UNIVERSIDADE FEDERAL DO ESPIRITO SANTO

Segundo Trabalho de Programação III

Relatório

Trabalho referente à disciplina de Programação III do curso de Ciência de Computação do Departamento de Informática da Universidade Federal do Espírito Santo.

Lucas Moraes Soares de Souza
Lucas Mendonça Emery Cade

Vitória
Novembro de 2018

Sumário

1 Introdução

O segundo trabalho de implementação tinha como objetivo exercitar os conceitos básicos de programação orientada a objeto aprendidos no primeiro trabalho, aprender os conceitos básicos da linguagem C++ e aprender como entender críticas e melhorias possíveis para o sistema implementado da primeira vez, para a segunda implementação.

Como no primeiro trabalho, a partir de um arquivo ".csv" referente a uma eleiçao municipal, o programa deveria:

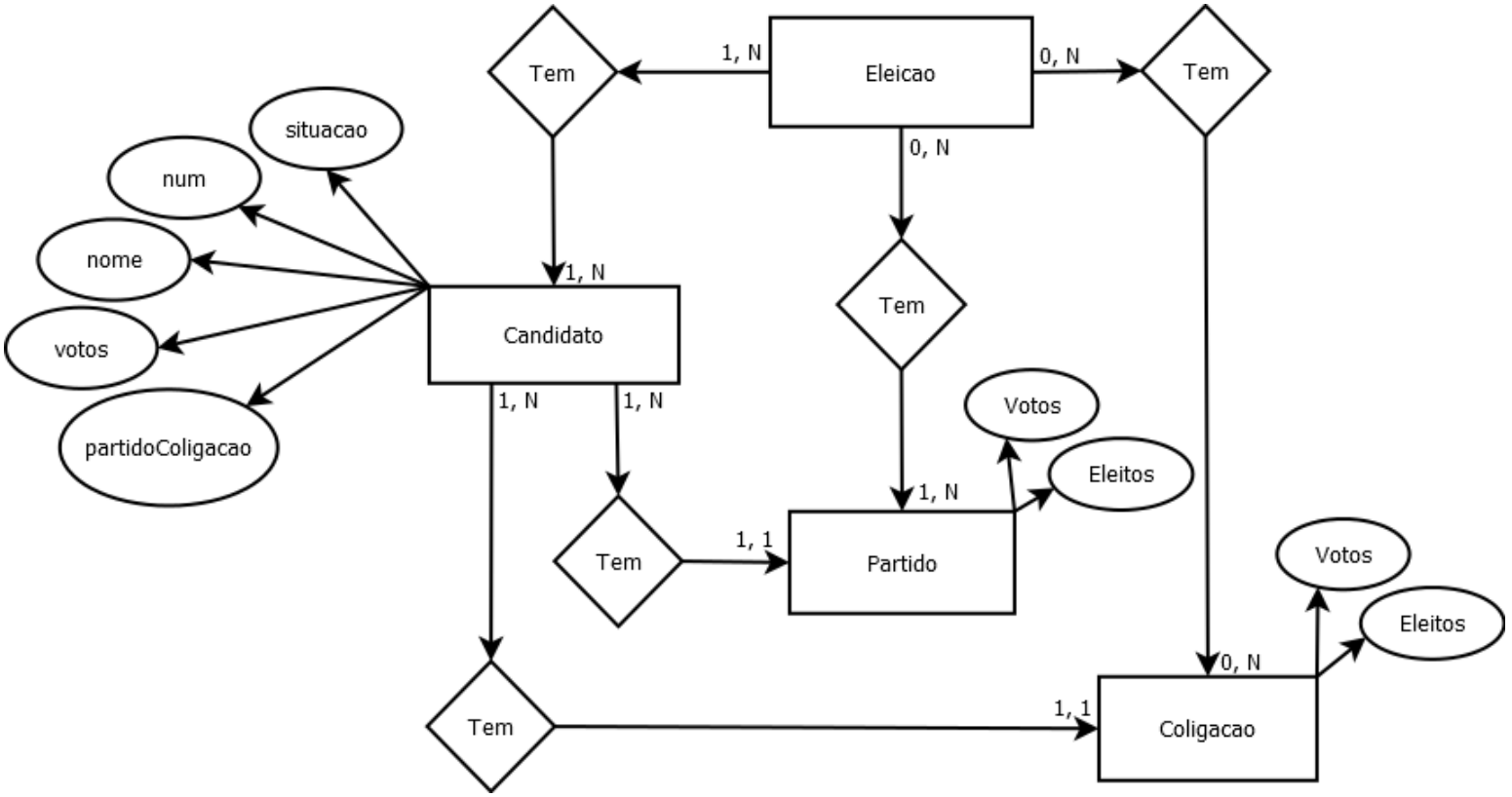
1. Armazenar as informações lidas no arquivo.
2. Gerar as seguintes informações, ordenadas, na saída padrão:
    a. Número de vagas.
    b. Candidatos eleitos (sempre indicando partido, número de votos e coligação).
    c. Candidatos mais votados, dentro do número de vagas.
    d. Candidatos não eleitos, mas que seriam eleitos caso fosse usado o sistema majoritário.
    e. Candidatos eleitos no sistema vigente, e que não seriam eleitos caso fosse usado o sistema majoritário.
    f. As coligações e seus respectivos totais de votos.
    g. Os partidos e seus respectivos totais de votos.
    h. O total de votos nominais da eleição.
3. Liberar a memória alocada para a execução do código.

