

**UNEB - Universidade do Estado da Bahia**  
**Departamento de Ciências Exatas e da Terra**  
**Linguagem de Programação II**  
**Professor Marco Antônio Chaves Câmara**

**LISTA DE EXERCÍCIOS I**

O objetivo destes exercícios é verificar o aprendizado dos conceitos básicos apresentados nas primeiras aulas, além de oferecer atividades a serem desenvolvidas durante as próximas aulas :

1. Dados os trechos de código fonte em C abaixo, responda as perguntas associadas :

a) A função abaixo possui um erro grave na declaração de variáveis. Qual é ?

```
void faznada(void)
{
    int zero
    zero = 0;
    int um
    um = 1;
}
```

---

---

---

b) A função abaixo, descrita no livro-texto, identifica a presença de um caracter dentro de uma *string* (vetor de caracteres). Analise a função e responda às perguntas da próxima página :

```
is_in( char *s, char c)
{
    while( *s )
        if( *s == c ) return 1;
        else s++;
    return 0;
}
```

b.1) Como veremos diversas vezes durante o estudo, neste exemplo a string é passada para a função através de um ponteiro (**s**). Com base nisto, qual a garantia de interrupção do *loop* ao final da *string* ? Por que ?

\* Lembre-se que, em C, não há checagem de tamanho de vetor.

---

---

---

b.2) No laço *while*, qual a função do sinal de ponto-e-vírgula ao final do incremento de **s** ?

---

---

b.3) Considerando o valor lógico de retorno para **is\_in()**, a função abaixo pode substituí-la ? Se não puder, explique o motivo.

```
is_in( char *s, char c )
{
    while( *s && ( *s != c ) ) s++;
    return (*s);
}
```

---

---

---

b.4) Ao decidir não declarar o tipo da função, podemos considerar que ela retornará valores de que tipo ? Este tipo é compatível com o valor lógico de retorno pretendido para a função? Como são interpretados estes valores ?

---

---

---

2. Utilizando os conhecimentos adquiridos durante as aulas e no livro-texto indicado, desenvolver e apresentar código fonte dos seguintes programas (enviar o fonte via e-mail para [mcamara@logicsoft.com.br](mailto:mcamara@logicsoft.com.br)).

## BÁSICO

2.a) Um número primo só é divisível por ele mesmo e por 1. Os primeiros números primos conhecidos são 2,3,5,7 e 11. Desenvolva um programa que encontra os 100 próximos números primos a partir de um valor específico.

2.b) Desenvolva uma função que transforme uma *string* de caracteres, forçando que todas as letras iniciais de cada palavra seja maiúscula, enquanto que as outras sejam minúsculas. Após isto, desenvolva um programa que leia uma *string* e mostre a mesma *string* após a transformação.

Dica : Em ASCII, uma letra minúscula corresponde ao código da maiúscula correspondente mais 32 (que é o código do espaço em branco, usado para separar as palavras na string). O primeiro código ASCII útil para este programa é o da letra A maiúscula(65).

## AVANÇADO

2.c) Na configuração de estações de trabalho baseadas no protocolo TCP/IP, é muito comum utilizarmos máscaras de sub-rede. Com base na teoria descrita na Apostila 13 do meu curso de redes na UCSAL (disponível no site [www.logicsoft.com.br/mcamara](http://www.logicsoft.com.br/mcamara)), desenvolva um programa capaz de obter os seguintes resultados após a leitura de um endereço IP e sua máscara :

Endereço de rede

Endereço de sub-rede

Endereço de *host*

Número de sub-redes

Faixa de valores válidos da sub-rede em questão

2.d) Usando o método "bolha" de classificação de vetores, leia um vetor de 5 *strings* de uma palavra (com até 20 caracteres cada uma) e apresente-as posteriormente em ordem alfabética.

Dica : o método "bolha" considera que são feitas comparações seqüenciais dos elementos de um vetor (o primeiro é comparado com o segundo, o segundo com o terceiro e assim sucessivamente). Após cada comparação, são invertidos aqueles elementos que se encontram fora de ordem. Ao final do vetor, se foi realizada alguma troca, volta-se ao início novamente.

2.e) Dado um número entre 0 e 9, mostre seu correspondente em um *display* de sete segmentos, da mesma forma que em uma calculadora.

Dica : uma calculadora utiliza sete segmentos para codificar todos os dígitos de 0 a 9. Os segmentos serão identificados pelas letras de **a** até **g**, começando pelo segmento da esquerda superior, que é a letra **a**, girando no sentido horário até finalizar no segmento horizontal do centro, que será a letra **g**.