Laboratório de Computação I
Departamento de Engenharia Eletrônica e de Computação - DEL-Poli/UFRJ
Prof.: Pedro Braconnot Velloso

Trabalho: Diário de notas

Escreva um programa para armazenar o nome dos alunos, o DRE, notas de 2 provas, média, número de faltas, e o status em um programa para auxiliar professores. Esse programa deve permitir interativamente as seguintes operações:

1. **abrir/fechar turma a partir do nome. Caso a turma não exista, perguntar se quer ser criada;**
2. **remoção de uma turma: apaga o arquivo associado àquela turma;**
3. **inclusão de um aluno em uma determinada turma (nome, DRE, notas, falta) em ordem alfabética;**
4. **remoção de um aluno específico de uma determinada turma;**
5. **edição das notas e faltas de um aluno (a busca deve ser feita pelo nome);**
6. **consulta os dados de uma turma:**
    a. **um aluno, mostrando: DRE, nome, P1, P2, média, faltas;**
    b. **alunos aprovados (média ≥ 7,0), mostrando: DRE, nome, média;**
    c. **alunos reprovados mostrando: DRE, nome, média;**
7. **listagem de todos os alunos de uma determinada turma, com todos seus dados;**

O programa deve armazenar os alunos de cada turma em um arquivo de registros. Assim, cada turma será armazenada em um arquivo diferente. Os descritores de arquivo para cada turma devem ser armazenados em uma lista de turmas ativas, que será uma lista duplamente encadeada. Esta lista armazena todos as turmas que estão sendo usadas pelo usuário, com seu respectivo arquivo. O usuário pode acessar quantas turmas quiser, sem ter que abrir e fechar outra turma. Portanto, para acessar uma determinada turma deve-se primeiro “abrir” esta turma (que será colocada na lista de turmas ativas) e procurar o descritor de arquivo que está associado a ela na lista encadeada. Ao terminar o acesso a turma, ela deve ser fechada, ou seja, removida da lista de turmas ativas. A lista de turmas ativas deve estar ordenada por nome da turma.

Um menu deve também estar disponível para que o usuário saiba como utilizar o programa. Lembre-se de inserir testes de pertinência sobre os dados inseridos. Todas as opções devem estar disponíveis no menu oferecido ao usuário, inclusive a de sair do programa.

A **inclusão de um aluno deve ser feita já de maneira ordenada**. Isso quer dizer que os dados não podem ser inseridos e posteriormente ordenados.

ATENÇÃO: O único vetor do programa deve ser o de notas dentro do *record*.

Requisitos do programa:

- modularidade: o programa deve usar funções e procedimentos sempre que possível a fim de organizar o código;
- os parâmetros devem ser passados por valor ou por referencia de acordo com a necessidade e não aleatoriamente;
- não usar variáveis globais.
- indentação adequada;
- nomes de variáveis intuitivos;
- comentários ao longo do programa;
- a compilação não deve dar “warnings”;

- usar *Units* para implementar os tratamentos de erros das entradas.

Entrega do programa:

O trabalho deve ser enviado ao e-mail velloso@poli.ufrj.br até o dia **11 de fevereiro**, às 23h55. Trabalhos entregues após esta data serão desconsiderados. O aluno deve colocar o nome do arquivo .pas com o seu nome, exemplo: “pedro−velloso.pas”. O arquivo do programa principal, bem como a Unit, devem estar dentro de um diretório com o nome do aluno, exemplo: pedro−velloso. Este diretório deve ser zipado e enviado para o e-mail especificado. O diretório DEVE conter apenas esses dois arquivos.

Importante: Os trabalhos devem ser feitos **individualmente, compilados e executados corretamente em sala de aula e entregues junto do código fonte**. O programa deve ser apresentado ao Professor Pedro em sala de aula até o dia:

- **12/12/2016** -- turma de segunda-feira
- **13/12/2016** -- turmas de terça-feira.

Trabalhos com código fonte de leitura difícil (variáveis com nomes pouco intuitivos e ausência de identação) perderão pontos. Além disso, os **trabalhos copiados receberão nota ZERO**. Alguns alunos poderão ser escolhidos para explicar o programa desenvolvido oralmente.

O programa **DEVE** usar a seguinte estrutura de dados (estrutura exatamente igual):

```
registro = RECORD
        Nome: string;
        DRE: integer;
        Notas: array[1..2] of real;
        Media = real;
        Faltas = integer;
end;
regFile = file of registro;
```

Dicas:

- Implementar o programa de acordo com as especificações dadas, pois cada item fora da especificação acarretará a perda de pontos;
- outras funcionalidades que não foram especificadas que ajudem e facilitem a utilização do programa serão compensadas com pontos extras;
- Para formatar a impressão da tabela usar o TAB. Basta colocar “**#9**” no ***writeln***, pois o valor ASCII para o TAB é 9. Ele deve ficar de fora das apóstrofes. Ex:
    - writeln(‘Nome’, #9, ‘Idade’);