29/04/2015

Trabalho de Programação 3

Prof. Flávio Miguel Varejão

I. Descrição

A comunidade da UFES é composta por um corpo de funcionários (professores e servidores) e estudantes (de graduação ou pós-graduação). Todos possuem como características comuns seu nome, idade, sexo, matrícula e data de entrada (data de assinatura do contrato, no caso de funcionários, e data de primeira matrícula, no caso de estudantes).

Além das características comuns, os funcionários possuem como característica adicional seu salário. Servidores apresentam ainda uma característica específica: podem ser empregados da UFES ou terceirizados. Professores, por sua vez, têm como característica específica a sua carga de trabalho (DE, 20 ou 40 horas).

Estudantes estão sempre vinculados a um curso e possuem um coeficiente de rendimento. Enquanto a característica específica dos estudantes de graduação é a pontuação obtida pelo aluno no vestibular, a característica específica dos estudantes de pós-graduação é o curso de graduação no qual se formou.

É importante destacar que funcionários também podem eventualmente ser estudantes de apenas 1 (UM) curso de graduação ou pós-graduação. Neste caso, eles possuirão tanto dados de funcionário quanto de estudante. Note ainda que nome, idade e sexo sempre serão únicos, enquanto funcionários-estudantes terão duas matrículas e datas de entrada (uma como funcionário e outra como estudante).

O diretor do Núcleo de Tecnologia da Informação (NTI) da UFES resolveu criar um sistema para extrair algumas informações a respeito do corpo de membros dessa comunidade. Para tanto, ele designou um analista para realizar a especificação deste sistema. O analista produziu um documento (veja a seguir) descrevendo, de uma forma geral, a especificação. O diretor decidiu, então, te contratar, para implementar este sistema.

DOCUMENTO DO ANALISTA

II. Especificação do Sistema

Funcionalidades a serem implementadas:

1. Leitura de dados dos membros existentes de um arquivo texto (arquivo mestre), denominado "mestre.txt".
2. Leitura de operações de atualização sobre o arquivo mestre de um arquivo texto (arquivo de atualização), denominado "atual.txt".
3. Obtenção das seguintes listagens:
   a) Listagem ordenada crescentemente por matrícula de todos os professores de regime 40 horas da UFES. Em membros com duas matrículas, utilizar a matrícula de professor.
   b) Listagem ordenada decrescentemente por idade de todos os estudantes de pós-graduação da UFES. Em caso da ocorrência de estudantes com a mesma idade, colocar em ordem decrescente de matrícula. Em membros com duas matrículas, utilizar a matrícula de estudante.
   c) Listagem ordenada decrescentemente por coeficiente de todos os estudantes da UFES. Em caso da ocorrência de estudantes com o mesmo coeficiente, colocar em ordem decrescente de matrícula. Em membros com duas matrículas, utilizar a matrícula de estudante.
   d) Listagem ordenada decrescentemente por salário dos funcionários da UFES. Em caso da ocorrência de funcionários com o mesmo salário, colocar em ordem decrescente de matrícula. Em membros com duas matrículas, utilizar a matrícula de funcionário.
   e) Listagem ordenada crescentemente por nome de todos os membros da comunidade UFES. Em caso da ocorrência de membros com o mesmo nome, colocar em ordem crescente de matrícula. Em membros com duas matrículas, usar a maior das duas.
4. Gravação de relatório, no arquivo "relat.txt", da soma das matrículas do i-ésimo elemento da listagem do item a, do j-ésimo elemento da listagem do item b, do k-ésimo elemento da listagem do item c, do l-ésimo elemento da listagem do item d e do m-ésimo elemento da listagem do item e. Os valores de i, j, k, l e m serão lidos de um arquivo texto, denominado "espec.txt". Em caso de não existir o elemento desejado na listagem (por exemplo, se o valor i, j, k, l ou m for superior ao índice do último elemento da listagem), considerar o valor da matrícula igual a zero na soma. Em caso de membros com duas matrículas, usar a matrícula utilizada nos critérios de desempate de cada listagem.

Formato dos Dados do Sistema:

```
Nome:              até 30 caracteres
Idade:             inteiro não negativo
Sexo:              1 caracter (M ou F)
Matrícula:         inteiro não negativo grande
Data de Entrada:   3 inteiros não negativos
Salário:           ponto flutuante de precisão simples
```

```
Terceirização:      1 caracter (V ou F)
Regime:             2 caracteres (DE, 20 ou 40)
Curso:              até 30 caracteres
Coeficiente:        ponto flutuante de precisão simples
Pontuação no Vest.: inteiro não negativo
Curso de Formação:  até 30 caracteres
```

Entrada de Dados:

A entrada de dados será realizada a partir de três arquivos texto (*mestre.txt* com dados dos membros da universidade; *atual.txt*, com dados sobre as operações a serem realizadas sobre o arquivo mestre; e *espec.txt*, com dados a respeito da soma do relatório a ser gerado pelo programa). Por simplificação, cada registro em cada um destes arquivos será separado por uma linha em branco. Além disso, cada dado de um registro destes arquivos corresponderá a uma linha do arquivo e nunca faltará qualquer dado.

