Grupo:
- Augusto Vinícius Ferreira de Sales. Matrícula: 20200001829
- Leandro Lucas de Oliveira Bandeira. Matrícula: 2021002509
- Renata Avelino de Andrade. Matrícula: 20210025421

Documentação

1. Empresa

Classe composta pelas classes filhas de Funcionario: operador, gerente, diretor e presidente

Os métodos *public* são:

+ addFuncionario(Funcionario*, int) : void

Recebe um ponteiro para um funcionário que já foi instanciado para o tipo de designação correto e o adiciona ao vector do seu tipo ou no caso do presidente faz o ponteiro de presidente apontar para ele.

+ modificarFuncionario(int, int, string) : void

Recebe o código do funcionário a ser modificado a opção do que ele vai modificar e no caso dessa sobrecarga da função recebe o atributo string para modificar atributos do funcionário que são strings.

+ modificarFuncionario(int, int, int) : void

Recebe o código do funcionário a ser modificado a opção do que ele vai modificar e no caso dessa sobrecarga da função recebe o atributo int para modificar atributos do funcionário que são inteiros.

+ modificarFuncionario(int, int*) : void

Recebe o código do funcionário a ser modificado a opção do que ele vai modificar e no caso dessa sobrecarga da função recebe o atributo int* que aponta para o array contendo a data para modificar o atributo data ingresso.

+ excluirFuncionario(int) : void

Recebe o código do funcionário para excluir, o funcionário é tirado do seu vector de acordo com a sua designação e depois a memória reservada para ele é liberada, no caso do presidente que é apenas um ponteiro a sua memória é liberada.

+ exibirFuncionario(int) : void

Recebe o código do funcionário para exibi-lo.

+ exibirTodosFuncionarios( ) : void

A função percorre os três vectors dos tipos de designações exibindo os funcionários armazenados neles e por último é exibido o presidente que é apenas um ponteiro.

+ exibirFuncionariosPorTipo(int) : void

Recebe o tipo de designação na forma de inteiro, e através da designação é percorrido o vector respectivo exibindo os funcionários ali armazenados.

+ concederAumentoSalarial( ) : void

A função percorre todos os vectors das designações e o ponteiro de presidente chamando o método getFolhaSalarial(int) e depois chamando o método aumentarSalarioBase( ) que é um método da classe FolhaSalarial para cada funcionário ali armazenado.

+ calcularFolhaSalarial(int) : void

A função percorre todos os vectors das designações e o ponteiro de presidente chamando a função calcularSalarioMensal(int) para cada um ali armazenado.

+ imprimirFolhaSalarialFuncionario(int) : void

Nessa sobrecarga recebe o código do funcionário e exibe as folhas salariais que já foram calculadas usando a função calcularFolhaSalarial(int).

+ imprimirFolhaSalarialFuncionario(string) : void

Nessa sobrecarga recebe o nome do funcionário e exibe as folhas salariais que já foram calculadas usando a função calcularFolhaSalarial(int).

+ imprimirFolhaSalarialEmpresa(int) : void

Recebe a opção como um inteiro, a opção pode ser imprimir a folha salarial da empresa para o ano ou para um mês específico, para a primeira é percorrida os 12 meses de folhas salariais dos funcionários e somado os valores de salário base, salário líquido, descontos da previdência social e descontos do imposto de renda e ao fim é exibido a soma desses atributos, para a opção de imprimir para um mês específico é feito o mesmo algoritmo da opção um só que dessa vez apenas para o mês solicitado pelo o usuário

+ buscarFuncionario(int) : Funcionario*

Nessa sobrecarga recebe o código do funcionário a ser buscado, com esse código é percorrido os vectors das designações e o ponteiro do presidente em busca de um funcionário que o código corresponda com o código solicitado para a busca, se o funcionário for encontrado é retornado um ponteiro para ele, se não for encontrado nenhum funcionário com aquele código é retornado um ponteiro nulo (nullptr).

+ buscarFuncionario(int, int*, int*) : Funcionario*

Nessa sobrecarga recebe o código do funcionário a ser buscado, um ponteiro para o índice dele no vector e um ponteiro para a sua designação, com esse código é percorrido os vectors das designações e o ponteiro do presidente em busca de um funcionário que o código corresponda com o código solicitado para a busca, se o funcionário for encontrado é retornado um ponteiro para ele e os ponteiros de indice e designação vão apontar para o indice e designação do funcionário encontrado, se não for encontrado nenhum funcionário com aquele código é retornado um ponteiro nulo (nullptr).

