Departamento de Ciência da Computação – IME-USP

MAC0115 – Introdução à Computação para Ciências Exatas e Tecnologia

INSTITUTO DE FÍSICA — TURMA 21 — SEGUNDO SEMESTRE DE 2008

Segundo Exercício-Programa Data de entrega: até **21 de novembro de 2008.**

Fazendo Média ☺

Todo fim de semestre cada professor tem que calcular a média final das suas turmas. Este ano, o Departamento de Ciência da Computação da USP resolveu uniformizar este processo, contratando vocês para implementarem um programa para o cálculo das médias.

Escreva, na linguagem C, um programa que leia de um arquivo os dados de uma turma e imprima a média final de acordo com o critério também descrito no arquivo de entrada.

No arquivo de entrada estão os seguintes dados, nesta ordem:

- três inteiros positivos n, p e m, representando respectivamente o número de alunos da turma, o número de notas de provas e o número de notas de exercícios-programa;
- uma seqüência de p inteiros positivos representando o peso de cada prova;
- uma seqüência de m inteiros positivos representando o peso de cada exercício-programa;
- dois inteiros positivos *pp* e *pep* representando o peso da média de provas e da média de exercícios-programa para o cálculo da média final;
- uma seqüência de n linhas, cada uma com os dados de um dos n alunos da turma: nome, notas de provas e notas de exercícios-programa.

Vocês podem supor que o nome de cada aluno contém no máximo 29 caracteres. Mais especificamente vocês podem supor que os 30 primeiros caracteres de cada linha com os dados de um aluno contém um nome seguido de brancos (haverá pelo menos um branco depois do nome) e que na coluna 31 começam as notas deste aluno. Cada nota deve ser um número real entre 0 e 10.

Exemplo de arquivo de entrada:

```
3 3 5
1 2 2
1 1 2 2 3
1 2
Cassandra Avestruz da Silva   5.0  6.6  4.2 10.0  9.0  8.0  7.0  6.0
Nicolau dos Santos Neto       3.2  2.0  1.5  4.2  3.1  2.4  1.5  0.0
Romário de Souza Faria        8.0  3.0  7.7  9.5  8.0 10.0  9.5  9.0
```

O seu programa deve pedir que o usuário digite o nome do arquivo de entrada e deve ter como saída um arquivo de nome `"saida.txt"` contendo as seguintes informações:

- uma linha inicial com os rótulos das várias colunas a serem impressas (veja no exemplo abaixo);

- uma linha para cada aluno, contendo o seu nome, as suas notas de prova, a sua média de prova usando os pesos dados no arquivo de entrada, as suas notas dos exercícios-programa, a sua média de exercício-programa usando os pesos dados no arquivo de entrada e finalmente a sua média final;
- uma linha contendo a média da turma em cada prova, a média das médias das provas, a média de cada exercício-programa, a média das médias dos exercícios-programa e a média das médias finais;
- no final, o número de alunos aprovados (com média final maior ou igual a 5.0), o número de alunos que ficaram de recuperação (com média final maior ou igual a 3.0 e menor que 5.0) e o número de reprovados (com média final menor que 3.0).

Exemplo de saída (para o arquivo de entrada já visto):

```
Nome                            p1   p2   p3   mp  ep1  ep2  ep3  ep4  ep5  mep   mf
Cassandra Avestruz da Silva    5.0  6.6  4.2  5.3 10.0  9.0  8.0  7.0  6.0  7.4  6.7
Nicolau dos Santos Neto        3.2  2.0  1.5  2.0  4.2  3.1  2.4  1.5  0.0  1.7  1.8
Romário de Souza Faria         8.0  3.0  7.7  5.9  9.5  8.0 10.0  9.5  9.0  9.3  8.1
Medias                         5.4  3.9  4.5  4.4  7.9  6.7  6.8  6.0  5.0  6.1  5.6

Aprovados:    2
Recuperacao:  0
Reprovados:   1
```

Leitura e gravação de um arquivo

Para fazer a leitura de um arquivo de entrada e a gravação da saída em um arquivo, você precisa abrir o arquivo de entrada e o arquivo de saida. Faça isso utilizando a seguinte receita no seu programa, dentro da função `main()`:

```
/* Declaração das variáveis para leitura e gravação em arquivos */
char nome_arq_entrada[40]; /* para o nome do arquivo de entrada */
FILE *entrada, *saida;

/* Primeiros comandos do seu programa */
/* Abertura do arquivo de entrada */
printf("Digite o nome do arquivo de entrada: ");
scanf("%s", nome_arq_entrada);
if ((entrada = fopen(nome_arq_entrada, "r")) == NULL) {
    printf("Arquivo de entrada nao encontrado!\n");
    exit(1);
}
/* Abertura do arquivo de saída */
if ((saida = fopen("saida.txt", "w")) == NULL) {
    printf("Erro na abertura do arquivo de saida!\n");
    exit(1);
}
```

Tendo feito isso, a variável `entrada` se referirá ao arquivo de entrada e a variável `saida` se referirá ao arquivo de saída. No restante do seu programa utilize a função `fscanf` para ler dados do arquivo de entrada e a função `fprintf` para escrever dados no arquivo de saída. Estas funções funcionam de forma semelhante ao `scanf` e ao `printf`, exceto que elas possuem um parâmetro a mais, que especifica o arquivo do qual o `fscanf` lê os dados ou no qual o `fprintf` escreve os dados. Para informar ao `fscanf` qual é o arquivo de entrada, passe como parâmetro a variável `entrada`. Para informar ao `fprintf` qual é o arquivo de saída, passe como parâmetro a variável `saida`.

