Especificação do trabalho final

A atividade final é a implementação de uma solução que use Multilista e/ou Lista invertida como forma de indexação de dados não-únicos. A estrutura de dados que estiver "por baixo" da multilista ou da lista invertida é de escolha da equipe. Poderá ser usada uma tabela, uma árvore binária, um hash, etc.

A decisão pelo conjunto de dados que será trabalhado (assim como sua formação) é de responsabilidade da equipe (ex: seleções da copa, alunos da ufsc, torcedores dos times tais, ...)

O programa deverá indexar, no mínimo 3 colunas de dados do conjunto, oferecendo busca simples a partir das 3 colunas bem como busca composta por 2 das colunas. Exemplos de busca simples: torcedores do Figueirense, estudantes de Criciúma, etc. Exemplo de busca composta: torcedores do Figueirense que são de Criciúma. Enquanto duas colunas de dados deverão ser de dados discretos, a terceira deverá conter valores contínuos e, por isso, a indexação deverá ser feita a partir de faixas de valores pré-definidas.

A solução do problema deverá ter uma opção de "carga de dados", em que um conjunto prévio de dados será introduzido na estrutura de dados escolhida, bem como uma opção para inclusão de novos dados pelo usuário (ou seja, eu).

O usuário deve poder executar as seguintes operações:

- Carga de dados
- Busca simples -> escolha de uma coluna de dados e especificação de um valor para busca -> com exibição do resultado da busca (exemplo: escolha da coluna Times, seguido da especificação do time "Figueirense")
- Busca composta -> especificação de dois valores para a busca -> com exibição do resultado da busca
- Inclusão de novo elemento
- Remoção de elemento existente
- Exibir todos os dados

Essas operações serão feitas a partir de uma interface com menus - gráfica ou não, não importa.

O trabalho final da disciplina deve obedecer a seguinte orientação:

- Deve ser implementado por uma equipe de até 2 3 integrantes (sem choro)
- Deve ser implementado em uma linguagem de programação de alto nível, obedecendo preceitos de boa programação e de orientação a objetos
- Deve ser entregue funcionando e com todos os arquivos necessários para a avaliação do professor quanto ao funcionamento, permitindo que seja compilado em ambiente padrão
- Deve ser acompanhado por documentação que apresente, no mínimo: o projeto de solução do problema e as decisões de projeto ("escolhemos usar multilista porque blablabla", "dividimos em 3 classes x, y e z porque assim blebleblé"...), a modularização adotada. As decisões devem ser não somente descritas, mas justificadas/defendidas.

Relatório da Implementação do programa “Gerenciador de Cadastro de pacientes”.

A solução do problema apresentado na atividade final da disciplina, foi resolvido através da implementação de uma classe gerenciadora nomeada: Gerenciador. Uma instância dessa classe é capaz de manipular objetos do tipo Usuário, este, um tipo de Pessoa, através de seu armazenamento em uma entidade nomeada de Box. Esta entidade é capaz de se organizar de forma linear formando uma sequencia ou lista de objetos duplamente encadeados.

A aplicação por mim criada é apenas um módulo de um sistema cujas informações são alimentadas por outros módulos específicos para outros profissionais, médicos, ou contadores, por exemplo. O uso desse modulo implementado é voltado para algum setor administrativo que efetua o cadastro, e sua manutenção, dos usuários de um estabelecimento de saúde, poderia ser uma clínica particular ou uma unidade publica de saúde.

As estruturas utilizadas na implementação formam a lista duplamente encadeada e a lista invertida. Para a organização da base de dados do programa foi usada a lista duplamente encadeada, pois sendo uma estrutura de fácil implementação, fácil manipulação dos dados, já que se pode com ela não ser necessária muita preocupação com seu aumento de tamanho, foi a melhor solução encontrada para quantidade de dados que a aplicação iria trabalhar neste exemplo. Para a otimização das pesquisas realizadas nos dados do programa, foi utilizada a lista invertida. Ela foi escolhida devido a rapidez e facilidade que se pode ter acesso direto aos dados, já que a mesma armazena os endereços dos objetos. A opção por esta estrutura de dados foi feita pois acredito que ela possui vantagens se comparado a multilista, já que a sua estrutura é criada a parte, sua única relação com os objetos da base de dados são seus endereços. Sendo assim eu posso manipulá-la sem haver necessidade de acessar a estrutura que armazena os dados da base de dados.

