

Algoritmos e Programação 2

Atividade 3

Observações Importantes:

- Envie, para cada exercício, um arquivo contendo seu código fonte. O nome dos arquivos deverão seguir o seguinte padrão: <RA>_EX<número do exercícios>.c. Exemplo: 99999_EX1.c
- Todos os arquivos enviados deverão conter um cabeçalho no seguinte formato:

```
/*
Algoritmos e Programacao II
Nome: <nome do aluno>
RA: <ra do aluno>
Exercicio: Atividade 3 - <número do exercício>
Maio/2016
*/
```

- Exercícios com erros de compilação serão desconsiderado.
- Exercícios com suspeita de fraude serão desconsiderados (nota zero para todos os envolvidos).

Tarefa:

1. Crie um programa para manter um histórico escolar. O seu programa deverá oferecer as seguintes funcionalidades (menu):

1. Cadastrar
2. Alterar
3. Remover
4. Buscar
5. Exibir histórico sujo
6. Exibir histórico limpo
7. Sair

Todos os dados deverão ser salvos no arquivo-texto `historico.dat` com os registros ordenados pelo nome da disciplina. Caso o arquivo já exista, o seu programa deverá ler os dados já existentes. Mantenha os dados das disciplinas em uma estrutura `Disciplina` com os seguintes campos: nome (*string* - 100 posições), código da disciplina (*int*), quantidade de créditos (*int*), status (que pode ser: “APROVADO” ou “REPROVADO” - *string*), nota (*int*) e frequência (*int*).

Cada opção deverá executar o seguinte procedimento:

- (a) **Cadastrar:** solicita todos os dados da disciplina. Caso a disciplina já tenha sido cadastrada, informar na tela que a disciplina já está cadastrada e retornar ao menu;
- (b) **Alterar:** solicitar o código da disciplina. Caso ele seja encontrado, solicitar novamente os campos: nome, quantidade de créditos, status, nota e frequência. Caso contrário, emitir mensagem de disciplina não cadastrada e retornar ao menu.
- (c) **Remover:** solicitar o código da disciplina. Caso ela seja encontrada, remover o registro. Caso contrário, emitir mensagem de disciplina não cadastrada e retornar ao menu.
- (d) **Buscar:** solicitar o código da disciplina. Caso ela seja encontrada, exibir todos os campos do registro. Caso contrário, emitir mensagem de disciplina não cadastrada e retornar ao menu.
- (e) **Exibir histórico sujo:** imprimir na tela todos os campos de todos os registros existentes ordenados pelo nome da disciplina e o total de créditos cursados.
- (f) **Exibir histórico limpo:** imprimir na tela todos os campos das disciplinas cujo status seja “APROVADO” e o total de créditos cursados (considere apenas os créditos das disciplinas aprovadas).
- (g) **Sair:** sobrescrever o arquivo com dados atualizados, liberar memória e fechar o programa.

O seu programa deverá carregar/criar o arquivo na inicialização, manipular os dados em memória (sempre mantendo os registros ordenados pelo nome da disciplina) e sobrescrever o arquivo na finalização.

Os campos dos registros deverão ser separados pelo delimitador @ e cada registro deverá ocupar uma linha do arquivo de dados.

Trate os possíveis erros que o usuário possa cometer, tais como: informar valores não numéricos nos campos `int`, deixar campos em branco, digitar uma opção inválida no menu, etc.

2. O jogo Sudoku é um quebra-cabeça baseado na colocação lógica de números. O objetivo do jogo é a colocação de números de 1 a 9 em cada uma das células vazias numa grade 9x9, constituída por subgrades 3x3 chamadas regiões.

Este jogo espalhou-se rapidamente por todo o mundo, tornando-se hoje um passatempo muito popular. Muitas pessoas, entretanto, preenchem a grade de forma incorreta, desrespeitando as restrições do jogo.

Para ser uma solução do problema, cada linha e coluna da grade deve conter todos os números de 1 a 9. Além disso, cada uma das subgrades também deve conter os números de 1 a 9. O quebra-cabeça contém algumas pistas iniciais, que são números inseridos em algumas células, de maneira a permitir uma indução ou dedução dos números em células que estejam vazias. As figuras a seguir mostram, respectivamente, as pistas iniciais para resolvermos um exemplo de jogo e a solução final:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

(a) Grade inicial.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

(b) Grade solucionada.

Faça um programa que leia uma matriz que representa um jogo Sudoku em sua configuração inicial e imprima na tela o jogo resolvido. Caso a matriz de entrada não tenha solução, imprima a mensagem “SEM SOLUÇÃO”. Para encontrar uma solução utilize **funções recursivas**.

Exemplo de entrada e saída:

Entrada:

```
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

Saída:

```
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Observação: Imprima APENAS o que está sendo pedido. Ou seja, a grade solucionada ou a mensagem “SEM SOLUÇÃO”.