As listas estão ordenadas pelas especificações do item em questão ou em ordem decrescente de votos.

Foram usadas as classes Eleição[3.1], Candidato[3.2], Partido[3.3], Coligação[3.4], Entrada[3.5] e Saída[3.6] (os últimos dois sendo classes organizacionais que só servem para entrada e saída de dados). As classes são divididas em um arquivo cabeçalho "[Classe].h" e um arquivo de implementação "[Classe].cpp", sem acentos e símbolos especiais nos nomes. As classes serão explicadas em suas respectivas seções do relatório.

2 Diagrama

As classes não-organizacionais e seus atributos são organizadas da seguinte forma:



3 Classes

3.1 Eleição

A classe eleição é a classe geral que engloba todas as estruturas e listas do trabalho. Seus atributos são:

- candidatos;

  Uma lista de ponteiros para instâncias da classe Candidato.

- partidos;

  Uma lista de ponteiros para instâncias da classe Partido.

- coligacoes;

  Uma lista de ponteiros para instâncias da classe Coligação.

- vagas;

  Um inteiro que armazena o número de vagas da eleição

Todos os atributos apresentam um método “get” próprio.

O único método “set” é o “setvagas()”, que percorre a lista de candidatos e calcula o número de candidatos eleitos, que correspone ao número de vagas.

O método “partidoColigacaoVagas()” executa o “setvagas()” e percorre a lista de candidatos executando, para cada candidato, o método seguinte.

O método “adicionaPartidoColigacao(Candidato*)” divide a string referente ao partido e a coligação do candidato (ou identifica que o candidato não tem uma coligação) e chama as duas funções seguintes.

Os métodos “adicionaPartido(Candidato*, string)” e “adicionaColigacao(Candidato*, string)” fazem o mesmo processo para adicionar um partido e uma coligação, respectivamente, nos atributos do candidato e nas listas da eleição.

O método “adicionaCandidato(Candidato*)” somente adiciona um candidato à lista de candidatos referente à eleição.

3.2 Candidato

A classe candidato é a classe com o maior número de atributos, sendo o foco da implementação do trabalho. Seus atributos são:

- situacao;

  Um caracter que representa a situação do candidato (‘*’ representa eleito, ‘#’ representa votos não contabilizados e um número representa situação normal não eleito).

- num;

  Um inteiro que representa o número único do candidato.

- nome;

  Uma string com o nome do candidato.

- partido;

  Um ponteiro para uma instância da classe Partido, à qual o candidato é associado.

- coligacao;

  Um ponteiro para uma instância da classe Partido, à qual o candidato é associado. Se o partido do candidato não tiver coligação, a coligação é o partido individualmente.

- votos;

  Um inteiro que representa a contagem de votos do candidato.

- partidoColigacao;

Uma string auxiliar que armazena a string de entrada “[Partido] – [Coligação]” referente ao candidato, usada no método “adicionaPartidoColigacao(Candidato*)”.

A classe Candidato tem, como métodos, só os “getters” e “setters” dos atributos

3.3 Partido e 3.4 Coligação

As classes partido e coligação são funcionalmente iguais, sua única diferença sendo o meio como elas são criadas (no método “adicionaPartidoColigacao(Candidato* c)”). Seus atributos são:

- nome;

  String que armazena o nome do partido ou uma string com os partidos participantes de uma coligação.

- votos;

  Inteiro que armazena o somatório dos votos de todos os candidatos pertencentes a um partido ou a uma coligação.

- eleitos;

  Inteiro que armazena o número de candidatos eleitos pertencentes a um partido ou a uma coligação.

Os únicos métodos da classe são os ”getters” e “setters” dos atributos.

3.5 Entrada

A classe entrada é uma classe meramente organizacional, que comporta os métodos utilizados para a leitura do arquivo “divulga.csv” de entrada. Ela não tem atributos.

