " configurada!";
982       Wait1();
983     }
984 
985     //ADICIONAR JANELA
986     if(esc == 3){
987 
988       string nome = "";
989       cout<< "Digite o nome da nova janela: ";
990       cin.ignore();
991       getline(cin, nome);
992       comodo.AdicionarDispositivo(5, nome);
993       cout << "Janela " << nome << " salva!";
994       Wait1();
995     }
996 
```

```
997     //REMOVER JANELA
998     if(esc == 4){
999
1000       cout << "Janelas:" << endl;
1001       comodo.ListarDispositivos(5);
1002       // USUÁRIO ESCOLHE A JANELA QUE QUER REMOVER
1003       bool ex = 0;
1004       string janela = "";
1005       while(!ex){
1006
1007       cout<<"Digite o nome da janela que deseja remover: ";
1008       cin.ignore();
1009         getline(cin, janela);
1010
1011         for(auto it : comodo.janelas_){
1012           if(it.first == janela){
1013             ex = 1;
1014           }
1015         }
1016         if(!ex){cout<<"Essa janela não existe!\n";}
1017       }
1018
1019       comodo.RemoverDispositivo(5, janela);
1020       cout << "Janela " << janela << " removida!";
1021       Wait1();
1022     }
1023
1024
1025     //CONFIGURAR TODAS AS JANELAS
1026     if(esc == 5){
1027       comodo.ConfigurarTodos(5);
1028       cout << "Janelas configuradas!";
1029       Wait1();
1030     }
1031   }
1032 }
```

void MenuLampada (Comodo & *comodo*)

```
405                                 {
406
407   int esc = 0;
408   while(esc != 6){
409
410     system("clear");
411     Titulo("LÂMPADAS DE " + comodo.Nome());
412     cout<<"  1 - Listar Lâmpadas\n";
413     cout<<"  2 - Configurar Lâmpadas\n";
414     cout<<"  3 - Adicionar Lâmpadas\n";
415     cout<<"  4 - Remover Lâmpadas\n";
416     cout<<"  5 - Configurar todas as Lâmpadas\n";
417     cout<<"  6 - Voltar\n";
418
419     esc = 0;
420     while(esc < 1 || esc > 6){
421       cout<<"Digite sua escolha: ";
422       cin>>esc;
423       if(esc < 1 || esc > 6){
424         cout<<"Escolha invalida!"<<endl;
425       }
426     }
427
428     if((esc == 2 || esc == 4 || esc == 5) && comodo.lampadas_.size() == 0){
429       cout<<"Não há lâmpadas! Escolha inválida!"<<endl;
430       esc = 0;
431       Wait1();
432     }
433
434     // listar as lampadas
435     if(esc == 1){
436       cout << "Lâmpadas:" << endl;
437       comodo.ListarDispositivos(1);
438       Wait1();
439     }
440
441     //Configurar uma lâmpada
442     if(esc == 2){
```

```
443
444       cout<<"Lâmpadas para configurar: \n";
445       comodo.ListarDispositivos(1);
446       string lampada = "";
447
448       // USUÁRIO ESCOLHE A LAMAPADA QUE QUER ALTERAR
449       bool ex = 0;
450       while(!ex){
451
452         cout<<"Digite o nome da lâmpada que deseja configurar: ";
453         cin.ignore();
454         getline(cin, lampada);
455
456         for(auto it : comodo.lampadas_){
457           if(it.first == lampada){
458             ex = 1;
459           }
460         }
461         if(!ex){cout<<"Essa lâmpada não existe!\n";}
462       }
463
464       int intensidade = 0;
465       string cor = "";
466       //setando a intensidade
467       cout<< "Digite a intensidade da lâmpada de 0 a 100: ";
468       cin>>intensidade; //lampada.cc já trata essa excessão
469       comodo.lampadas_[lampada].SetIntensidade(intensidade);
470
471       //setando a cor
472       cout<< "Digite a cor da lâmpada (Amarelo, Vermelho Azul, Branco, Laranja,
Verde ou Roxo): ";
473       cin>>cor;
474       comodo.lampadas_[lampada].SetCor(cor);
475
476       cout << "Lâmpada " << lampada << " configurada!";
477       Wait1();
478     }
479
480     //ADICIONAR LAMPADA
481     if(esc == 3){
482
483       string nome = "";
484       cout<< "Digite o nome da nova lâmpada: ";
485       cin.ignore();
486       getline(cin, nome);
487       comodo.AdicionarDispositivo(1, nome);
488
489       cout << "Lâmpada " << nome << " salva!";
490       Wait1();
491     }
492
493     //REMOVER LAMPADA
494     if(esc == 4){
495
496       cout<< "Lâmpadas disponíveis:"<<endl;
497       comodo.ListarDispositivos(1);
498
499       // USUÁRIO ESCOLHE A LAMPADA QUE QUER REMOVER
500       bool ex = 0;
501       string lampada = "";
502       while(!ex){
503
504         cout<<"Digite o nome da lâmpada que deseja remover: ";
505         cin.ignore();
506         getline(cin, lampada);
507
508         for(auto it : comodo.lampadas_){
509           if(it.first == lampada){
510             ex = 1;
511           }
512         }
513         if(!ex){cout<<"Essa lâmpada não existe!\n";}
514       }
515
516       comodo.RemoverDispositivo(1,lampada);
517       cout << "Lâmpada " << lampada << " removida!";
518       Wait1();
```