A seguir, apresentam-se as especificações dos registros de cada um destes arquivos com pequenos exemplos correspondentes.

mestre.txt

Especificação do registro:

```
<categoria>    (prof, serv, pos, grad, prof_pos, prof_grad,
                serv_pos, serv_grad)
<nome>
<idade>
<sexo>
<matricula>
<data de entrada>        % no formato dd/mm/aaaa)
<<dados_categoria>>      % dados especificos da categoria
<<dados_categ_adic>>     % dados de categoria adicional (
                             % campos opcionais)


Campos específicos da categoria prof:
<salario>
<regime>

Campos específicos da categoria serv:
<salario>
<terceirizado>

Campos específicos da categoria pos:
<nome do curso>
<coeficiente>
<codigo de curso formado>

Campos específicos da categoria grad:
```

<nome do curso>
<coeficiente>
<pontuação>

Exemplo de arquivo:

```
prof
Flavio Miguel Varejao
35
M
5432
28/12/1989
3000.00
DE

prof_grad
Saulo Bortolon
34
M
5420
28/06/1989
3000.00
40
214365
12/03/2000
Medicina
9.9
2000

grad
Fernanda Tommasi
19
F
124203
01/03/1998
Ciência da Computação
9.5
1890

pos
Nelson dos Santos
32
M
123456
```

```
07/04/1999
Informática
8.7
Matemática

prof
Berilhes Borges Garcia
35
M
5432
28/07/1993
3000.00
DE

grad
Isabel Rosseti
21
F
923456
01/03/1997
Engenharia da Computação
9.6
2100
```

atual.txt

Especificação do registro:

```
<operação>          % i(inclusão);e(exclusão);a(alteração)
<dados da operação> % dados específicos para operação
```

Nas operações de inclusão e alteração, os dados específicos têm formato idêntico aos dos registros do arquivo mestre. No caso da alteração, a matrícula identificará o membro a ser alterado. Todos os dados do membro serão alterados (com exceção da(s) matrícula(s), por motivos óbvios). Na operação de exclusão, o único dado específico é a matrícula do membro que será excluído. Se o membro possuir duas matrículas, só deve ser excluído os dados relativos a matrícula indicada e o membro continua existindo como elemento da categoria cuja matrícula não foi especificada.

Exemplo de arquivo:

```
i
pos
Juliana Pezzin
21
```

```
F
923456
01/06/2002
Artes Plásticas
9.6
1089

e
123456

a
grad
Fernanda Barcellos Tommasi
19
F
124203
01/03/1998
Odontologia
9.5
1800
```

espec.txt

Especificação do registro:

<valor de i>
<valor de j>
<valor de k>
<valor de l>
<valor de m>

Exemplo de arquivo:

```
1
7
3
12
8
```

Saída de Dados:

O arquivo *relat.txt* conterá apenas o valor da soma das matrículas dos alunos selecionados em *espec.txt*.

Especificação do registro:

<valor da soma>

relat.txt

Exemplo de arquivo:

932789

III. Requisitos da implementação

1. Modularize seu código adequadamente. O uso de variáveis globais é proibido;
2. Crie códigos claros e organizados. Utilize um estilo de programação consistente.
3. Os arquivos ***mestre.txt, atual.txt*** e ***espec.txt*** devem ser lidos na mesma pasta onde se encontram os arquivos fonte do seu programa. O arquivo ***relat.txt*** deve ser gerado nesta mesma pasta.
4. Você deve implementar uma lista encadeada genérica para usar no armazenamento dos membros da UFES. Isto é, você não pode usar listas fornecidas por bibliotecas da linguagem. Lembre-se que uma estrutura de dados genérica contém operações que são INDEPENDENTES do tipo do dado armazenado nela. Se, em sua estrutura, alguma operação faz referência a algum ATRIBUTO de um dado armazenado nela, esta estrutura NÃO está genérica e, portanto, está INCORRETA.
5. A lista encadeada deve utilizar um único método de ordenação genérico que será usado para todos os tipos de ordenações requeridas (tanto crescentes quanto decrescentes). Se implementar um método de inserção ordenada, este também deverá ser genérico tal qual o método de ordenação.
6. Você deve criar uma hierarquia de classses para representar os diferentes tipos de membros da UFES.
7. Você deve sobrecarregar os operadores += e ++ na implementação das operações de inclusão de elemento ao final da lista e de retornar o elemento corrente e avançar para o próximo elemento da lista.

