Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Computação Básica

Atividades a serem desenvolvidas nas sessões de Laboratório

Sessão 7:

Objetivos:
- Aprender a elaborar programas que utilizam **módulos-funções com parâmetros**.
- Exercitar a elaboração de programas que implementam algoritmos de **busca** e **ordenação** em vetores.

1. Criar o programa abaixo:

Exemplo 1:

```
#include <stdio.h>

float fat(int x){
int i;
float p;
        p=1;
        for (i=1;i<=x;i++){
                p=p*i;
        }
        return p;

}
int main(void){
int n,k;
float c;

        printf("Forneça o valor de n: ");
        scanf("%d",&n);
        printf("\n");

        printf("Forneça o valor de k: ");
        scanf("%d",&k);
        printf("\n");

        c=fat(n)/(fat(k)*fat(n-k));
        printf("%f\n",c);
        return 0;
}
```

a) Corrija todos os erros sintáticos;
b) Compile, execute e verifique o resultado do programa;

2. Procure prever o comportamento do programa abaixo e depois teste o programa e indique qual é o valor das variáveis A, B, C, e Z que é escrito na tela – dentro da função e no programa principal.

Variável	A	B	C	Z
Antes da funcao:				
Dentro da funcao (inicio):				
Dentro do funcao (fim):				
Depois da funcao:				

```
#include <stdio.h>

void trocaValor (int *u, int *v, int *x, int *y ) {
        *u = 1;
        *v = 2;
        *x = 3;
        *y = 4;
}

int main() {
        int a, b,c,z;
        a = 1;
        b = 2;
        c = 3;
        z = 100;
        printf ("a = %d, b = %d, c = %d, z= %d \n", a,b,c,z);
        printf ("apos a funcao\n");
        trocaValor(&a, &b, &c, &z);
        printf ("a = %d, b = %d, c = %d, z= %d \n", a,b,c,z);
        return(0);
}
```

Corrija os erros de compilação!

3. Faça uma função que ordene 10 valores numéricos armazenados em um vetor utilizando um dos algoritmos de ordenação. Depois fazer um programa que lê 5 vetores contendo 10 números cada, e para cada um mostre o mesmo ordenado (Utilizar a função de

ordenação bolha). Observação: devem ser exibidos na tela os elementos do vetor antes e após a ordenação.

4. Encontrar informações em um vetor não-ordenado requer uma **pesquisa seqüencial**, começando no primeiro elemento e parando quando o elemento procurado ou o final do vetor for encontrado. Este método, usado em dados não-ordenados, também pode ser aplicado a dados ordenados. No entanto, se os dados já estiverem ordenados, pode-se usar uma **pesquisa binária**, que ajuda a localizar o valor mais rapidamente. A pesquisa binária utiliza uma técnica computacional denominada "dividir e conquistar". Neste caso, o algoritmo inicialmente deve verificar o valor do elemento que está no "meio" do vetor. Se este elemento for maior que o valor procurado, deve ser testado o elemento central da primeira metade do vetor; caso contrário, deve-se testar o elemento central da segunda metade. Esse procedimento é repetido até que o elemento seja encontrado ou que não haja mais elementos a testar.

   Por exemplo, para encontrar o elemento 4 no vetor abaixo, utilizando pesquisa binária,

   1 2 3 4 5 6 7 8 9

   primeiro o algoritmo testa o elemento do meio, neste caso 5. Como ele é maior que 4, a pesquisa continua com a primeira metade,

   1 2 3 4 5

   O elemento central agora é 3, que é menor que 4, então, a primeira metade pode ser descartada, e a pesquisa continua com a segunda metade do vetor,

4 5

   Neste ponto, o elemento é encontrado.

   Com base nessas informações:

a) Faça uma função que ordene um vetor (utilizando o algoritmo de ordenação de sua preferência) e imprima o mesmo. Observação: devem ser exibidos na tela os elementos do vetor antes e após a ordenação.

b) Faça uma função que implemente uma pesquisa binária de um valor numérico dentro de um vetor de 10 posições (previamente ordenado). A função deve retornar se o elemento foi encontrado ou não.

c) Faça um programa que leia 3 vetores diferentes contendo 10 números, e para cada um leia também um número a ser pesquisado dentro do vetor. Utilizar as funções desenvolvidas nos itens (a) e (b) para ordenar os vetores, e fazer uma pesquisa binária do valor informado. Por fim, o programa deve informar, para cada vetor, se o número pesquisado existe ou não dentro daquele vetor.

5. a) Escreva uma função de nome SomaPares que recebe, como parâmetro, um número inteiro e retorna a soma de todos os números pares menores ou iguais ao N. (Obs: não existem números pares negativos).
   b) Escreva um programa principal que leia um número N inteiro e, se ele for positivo, imprima a soma de todos os números pares menores ou iguais a N, chamando, para isso, a função do item (a); caso contrário, imprimir a mensagem 'Número não é positivo'.

6. Escrever uma função que receba como parâmetros:
   - um vetor M de 50 elementos do tipo inteiro;
   - o número n de elementos de M;
   - um valor x;
   e devolva o valor 1 (um) se x está dentro do vetor M ou 0 (zero), caso contrário.
   Escrever um programa que leia um conjunto de 50 números inteiros, e armazene no vetor V. Em seguida, o algoritmo deve ler um conjunto de 10 números inteiros e imprimir quais destes números pertencem a V utilizando a função desenvolvida.

7. Escrever uma função que receba como parâmetros dois números inteiros positivos e determine o seu produto, utilizando o seguinte método de multiplicação:
   - dividir sucessivamente o primeiro número por 2 (divisão inteira), até obter 1 como quociente;
   - paralelamente, dobrar o segundo número;
   - somar os números da segunda coluna, que tenham um número ímpar na primeira coluna. O total obtido desta soma é o produto procurado.
   Exemplo: multiplicar 9 por 6:

9	6	→	6
4	12		
2	24		+
1	48	→	48
			54

   Escreva um programa que leia 10 pares de números, e imprima os números lidos e os respectivos produtos, utilizando a função acima.