Leme/SP, 23 de junho de 2023.

Fatec
Faculdade de Tecnologia

Documentação do Projeto: A festa da democracia.

HISTÓRICO DE VERSÃO

Versão	Modificação	Data	Modificado por
v1.0	Documento criado	23-06-2023	Igor Ferreira

Sumário

1. PROJETO

1.1 Resumo

O projeto "A festa da democracia" consiste em um simulador de votação, onde os usuários podem cadastrar candidatos e eleitores, registrar votos e gerar relatórios com os resultados da eleição. O projeto foi desenvolvido utilizando a linguagem C++ e não faz uso de bibliotecas que encapsulam o desenvolvimento das estruturas de dados, como a STL.

2. CÓDIGO

2.1 Bibliotecas

A biblioteca iostream fornece classes e objetos para entrada e saída de dados no console. Ela inclui as classes std::istream e std::ostream, que são responsáveis pela entrada e saída de dados, respectivamente. As principais classes da biblioteca iostream são std::cin e std::cout,

que permitem a leitura de dados do usuário a partir do console e a exibição de mensagens na saída padrão.

A biblioteca fstream fornece classes e objetos para manipulação de arquivos. Ela inclui as classes std::ifstream e std::ofstream, que são derivadas das classes std::istream e std::ostream, respectivamente, permitindo a leitura e gravação de dados em arquivos. A biblioteca fstream permite a abertura, fechamento, leitura e escrita de arquivos por meio de objetos como std::ifstream para leitura e std::ofstream para escrita.

A biblioteca string fornece classes e funções para manipulação de strings. Ela inclui a classe std::string, que representa uma sequência de caracteres e fornece operações para manipular e acessar os dados da string. A biblioteca string permite a criação, concatenação, comparação, extração de substrings e outras operações com strings por meio de métodos e funções fornecidos pela classe std::string.

Essas bibliotecas são amplamente utilizadas no código para permitir a interação com o usuário por meio do console, manipulação de arquivos para gerar relatórios e armazenamento e manipulação de dados do tipo string para representar nomes de candidatos e eleitores.

2.2 Structs

As structs são utilizadas no código para representar os candidatos, eleitores e para criar listas encadeadas que armazenam os candidatos e eleitores cadastrados. Cada nó dessas listas contém os dados de um candidato ou eleitor, além de um ponteiro para o próximo nó da lista, permitindo a organização e manipulação dos dados de forma estruturada.

2.2.1 Candidato

Representa um candidato e possui os seguintes campos:

- nome (string): armazena o nome do candidato.
- numero (int): armazena o número do candidato.
- votos (int): armazena a quantidade de votos recebidos pelo candidato.

2.2.2 Eleitor

Representa um eleitor e possui os seguintes campos:

- nome (string): armazena o nome do eleitor.
- titulo (int): armazena o número de título do eleitor.
- idade (int): armazena a idade do eleitor.
- votou (bool): indica se o eleitor já votou ou não.

2.2.3 ListaCandidatos

É usada para criar uma lista encadeada de candidatos e possui os seguintes campos:

- candidato (do tipo Candidato): armazena as informações do candidato.
- proximo (do tipo ListaCandidatos*): ponteiro para o próximo nó da lista encadeada.

2.2.4 ListaEleitores

É usada para criar uma lista encadeada de eleitores e possui os seguintes campos:

- eleitor (do tipo Eleitor): armazena as informações do eleitor.
- proximo (do tipo ListaEleitores*): ponteiro para o próximo nó da lista encadeada.

2.3 Funções

2.3.1 inserirCandidato

A função inserirCandidato recebe como parâmetros o nome e o número de um candidato e tem a finalidade de inserir esse candidato em uma lista de candidatos.

Primeiro, a função cria um novo objeto do tipo ListaCandidatos por meio da alocação dinâmica de memória. Em seguida, são atribuídos os valores do nome, número e quantidade de votos ao candidato representado pelo novo objeto.

Após isso, é verificado se a lista de candidatos está vazia, ou seja, se o ponteiro listaCandidatos aponta para nullptr. Se a lista estiver vazia, o novo candidato é atribuído como o primeiro elemento da lista.