+ buscarFuncionariosIntervaloTempo(int*, int*) : void

Recebe um ponteiro para um array contendo a data inicial do intervalo e um ponteiro para a data final do intervalo, com essas datas é feito uma checagem no algoritmo implementado percorrendo os vectors das designações e o ponteiro de presidente, os funcionários que estiverem no intervalo de tempo são exibidos para o usuário

+ buscarFuncionariosParcial(string, int) : void

Recebe o atributo string que pode ser o nome ou o CEP e recebe também a opção de busca parcial que é um inteiro, a opção pode ser buscar por nome ou por endereço, com isso é percorrido os vectors das designações e o ponteiro de presidente em busca de funcionários que correspondam com os parâmetros de busca, os que correspondem são exibidos para o usuário.

Os atributos são:

- qtdFuncionario : int [ ] = 0
- operadores : vector<Funcionario*>
- gerentes : vector<Funcionario*>
- diretores : vector<Funcionario*>
- presidente : Funcionario*
- dadosArquivos : Arquivo

2. Pessoa

Pessoa é a classe mãe de Funcionario.

Os métodos *public* são:

+ Pessoa(string, string, string, int, string, int)

  Recebe os parâmetros e instância através dos set's

+ validaNome(string) : void

  Recebe uma string como parâmetro e faz o tratamento de erro dela antes de instanciá-la.

+ validaIdade(string) : int

  Recebe uma string como parâmetro e faz o tratamento de erro seguindo a idade correta de um funcionário. Após o tratamento, repassa um valor int.

Os atributos são:

- nome : char [ ]
- telefone : char [ ]
- CPF : char [ ]
- idade : int
- endereco : Endereco

2.1 Funcionario

Funcionario é uma subclasse de Pessoa.

Os métodos *public* são:

+ Funcionario(int, string, string, int, string, int, string, int*, int)

  Recebe os parâmetros e repassa para o construtor de Pessoa, além de instanciar seus atributos próprios através dos métodos de set.

+ calcularSalarioMensal(int) : void

  É um método que é sobrescrito pelas suas classes filhas.

+ validaDesignacao(string) : int

  Recebe uma string como parâmetro e a trata até ficar compatível com o número da designação dos funcionários. 0 = Operador. 1 = Gerente. 2 = Diretor. 3 = Presidente

+ validaDataIngresso(string, int*): void

  Recebe uma string como parâmetro e trata até ficar compatível com o padrão esperado de data (XX XX XXXX), além de verificar se o dia, mês e ano digitado são válidos.

+ validaCodigoFuncionario(string): int

  Recebe uma string como parâmetro e trata até ficar um inteiro positivo não nulo.

Os métodos *protected* são:

gerarAleatorio(int) : int

Esse método recebe um inteiro e gera um valor aleatório de 0 ao número passado.

Os atributos são:

- horasTrabalhadas : int
- codigoFuncionario : int
- designacao : int
- folhaSalarial : FolhaSalarial [ ]
- dataIngresso : Data

2.1.1 Operador

Operador é uma subclasse de Funcionario.
Os métodos *public* são:

+ Operador(int, string, string, int, string, int, string, int*, int)

  O construtor de Operador recebe os atributos e passa para Funcionario para serem inicializadas dentro do construtor da classe mãe.

+ calcularSalarioMensal(int) : void

  Esse método chama o gerarAleatorio(int) para instanciar as horas trabalhadas do mês (caso já não tenham sido instanciadas) e calcula o salário do mês, verificando se o operador fez hora extra.

Os atributos são:

- HORA_EXTRA : const double = 14,10

2.1.2 Gerente

Gerente é uma subclasse de Funcionario.
Os métodos *public* são:

+ Gerente(int, string, string, int, string, int, string, int*, int, string)

O construtor de Gerente recebe os atributos e passa para Funcionario para serem inicializadas dentro do construtor da classe mãe. Ele também recebe atributo extra (areaSupervisao) e instância através do setAreaSupervisao(string).

+ calcularSalarioMensal(int) : void

Esse método chama o gerarAleatorio(int) para instanciar as horas trabalhadas do mês (caso já não tenham sido instanciadas) e calcula o salário do mês, verificando se o operador fez hora extra.

