Algoritmos e Programação I: Lista de Exercícios 05 - Orientações para submissão no Moodle e BOCA. *

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
79070-900 Campo Grande, MS
`http://moodle.facom.ufms.br`

Após submeter no BOCA, não se esqueça de enviar o programa para o Moodle, usando o seguinte nome: pxxloginname.c, onde xx é o número do problema e o loginname é o seu login.

Compile o seu programa usando:

```
gcc -Wall -pedantic -std=c99 -o programa programa.c [-lm]
```

A flag `-lm` deve ser usada quando o programa incluir a biblioteca `math.h`.

Uma forma eficiente de testar se o seu programa está correto é gerando arquivos de entrada e saída e verificar a diferença entre eles. Isso pode ser feito da seguinte forma:

```
./programa < programa.in > programa.out
```

O conjunto de entrada deve ser digitado e salvo num arquivo `programa.in`. O modelo da saída deve ser digitado e salvo num arquivo `programa.sol`.

Verifique se a saída do seu programa `programa.out` é EXATAMENTE igual ao modelo de saída `programa.sol`. Isso pode ser feito com a utilização do comando `diff`, que verifica a diferença entre dois arquivos.

```
diff programa.out programa.sol
```

O seu programa só está correto se o resultado de `diff` for vazio (e se você fez todos os passos como indicado).

A solução dos exercícios deve seguir a metodologia descrita em sala:

a. Diálogo

b. Saída/Entrada (pós e pré)

c. Subdivisão

d. Abstrações

*Este material é para o uso exclusivo da disciplina de Algoritmos e Programação I da FACOM/UFMS e utiliza as referências bibliográficas da disciplina (B. Forouzan e R. Gilbert, A. B. Tucker et al. e S. Leestma e L. Nyhoff, Lambert et al., H. Farrer et al., K. B. Bruce et al., Material Didático dos Profs. Edson Takashi Matsubara e Fábio Henrique Viduani Martinez)

e. Implementação

f. Teste

Na submissão ao BOCA não se esqueça de comentar todos os `printf`'s da entrada e que a saída não pode conter acentuação ou *ç*.

Para utilização de strings, crie um tipo string e defina as variáveis usando esse tipo:

```
// declaração de constantes
#define LIMSTRING 10 // use o tamanho adequado para o seu programa
.
.
.
// declaração de tipos
typedef char string[LIMSTRING];
.
.
.
// declaração de variáveis
string nome_variável;
```

Exercícios

1. Assuma que as seguintes declarações tenham sido feitas:

   ```
   // Declaração de tipos
   typedef enum {vermelho, amarelo, azul, verde, branco, preto} Cores;
   typedef char palavra[6];
   typedef char string[10];

   // Declaração das variáveis
   palavra           CodigoCor;
   int               i,j;
   Cores             Cor;
   char Ch;
   string            TextoLinha,
                     Caracteres;
   ```

   Assuma também que os seguintes dados são fornecidos (onde □ denota um espaço em branco e ®é o caractere que indica final de linha gerado pela tecla ENTER ou RETURN):

   AB□CDEF®

   GH$®

   Para cada um dos seguintes itens; diga qual o valor (se algum) será atribuído a cada elemento do string, ou explique porque ocorre erro.

a.
```
for (i = 0; i <= 6; i++) {
   scanf("%c", &Caracteres[i]);
}
scanf("%c", &Ch);
for (i = 7; i <= 9; i++) {
   scanf("%c", &Caracteres[i]);
}
```

b.
```
scanf("%s", Caracteres);
```

c.
```
fgets(TextoLinha, sizeof(TextoLinha), stdin);
```

d.
```
for (Cor = vermelho; Cor <= preto; Cor++) {
   scanf("%d", &CodigoCor[Cor]);
}
```

e.
```
for (i = 0; i <= 9; i++) {
   scanf("%c", &TextoLinha[i]);
}
```

f.
```
for (i = 9; i >= 0; i--) {
   scanf("%c", &TextoLinha[i]);
}
```

g.
```
Cor = vermelho;
while (Cor < preto) {
   scanf("%d", &CodigoCor[Cor]);
   Cor++;
}
scanf("%d", &CodigoCor[Cor]);

for (Cor = vermelho; Cor <= preto; Cor++) {
   Caracteres[Cor] = CodigoCor[Cor];
}
scanf("%c", &Ch);
scanf("%c", &Ch);
for (i = 6; i <= 9; i++) {
   scanf("%c", &Caracteres[i]);
}
```