Caso a lista não esteja vazia, é percorrida até o final por meio de um loop while, buscando o último elemento da lista. O ponteiro temp é utilizado como um iterador para percorrer os elementos da lista até encontrar o último nó, cujo ponteiro proximo é nullptr. Em seguida, o ponteiro proximo desse último elemento é atualizado para apontar para o novo candidato, inserindo-o no final da lista.

Por fim, uma mensagem é exibida indicando que o candidato foi inserido com sucesso.

2.3.2 removerCandidato

A função removerCandidato recebe como parâmetro o número de um candidato e tem como objetivo remover esse candidato da lista de candidatos.

Inicialmente, a função verifica se a lista de candidatos está vazia, ou seja, se o ponteiro listaCandidatos aponta para nullptr. Se a lista estiver vazia, uma mensagem é exibida informando que não há candidatos cadastrados, e a função retorna.

Em seguida, a função verifica se o primeiro candidato da lista possui o número fornecido como parâmetro. Se for o caso, o ponteiro listaCandidatos é atualizado para apontar para o próximo elemento da lista, o candidato removido é liberado da memória e uma mensagem é exibida indicando que o candidato foi removido com sucesso. A função então retorna.

Caso o primeiro candidato não corresponda ao número fornecido, a função prossegue com a busca do candidato a ser removido. Dois ponteiros auxiliares, temp e tempAnterior, são utilizados para percorrer a lista. O ponteiro temp é inicializado como o segundo elemento da lista (listaCandidatos->proximo), enquanto o ponteiro tempAnterior é inicializado como o primeiro elemento da lista (listaCandidatos).

A função percorre a lista por meio de um loop while, verificando se o número do candidato representado pelo nó atual (temp->candidato.numero) corresponde ao número fornecido. Se houver correspondência, o ponteiro proximo do nó anterior (tempAnterior->proximo) é atualizado para apontar para o próximo nó da lista (temp->proximo), o candidato removido é liberado da memória e uma mensagem é exibida indicando que o candidato foi removido com sucesso. A função então retorna.

Caso o número fornecido não seja encontrado em nenhum candidato da lista, a função percorre toda a lista e não encontra correspondência. Nesse caso, uma mensagem é exibida informando que não foi encontrado um candidato com o número fornecido.

2.3.3 listarCandidatos

A função listarCandidatos tem como objetivo exibir na tela os candidatos cadastrados na lista de candidatos.

Inicialmente, a função verifica se a lista de candidatos está vazia, ou seja, se o ponteiro listaCandidatos aponta para nullptr. Se a lista estiver vazia, uma mensagem é exibida informando que não há candidatos cadastrados, e a função retorna.

Caso a lista não esteja vazia, a função inicia um loop while para percorrer a lista de candidatos. Um ponteiro auxiliar temp é inicializado com o valor do ponteiro listaCandidatos, que aponta para o primeiro candidato da lista.

Dentro do loop, as informações de cada candidato são exibidas na tela. Isso inclui o nome do candidato (temp->candidato.nome), o número do candidato (temp->candidato.numero) e a quantidade de votos recebidos pelo candidato (temp->candidato.votos). Em seguida, é exibida uma linha separadora.

Após exibir as informações do candidato atual, o ponteiro temp é atualizado para apontar para o próximo candidato da lista, avançando para a próxima iteração do loop. O loop continua até que o ponteiro temp aponte para nullptr, indicando o final da lista.

Dessa forma, a função percorre a lista de candidatos e exibe as informações de cada um deles na tela, permitindo visualizar os candidatos cadastrados.

2.3.4 inserirEleitor

A função inserirEleitor tem como objetivo adicionar um novo eleitor à lista de eleitores.

Inicialmente, a função cria um novo nó da lista de eleitores utilizando a estrutura ListaEleitores, e o ponteiro novoEleitor é configurado para apontar para esse novo nó. Em seguida, são atribuídos os valores fornecidos aos campos da estrutura Eleitor dentro desse nó. O nome do eleitor é atribuído a novoEleitor->eleitor.nome, o título eleitoral a novoEleitor->eleitor.titulo, a idade a novoEleitor->eleitor.idade e o status de voto (votou) a novoEleitor->eleitor.votou, que é definido como false para indicar que o eleitor ainda não votou. Além disso, o ponteiro novoEleitor->proximo é configurado como nullptr, indicando que esse novo nó será o último da lista.