Os atributos são:

- areaSupervisao : char [ ]
- HORA_EXTRA : const double = 40,16

2.1.3 Diretor

Diretor é uma subclasse de Funcionario.
Os métodos *public* são:

+ Diretor(int, string, string, int, string, int, string, int*, int, string, string)

O construtor de Diretor recebe os atributos e passa para Funcionario para serem inicializadas dentro do construtor da classe mãe. Ele também recebe atributos extras (areaSupervisao e areaFormacao) e instância através do setAreaSupervisao(string) e setAreaFormacao(string).

+ calcularSalarioMensal(int) : void

Esse método chama o gerarAleatorio(int) para instanciar as horas trabalhadas do mês (caso já não tenham sido instanciadas) e calcula o salário do mês, verificando se o operador fez hora extra.

Os atributos são:

- areaSupervisao : char [ ]
- areaFormacao : char [ ]
- HORA_EXTRA : const double = 74,48

2.1.4 Presidente

Presidente é uma subclasse de Funcionario.

Os métodos *public* são:

+ Presidente(int, string, string, int, string, int, string, int*, int, string, string)

O construtor de Presidente recebe os atributos e passa para Funcionario para serem inicializadas dentro do construtor da classe mãe. Ele também recebe atributos extras (areaFormacao e formacaoMax) e instância através do setAreaFormacao(string) setFormacaoMax(string).

+ calcularSalarioMensal(int) : void

Esse método chama o gerarAleatorio(int) para instanciar as horas trabalhadas do mês (caso já não tenham sido instanciadas) e calcula o salário do mês, verificando se o operador fez hora extra.

Os atributos são:

- areaFormacao : char [ ]
- formacaoMax : char [ ]
- HORA_EXTRA : const double = 97,24

3. Endereco

A classe Endereco é uma composição da classe Pessoa.

Os métodos *public* são:

+ validaCEP(string): void

Recebe uma string e faz um tratamento de erro, até a string ser válida para ser testada. Após isso é enviado e baixado o arquivo json (do site viacep) que contém os atributos do CEP digitado, caso não seja válido é enviado uma mensagem avisando. Mas, caso seja válido, os atributos do viacep são salvos no Endereco.

+ getInformacao( ): string

Retorna o endereço completo em forma de string.

+ comparaEndereco(string): bool

Recebe uma string e faz um tratamento de erro até a string ser válida para ser testada. Após isso ela verifica se o CEP digitado é igual ao CEP armazenado no Endereco.

+ validaNumero(string): int

Recebe uma string e faz um tratamento de erro até ela pode ser um número de residência válido (inteiro acima de 0).

Os atributos são:

- CEP: char [ ]
- rua: char [ ]
- bairro: char [ ]
- cidade: char [ ]
- estado: char [ ]
- numero: int

4. Arquivo

Classe responsável pela manipulação de arquivo

Os métodos *public* são:

+ atualizaArquivoFolha(vector < Funcionario * >&) : void
  Atualiza o arquivo de folha.csv a partir do vector recebido.

+ atualizaArquivoFolha(Funcionario *) : void
  Atualiza o arquivo de folha.csv somente para um funcionário.

+ AtualizaBaseDadosCsv(const vector < Funcionario * > &) : void
  Atualiza a base de dados a partir de uma lista de funcionários.

+ addPresidenteBaseDadosCsv(Funcionario *) : void
  Adiciona somente um funcionário na base de dados.

+ criaArquivoBaseDadosZerado() : void
  Recria o arquivo de base de dados.

+ carregaDadosCsv(vector < Funcionario * > &operadores, vector < Funcionario * > &diretores, vector < Funcionario * > &gerentes, Funcionario **presidente) : void
  Carrega os dados contido no arquivo Dados.csv e manda para as listas recebidas nos parâmetros.

+ HistoricoArquivo *getHistoricoArquivo(): void
  Retorna um objeto do tipo HistoricoArquivo, o qual é um objeto responsável pela composição da classe HistoricoArquivo.

  Os atributos são:
  - path : string
  - historico: HistoricoArquivo

5. HistoricoArquivo

Classe responsável por manipular o histórico de dados, o qual é feita por composição pela classe arquivo.

Os métodos *public* são:

+ setDataModificacao(int): void

Coloca a data de modificação do funcionário relativo à data do computador, recebe o índice do tipo de funcionário.

+ setModificacao(int , string) : void

Coloca a modificação realizada no funcionário, recebe o tipo de funcionário.

+ setCodigo(int, int) : void

Coloca o código para o qual foi feita a modificação e o tipo do funcionário.