h.
```
for (i = 0; i <=9; i++) {
   scanf("%c", &Caracteres[i]);
}
strcpy(TextoLinha, Caracteres);
```

i.
```
for (i = 0; i <= 9; i++) {
   scanf("%c", &Ch);
   if (Ch != '\n') {
      TextoLinha[i] = Ch;
   } // fim if
   else {
      TextoLinha[i] = 'R';
   } // fim else
} // fim for
```

j.
```
for (i = 0; i <= 5; i++) {
   scanf("%c", &Ch);
```

```
         if (Ch != '\n') {
            Caracteres[i] = Ch;
            scanf("%c", &Caracteres[i+5]);
         } // fim if
         else {
            Caracteres[i] = '$';
            Caracteres[i+5] = 'R';
         } // fim else
      } // fim for
```

k.
```
      i = 0;
      for (j = 0; j <= 1; j++) {
         scanf("%c", &Ch);
         while (Ch != '\n') {
            TextoLinha[i] = Ch;
            i++;
            scanf("%c", &Ch);
         } // fim while
         scanf("%c", &Ch);
      } // fim for
```

l.
```
      i = 0;
      do {
         scanf("%c", &Ch);
         while (Ch != '\n') {
            TextoLinha[i] = Ch;
            i++;
            scanf("%c", &Ch);
         } // fim while
         scanf("%c", &Ch);
      } while (TextoLinha[i] != '$'); // fim do
```

2. Escreva um programa que receba como entrada um nome consistindo de um primeiro nome, um nome do meio ou inicial, e um último nome, nesta ordem, e então imprima o último nome, seguido de uma vírgula, e as iniciais do primeiro nome e do nome meio. Por exemplo, A entrada `Daniel Fabio Freitas` deve produzir `Freitas, D. F.`. O programa deve ser executado para `n` nomes.

```
Entrada:
4
Daniel Fabio Freitas
Alvaro Malaquias Urbieta
Caroline Andrea Martinez
Eduardo Perin Raimundo
```

O número 4 indica o número `n` de vezes que o programa será executado.

```
Saída:
Freitas, D. F.
Urbieta, A. M.
Martinez, C. A.
```

```
Raimundo, E. P.
```

3. Escreva um programa que conte todas as ocorrências de duas letras seguidas em uma dada linha de um texto lido de um arquivo. O programa deve ser executado para `n` linhas.

```
Entrada:
3
The Commodore 64, an 8-bit, mass-produced machine that brought
personal computing into the home for millions of  users in the
early- and mid-1980s.
Commodore sold more than 17 million of its C64 systems, according to
the company.
The new Commodore 64's hardware includes a dual-core 1.8 Ghz Atom
processor, Nvidia Ion2 graphics chipset, up to 4 GB of RAM and HDMI
output for desktop viewing on TV.
```

O número 3 indica o número `n` de vezes que o programa será executado.

```
Saída:
3
3
2
```

4. O sistema de codificação existente no programa `codifica.c` utilizado para produzir um criptograma simplesmente consiste em adicionar um dado inteiro para o código de cada caractere da mensagem. Isto é um caso especial da uma técnica conhecida como codificação por Palavra Chave, na qual uma sequência de inteiros correspondentes aos caracteres de uma dada palavra chave é adicionada ordenadamente aos códigos dos caracteres da mensagem. Para ilustrar, se a palavra chave é `ABC` e a mensagem é `ESTOUNORU`, os códigos para `A`, `B` e `C` (1, 2 e 3) são adicionados aos códigos de `E`, `S` e `T`, respectivamente, e então adicionados aos códigos para `O`, `U` e `N`, respectivamente, e assim por diante, produzindo o criptograma `FUWPWQPTX` (assumindo que estamos trabalhando com ASCII). Escreva um programa para implementar este método de codificação da palavra chave. O programa deve ser executado para `n` palavras chaves e `n` mensagens.

5. Um outro método de codificação utiliza substituição, por exemplo:

Letra:	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Substituta:	T F H X Q J E M U P I D C K V B A O L R Z W G N S Y

Cada letra da mensagem de entrada é substituída com sua substituta, por exemplo:

Mensagem:	T H E R E D C O A T S A R E C O M I N G
Criptograma:	R M Q O Q X H V T R L T O Q H V C U K E

Escreva um programa que receba como entrada uma mensagem ou um criptograma e então codifica ou decodifica a entrada usando substituição.

