ESTRUTURA DE DADOS I
EXERCÍCIOS: **PONTEIROS**
Profª Lucília Ribeiro

01 Crie um programa contendo o seguinte trecho de código e responda:

```
int a = 25;
int *pa = &a;
int b = *pa + a;
printf("\n%d - %d - %d - %d - %d - %d\n", a, pa, &a, *pa, b, &b);
```

a) Qual o resultado da execução do programa?
b) Qual o significado de cada um dos valores escritos na tela?

02 Crie um programa para calcular a área e o perímetro de um hexágono. O seu programa obrigatoriamente deverá possuir um módulo chamado Geometria (Geometria.h e Geometria.c). Neste módulo, implemente uma função chamada calculaHexagono que calcule a área e o perímetro de um hexágono regular de lado l. A função deve obedecer o seguinte protótipo:

```
void calculaHexagono(float l, float *area, float *perimetro);
```

Lembrando que a área e o perímetro de um hexágono regular são dados por:

$$A = \frac{3l^2\sqrt{3}}{2} \qquad P = 6l$$

Para os cálculos, obrigatoriamente você deve utilizar as funções `sqrt` e `pow` da biblioteca `math.h`.

Em seguida crie um programa que utilize o módulo Geometria e a função calculaHexagono para calcular a área e o perímetro de um hexágono de lado l informado pelo usuário.

03 Escreva uma função que determina a média e a situação de um aluno em uma disciplina. A função recebe como parâmetros as três notas de um aluno (p1, p2 e p3), seu número de faltas (faltas), o número total de aulas da disciplina (aulas) e o ponteiro para uma variável (media), conforme o seguinte protótipo:

```
char situacao(float p1, float p2, float p3, int faltas, int aulas,
float *media);
```

Na variável indicada pelo ponteiro media, a função deve armazenar a média do aluno, calculada como a média aritmética das três provas. Além disso, a função deve retornar um caractere indicando a situação do aluno no curso, definido de acordo com o seguinte critério:

Número de Faltas	Média	Situação	Retorno
Menor ou Igual a 25% do número total de aulas	>= 6.0	Aprovado	A
	< 6.0	Reprovado	R
Maior que 25% do número total de aulas	Qualquer	Reprovado por faltas	F

Em seguida, escreva um programa que utiliza a função anterior para determinar a situação de um aluno. O programa deve:

a) Ler do teclado três números reais e dois números inteiros, representando as notas da p1, p2 e p3, o número de faltas e o número de aulas, respectivamente;

b) Chamar a função desenvolvida na primeira questão para determinar a média e a situação do aluno na disciplina;

c) Exibir a média (com apenas uma casa decimal) e a situação do aluno, isto é, "APROVADO", "REPROVADO" ou "REPROVADO POR FALTAS", dependendo do caractere retornado pela função, conforme a tabela acima.

04 Crie um programa para manipular vetores. O seu programa obrigatoriamente deverá possuir um módulo chamado Vetor (Vetor.h e Vetor.c). Neste módulo, implemente uma função chamada `inverteVetor`, que recebe como parâmetro dois vetores V1 e V2, ambos de tamanho N. A função deve copiar os elementos de V1 para V2 na ordem inversa. Ou seja, se a função receber V1 = {1,2,3,4,5}, a função deve copiar os elementos para V2 na seguinte ordem: V2 = {5,4,3,2,1}. Além disso, a função também deve retornar o maior valor encontrado em V1.
A função deve obedecer ao seguinte protótipo:

```
int inverteVetor(int *v1, int *v2, int n);
```

Em seguida, implemente no mesmo módulo outra função chamada multiplicaEscalar, que recebe como parâmetro dois vetores V1 e V2 (ambos de tamanho N), e um número inteiro X. A função deve multiplicar cada um dos elementos de V1 por X e armazenar os resultados em V2. A função deve obedecer ao seguinte protótipo:

```
void multiplicaEscalar(int *v1, int *v2, int x, int n);
```

No mesmo módulo, implemente também uma função chamada obtemMaiores, que recebe como parâmetro um vetor de inteiros V ordenado crescentemente (de tamanho N), e um número inteiro X. A função deve retornar um ponteiro, ou seja, o endereço, do primeiro elemento do vetor maior do que X. A função deve obedecer ao seguinte protótipo:

```
int* obtemMaiores(int *v, int x, int n);
```

Em seguida, crie um programa que utilize o módulo Vetor e as funções inverteVetor e multiplicaEscalar para inverter um vetor de tamanho 10 fornecido pelo usuário e em seguida multiplicar esse vetor por um escalar também fornecido pelo usuário. O programa deverá exibir o vetor resultante da multiplicação pelo escalar.

Por último, o programa deve pedir para o usuário fornecer um novo vetor (ordenado crescentemente) e um valor X a ser buscado. Utilizando a função obtemMaiores, o programa deve exibir na tela todos os valores maiores que X existentes no novo vetor.

05 Crie um programa para manipular vetores. O seu programa obrigatoriamente deverá possuir um módulo chamado Vetor (Vetor.h e Vetor.c). Neste módulo, implemente uma função que receba um vetor de inteiros V e retorne um outro vetor de inteiros alocado dinamicamente com todos os valores de V que estejam entre o valor mínimo e máximo (que também são passados como parâmetro para a função). A função deve obedecer ao seguinte protótipo

```
int* valoresEntre(int *v, int n, int min, int max, int *qtd);
```

A função recebe:

v: vetor de números inteiros;
n: a quantidade de elementos do vetor v;
min: valor mínimo a ser buscado;
max: valor máximo a ser buscado;

A função deve:

Verificar a quantidade de elementos do vetor que sejam maiores do que min e menores que max;

Caso a quantidade seja maior do que 0 (zero), alocar dinamicamente uma área do exato tamanho necessário para armazenar os valores;

Copiar os elementos do vetor que sejam maiores do que min e menores que max para a área alocada dinamicamente.