+ getCodigo(int) : void

Retorna o código do funcionário para o qual foi realizada a modificação, recebendo o tipo dele.

+ setNome(int, string) : void

Coloca o nome do funcionário em que foi feita a modificação, recebendo assim seu tipo.

+ getNome(int) : void

Retorna o nome do funcionário modificado, recebe o tipo dele como parâmetro.

+ escreveArquivoModificacoes(int) : void

Escreve todas as modificações referente ao tipo dele, recebido como parâmetro.

Os atributos são:

- string modificacoes: []
- int codigos: []
- string datasModificacoes: []
- string nome: []
- saidaHistorico: ofstream
- entradaHistorico: ifstream
- path: string

6. Data

Classe responsável pelo geralmente de todas as datas do programa, o qual é feita por composição com a classe funcionário.

Os métodos *public* são:

+ retornaStringData() : string
  Retorna uma string de data salva.

+ retornaDataComputador(): string
  Retorna a data do computador em formato de string.

+ comparaDatas(int *) : bool
  Compara duas datas recebidas

  Os atributos são:
  - dia: int
  - mes: int
  - ano: int

7. Interface

Classe responsável por exibir os menus e receber os inputs do programa.

Os métodos *public* são:

+ menu( ) : int

menu recebe o input dizendo o que o usuário quer executar no programa, o valor recebido é tratado para não ter erros em tempo de execução e depois é retornado.

+ menuTexto( ) : void

menuTexto contém o texto exibido no menu( ).

+ lerAtributosFuncionario( ) : Funcionario*

Lê todos os atributos necessários para instanciar o funcionário de acordo com sua designação, e é feito o tratamento de possíveis erros para todos os atributos lidos, o funcionário é instanciado e depois o método retorna um ponteiro para ele.

+ lerCodigoParaModificarFuncionario( ) : int

Lê o código do funcionário que deseja ser modificado e retorna.

+ lerOpcaoParaModificarFuncionario( ) : int

Lê a opção de modificação de atributo do funcionário e retorna.

+ lerNovoAtributoStrParaModificarFuncionario(int) : string

Lê um atributo string para modificar no funcionário e retorna.

+ lerNovoAtributoIntParaModificarFuncionario(int) : int

Lê um atributo int para modificar no funcionário e retorna.

+ lerNovaDataParaModificarFuncionario(int*) : void

Lê uma data para um array de int e retorna para modificar no funcionário.

+ lerCodigoParaExibirFuncionario( ) : int

Lê o código do funcionário que vai ser exibido e retorna.

+ lerCodigoParaExcluirFuncionario( ) : int

Lê o código do funcionário que vai ser excluído e retorna.

+ lerTipoParaExibirFuncionarios( ) : int

Lê o tipo de designação para exibir funcionários por tipo e retorna

+ lerMesParaCalcularFolhaSalarialEmpresa( ) : int

Lê o mês para calcular a folha salarial da empresa e retorna.

+ lerNomeParaImprimirFolhaSalarialFuncionario( ) : string

Lê o nome do funcionário que vai ser exibida a folha salarial e retorna.

+ lerCodigoParaImprimirFolhaSalarialFuncionario( ) : int

Lê o código do funcionário que vai ser exibida a folha salarial e retorna.

+ lerTipoAtributoParaImprimirFolhaSalarialFuncionario( ) : int

Lê o tipo de atributo que vai ser usado para imprimir a folha salarial do funcionário e retorna.

+ lerOpcaoParaImprimirFolhaSalarialEmpresa( ) : int

Lê a opção para imprimir a folha salarial da empresa e retorna, as opções podem ser imprimir a folha do ano ou de um mês específico.

+ lerOpcaoParaBuscarFuncionario( ) : int

Lê a opção de buscar funcionário e retorna, as opções podem ser por busca parcial ou por intervalo de tempo.

+ lerTipoInformacaoStrParaBuscarFuncionario(int) : int

Lê uma informação do tipo string que vai ser usada na busca parcial e retorna.

+ lerDataParaBuscarFuncionario(int*, int*) : void

Lê as datas iniciais e finais da opção de busca por intervalo de tempo e armazena em arrays depois os ponteiros apontam para os arrays.

+ validaMes(string) : int

Recebe a string do mês e verifica se é um mês valido depois retorna o mês na forma de int.

+ validaInteiro(string) : int

Recebe uma string e verifica se é um numero depois retorna o valor na forma de int.