Sinta-se a vontade para escrever tanto no arquivo (com o `fprintf`) quanto na tela (com o `printf`): escrever também na tela pode lhe ajudar na fase de teste do seu programa, por exemplo na verificação de que a leitura do arquivo está funcionando corretamente, etc.

Para fazer a leitura do nome, utilize a função `fgets`, que lê uma *string* de um arquivo. Essa função recebe como parâmetros (nesta ordem) o vetor de `chars` onde será guardada a *string* lida, o comprimento desse vetor e um argumento que especifica o arquivo de onde a leitura deve ser feita. O comprimento do vetor é um a mais que o número máximo de caracteres que se deseja ler. No nosso caso, esse comprimento é 30 — lembre-se de utilizar o comando `#define MAX_NOME 30` no início do programa. A função `fgets` lê uma seqüência de caracteres do arquivo até encontrar uma marca de nova linha, ou atingir o final de arquivo, ou atingir o número máximo de caracteres que se deseja ler.

Exemplos:

```
/* Lendo do arquivo de entrada */
fscanf(entrada, "%d %d %d ", &n, &p, &m);

/* Verificando os valores que foram lidos - impressão na tela */
printf("n = %d   p = %d   m = %d\n", n, p, m);

/* Leitura de um nome do arquivo de entrada */
fgets(nome, MAX_NOME, entrada);

/* Gravando no arquivo de saida */
fprintf(saida, "%s %4.1f\n", nome, mf);
```

Observações:

1. O seu programa não precisa fazer consistência de dados, ou seja,não precisa verificar se os dados no arquivo de entrada são válidos. Mais especificamente, você não precisa verificar se os nomes de fato tem no máximo 30 caracteres, se as notas de fato estão entre 0 e 10, etc. Suponha que o arquivo satisfaz estas restrições.

2. Note que o critério acima difere do nosso critério de aprovação, que exige que o aluno tenha média de EP e média de prova pelo menos 5.0 para ser aprovado.

3. **IMPORTANTE!** Para não ter problemas na leitura dos dados, deixe sempre um espaço em branco depois do formato de leitura de cada número inteiro ou real. Isso é essencial para você não ter problemas com a leitura dos nomes. Assim, em vez de escrever

   ```
   fscanf(entrada, "%d %d %d", &n, &p, &m);
   ```

   escreva

```
fscanf(entrada, "%d %d %d ", &n, &p, &m);
```

(Note o branco depois do último `%d`!)

Observações Importantes sobre os Exercícios-Programa

1) Todos os exercícios-programa devem ter um cabeçalho como o seguinte:

```
/*****************************************************/
/* Aluno: Fulano de Tal                              */
/* Número USP: 12345678                              */
/* Exercicio-Programa 3 -- Fazendo Media             */
/* MAC115 -- 2008 -- IFUSP, turma 21 -- Prof. Reverbel  */
/* Compilador: ... (gcc ou DevC++) versão ...        */
/*****************************************************/
```

2) O exercício-programa é estritamente individual. Exercícios copiados (com ou sem eventuais disfarces) receberão nota ZERO.
3) Exercícios atrasados não serão aceitos.
4) Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota ZERO. Seu programa deve ser compilável sem erros ou *warnings*, com o compilador no modo em que todos os *warnings* possíveis são emitidos. Caso você use o `gcc`, passe ao compilador (na linha de comando) as opções "`-Wall -ansi -pedantic -O2`". Caso você use o `DevC++`, clique em "Ferramentas" (ou "Tools") e "Opções do Compilador" (ou "*Compiler Options*") e, na tela de opções do compilador, marque como selecionada a opção "Adicione os seguintes comandos quando chamar o compilador" (ou "*Add the following commands when calling compiler*"). Na caixa de texto que aparece logo depois dessa opção, digite "`-Wall -ansi -O2`". (Não use `-pedantic` com o DevC++.)
5) É muito importante que seu programa tenha comentários e esteja bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa (conforme visto em aula). A avaliação dos exercícios-programa levará isto em conta.
6) Cada programa deve ter sido executado tantas vezes quantas forem necessárias para testar os vários casos possíveis para as entradas.
7) Você entregará seu exercício-programa através do sistema Paca/Moodle (`http://paca.ime.usp.br`).
8) Entregue apenas o programa fonte em C, num arquivo com nome `ep3-`<*seu-número-USP*>`.c`. (Exemplo: se seu número USP for 12345678, você deverá entregar um arquivo `ep3-12345678.c`.)
9) Enquanto o prazo de entrega não expirar, você poderá entregar várias versões do mesmo exercício-programa. Apenas a última versão entregue será guardada pelo sistema. Encerrado o prazo, o sistema não aceitará mais a entrega de exercícios-programa. Não deixe para entregar seu exercício na última hora!
10) Guarde uma cópia do seu exercício-programa pelo menos até o final do semestre.

Bom trabalho!