Em seguida, a função verifica se a lista de eleitores está vazia, ou seja, se o ponteiro listaEleitores aponta para nullptr. Se a lista estiver vazia, o ponteiro listaEleitores é atualizado para apontar para o novo eleitor criado. Caso contrário, a função percorre a lista de eleitores até encontrar o último nó, onde o ponteiro proximo é nullptr. O ponteiro temp é utilizado para percorrer a lista, começando pelo primeiro nó (listaEleitores), e avançando para o próximo nó até encontrar o último nó da lista. Em seguida, o ponteiro proximo do último nó é atualizado para apontar para o novo eleitor, inserindo-o no final da lista.

Após adicionar o eleitor com sucesso, a função exibe uma mensagem informando que o eleitor foi inserido com sucesso.

Dessa forma, a função permite adicionar um novo eleitor à lista de eleitores, garantindo que ele seja colocado no final da lista caso ela já possua outros eleitores.

2.3.5 removerEleitor

A função removerEleitor tem como objetivo remover um eleitor da lista de eleitores com base no número de título fornecido.

Inicialmente, a função verifica se a lista de eleitores está vazia, ou seja, se o ponteiro listaEleitores aponta para nullptr. Se a lista estiver vazia, a função exibe uma mensagem informando que não há eleitores cadastrados e retorna.

Em seguida, a função verifica se o primeiro eleitor da lista possui o número de título igual ao número fornecido. Se for o caso, o ponteiro eleitorRemovido é configurado para apontar para o primeiro nó da lista (listaEleitores). O ponteiro listaEleitores é então atualizado para apontar para o próximo nó, removendo o primeiro eleitor. O nó removido é liberado da memória utilizando o operador delete, e a função exibe uma mensagem informando que o eleitor foi removido com sucesso. Em seguida, a função retorna.

Caso o primeiro eleitor da lista não corresponda ao número de título fornecido, a função utiliza dois ponteiros, temp e tempAnterior, para percorrer a lista a partir do segundo nó. O ponteiro temp aponta para o nó seguinte ao primeiro (listaEleitores->proximo), e o ponteiro tempAnterior aponta para o primeiro nó da lista (listaEleitores). A função percorre a lista enquanto o ponteiro temp não for nullptr.

Dentro do loop, a função verifica se o número de título do eleitor apontado por temp corresponde ao número fornecido. Se for o caso, o ponteiro proximo do nó anterior (tempAnterior->proximo) é atualizado para apontar para o nó seguinte ao nó atual (temp->proximo), removendo assim o eleitor da lista. O nó removido é liberado da memória utilizando o operador delete, e a função exibe uma mensagem informando que o eleitor foi removido com sucesso. Em seguida, a função retorna.

Se o número de título fornecido não corresponder a nenhum eleitor na lista, a função exibe uma mensagem informando que não foi encontrado um eleitor com o número de título fornecido.

Dessa forma, a função permite remover um eleitor da lista de eleitores com base no número de título fornecido, mantendo a integridade da lista.

2.3.6 listarEleitores

A função listarEleitores tem como objetivo exibir na tela os eleitores cadastrados na lista de eleitores.

Inicialmente, a função verifica se a lista de eleitores está vazia, ou seja, se o ponteiro listaEleitores aponta para nullptr. Se a lista estiver vazia, a função exibe uma mensagem informando que não há eleitores cadastrados e retorna.

Em seguida, a função utiliza um ponteiro temp para percorrer a lista de eleitores. O ponteiro temp é inicializado com o endereço do primeiro nó da lista (listaEleitores).

Dentro do loop, a função exibe as informações de cada eleitor através do ponteiro temp. São exibidos o nome do eleitor (temp->eleitor.nome), o número de título (temp->eleitor.titulo), a

idade (temp->eleitor.idade) e se o eleitor votou ou não (temp->eleitor.votou). A informação sobre se o eleitor votou ou não é exibida de forma condicional, sendo exibido "Sim" se temp->eleitor.votou for true e "Não" se for false.

Após exibir as informações de um eleitor, o ponteiro temp é atualizado para apontar para o próximo nó da lista (temp = temp->proximo), permitindo percorrer todos os eleitores da lista.

