**Gerenciamento de Funcionários**

# Descrição do sistema

O programa que desenvolvemos tem como intuito ajudar uma empresa a gerenciar o cadastro de seus funcionários, com dados como o cargo dos mesmos, salário, endereço, etc. O que foi desenvolvido até aqui é parte de um sistema mais robusto, ainda em desenvolvimento, que será efetivamente utilizado na vida real, na empresa do pai de Glayson.

# Especificação do Sistema

Utilizamos apenas a linguagem C++ e a biblioteca iostream, que é o foco de nosso trabalho, além da namespace std para facilitar as funções gerais do código.

É possível:

**Através do método executar() :**

**Imprimir a lista de funcionários** ( que utilizará da classe Exibir para imprimir toda a lista de funcionários presente);

**Adicionar um funcionário** (o qual será inserido pelo usuário no terminal utilizando do cin e do cout para capturar as informações fornecidas e depois exibi-las);

**Sair do sistema** ( se for da escolha do usuário ele poderá selecionar a opção de sair do sistema, encerrando a execução do programa).

\***Algumas funções extras:**

Caso o programa esteja sem funcionários cadastrados em sua lista, ele avisará que a mesma está vazia, do contrário, caso esteja cheia, o programara também irá avisar ao usuário. Além disso, para um cadastro realizado de forma correta, o programa dá o feedback de que o funcionário foi cadastrado com sucesso. Por fim, em caso de equivoco na escolha das opções fornecidas pelo programa, o mesmo avisa que tal opção escolhida não existe.

# Conteúdos usados no projeto

**Separando a interface da classe de sua implementação:**

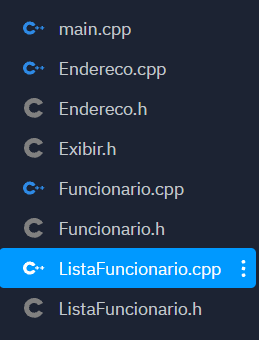


Figura 1. Separando a interface da classe

Como visto na figura 1, o programa foi separado em .h e .cpp com o intuito de facilitar a manuntenção de código, como também assegurar o encapsulamento do mesmo.

**Encapsulamento:**

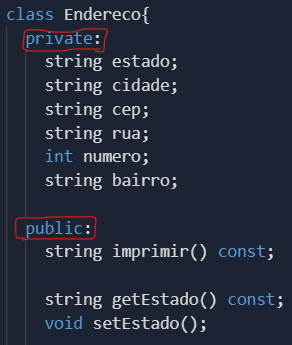


Figura 2. Protegendo as variáveis do programa

Na figura 2, podemos perceber de fato o encapsulamento sendo utilizado, o qual protege as variáveis de uma classe base para que só classes derivadas possam usufruir das mesmas.

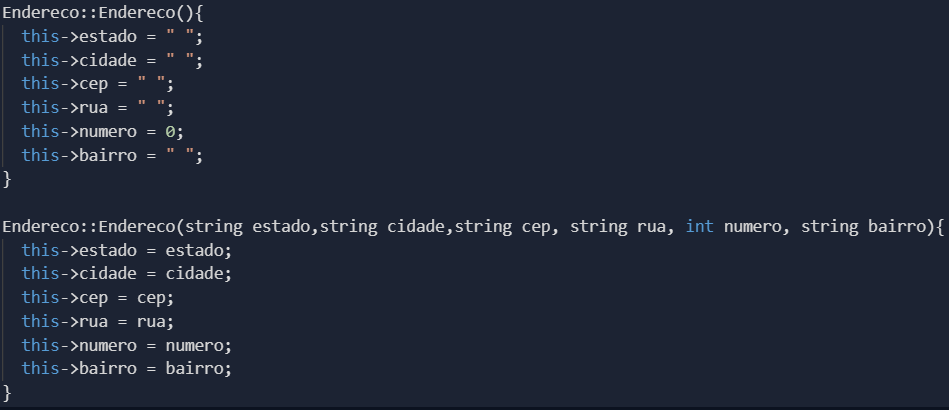
**Uso de construtores:**

Figura 3. Construtor vazio e construtor completo

A figura 3 mostra a inicialização das variáveis vazias através do construtor vazio e a inicialização das variáveis do construtor preencchido, utilizando do this para apontar quem é variável e quem é parâmetro.

**Funções- membro const:**

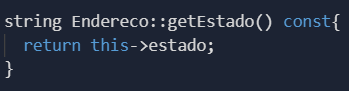


Figura 4. Função Const

Na figura 4, vemos uma função que tem como objetivo sempre retornar uma constante, em razão disso, a elencamos como const.

**Objeto const:**



Figura 5. Objeto Const

Na figura 5 temos uma constante que serve como parâmetro para o tamanho da lista de funcionários.

**Composição:**

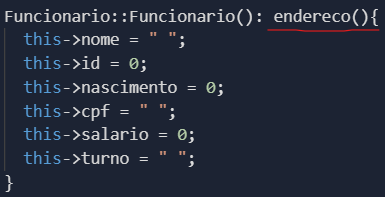
****

Figura 6. Composição na classe Funcionario

Na figura 6 podemos observara relação de que todo Funcionário “tem um” endereço, o que caracteriza a utilização da técnica de composição entre classes.

**Herança:**

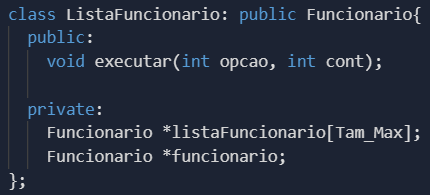


Figura 7. Herança presente na classe ListaFuncionario

Na figura 7, em class ListaFuncionario temos uma herança com tipo de acesso público, na qual ListaFuncionario herda a classe Funcionario.

**Polimorfismo de inclusão e classe abstrata:**

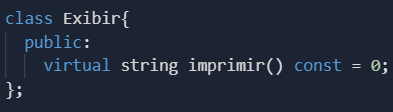


Figura 8. Polimorfismo na class Exibir

Na figura 8 temos o Polimorfismo de Inclusão presente no método imprimir que é precedido da palavra “virtual”.

# Conclusão

O projeto desse sistema computacional foi um desafio realizado com sucesso. Os conhecimentos de programação orientada a objetos foram postos em prática e isso pôs à prova as dificuldades enfrentadas durante o processo de implementação, bem como o mérito na resolução de problemas com o desenvolvimento do código.

O projeto pode ser atualizado e refatorado para que se torne ainda mais robusto. Cabem novas funcionalidades para torná-lo versátil e completo. Como Glayson utilizará esse projeto no sistema da empresa de seu pai, outras informações passarão a incorporar o código. Como implementação futura, será necessário um método que remova funcionários cadastrados. A classe funcionários também precisará de novas características, como adicional de salário baseado em quantidade de filhos.

Por fim, conclui-se que a programação orientada a objetos é uma ferramenta poderosa para a implementação de sistemas como o que foi desenvolvido aqui e a absorção de novos conteúdos desse tópico será necessária para futuras realizações de novos programas.