Programação Imperativa – Exercícios sobre Estruturas de Dados
Prof. Alcides Calsavara – PUCPR

(C.1) Considere o seguinte programa:

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

#define NUM_MESES 12
#define NUM_PINTORES 20
#define MAX_NOME 50

char* mes_str[NUM_MESES] = { "janeiro", "fevereiro", "marco",
                             "abril", "maio", "junho",
                            "julho", "agosto", "setembro",
                            "outubro", "novembro", "dezembro" };

typedef
    enum {JAN, FEV, MAR, ABR, MAI, JUN, JUL, AGO, SET, OUT, NOV, DEZ}
    Mes;

typedef
    struct
    {
        char nome[MAX_NOME]; // nome da pessoa
        Mes mes; // mês de aniversário da pessoa
    }
    Pessoa;

int main()
{
    Pessoa pintor[NUM_PINTORES];

    pintor[ 0].mes = JAN; strcpy(pintor[ 0].nome, "Leonardo da Vinci");
    pintor[ 1].mes = ABR; strcpy(pintor[ 1].nome, "Sandro Botticelli");
    pintor[ 2].mes = AGO; strcpy(pintor[ 2].nome, "Georges Seurat");
    pintor[ 3].mes = ABR; strcpy(pintor[ 3].nome, "Vincent van Gogh");
    pintor[ 4].mes = SET; strcpy(pintor[ 4].nome, "Paul Gauguin");
    pintor[ 5].mes = JUN; strcpy(pintor[ 5].nome, "Edouard Manet");
    pintor[ 6].mes = OUT; strcpy(pintor[ 6].nome, "Paul Cezanne");
    pintor[ 7].mes = JUN; strcpy(pintor[ 7].nome, "Auguste Renoir");
    pintor[ 8].mes = JUN; strcpy(pintor[ 8].nome, "Claude Monet");
    pintor[ 9].mes = AGO; strcpy(pintor[ 9].nome, "Pablo Picasso");
    pintor[10].mes = DEZ; strcpy(pintor[10].nome, "Edgar Degas");
    pintor[11].mes = ABR; strcpy(pintor[11].nome, "Edvard Munch");
    pintor[12].mes = JAN; strcpy(pintor[12].nome, "Michelangelo Merisi
(Caravaggio)");
```

```
    pintor[13].mes = JAN; strcpy(pintor[13].nome, "Michelangelo
Buonarroti");
    pintor[14].mes = SET; strcpy(pintor[14].nome, "Tarsila do Amaral");
    pintor[15].mes = AGO; strcpy(pintor[15].nome, "Gustav Klimt");
    pintor[16].mes = NOV; strcpy(pintor[16].nome, "Rembrandt van
Rijn");
    pintor[17].mes = MAI; strcpy(pintor[17].nome, "Amedeo Modigliani");
    pintor[18].mes = SET; strcpy(pintor[18].nome, "Caillebotte");
    pintor[19].mes = SET; strcpy(pintor[19].nome, "Joseph Turner");

    return 0;
}
```

Estenda esse programa de modo que permita o usuário pesquisar todos os pintores nascidos em certo mês. O programa deverá apresentar uma interface de usuário em conformidade com os exemplos de interação a seguir.

Exemplo 1:

```
Meses para a pesquisa:
 ( 1) janeiro
 ( 2) fevereiro
 ( 3) março
 ( 4) abril
 ( 5) maio
 ( 6) junho
 ( 7) julho
 ( 8) agosto
 ( 9) setembro
 (10) outubro
 (11) novembro
 (12) dezembro

 Escolha um mês pelo seu número: 6

 Pintores nascidos no mês de junho:
 (1) Edouard Manet
 (2) Auguste Renoir
 (3) Claude Monet

 ==> 3 pintores foram encontrados.
```

Exemplo 2:

```
Meses para a pesquisa:
( 1) janeiro
( 2) fevereiro
( 3) março
( 4) abril
( 5) maio
( 6) junho
( 7) julho
( 8) agosto
( 9) setembro
(10) outubro
(11) novembro
(12) dezembro

Escolha um mês pelo seu número: 5

Pintores nascidos no mês de maio:
(1) Amedeo Modigliani

==> Apenas um pintor foi encontrado.
```

Exemplo 3:

```
Meses para a pesquisa:
( 1) janeiro
( 2) fevereiro
( 3) março
( 4) abril
( 5) maio
( 6) junho
( 7) julho
( 8) agosto
( 9) setembro
(10) outubro
(11) novembro
(12) dezembro

Escolha um mês pelo seu número: 2

Pintores nascidos no mês de fevereiro:

==> Nenhum pintor foi encontrado.
```

Exemplo 4:

```
Meses para a pesquisa:
( 1) janeiro
( 2) fevereiro
( 3) março
( 4) abril
( 5) maio
( 6) junho
( 7) julho
( 8) agosto
( 9) setembro
(10) outubro
(11) novembro
(12) dezembro

Escolha um mês pelo seu número: 15

Escolha inválida!
```

(C.2) Escreva um programa na linguagem C que permita realizar operações sobre o catálogo de produtos de uma loja. No catálogo, constam os seguintes campos para cada produto:

1. código: um valor inteiro positivo automaticamente atribuído pelo programa (a cada novo produto inserido no catálogo, o código deve ser incrementado)

2. descrição: uma string de, no máximo, 30 caracteres
3. estoque: um valor inteiro correspondente à quantidade de itens do produto
4. categoria: um dos seguintes valores:
    a. ELETRODOMÉSTICO
    b. FERRAMENTA
    c. VESTUÁRIO

O programa deve fornecer uma interface para o usuário escolher continuamente uma das seguintes operações:

1. listar todo o catálogo
2. inserir um novo produto no catálogo, dada a sua descrição e a sua categoria, com estoque inicial igual a zero
3. remover um produto do catálogo, dado o seu código
4. atualizar o estoque de um produto, dado o seu código e uma quantidade
5. aumentar o estoque de um produto, dado o seu código e uma quantidade
6. diminiuir o estoque de um produto, dado o seu código e uma quantidade
7. sair do programa

Requisitos de implementação:

1. Deve ser definido um **enum** para representar as categorias de produtos.
2. Deve ser definido um **struct** para representar os campos de um produto.
3. Deve ser definido um vetor (*array*) para representar o catálogo de produtos.
4. Deve ser definida uma constante para estabelecer o número máximo de produtos no catálogo.
5. Deve ser definida uma constante para estabelecer o tamanho máximo (em caracteres) da descrição de um produto.
6. Deve ser definida uma função para cada tipo de operação aplicável ao catálogo, incluindo:
    a. **listar_catalogo**
    b. **inserir_produto**: recebe a descrição e a categoria do produto
    c. **remover_produto**: recebe o código do produto
    d. **atualizar_estoque**: recebe o código do produto e uma quantidade
    e. **aumentar_estoque**: recebe o código do produto e uma quantidade
    f. **diminuir_estoque**: recebe o código do produto e uma quantidade
7. Todas as funções acima que correspondem a operações sobre o catálogo, exceto a operação **listar_catalogo**, devem retornar um valor do tipo **bool** para indicar se a operação foi realizada com sucesso ou se houve alguma falha.
8. A operação listar_catalogo deve exibir todos os campos de todos os produtos do catálogo em formato adequado para o usuário, preferencialmente, em forma de colunas (uma coluna para cada campo).
9. A inserção de um novo produto deve ser feita sempre no final do catálogo.
10. A remoção de um produto do catálogo deve fazer com que todos os produtos que estejam na sequência desse produto sejam adiantados em uma posição do catálogo.

11. A função main deve implementar a interação com o usuário (exibição das operações sobre o catálogo, escolha da opção do usuário e ativação da função correspondente) por meio do comando **switch**.
12. Deve ser definida uma função para tratar cada uma das opções disponíveis para o usuário, exceto para a opção de listar todo o catálogo, com a seguinte nomenclatura:
    a. IHC_inserir_produto
    b. IHC_remover_produto
    c. IHC_atualizar_estoque
    d. IHC_aumentar_estoque
    e. IHC_diminuir_estoque

Todas as funções de interação humano-computador (IHC) devem dar mensangens claras para o usuário a respeito do sucesso da operação ou de uma eventual falha.