Dessa forma, a função permite listar os eleitores cadastrados, exibindo suas informações na tela de forma organizada. Caso a lista esteja vazia, uma mensagem apropriada é exibida.

2.3.7 votar

A função votar tem como objetivo registrar o voto de um eleitor em um candidato específico.

A função começa percorrendo a lista de eleitores, utilizando o ponteiro tempEleitor para percorrer os nós. Enquanto houver nós na lista (ou seja, tempEleitor != nullptr), a função verifica se o número de título do eleitor atual é igual ao número de título fornecido como parâmetro. Se houver uma correspondência, significa que o eleitor foi encontrado na lista.

Dentro desse bloco condicional, a função verifica se o eleitor já votou, verificando o valor do atributo votou do eleitor (tempEleitor->eleitor.votou). Se o eleitor já tiver votado, a função exibe uma mensagem informando que o eleitor já votou e retorna, encerrando a função.

Caso o eleitor não tenha votado, a função atualiza o valor do atributo votou para true, indicando que o eleitor votou. Em seguida, o loop é interrompido com break.

Após percorrer a lista de eleitores, a função passa a percorrer a lista de candidatos, utilizando o ponteiro tempCandidato. Da mesma forma, enquanto houver nós na lista (ou seja, tempCandidato != nullptr), a função verifica se o número do candidato atual é igual ao número fornecido como parâmetro. Se houver uma correspondência, significa que o candidato foi encontrado na lista.

Dentro desse bloco condicional, a função incrementa o número de votos do candidato (tempCandidato->candidato.votos++), indicando que o candidato recebeu mais um voto. Em seguida, a função exibe uma mensagem informando que o voto foi registrado com sucesso e retorna, encerrando a função.

Se nenhum candidato for encontrado com o número fornecido, a função exibe uma mensagem informando que não foi encontrado um candidato com o número fornecido.

Dessa forma, a função votar permite registrar o voto de um eleitor em um candidato, atualizando as informações correspondentes na lista de eleitores e na lista de candidatos.

2.3.8 gerarRelatorio

A função gerarRelatorio tem como objetivo gerar um relatório em formato de arquivo de texto contendo informações sobre a eleição.

A função começa abrindo um arquivo chamado "relatorio.txt" para escrita, utilizando um objeto da classe ofstream chamado arquivo. Esse arquivo será utilizado para armazenar o relatório.

A primeira seção do relatório é a "Classificação dos Candidatos". A função percorre a lista de candidatos utilizando o ponteiro tempCandidato, e enquanto houver nós na lista (ou seja, tempCandidato != nullptr), a função escreve no arquivo as informações do candidato atual, incluindo o nome, o número e a quantidade de votos recebidos. Em seguida, é adicionada uma linha de separação para melhor legibilidade. Esse processo é repetido para todos os candidatos da lista.

Após a seção de classificação dos candidatos, a função adiciona uma linha em branco no arquivo e inicia a seção "Votos por Candidato". Novamente, a função percorre a lista de candidatos utilizando o ponteiro tempCandidato, escrevendo no arquivo as informações do candidato atual, incluindo o nome, o número e a quantidade de votos recebidos. Uma linha de separação é adicionada após cada candidato.

A próxima seção do relatório é "Eleitores que Faltaram". A função conta quantos eleitores faltaram na eleição percorrendo a lista de eleitores utilizando o ponteiro tempEleitor. Para cada eleitor da lista, se ele não tiver votado (ou seja, !tempEleitor->eleitor.votou), a variável eleitoresFaltantes é incrementada. Após percorrer a lista completa, a função escreve no arquivo a quantidade de eleitores faltantes.

Após concluir a escrita no arquivo, a função fecha o arquivo utilizando o método close() do objeto arquivo.

Por fim, a função exibe uma mensagem indicando que o relatório foi gerado com sucesso.

Dessa forma, a função gerarRelatorio realiza a criação de um arquivo de texto contendo informações relevantes sobre a eleição, incluindo a classificação dos candidatos, os votos por candidato e a quantidade de eleitores que faltaram.

2.3.9 liberarMemoriaCandidatos

A função liberarMemoriaCandidatos é responsável por desalocar a memória utilizada pelos nós da lista encadeada de candidatos.