```
519     }
520
521
522     //CONFIGURAR TODAS AS LAMPADAS
523     if(esc == 5){
524       comodo.ConfigurarTodos(1);
525       cout << "Lampadas configuradas!";
526       Wait1();
527     }
528   }
529 }
```

void MenuModos (Casa & *house*)

```
1035                            { //tem que criar esse map de modos ainda
1036   int esc = 0;
1037   while(esc != 6){
1038
1039     system("clear");
1040     Titulo("MENU DE MODOS - "+ house.Nome());
1041     cout<<"  1 - Adicionar modo\n";
1042     cout<<"  2 - Ativar modo\n";
1043     cout<<"  3 - Remover modo\n";
1044     cout<<"  4 - Configurar modo\n";
1045     cout<<"  5 - Mudar nome de um modo\n";
1046     cout<<"  6 - Voltar\n";
1047
1048     esc = 0;
1049     while(esc < 1 || esc > 6){
1050       cout<<"Digite sua escolha: ";
1051       cin>>esc;
1052       if(esc < 1 || esc > 6){cout<<"Escolha invalida!"<<endl;}
1053     }
1054
1055     if((esc == 2 || esc == 4) && house.modos_.size() == 0){
1056       cout<<"Não há modos! Escolha inválida!"<<endl;
1057       esc = 0;
1058       Wait1();
1059     }
1060
1061     // Adicionar o modo
1062     if(esc == 1){
1063
1064       Casa modo;
1065       string nome;
1066       modo.comodos_ = house.comodos_;
1067
1068       cout << "Digite o nome do modo que deseja criar: ";
1069       cin.ignore();
1070       getline(cin, nome);
1071       modo.SetNome(nome);
1072       house.modos_[nome] = modo;
1073
1074       cout << "Modo " << nome << " salvo!";
1075       Wait1();
1076     }
1077
1078     // Ativar o modo
1079     else if(esc == 2){
1080
1081       string nome_modo;
1082       cout << "Modos disponíveis:" << endl;
1083       house.ListarModos();
1084       cout << "Digite o nome do modo que deseja ativar: ";
1085       cin.ignore();
1086       getline(cin, nome_modo);
1087       house.AtivarModo(nome_modo);
1088       Wait1();
1089     }
1090
1091     // Remover o modo
1092     else if(esc == 3){
1093
1094       string nome_modo;
1095       cout << "Modos disponíveis:" << endl;
1096       house.ListarModos();
1097
```

