Universidade Federal do Ceará - Campus Quixadá
QXD0115 – Estrutura de Dados Avançada – Turma 01A
Curso de Ciência da Computação
Prof. Atílio Gomes Luiz

Avaliação Parcial 02 — Aplicação de Árvore AVL

1. (10 points) Nesse trabalho, deverá ser desenvolvida uma aplicação que utilize a estrutura de árvore binária de busca balanceada AVL.

   A aplicação a ser desenvolvida consiste na leitura de um arquivo contendo informações de pessoas e armazenamento destes dados na memória principal de modo a possibilitar rápido acesso aos registros por meio de consultas por campos individuais.

   O trabalho deverá passar por três etapas distintas: (1) entendimento do problema a ser abordado, (2) construção da implementação, documentação, depuração e (3) escrita de relatório final.

Descrição do Problema

Deverá ser desenvolvida uma aplicação que seja capaz de carregar um arquivo de entrada contendo as seguintes informações de pessoas e na seguinte ordem: número do CPF (numérico), primeiro nome da pessoa (string), sobrenome da pessoa (string), data de nascimento da pessoa (data no formato MM/DD/AAAA) e nome da cidade onde nasceu (string).

O arquivo a ser lido estará em um formato CSV, de modo que os campos de cada registro sejam separados por vírgula (,). Seguem algumas linhas de um exemplo de como será o arquivo a ser lido:

```
388.624.732-57,Tiago,Cunha,7/19/1989,Rio Branco
992.809.969-32,Kai,Gomes,10/29/1953,Jundiaí
518.376.278-35,Estevan,Azevedo,1/3/1938,Nilópolis
```

O arquivo de exemplo completo está no Moodle, juntamente a esta descrição do projeto. Após carregar o arquivo em objetos `Pessoa`, deverá ser possível realizar as seguintes consultas:

- Consultar uma única pessoa pelo seu CPF e exibir seus dados na tela.
- Consultar todas as pessoas cujo nome comece com uma string informada pelo usuário e exibir na tela todos os dados dessas pessoas na forma de lista.
- Consultar todas as pessoas cuja data de nascimento esteja em um intervalo estabelecido pelo usuário e exibir na tela todos os dados dessas pessoas na forma de lista.

Implementação

O programa deverá ser escrito seguindo as seguintes restrições:

- O programa deve ser desenvolvido na linguagem de programação C++.
- Deve usar o paradigma de programação orientado a objeto.
- **Disposição dos dados na memória:** Os registros lidos do arquivo deverão ser, obrigatoriamente, indexados pelos campos supracitados de modo que todas as consultas possam ser executadas de forma bem rápida. Para isso, será necessária e OBRIGATÓRIA a criação dessa indexação em cada um dos campos já mencionados de uma Pessoa.

  Utilize a estrutura de árvore binária de busca balanceada AVL, sendo que, para cada campo indexado (no caso, CPF, nome da pessoa e data de nascimento), crie uma árvore binária de busca AVL. Observe que não está sendo solicitada a implementação de três classes de árvores distintas. Nesse ponto, a criação se refere à instanciação da árvore AVL três vezes, sendo uma voltada para cada campo. Assim sendo, o aluno pode e é incentivado a implementar sua árvore AVL utilizando templates e deixando a implementação genérica o suficiente para ser reutilizada em qualquer uso de propósito geral.
- **Observação:** Os registros (objetos `Pessoa`) não deverão estar duplicados na memória. A indexação é feita apenas com chaves e endereços dos registros, e não com cópias inteiras dos registros na memória.
- **Interface com o usuário:** Deverá ser criada uma interface minimamente navegável (via terminal) possibilitando a realização das consultas e visualização dos dados retornados.

O que deve ser entregue?

- Código fonte em C++ (bem indentado e comentado). Códigos não documentados terão redução na nota.
- **Relatório do trabalho.** Entre outras coisas, o relatório deve conter:
  1. **Introdução:** descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. **Implementação:** descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada, o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.

     Importante: os códigos utilizados nas implementações devem ser inseridos na documentação.

3. **Análise de Complexidade:** estudo da complexidade de tempo e espaço das funções implementadas e do programa como um todo (notação O).
4. **Conclusão:** comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
5. **Bibliografia:** bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso. Uma referência bibliográfica deve ser citada no texto quando da sua utilização.
6. **Formato:** mandatoriamente em PDF. Para quem sabe um pouco de LaTeX, vários templates de relatórios podem ser encontrados na internet, em particular, aqui: `https://pt.overleaf.com/latex/templates`

Entrega

A entrega deve ser feita via Moodle. Os arquivos de código fonte e o relatório deverão ser compactados em um único arquivo (ZIP, RAR, 7Z, etc.) e enviados **até o dia 27/09/2020, 23h59**. Deverá conter na raiz do arquivo compactado, um arquivo de nome LEIAME.TXT explicando como compilar e executar o programa. Adicione scripts de compilação/execução caso seja possível/necessário.

Avaliação

A avaliação será realizada em duas etapas: uma referente ao programa e a outra referente ao relatório.

Na avaliação, o professor considerarará o funcionamento correto do programa, o atendimento de todos os requisitos e a obrigatoriedade da implementação correta da árvore AVL. Também será considerada a qualidade do relatório: formato, e explicação do aluno acerca do código desenvolvido e da solução projetada.

Informações adicionais

- Clareza, indentação e comentários no programa serão considerados na nota;
- O trabalho deve ser feito em **dupla** ou **individualmente** (grupo de **no máximo DOIS** alunos);
- **Não copie trabalho.** Trabalhos copiados terão nota **ZERO**;
- Trabalhos entregues em atraso serão aceitos, todavia será descontado 0,1 pontos por hora de atraso.