Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II
Prof. Felipe Domingos da Cunha / Prof. Max do Val Machado

Trabalho Prático III - Estruturas Sequenciais

1 Regras Básicas

1. extends TP1RegrasBasicas;
2. Todas as classes devem ser entregues em um único arquivo. Essa regra será necessária para a submissão de trabalhos no Verde e no identificador de plágio utilizado na disciplina.
3. Use o padrão Java para comentários.
4. Na leitura de um número inteiro, se seu valor for vazio ou não número, ele recebe zero.

2 Introdução

O Instituto Brasileiro de Geografia e Estatística (IBGE) publica anualmente uma pesquisa identificando o Perfil dos Municípios Brasileiros. A última publicação é refente ao ano de 2015[1]. Nela, o IBGE divulga os resultados da Pesquisa de Informações Básicas Municipais realizada, em 2015, nas prefeituras dos 5.570 municípios brasileiros. Os dados apresentados estão agregados por classes de tamanho da população, estados e regiões, tendo como norte a ampliação e a atualização permanente das variáveis investigadas pela pesquisa desde 1999 (primeira edição). Na versão atual, a publicação está organizada em seis capítulos que destacam aspectos relevantes da gestão e da estrutura dos municípios a partir dos seguintes eixos: recursos humanos das administrações municipais, legislação e instrumentos de planejamento, recursos para a gestão, gestão ambiental, articulação interinstitucional - já investigados anteriormente -, além de terceirização e informatização, tema incluído na presente edição em razão de sua importância para a avaliação dos serviços prestados pelas prefeituras. O permanente esforço de atualização temática da pesquisa tem renovado o interesse de diferentes agentes e organizações na obtenção de informações que contribuam para a implementação de políticas setoriais eficientes, mediante a compreensão da dinâmica que os respectivos fenômenos vêm adquirindo na escala local.

[1] Disponível em *www.ibge.gov.br/home/estatistica/economia/perfilmunic/*2015/*default.shtm*

Os dados dessa pesquisa (na verdade, quase todos) foram armazenados em arquivos txt disponíveis na coleção municipio.zip. Observe que os arquivos podem apresentar diferentes *encodings* (ver unidade01j_conceitosBasicos_formatacao.pdf). O arquivo dicionario.txt mostra os dados disponíveis e sua localização nos demais arquivos. Essa localização está disponível na primeira linha de cada arquivo. Por exemplo, veremos que a localização da Região é a A199 em variaveisexternas.txt e o ano da última atualização do plano diretor é o A19 em planejamentourbano.txt. Nos arquivos txt (exceto em dicionario.txt), a primeira linha indica o campo de localização e as demais, o dado de cada localização para cada município. Veja que a ordem dos município é a mesma em todos os arquivos. Por exemplo, a primeira cidade de todos eles é ALTA FLORESTA DOESTE/RO.

A coleção municipio.zip deve ser descompactada na pasta /tmp/. Quando reiniciamos o Linux, ele normalmente apaga os arquivos existentes na pasta /tmp/.

3 Descrição

1. **Classe Município**: Crie uma classe *Município* seguindo todas as regras apresentadas no slide unidade01g_conceitosBasicos_introducaoOO.pdf. Sua classe terá os atributos privados ID (int, A1), nome (String, A202), UF (String, A201), codigoUF (int, A200), população (int, A204), númeroFuncionários (int, A2 - total de funcionários ativos da administração direta), númeroComissionados (int, A5 - total de funcionários ativos da administração direta - Somente comissionados), escolaridade (String, A16 - escolaridade do gestor), estágio (String, A143 - estágio agenda 21), atualizaçãoPlano (int, A19 - ano da última atualização do plano diretor), região (String, A199), atualizaçãoCadastro (int, A63 - ano da última atualização completa do cadastro) e consórcio (boolean, A152 - se o município faz parte de consórcio público na área de Educação Intermunicipal). Ela terá também pelo menos dois construtores, e os métodos gets, sets, clone e imprimir e ler. O método imprimir mostra a String "ID nome UF codigoUF população númeroFuncionários númeroComissionados escolaridade estágio atualizaçãoPlano região atualizaçãoCadastro consórcio", contendo todos os atributos da classe. O método ler deve efetuar a leitura dos atributos de um registro. Veja que os atributos estão divididos em vários arquivos.