Ela percorre a lista a partir do primeiro nó, representado pelo ponteiro listaCandidatos. O processo é realizado por meio de um loop while, onde o ponteiro temp é inicializado com o endereço do primeiro nó.

Enquanto o ponteiro temp não for nulo, ou seja, enquanto houver nós na lista, o seguinte processo é executado:

1. Um ponteiro auxiliar, chamado candidatoRemovido, é criado e recebe o valor do ponteiro temp. Isso é feito para garantir que o nó atual seja preservado antes de avançar para o próximo nó.
2. O ponteiro temp é atualizado para apontar para o próximo nó da lista, acessado através do campo proximo do nó atual.
3. A memória alocada para o nó anteriormente apontado por candidatoRemovido é liberada utilizando o operador delete. Isso garante que a memória seja desalocada corretamente e evita vazamentos de memória.

Esse processo é repetido até que todos os nós da lista tenham sido percorridos e a memória alocada para cada nó tenha sido liberada.

Essa função é fundamental para garantir o uso eficiente da memória, evitando vazamentos de memória e liberando recursos quando eles não forem mais necessários. É importante chamar essa função quando não for mais necessário utilizar a lista encadeada de candidatos, a fim de evitar problemas de vazamento de memória e desperdício de recursos.

2.3.10 liberarMemoriaEleitores

A função liberarMemoriaEleitores tem como objetivo desalocar a memória utilizada pelos nós da lista encadeada de eleitores.

A função começa definindo um ponteiro temp e o inicializa com o endereço do primeiro nó da lista de eleitores, que é armazenado na variável listaEleitores.

Em seguida, um loop while é executado enquanto o ponteiro temp não for nulo, indicando que ainda há nós na lista. Dentro do loop, o seguinte processo ocorre:

1. Um ponteiro auxiliar chamado eleitorRemovido é criado e recebe o valor do ponteiro temp. Isso é feito para preservar o nó atual antes de avançar para o próximo nó da lista.
2. O ponteiro temp é atualizado para apontar para o próximo nó da lista, acessado através do campo proximo do nó atual.
3. A memória alocada para o nó anteriormente apontado por eleitorRemovido é desalocada usando o operador delete. Isso garante que a memória seja liberada

corretamente e evita vazamentos de memória.

Esse processo é repetido até que todos os nós da lista tenham sido percorridos e a memória alocada para cada nó tenha sido liberada.

A função liberarMemoriaEleitores é importante para garantir uma gestão adequada da memória e evitar vazamentos de memória. É recomendado chamar essa função quando não houver mais necessidade de utilizar a lista encadeada de eleitores, a fim de evitar problemas de desperdício de recursos e manter o sistema eficiente.

2.4 Main

A função main é o ponto de entrada do programa e controla o fluxo principal do sistema de votação. Começando com a declaração da variável opcao do tipo int para armazenar a opção escolhida pelo usuário, em seguida, é iniciado um loop do-while que executa o código dentro dele pelo menos uma vez e continua repetindo enquanto a opção escolhida pelo usuário for diferente de zero (0).

Dentro desse loop, são exibidas no console as opções disponíveis do sistema de votação, como inserir candidato, remover candidato, listar candidatos, inserir eleitor, remover eleitor, listar eleitores, votar e gerar relatório, além da opção para sair do programa. O usuário é solicitado a escolher uma opção através da entrada padrão (cin) e o valor é armazenado na variável opcao.

Em seguida, um switch-case é utilizado para executar o código correspondente à opção escolhida pelo usuário. Para cada opção, um código específico é executado. Por exemplo, se a opção escolhida for 1, o usuário é solicitado a digitar o nome e o número do candidato, e a função inserirCandidato é chamada com os valores digitados.

O loop continua até que o usuário escolha a opção 0, que representa sair do sistema. Quando isso acontece, a mensagem "Saindo..." é exibida no console. Após sair do loop, as funções liberarMemoriaCandidatos e liberarMemoriaEleitores são chamadas para liberar a memória alocada para as listas de candidatos e eleitores, respectivamente.

Finalmente, a função main retorna 0, indicando que o programa foi executado com sucesso. Em resumo, essa função é responsável por controlar a interação com o usuário, exibir as opções disponíveis, chamar as funções correspondentes e garantir o funcionamento adequado do sistema de votação.