Algoritmos e Estruturas de Dados I (DCC/003) – 2018/1

Aula Prática 8 – 16/05/2018

Preparem os exercícios de forma que:

1. Os aquivos utilizem a extensão "**.c**"
2. Não seja utilizada função **system**("pause")
3. A função **printf** deve ser utilizada apenas para imprimir a saída do programa.

Atenção: Para esta lista de exercícios é necessário utilizar funções conforme pedem as atividades, caso o contrário, a nota será **0**.

1) Faça um programa que simule cadastros em uma agenda de eventos. Para isto deverá ser criadas duas estruturas (structs) para representar uma data e um evento. Estas estruturas devem armazenar valores dos tipos mostrados na tabela abaixo:

Nome da estrutura	**Campos da estrutura**
Data	• dia – valor inteiro • mes – valor de um tipo ***enum*** • ano – valor inteiro
Evento	• nome – string de até 100 caracteres • local – string de até 100 caracteres • data – valor do tipo **Data**

O programa deve funcionar da seguinte forma:

1. Solicitar a leitura de ***3*** registros do tipo **Evento**, estes registros devem ser armazenados em um vetor de **Evento**. Esta leitura deve ser feita em uma função chamada:

```
void cadastrar_eventos(Evento agenda[], int n);
```

2. Após a leitura dos ***3*** registros, o programa deve realizar a leitura de um registro do tipo **Data** e imprimir na tela todos os eventos daquela data na ordem que foram cadastrados. Essa impressão deve ser feita usando uma função chamada:

```
void imprimir_eventos(Evento agenda[], Data d, int n);
```

Para a leitura de *strings* deve ser utilizada a função **scanf**, outras funções não permitem a correta avaliação do Prático.

Exemplo:

Entrada:	evento1 local1 2 6 2014 evento2 local2 13 7 2014 evento3 local3 13 7 2014 13 7 2014
Saída:	evento2 local2 evento3 local3

- **Entrada**:
  - **3** registros, onde cada registro possui duas strings e três valores inteiros (dia, mês e ano),
  - três números inteiros (dia, mês e ano).
- **Saída**: **uma** das seguintes mensagens:
  - "Nenhum evento encontrado!\n"
  - "`%s %s\n`", sendo a primeira *string* o nome do evento e a segunda o nome do local do evento, para cada evento encontrado.

2) Faça um programa que leia as informações de um Aluno no semestre. O Aluno deve ser criado como uma struct. Cada aluno possuí no registro o primeiro nome, o número de matricula e notas de 4 matérias. Como será necessário calcular a média de cada um dos alunos, a struct também deverá ter um valor média.
O programa deverá funcionar da seguinte maneira:

1. Para cada um dos 5 alunos, solicitar as informações do registro de cada um deles, na seguinte ordem: Primeiro nome, matrícula, nota das 4 matérias;
2. Após isso, calcular a média de cada um dos alunos, salvando na respectiva struct;
3. Imprimir as informações do aluno com maior média e do com menor média, usando as seguintes mensagens:
   "Maior media\nAluno: %s - %d\nMedia: %.2f\n"
   "Menor media\nAluno: %s - %d\nMedia: %.2f\n"
4. Após imprimir a maior média e a menor, imprimir a média geral da turma com a seguinte mensagem:
   "Media geral: %.2f"

Exemplo

• **Entrada:**	• Silvio 123456 100 100 100 100 • Xuxa 654321 40 60 80 100 • Faustao 528496 80 80 80 80 • Gugu 332548 60 80 100 80 • Eliana 332211 100 100 100 99
• **Saída:**	• Maior media Aluno: Silvio – 123456

	Media: 100.00 • Menor media Aluno: Xuxa – 654321 Media: 70.00 • Media geral: 85.95

3) Em Aprendizado de Máquina uma forma simples, e ainda efetiva, de resolver alguns problemas de classificação é encontrar o registro de treinamento mais parecido com o registro que se deseja classificar. Para medir o quão parecido um registro com valores numéricos é de outro, podemos utilizar a distância euclidiana dada pela fórmula abaixo:

$$\sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

onde p_i e p_i representam elementos dos vetores $P(p_1, p_2, \ldots p_n)$ e $Q(q_1, q_2, \ldots q_n)$ respectivamente.

Neste exercício você deverá fazer um programa para identificar o tipo de uma flor do gênero íris, esse é um problema clássico utilizado para avaliar métodos de classificação (http://archive.ics.uci.edu/ml/datasets/Iris). Para isto o programa deve:

1. Implementar uma estrutura chamada **iris** que contém os seguintes campos:
   - *comprimento da sépala* – valor numérico real
   - *largura da sépala* – valor numérico real
   - *comprimento da pétala* – valor numérico real
   - *largura da pétala* – valor numérico real
   - *tipo* – string de até 50 caracteres
2. Realizar a leitura de **6** registros do tipo **iris**, para isto deve ler primeiro os 4 valores reais e depois uma string.
3. Realizar a leitura de 4 valores reais, representando os dados de uma flor que não foi identificada.
4. Encontrar o registro que possui a menor distância euclidiana em relação aos valores lidos e imprimir o tipo da flor.

A distância euclidiana deve ser calculada considerando os 4 valores reais dos registros. O programa deve implementar e usar a função definida pelo protótipo abaixo:

```
void classificar(iris *nao_identificada, iris registros_identificados[], int n);
```

Esta função recebe como parâmetro um ponteiro para o registro **iris** que contém os valores lidos da flor não identificada e um vetor de **iris** que contem os ***6*** registros lidos no início do programa.

Exemplo:

Entrada:	5.1 3.5 1.4 0.2 Iris-setosa 5.1 2.2 3.8 0.9 Iris-versicolor 6.5 3.2 5.1 2.0 Iris-virginica 6.3 2.9 5.6 1.8 Iris-virginica 5.6 2.9 3.6 1.3 Iris-versicolor 5.3 3.7 1.5 0.2 Iris-setosa 5.0 2.0 3.5 1.0
Saída:	Tipo de flor: Iris-versicolor

- **Entrada**:
  - ***6*** registros do tipo ***iris***, cada registro formado por 4 valores reais e uma string,
  - 4 valores reais.
- **Saída**: a seguinte mensagem:
  - “Tipo de flor: %s\n”.