Universidade Federal do Espírito Santo
Centro Tecnológico
Departamento de Informática



Prof. Vítor E. Silva Souza

# Exercícios de Revisão de C

Os exercícios a seguir são um passo a passo para construção de um programa que lê uma lista de alunos e suas notas e imprime o nome e as notas dos alunos que estiverem acima da média da turma.

Para todos os exercícios assume-se que os dados são passados de forma correta e que não é necessário verificar erros do usuário, a menos que isso seja explicitamente indicado.

1) Escreva um programa em C que receba como argumento o nome de um arquivo e imprima seu conteúdo na tela. Caso não seja passado nenhum argumento, o programa deve pedir ao usuário que digite o nome do arquivo na entrada padrão.

2) Para armazenar os dados dos alunos (nome e nota), crie uma estrutura de dados (*struct*) tAluno e as seguintes operações:

a) Dados um nome e uma nota, retorna um aluno;

b) Obtém o nome de um aluno;

c) Obtém a nota de um aluno.

A estrutura e as funções devem ser implementadas observando princípios de modularidade do código, ou seja, separadamente do arquivo de código do programa principal e divididos em arquivo cabeçalho (*header* ou .h) e código (.c).

3) Modifique o programa principal (desenvolvido no exercício 1) para que leia os dados do arquivo, monte um vetor de alunos para, em seguida, imprimir a frase "O aluno <nome> tirou a nota <nota>" para cada aluno no vetor, substituindo <nome> e <nota> pelos dados do aluno.

4) Modifique o programa principal para que calcule a média das notas dos alunos e imprima na tela apenas o nome dos alunos que possuem nota acima da média da turma.

5) Considerando que a média para passar direto é 7, faça com que o programa escreva num arquivo "saida.csv" o nome, a nota e a situação de cada aluno em formato CSV, como no exemplo abaixo:

```
Nome,Nota,Situação
Fulano,0.00,Prova Final
Ciclano,10.00,Aprovado
Beltrano,3.00,Prova Final
```

## Problemas avançados

6) Esconda a estrutura de dados do tipo tAluno no arquivo de implementação (.c) definindo o tipo tAluno como um ponteiro para esta estrutura.

7) Ordene o vetor de alunos pelo nome do aluno antes de imprimir as informações na tela e nos arquivos. Dica: use a função qsort() da biblioteca do C.

# Resolução dos Exercícios

1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Programa principal. */
int main(int argc, char **argv) {
	char *nomeArquivo;
	char linha[255];

	// Verifica se o nome do arquivo foi passado como argumento.
	if (argc > 1) {
		nomeArquivo = argv[1];
	}

	// Não foi passado o nome do arquivo por argumento. Solicita via entrada padrão.
	else {
		printf("Arquivo de entrada: ");
		nomeArquivo = (char *)malloc(255 * sizeof(char));
		scanf("%s", nomeArquivo);
	}

	// Lê o conteúdo do arquivo.
	FILE *arquivo = fopen(nomeArquivo, "r");
	while (fgets(linha, sizeof(linha), arquivo)) {
		linha[strlen(linha) - 1] = '\0';
		printf("%s\n", linha);
	}
	fclose(arquivo);
}
```

---

2)

```c
#ifndef TALUNO_H_
#define TALUNO_H_

// Define a estrutura TAluno e o tipo tAluno como equivalente a ela (para facilitar o ex. 6).
typedef struct TAluno {
	char nome[50];
	float nota;
} tAluno;

// Operação de criação de um aluno.
tAluno criarAluno(char *nome, float nota);

// Operação de obtenção do nome de um aluno.
char* obterNomeAluno(tAluno aluno);

// Operação de obtenção da nota de um aluno.
float obterNotaAluno(tAluno aluno);

#endif
```

---

```c
#include "tAluno.h"

// Operação de criação de um aluno.
tAluno criarAluno(char *nome, float nota) {
	tAluno aluno;
```

```
	strcpy(aluno.nome, nome);
	aluno.nota = nota;
	return aluno;
}

// Operação de obtenção do nome de um aluno.
char* obterNomeAluno(tAluno aluno) {
	return aluno.nome;
}