```
1098        cout << "Digite o nome do modo que deseja remover: ";
1099        cin.ignore();
1100        getline(cin, nome_modo);
1101
1102        bool existe = false;
1103        for(auto it : house.modos_){
1104          if(it.first == nome_modo){
1105            existe = 1;
1106          }
1107        }
1108        if(existe){
1109          house.modos_.erase(nome_modo);
1110          cout << "Modo " << nome_modo << " removido!";
1111          Wait1();
1112        }
1113        else{
1114          cout << "Este modo não existe!" << endl;
1115          Wait1();
1116        }
1117      }
1118
1119      //Configura o modo
1120      else if(esc == 4){
1121
1122        cout << "Modos disponíveis:" << endl;
1123        house.ListarModos();
1124
1125        string nome_modo;
1126        cout << "Digite o nome do modo que deseja configurar: ";
1127        cin.ignore();
1128        getline(cin, nome_modo);
1129
1130        bool existe = false;
1131        for(auto it : house.modos_){
1132          if(it.first == nome_modo){
1133            existe = 1;
1134          }
1135        }
1136        if(existe){
1137          MenuComodos(house.modos_[nome_modo]);
1138        }
1139        else{
1140          cout << "Este modo não existe!" << endl;
1141          Wait1();
1142        }
1143      }
1144
1145      // Mudar nome de modo
1146      else if(esc == 5){
1147
1148        string nome modo;
1149        cout << "Digite o nome do modo que deseja configurar: ";
1150        cin.ignore();
1151        getline(cin, nome_modo);
1152
1153        bool existe = false;
1154        for(auto it : house.modos_){
1155          if(it.first == nome_modo){
1156            existe = 1;
1157          }
1158        }
1159        if(existe){
1160
1161          string novo_nome = "";
1162          cout<<"Digite o novo nome do modo: ";
1163          cin>>novo_nome;
1164
1165          house.modos_[nome_modo].SetNome(novo_nome);
1166          house.modos_[novo_nome] = move(house.modos_[nome_modo]);
1167          house.modos_.erase(nome_modo);
1168          cout << "Modo " << nome_modo << " renomeado!";
1169          Wait1();
1170        }
1171        else{
1172          cout << "Este modo não existe!" << endl;
1173          Wait1();
1174        }
```

```
1175     }
1176   }
1177 }
```

void MenuTrancas (Comodo & *comodo*)

```
778                                 {
779
780   int esc = 0;
781   while(esc != 6){
782
783     system("clear");
784     Titulo("TRANCAS DE " + comodo.Nome());
785     cout<<"  1 - Listar Trancas\n";
786     cout<<"  2 - Configurar Trancas\n";
787     cout<<"  3 - Adicionar Trancas\n";
788     cout<<"  4 - Remover Trancas\n";
789     cout<<"  5 - Configurar todas as Trancas\n";
790     cout<<"  6 - Voltar\n";
791
792     esc = 0;
793     while(esc < 1 || esc > 6){
794       cout<<"Digite sua escolha: ";
795       cin>>esc;
796       if(esc < 1 || esc > 6){cout<<"Escolha invalida!"<<endl;}
797     }
798
799     if((esc == 2 || esc == 4 || esc == 5) && comodo.trancas_.size() == 0){
800       cout<<"Não há trancas! Escolha inválida!"<<endl;
801       esc = 0;
802       Wait1();
803     }
804
805     // listar as trancas
806     if(esc == 1){
807
808       cout << "Trancas:" << endl;
809       comodo.ListarDispositivos(4);
810       Wait1();
811     }
812
813     //Configurar uma tranca
814     if(esc == 2){
815
816       cout<<"Trancas para configurar: \n";
817       comodo.ListarDispositivos(4);
818        string tranca = "";
819
820       // USUÁRIO ESCOLHE A TRANCA QUE QUER ALTERAR
821       bool ex = 0;
822       while(!ex){
823
824         cout<<"Digite o nome da tranca que deseja configurar: ";
825         cin.ignore();
826         getline(cin, tranca);
827
828         for(auto it :comodo.trancas_){
829           if(it.first == tranca){
830             ex = 1;
831           }
832         }
833         if(!ex){cout<<"Essa tranca não existe!\n";}
834       }
835
836       bool trancada = false;
837       string resposta ="";
838       //abrindo ou fechando
839       cout<< "Digite se quer abrir ou fechar a tranca: ";
840       cin>>resposta;
841       if(resposta =="FECHAR" || resposta == "Fechar" || resposta == "fechar" ){
842         trancada = true;
843       }
844       else { trancada = false; }
845       comodo.trancas_[tranca].SetAtiva(trancada);
846       cout << "Tranca " << tranca << " configurada!";
847       Wait1();
848     }
```