O método “leEntrada(ifstream&, Eleicao&)” lê o arquivo de entrada e, para cada linha do arquivo, cria um candidato e executa a função de

criação de candidato a seguir. Depois, ela insere o candidato na lista de candidatos da eleição e preenche as listas de partidos e coligações.

O método “Candidato* criaCandidato(string&)” lê uma linha do arquivo de entrada referente às informações de um candidato e armazena elas em strings ou inteiros, utilizando o método “int tiraPontos(string&)” a seguir, como necessário. Depois, ela envia esses dados para o construtor da classe Candidato.

O método privado “int tiraPontos(string&)” verifica se uma string referente a um número inteiro tem pontos de divisão de milhar e, se tiver, retira os pontos da string. Depois, ele transforma a string em um número inteiro, que é retornado.

3.6 Saída

A classe saída é outra classe meramente organizacional, comportando as funções utilizadas para impressão na saída padrão, nos modelos fornecidos. Ela tem duas listas de ponteiros para a classe Candidato, que são usadas para gerar as outras listas a serem impressas. Essas listas são:

- eleitos;

  Uma lista de ponteiros para classe Candidato, armazenando somente os candidatos eleitos.

- maisVotados;

  Uma lista de ponteiros para classe Candidato, armazenando todos os candidatos ordenados decrescentemente por número de votos.

O método principal dessa classe é o método “imprimeEleicao(Eleicao&)” que, em prática, executa todos os métodos abaixo.

Os métodos “imprimeCandidato(Candidato*)”, “imprimePartido(Partido*)” e “imprimeColigacao(Coligacao*)” imprimem, dentro dos modelos fornecidos, uma instância de uma das classes componentes do trabalho.

Os métodos “string criaEleitos(Eleicao&)”, “string criaMaisVotados(Eleicao&)”, “string criaEleitosMajoritaria(Eleicao&)” e “string criaBeneficiados(Eleicao&)” criam strings dentro dos modelos fornecidos para a impressão das listas de candidatos especificadas na Introdução. Para isso, alguns deles usam os métodos auxiliares a seguir.

Os métodos “string nFoiEleito(Eleicao&, Candidato*, int)” e “string foiEleito(Eleicao&, Candidato*)” são métodos auxiliares aos métodos acima, utilizados para facilitar a criação das listas especificadas.

Os métodos “string votacaoColigacao(list<Coligacao*>)” e “string votacaoPartido(list<Partido*>)” criam strings dentro dos modelos fornecidos para a impressão das listas de coligações e partidos especificadas na introdução.

O método “string totaisNominais(Eleicao&)” cria uma string dentro do modelo fornecido para a impressão do total de votos nominais especificado na introdução.

4. Testes

O código do trabalho foi testado, em um computador do LabGradII, nos exemplos fornecidos, utilizando a ferramenta de teste "test.sh" fornecida pelo professor. Os resultados foram:

2017100210@labgrad:~/Prog III/Prog3Trab2-master/2018-02-trab2$ ***./test.sh***
Script de teste PROG3 - Trabalho 2

[I] Testando src...
[I] Testando src: teste capela
[I] Testando src: teste capela, tudo OK
[I] Testando src: teste manaus
[I] Testando src: teste manaus, tudo OK
[I] Testando src: teste riodejaneiro
[I] Testando src: teste riodejaneiro, tudo OK
[I] Testando src: teste saopaulo
[I] Testando src: teste saopaulo, tudo OK
[I] Testando src: teste vilavelha
[I] Testando src: teste vilavelha, tudo OK
[I] Testando src: teste vitoria
[I] Testando src: teste vitoria, tudo OK
[I] Testando src: pronto!

5. Conclusão

A execução do segundo trabalho foi, mecânicamente, bem mais simples que a execução do primeiro trabalho, demonstrando a evolução dos integrantes do grupo no entendimento dos conceitos de programação orientada a objeto.

Algumas dificuldades ainda foram encontradas, pela diferença da linguagem C++ da linguagem Java, utilizada anteriormente, e de confusões com os conceitos da linguagem C, que é visualmente e organizacionalmente similar.

É visível, também, o amadurecimento por parte dos integrantes do grupo para com a organização dos métodos e atributos e reutilizabilidade do código, por terem sido aplicados formas de organização ensinadas em sala de aula e na matéria de Engenharia de Software.

Conclui-se, portanto, que o objetivo inicial de trabalhar os conceitos básicos de orientação a objeto e da linguagem de C++, além de trabalhar na organização do código em classes e entendimento de correções e críticas feitas ao código do primeiro trabalho foi cumprido.