// Operação de obtenção da nota de um aluno.
float obterNotaAluno(tAluno aluno) {
	return aluno.nota;
}
```

---

3)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "tAluno.h"

/* Programa principal. */
int main(int argc, char **argv) {
	char* nomeArquivo;
	char linha[255];

	// Verifica se o nome do arquivo foi passado como argumento.
	if (argc > 1) {
		nomeArquivo = argv[1];
	}

	// Não foi passado o nome do arquivo por argumento. Solicita via entrada padrão.
	else {
		printf("Arquivo de entrada: ");
		nomeArquivo = (char *)malloc(255 * sizeof(char));
		scanf("%s", nomeArquivo);
	}

	// Lê a primeira linha do arquivo, que contém o número de alunos.
	FILE* arquivo = fopen(nomeArquivo, "r");
	int numAlunos = 0;
	fscanf(arquivo, "%d\n", &numAlunos);

	// Aloca espaço para o número de alunos indicado e lê os dados dos alunos do arquivo.
	char nome[50];
	float nota;
	int idx = 0;
	tAluno* alunos = (tAluno*)malloc(numAlunos * sizeof(struct TAluno));
	while (fgets(linha, sizeof(linha), arquivo)) {
		linha[strlen(linha) - 1] = '\0';
		sscanf(linha, "%f %s", &nota, nome);
		alunos[idx++] = criarAluno(nome, nota);
	}
	fclose(arquivo);

	// Imprime o vetor de alunos.
	for (idx = 0; idx < numAlunos; idx++) {
		printf("O aluno %s tirou a nota %f\n", obterNomeAluno(alunos[idx]), obterNotaAluno(alunos[idx]));
	}
}
```

---

4)

```
/* Após a leitura dos alunos e construção do vetor. */

// Calcula a média da turma.
float media = 0;
for (idx = 0; idx < numAlunos; idx++) media += obterNotaAluno(alunos[idx]);
media /= numAlunos;

// Imprime os alunos que estão acima da média.
for (idx = 0; idx < numAlunos; idx++) {
        nota = obterNotaAluno(alunos[idx]);
        if (nota > media) printf("%s\n", obterNomeAluno(alunos[idx]));
}
```

---

5)

```
/* Após o cálculo da média. */

// Abre o arquivo saida.csv para escrita e escreve o cabeçalho.
arquivo = fopen("saida.csv", "w");
fprintf(arquivo, "Nome,Nota,Situação\n");

// Imprime na tela os alunos que estão acima da média e no arquivo todos os alunos.
for (idx = 0; idx < numAlunos; idx++) {
        char* aluno = obterNomeAluno(alunos[idx]);
        nota = obterNotaAluno(alunos[idx]);

        if (nota > media) printf("%s\n", aluno);

        if (nota >= 7.0) fprintf(arquivo, "%s,%.2f,Aprovado\n", aluno, nota);
        else fprintf(arquivo, "%s,%.2f,Prova Final\n", aluno, nota);
}
fclose(arquivo);
```

---

6)

```
/* Em tAluno.h: */
// Define o tipo tAluno como sendo equivalente a uma estrutura que será definida em tAluno.c.
typedef struct TAluno* tAluno;


/* ... */

// Operação de destruição de um aluno.
void destruirAluno(tAluno aluno);
```

---

```
/* Em tAluno.c: */
// Define a estrutura declarada no arquivo cabeçalho.
struct TAluno {
        char nome[50];
        float nota;
};

/* ... */


// Operação de destruição de um aluno.
```

```
void destruirAluno(tAluno aluno) {
	free(aluno);
}
```

---

```
/* Em FiltrarAlunos.c, só muda a alocação do vetor: */
tAluno* alunos = (tAluno*)malloc(numAlunos * sizeof(tAluno));
```

---

7)

```
/* Em tAluno.c, declarando a função em tAluno.h também: */
// Compara dois alunos por nome, funciona como strcmp().
int comparaPorNomeAluno(const void* p1, const void* p2) {
	tAluno a1 = *(tAluno*)p1;
	tAluno a2 = *(tAluno*)p2;
	return strcmp(a1->nome, a2->nome);
}

/* Em FiltrarAlunos.c, basta adicionar após a construção do vetor: */
	qsort(alunos, numAlunos, sizeof(tAluno), comparaPorNomeAluno);
```