6. Escreva um programa para converter números hindu-arábicos em números romanos e vice-versa. (I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 e M = 1000). Inicialmente, considere IIII = 4, VIIII = 9, etc.

7. Uma string é denominada palíndromo quando ela lida do início para o final tem o mesmo significado se a ordem dos caracteres é lida do final para o início. Por exemplo:

   MADAM
   463364
   ABLE WAS I ERE I SAW ELBA

   são palíndromos. Escreva um programa para ler uma string e então determinar se ela é um palíndromo.

8. Escreva um programa para determinar se uma determinada string ocorre em uma dada string, caso isso ocorra, imprima um asterisco (*) embaixo da primeira posição de cada ocorrência.

9. (`ana.c`) [etm] Dado uma frase de tamanho 9, verifique quantas vezes a palavra "ana"aparece dentro da frase.

   Entrada: A primeira linha contém o número de frases, as próximas linhas são frases com 9 caracteres.

   ```
   2
   anabanana
   guaranana
   ```

   Saída: Escrever quantas vezes a palavra "ana"foi encontrada. Obs: Não existe quebra de linha após o resultado da saída.

   ```
   3
   2
   ```

10. Escreva um programa para contar as ocorrências de uma especificada string em diversas linhas de texto.

11. Escreva um programa que receba duas strings e determine se uma é um anagrama da outra, isto é, se uma string é uma permutação dos caracteres da outra string. Por exemplo, "dear" é um anagrama de "read" como é "dare".

12. (`vogalconsoante.c`) [etm] Dado uma frase de tamanho 10, verifique quantas vogais e quantas consoantes tem a frase.

    Entrada: A primeira linha consiste do número de frases, as próximas linhas são frases com 10 caracteres. As frases só possuem letras minúsculas.

    ```
    3
    paaraleloo
    amarelinha
    algoritmos
    ```

Saída: Escrever quantas vogais e consoantes a frase tem. Obs: Não existe quebra de linha após o resultado da saída.

```
6 4
5 5
4 6
```

13. (`size.c`) [etm] Dado n strings, imprimir o tamanho delas. Neste exercício não pode ser usada a função `strlen`.

    Entrada: A primeira linha contém um número inteiro n, que equivale a quantidade de strings dadas, seguido das n strings.

    ```
    5
    oi
    tudo
    bem
    com
    voce
    ```

    Saída: A saída consiste em escrever para cada sequência a mensagem "crescente"ou "sem ordem"para sequências em ordem estritamente crescente ou que não sejam estritamente crescentes respetivamente.

    ```
    2
    4
    3
    3
    4
    ```

14. (`trazprafrente.c`) [etm] Dado n strings, imprimi-las de traz para frente.

    Entrada: A primeira linha contém um inteiro n indicando a quantidade de strings, seguido das n strings.

    ```
    2
    oi
    auga
    ```

    Saída: o tamanho de cada número

    ```
    io
    agua
    ```

15. (`quantidade.c`) [etm] Dado 1 caractere e n strings, verificar quantas vezes o caractere dado existe em cada string.

    Entrada: A primeira linha consiste do caractere dado, seguido de um número inteiro n, correspondente a quantidade de strings, seguido das n strings;

```
p
3
pipoca
pokemon
lua
```

Saída: A saída consiste em escrever para cada string, quantas vezes o caractere dado aparece nela.

```
2
1
0
```

16. (`equal.c`) [etm] Dados n pares de strings, imprimir 1 caso elas sejam iguais e 0 caso sejam diferentes. Neste exercício não pode ser usado a função `strcmp`.

    Entrada: A primeira linha contém um inteiro n que indica a quantidade de pares de strings, seguido de n linhas com 2 strings em cada linha.

    ```
    2
    oi oi
    boi oi
    ```

    Saída: A saída consiste em escrever 1 caso as strings sejam iguais e 0 caso sejam diferentes.

    ```
    1
    0
    ```

17. (`maiusculas.c`) [etm] Dado n strings compostas apenas de caracteres minúsculos, transformar essa string em uma string de caracteres maiúsculos.

    Entrada: A primeira linha contém um inteiro n indicando o número de strings, seguido das n strings.

    ```
    2
    celular
    algoritmos
    ```

    Saída: A saída corresponde as n strings de caracteres maiúsculos.

    ```
    CELULAR
    ALGORITMOS
    ```