Laboratório 9

9.1. Eventos

Faça um programa que simule cadastros em uma agenda de eventos. Para isto, duas estruturas (structs) deverão ser criadas para representar o tipo Data e o tipo Evento. Estas estruturas devem armazenar valores dos tipos mostrados na tabela abaixo:

Nome da estrutura	**Campos da estrutura**
Data	• dia – inteiro • mês – valor de um tipo ***enum*** • ano – 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. Inicialmente deve solicitar a leitura de um número inteiro ***n***
2. Em seguida deve solicitar a leitura de ***n*** registros do tipo **Evento**, estes registros devem ser armazenados em um vetor do tipo **Evento**. Esta leitura deve ser feita em uma função chamada:

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

3. Após a leitura dos ***n*** registros, o programa deve realizar a leitura de um registro do tipo **Data** e imprimir na tela todos os eventos daquela data na ordem em que foram cadastrados. Caso não haja eventos cadastrados naquela data, imprimir a mensagem "**Nenhum evento encontrado!**". Essa impressão deve ser feita usando uma função chamada:

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

Exemplo:

Entrada:	3 evento1 local1 2 6 2014 evento2 local2 13 7 2014 evento3 local3 13 7 2014 13 7 2014
Saída:	evento2 local2 evento3 local3

9.2. Vetor Multiplicado

Faça um programa que leia uma vetor que o usuário digitar, calcule a multiplicação deste vetor por um valor, e depois multiplique novamente o vetor de forma que ele volte a ter os valores originais. Deve-se definir as funções:

```
void imprime(int vetor[], int n);
```

```
void multiplica(int vetor[], int n, float v);
```

O programa deve funcionar da seguinte forma:

1. Perguntar ao usuário o tamanho do vetor;
2. Solicitar os valores do vetor;
3. Solicitar o valor a multiplicar o vetor;
4. Imprimir os valores do vetor utilizando uma função void imprime (int vetor[], int n);
5. Multiplicar todos os elementos do vetor pelo valor informado pelo usuário, usando a função void multiplica (int vetor[], int n, float v);
6. Imprimir novamente os valores do vetor utilizando uma função void imprime (int vetor[], int n);
7. Multiplicar todos os elementos do vetor por 1/v utilizando a chamada multiplica (vetor, n, 1/v);
8. Imprimir novamente os valores do vetor utilizando uma função void imprime (int vetor[], int n);

Exemplo:

Entrada:	5 87 4 36 42 1 4
Saída:	87 4 36 42 1 348 16 144 168 4 87 4 36 42 1

9.3. Iris

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 q_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. Este é um problema clássico utilizado para avaliar métodos de classificação (http://archive.ics.uci.edu/ml/datasets/Iris). O seu programa deve:

1. Implementar uma estrutura (struct) 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 um número inteiro ***n;***
3. Realizar a leitura de **n** registros do tipo **Iris;**
4. Realizar a leitura de um registro **Iris** adicional representando os dados de uma flor que não foi identificada;

5. Encontrar o registro que possui a menor distância euclidiana em relação aos valores lidos;
6. Imprimir o tipo da flor deste registro com a menor distância euclidiana.

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 o registro **Iris** da flor não identificada e um vetor de **Iris** que contém os ***n*** registros lidos no início do programa.

Exemplo:

Entrada:	3 5.1 3.5 1.4 0.2 Iris-setosa 5.9 3.2 4.8 1.8 Iris-versicolor 6.5 3.2 5.1 2.0 Iris-virginica 5.0 2.0 3.5 1.0
Saída:	Iris-versicolor