IV. Condições de Entrega

O trabalho deve ser feito individualmente e submetido por e-mail até as 23:59 horas das datas limites especificadas (25 de maio para o primeiro trabalho e 22 de junho para o segundo). Note que as datas limites já levam em conta um dia adicional de tolerância para o caso de problemas de submissão via rede. Isso significa que o aluno deve submeter seu trabalho até no máximo um dia antes da data limite. Se o aluno resolver submeter o trabalho na data limite, estará fazendo isso assumindo o risco do trabalho ser cadastrado no sistema após o prazo. Em caso de recebimento do trabalho após a data limite, o trabalho não será avaliado e a nota será ZERO. Aluno que receber zero por este motivo e vier pedir para o professor considerar o trabalho estará cometendo um ato de DESRESPEITO ao professor e estará sujeito a perda adicional de pontos na média.

V. Formato de Entrega dos Trabalhos

Este trabalho deverá ser entregue em duas versões (C++ e Java).

O recebimento dos trabalhos é automatizado. Portanto, as regras a seguir devem ser seguidas à risca para evitar que seu trabalho não possa ser avaliado.

O código-fonte de sua solução deverá ser compactado e entregue por e-mail (anexo ao e-mail) para o endereço fvarejao@ninfa.inf.ufes.br.

Serão aceitos trabalhos entregues até as 23h59 da data limite. O assunto do e-mail deverá ser o seguinte:

```
prog3:trab<id>:<nome>:
```

O termo "<id>" deve ser substituído pelo número correspondente do trabalho (1 ou 2). O termo "<nome>" deverá ser substituído pelo nome e o último sobrenome do aluno, sem acentos, til ou cedilha, como no exemplo abaixo:

```
lp:trab1:Flavio Varejao:
```

Atenção: o e-mail não deve ser enviado por servidores de emails que não seguem padrões normais de envio, tais como, TERRA, HOTMAIL ou BOL, pois o recebimento automatizado não consegue reconhecer seu trabalho.

O arquivo compactado deve estar no formato tar.gz com o nome trab<id>.tar.gz e conter apenas os arquivos fonte do programa (não deve conter executáveis ou arquivos compilados ou arquivos texto com dados). Para isso, abra um console, mude o diretório de trabalho para a pasta onde se encontra o código-fonte do trabalho e execute o seguinte comando (no caso do trabalho 1):

```
tar -zcvf trab1.tar.gz *
```

Preste bastante atenção para fazer com que o codigo fonte não seja colocado em subdiretórios dentro do arquivo compactado. Se isso ocorrer a compilação automática não funcionará e sua nota será ZERO. Atente também que os nomes usados no arquivo principal dos trabalhos DEVEM ser: trab1.cpp e trab2.java.

Um exemplo de um e-mail de envio do trabalho:

```
Para: fvarejao@ninfa.inf.ufes.br
De: Joao da Silva
Assunto: lp:trab1:Joao Silva:
Anexo: trab1.tar.gz
```

Para a compilação e execução dos programas, em princípio, serão utilizadas as versões instaladas no labgrad do g++ e java. Caso haja alguma modificação de versão de correção, ela será divulgada oportunamente.

Os comandos usados para compilação dos programas serão:

```
g++ -Wall -ansi -pedantic *.cpp -o trab1
javac trab2.java
```

Os programas serão compilados e executados no sistema operacional linux. Para que não haja problemas na correção do seu trabalho e você seja prejudicado, garanta que ele é compilado na versão de correção e executado no sistema operacional linux.

Se tudo correr bem, você receberá um e-mail de confirmação do recebimento do trabalho. Neste e-mail haverá um hash MD5 do arquivo recebido. Para garantir que o arquivo foi recebido sem ser corrompido, gere o hash MD5 do arquivo que você enviou e compare com o hash recebido na confirmação. Para gerar o hash, utilize o seguinte comando:

```
md5sum trab1.tar.gz
```

Caso você não receba o e-mail de confirmação ou caso o valor do hash seja diferente, envie o trabalho novamente.

VI. Avaliação

Os trabalhos terão nota zero se:

1. A data de entrega for fora do prazo estabelecido;
2. O trabalho não compilar;
3. O trabalho não gerar o arquivo com o resultado e formato esperado;
4. For detectada a ocorrência de plágio pelo sistema.

Ainda, os trabalhos poderão ser avaliados segundo os seguintes critérios:

5. Cumprimento das restrições estabelecidas no ítem III deste documento;
6. Modularização (considerando o uso de arquivos separados para os diversos tipos de dados);
7. Ausência de uso de variáveis globais;
8. Legibilidade (nomes de variáveis bem escolhidos, código bem formatado, uso de comentários quando necessário, etc.);
9. Consistência (utilização de um mesmo padrão de código);
10. Eficiência (sem exageros, tentar evitar grandes desperdícios de recursos);
11. Para programas em C++, evite vazamentos de memória (memory leaks). Utilize o programa Valgrind (http://valgrind.org) para detectar e eliminar memory leaks.

Observação importante:

Caso haja algum erro neste documento, serão publicadas novas versões e divulgadas erratas em sala de aula. É responsabilidade do aluno manter-se informado, freqüentando as aulas ou acompanhando as novidades na página da disciplina na Internet.