```
849
850     //ADICIONAR TRANCA
851     if(esc == 3){
852
853       string nome = "";
854       cout<< "Digite o nome da nova tranca: ";
855       cin.ignore();
856       getline(cin, nome);
857       comodo.AdicionarDispositivo(4, nome);
858       cout << "Tranca " << nome << " salva!";
859       Wait1();
860     }
861
862     //REMOVER TRANCA
863     if(esc == 4){
864
865       cout << "Trancas:" << endl;
866       comodo.ListarDispositivos(4);
867       // USUÁRIO ESCOLHE A TRANCA QUE QUER REMOVER
868       bool ex = 0;
869       string tranca = "";
870       while(!ex){
871
872         cout<<"Digite o nome da tranca que deseja remover: ";
873         cin.ignore();
874         getline(cin, tranca);
875
876         for(auto it : comodo.trancas_){
877           if(it.first == tranca){
878             ex = 1;
879           }
880         }
881         if(!ex){cout<<"Essa tranca não existe!\n";}
882       }
883
884       comodo.RemoverDispositivo(4, tranca);
885       cout << "Tranca " << tranca << " removida!";
886       Wait1();
887     }
888
889
890     //CONFIGURAR TODAS AS TRANCAS
891     if(esc == 5){
892
893       comodo.ConfigurarTodos(4);
894       cout << "Trancas configuradas!";
895       Wait1();
896     }
897   }
898 }
```

void RemoverComodo (Casa & *house*, string *nome*)

void RemoverModo (Casa & *house*, string *nome*)

void Titulo (string *titulo*)

```
155                               {
156     cout<<"================================================================\n";
157     cout<<"\t"<< titulo <<"\n";
158     cout<<"================================================================\n";
159     cout<<"\n";
160 }
```

void Wait1 ()

```
161             {
162
163   int ex = 0;
164   while(ex != 1){
165     cout<< " Digite 1 para voltar: ";
166     cin>> ex;
167   }
168 }
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/src/tranca.cc

```
#include "../include/tranca.h"
#include <string>
#include <iostream>
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/testes/teste_ar_condicionado.cc

```
#include <iostream>
#include "../include/ar_condicionado.h"
```

Funções

- int **main** ()

Funções

int main ()

```
6           {
7   ArCondicionado teste;
8   teste.SetNome("joao");
9   cout << "O nome do aparelho é " << teste.Nome() << endl;
10   teste.SetModo("cool");
11   cout << "O modo do aparelho é " << teste.Modo() << endl;
12   teste.SetLigar(true);
13   cout << "O estado do aparelho é: " << teste.Ligado() << endl;
14   teste.SetTemperatura(22);
15   cout << "A temperatura do aparelho é " << teste.Temperatura() << endl;
16
17 }
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/testes/teste_cortina.cc

```
#include <iostream>
#include <string>
#include "../include/cortina.h"
```

Funções

- int **main** ()

Funções

int main ()

```
7            {
8     int x;
9      string y;
10
11     cin>>x;
12     cin>>y;
13     Cortina cortina;
14     cortina.SetIntensidade(x);  // Define a intensidade da cortina
15     cortina.SetNome(y); //  Muda o nome do dispositivo.
16
17     // Imprime a intensidade usando cout
18     cout <<"A intensidade da cortina é = "<<  cortina.Intensidade() << endl;
19     cout <<"O nome da cortina é:"<<  cortina.Nome()<<endl;
20
21     return 0;
22 }
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/testes/teste_jalena.cc

```
#include <iostream>
#include "../include/janela.h"
```

Funções

- int **main** ()

Funções

int main ()

```
6           {
7     Janela teste;
8     teste.SetNome("janela1");
9     cout<<teste.Nome()<<endl;
10     teste.SetIntensidade(30);
11     cout<<teste.Intensidade()<<endl;
12     teste.SetTranca(0);
13     cout<<teste.Tranca()<<endl;
14     return 0;
15 }
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/testes/teste_lampada.cc

```
#include "../include/lampada.h"
#include <iostream>
```

Funções

- int **main** ()

Funções

int main ()

```
6           {
7     string nome = "joaozinho";
8     string cor = "Vermelho";
9     Lampada teste;
10     teste.SetIntensidade(85);
11     teste.SetCor(cor);
12     teste.ListarCores();
13     teste.SetNome(nome);
14     cout << teste.Nome() <<endl;
15     cout << teste.Intensidade() << endl;
16     cout << teste.Cor() << endl;
17
18 }
```

Referência do Arquivo D:/faculdade/material de leitura 2 semestre/Programação 2/TP-PDS2/testes/teste_tranca.cc

```
#include <iostream>
#include "../include/tranca.h"
```

Funções

- int **main** ()

Funções

int main ()

```
9           {
10      Tranca breno;
11      breno.SetNome("breno");
12      cout << breno.Nome() << endl;
13      breno.SetAtiva(true);
14      cout << breno.Ativa() << endl;
15
16  }
```

Sumário

INDEX