   A entrada padrão é composta por várias linhas e cada uma contém um inteiro indicando a linha do município a ser lido. Por exemplo, o número 1 indica a leitura de informações sobre ALTA FLORESTA DOESTE/RO. A última linha da entrada contém o número 0. A saída padrão também contém várias linhas, uma para cada registro contido em uma linha da entrada padrão.

2. **Lista com Alocação Sequencial**: Crie uma Lista de Municípios baseada na lista de inteiros vista na sala de aula. Sua lista deve conter todos os atributos e métodos existentes na lista de inteiros, contudo, adaptados para a classe *Município*. De toda forma, lembre-se que, na verdade, temos uma lista de ponteiros e cada um deles aponta para um objeto *Município*. Neste exercício,

faremos inserções, remoções e mostraremos os elementos de nossa lista.

Os métodos de inserir e remover devem operar conforme descrito a seguir, respeitando parâmetros e retornos. Primeiro, o *void inserirInicio(Município município)* insere um objeto na primeira posição da Lista e remaneja os demais. Segundo, o *void inserir(Município município, int posição)* insere um objeto na posição p da Lista, onde $p < n$ e n é o número de objetos cadastrados. Em seguida, esse método remaneja os demais objetos. O *void inserirFim(Município município)* insere um objeto na última posição da Lista. O *Município removerInicio()* remove e retorna o primeiro objeto cadastrado na Lista e remaneja os demais. O *Município remover(int posição)* remove e retorna o objeto cadastrado na p-ésima posição da Lista e remaneja os demais. O *Município removerFim()* remove e retorna o último objeto cadastrado na Lista.

A entrada padrão é composta por duas partes. A primeira é igual a entrada da primeira questão. As demais linhas correspondem a segunda parte. A primeira linha da segunda parte tem um número inteiro n indicando a quantidade de objetos a serem inseridos/removidos. Nas próximas n linhas, tem-se n comandos de inserção/remoção a serem processados neste exercício. Cada uma dessas linhas tem uma palavra de comando: II inserir no início, I* inserir em qualquer posição, IF inserir no fim, RI remover no início, R* remover em qualquer posição e RF remover no fim. No caso dos comandos de inserir, temos também um número inteiro indicando a linha (nos arquivos txt) do registro a ser inserido. No caso dos comandos de "em qualquer posição", temos também a posição. No Inserir, a posição fica imediatamente após a palavra de comando. A saída padrão tem uma linha para cada objeto removido sendo que essa informação será constituída pela palavra "(R)" e o atributo **nome**. No final, a saída mostra os atributos relativos a cada objeto cadastrado na lista após as operações de inserção e remoção.

3. **Pilha com Alocação Sequencial**: Crie uma Pilha de Municípios baseada na pilha de inteiros vista na sala de aula. Neste exercício, faremos inserções, remoções e mostraremos os elementos de nossa pilha. A entrada e a saída padrão serão como as da questão anterior, contudo, teremos apenas os comandos I para inserir na pilha (empilhar) e R para remover (desempilhar).

4. **Fila Circular com Alocação Sequencial**: Crie uma classe *Fila Circular* de *Município*. Essa fila deve ter tamanho cinco. Em seguida, faça um programa que leia vários registros e insira seus atributos na fila. Quando o programa tiver que inserir um objeto e a fila estiver cheia, antes, ele deve fazer uma remoção. A entrada padrão será igual à da questão anterior. A saída padrão será um número inteiro para cada registro inserido na fila. Esse número corresponde à média **arredondada** do atributo população dos registros contidos na fila após cada inserção. O final da saída padrão mostra os registros existentes na fila seguindo o padrão da questão anterior.

5. **Lista com Alocação Sequencial em C**: Refaça a questão 2, Lista com Alocação Sequencial, na linguagem C.