A função retorna:

O endereço da área alocada dinamicamente, preenchida com os números maiores do que min e menores que max, ou NULL, caso essa relação de números não tenha sido criada;

A quantidade de números carregados na área alocada dinamicamente, através do parâmetro qtd.

Em seguida, crie a função principal do programa para inicializar um vetor de inteiros, exibir esses valores na tela e pedir para o usuário digitar o valor mínimo e máximo a ser buscado. Em seguida o programa deverá chamar a função valoresEntre e exibir na tela os valores resultantes. Lembre-se de exibir uma mensagem de erro caso nenhum valor seja encontrado. Não se esqueça de liberar a memória alocada dinamicamente.

06 Adicione ao módulo Vetor criado no exercício anterior uma função que receba como parâmetros dois vetores de inteiros, v1 e v2, e as suas respectivas quantidades de elementos, n1 e n2. A função deverá retornar um ponteiro para um terceiro vetor, v3, com capacidade para (n1 + n2) elementos, alocado dinamicamente, contendo a união de v1 e v2.

Por exemplo, se
v1 = {11, 13, 45, 7}
v2 = {24, 4, 16, 81, 10, 12}
v3 irá conter {11, 13, 45, 7, 24, 4, 16, 81, 10, 12}.

O cabeçalho dessa função deverá ser o seguinte:

```
int* uniao(int *v1, int n1, int *v2, int n2);
```

Em seguida, modifique a função principal do programa criado anteriormente para que ele chame a função uniao e exiba na tela os elementos do vetor retornado. Não se esqueça de liberar a memória alocada dinamicamente.

07 Escreva um programa que contenha a função criaVetorDeProdutosZerados, que:

Recebe:

- Um vetor de inteiros com os códigos dos produtos vendidos em uma loja,
- Um vetor de inteiros com a quantidade em estoque de cada produto

(onde estoque[i] corresponde a codigo[i]) e o número de produtos.

- A função também recebe o endereço de uma variável onde deve ser devolvido o número de produtos com estoque zerado;

Retorna:

- Um novo vetor (ou seja, o endereço do primeiro elemento de um novo vetor) de inteiros, criado com o tamanho exato necessário, apenas com os códigos dos produtos com estoque igual a 0.
- A função deve também devolver, numa variável cujo endereço foi recebido como parâmetro, o número de produtos com estoque zerado.

Em seguida, implemente a função principal do programa para permitir que o usuário informe um conjunto de produtos (com código e estoque).
O programa deve utilizar a função criaVetorDeProdutosZerados para exibir na tela somente os produtos com estoque zerado.

08 Crie um programa que implemente o jogo “Bin-go”. Nesse jogo, o jogador deve selecionar a quantidade de números que ele gostaria de apostar (entre 1 e 20), e em seguida, informar os números escolhidos (valores entre 0 e 100). Após receber a aposta, o computador sorteia 20 números (entre 0 e 100) e compara os números sorteados com os números apostados, informando

ao apostador a quantidade de acertos e os números que ele acertou. O seu programa deverá implementar as funções `lerAposta`, `sorteiaValores` e `comparaAposta`.

A função `lerAposta` deve receber como parâmetro a quantidade de números que serão apostados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números apostados.
A função deve pedir para o usuário digitar os números apostados e armazena-los no vetor, garantindo que somente números dentro do intervalo de 0 a 100 sejam digitados. A função deve seguir o seguinte protótipo:

```
void lerAposta(int *aposta, int n);
```

A função `sorteiaValores` deve receber como parâmetro a quantidade de números que serão sorteados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números sorteados.
A função deve sortear aleatoriamente os números (entre 0 e 100) e armazena-los no vetor. A função deve seguir o seguinte protótipo:

```
void sorteiaValores(int *sorteio, int n);
```

A função `comparaAposta` deve receber como parâmetro o vetor com os números apostados (aposta), o vetor com os números sorteados (sorteio), juntamente com os seus respectivos tamanhos (na e ns) e um ponteiro para uma variável inteira (qtdAcertos), onde deve ser armazenada a quantidade de acertos.
A função deve retornar o ponteiro para um vetor alocado dinamicamente contendo os números que o apostador acertou. A função deve seguir o seguinte protótipo:

```
int* comparaAposta(int *aposta, int *sorteio, int *qtdacertos, int
na, int ns);
```

Em seguida, crie a função principal do programa utilizado as funções criadas anteriormente para implementar o jogo "Bin-go". Lembre-se que os vetores aposta, sorteio e acertos devem ser alocados dinamicamente e a memória alocada deve liberada quando ela não for mais ser utilizada.

Para sortear números aleatórios utilize a função `rand` da biblioteca `stdlib.h`. A função `rand` retorna um número aleatório em um determinado intervalo. Exemplo:

```
x = rand() % 10;     /* x vai receber um valor entre 0 e 10 */
```

Para garantir que novos números aleatórios sejam sorteados em cada execução do programa é necessário executar a função `srand` antes de sortear os números. Exemplo:

```
srand(time(NULL));
```

Para poder utilizar essas funções é necessário incluir no programa as bibliotecas `stdlib.h` e `time.h`. Exemplo de programa para sortear um número aleatório:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void) {
    int x;
    srand(time(NULL));
    x = rand() % 10;  /* x vai receber um valor entre 0 e 10 */
    printf("%d", x);
    return 0;
}
```