O conjunto de dados utilizados no programa foram instâncias da Classe Usuário. Ela é subclasse da classe pessoa. Um usuário no domínio da solução criada é uma pessoa que, possui os atributos básicos como Nome, RG, data de nascimento, dentre outras, e que também possui um número de prontuário médico e um histórico de atendimento no estabelecimento de saúde que ele frequenta. A chave primária adotada foi o número do prontuário.

A massa de dados é organizada na memória com o auxílio de instâncias da Classe Box. Elas organizam-se em forma de uma lista duplamente encadeada. Cada objeto desse tipo possui um compartimento capaz de receber um objeto da classe usuário, além de outros dois compartimentos que armazenam dois outros objetos desta mesma classe onde serão armazenados os endereços do anterior e do próximo Box que conterão os outros usuários. A estrutura foi montada dessa forma, pois, assim, pode-se percorrer sob os dados ou aumentar o tamanho da lista nos dois sentidos.

O problema de modelar uma estrutura de lista invertida foi resolvido pela classe Registro no diretório. Ela faz o papel de diretório onde serão indexados os dados e foi implementada com base em uma analogia a uma linha de uma tabela. Cada instância dessa classe representa uma linha do diretório. Um objeto dessa classe possui um compartimento para armazenar a cabeça de uma lista encadeada de objetos do tipo box que conterão os

endereços dos objetos do tipo usuários que foram indexados e classificados e pertencem a esta “linha” do diretório. Esta instância ainda possui um rótulo que classifica os objetos desta linha e dois outros objetos do seu mesmo tipo que fazem referência as outras linhas desse diretório. A implementação foi feita dessa forma para que se pudesse percorrer o diretório livremente dando a impressão de se estar percorrendo uma tabela. Dessa forma o diretório assemelha-se a uma lista encadeada que pode crescer em todas as direções, tanto em linhas de registro de diretório quanto em quantidade de indexações de objetos.

As informações pertinentes aos objetos da classe usuário que foram indexadas na técnica de lista invertida foram: a idade, o sexo e a raça do usuário. Foram escolhidas estas pois são características físicas importantes capazes de identificar e distinguir indivíduos. As características sexo e raça/cor são armazenadas de forma discreta em seus respectivos diretórios: em cada linha do diretório serão armazenados tipos físicos diferentes, uma para masculino outra para feminino, o diretório sexo, e uma para branco, outra para indígena, no diretório raça. A característica idade é um valor contínuo, não ficaria interessante indexar dessa forma o diretório pois poderá haver uma má distribuição dos dados, exemplo: caso fossem armazenados de forma continua no diretório e fosse necessário encontrar um usuário com 15 anos, teria de se percorrer as linhas do diretório de forma sequencial até chegar na linha com rótulo 15 para após percorrer a linha para localizá-lo, se caso esse usuário procurado estivesse no meio da linha eu já teria feito 15 comparações só para localizar a linha. Para evitar esse tipo de caso, que posso chamar de médio, foram criadas faixa etárias para as idades dos usuários: de 0 a 14(crianças), de 14 a 60(adultos) e de 60 acima (idosos). Essas faixas etárias foram feitas dessa forma, pois pode ajudar a identificar que tipo de profissional da área médica pode atender esse usuário futuramente, já que existem médicos especialistas em pediatria e geriatria.

A manipulação de todos os dados das estruturas implementadas é feita através um uma classe cujo objeto é o gestor que faz a comunicação com a interface gráfica com o usuário. Qualquer tipo de informação referente aos usuários é fornecedor por este objeto. Sendo assim, a ele pertence as estruturas criadas. Ele possui como atributos então:

- Um objeto Box, que seria a cabeça da lista básica de dados do programa;
- Três objetos Registro no diretório, que seriam cada um deles a cabeça, ou primeira linha dos diretórios: sexo, idade, e raça;
- Uma variável primitiva inteira, que armazenará a quantidade total de usuário já cadastrado;
- Uma variável primitiva inteira, que será o contador do número de prontuários.

Diagrama de classes do domínio do problema.