UNIVERSIDADE DE BRASILIA - FGA
INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO – ICC TURMA DD
2a Lista de Exercícios – 2/2011

Data de Entrega: 09/12/2011 (sexta-feira, antes da prova)

E-mail de entrega: iccturmadd.2.2011@gmail.com

Professor: Luiz Augusto F. Laranjeira

Escreva programas na linguagem C para resolver os exercícios listados abaixo.
Cada programa deverá estar funcionando (rodando) e deverá produzir um cabeçalho e mensagens esclarecendo a sua finalidade. Exemplo:

```
Universidade de Brasilia
113913 – Introdução à Ciência da Computação
Aluno:  <nome_do_aluno>

Programa para verificar a quantidade de alunos e alunas na sala.

Quantidade de alunos na sala: 8 (dado de entrada)
Quantidade de alunas na sala: 2 (dado de entrada)

Esta sala possui 10 alunos.
```

Itens de avaliação que deverão ser satisfeitos pelos programas:

a) Se existem comentários que documentem o código;
b) Se o código está devidamente indentado;
c) Se o programa produz mensagens adequadas solicitando entradas e escrevendo as saídas para que o usuario entenda a utilizacao e o resultado do programa;
d) Se as variáveis do programa têm nomes e tipos apropriados;
e) Se os dados de entrada são validados pelo programa (sempre que possível).

EXERCÍCIOS

1) Escreva um programa que calcule o termo de ordem ***n*** ($n \geq 1$) da seguinte sequência:

   a, b, a + b, a + 2b, 2a + 3b, ...

   Este programa deverá ler o valor de ***n*** como dado de entrada e escrever o valor de S(n) na tela. O programa deverá utilizar uma função recursiva para o cálculo de S(n).

2) Escreva um programa que leia os valores dos componentes de uma matriz M (3 x 3) e calcule e escreva na tela os componentes das matrizes:

   (a) Q = M x M;
   (b) D = 2 x M;
   (c) S = Q – D.

Os componentes de cada uma das matrizes Q, D e S deverão ser calculados por uma função que tenha por argumentos um ponteiro para o início da matriz M (que deverá ser declarada em ***main***), a linha da posição que se deseja calcular, e a coluna da posição que se deseja calcular.

Por exemplo a função que calcula um elemento de Q teria como protótipo algo como:

```
int Calcula_Q (int * p_M, int linha, int coluna);
```

Esta função deverá ser chamada em ***main*** da seguinte forma para calcular os valores de Q:

```
int M[3][3];
 <Cálculo de Q[i,j]> = Calcula_Q (&M[0][0], i, j);
```

O programa não deverá alocar memória para as matrizes Q, D e S. A saída do programa deverá ser da seguinte forma:

MATRIZ M

M_{11} M_{12} M_{13}
M_{21} M_{22} M_{23}
M_{31} M_{32} M_{33}

MATRIZ D = 2 x M

D_{11} D_{12} D_{13}
D_{11} D_{12} D_{13}
D_{11} D_{12} D_{13}

MATRIZ Q = M x M

Q_{11} Q_{12} Q_{13}
Q_{21} Q_{22} Q_{23}
Q_{31} Q_{32} Q_{33}

MATRIZ S = Q - D

S_{11} S_{12} S_{13}
S_{11} S_{12} S_{13}
S_{11} S_{12} S_{13}

3) Faça um programa que leia, via teclado, duas strings formadas apenas por letras (palavras) e espaços brancos. O programa deve imprimir uma lista das palavras que aparecem simultaneamente nas duas strings de entrada. Pode-se supor que em cada uma das strings não há palavras repetidas.

4) Faça um programa para copiar n caracteres da string s1 para a string s2 (no início desta), começando pela posição i da string s1. O usuário deve informar s1, n, e i. O programa deverá apresentar em sua saída as duas strings. Caso o valor de n ultrapasse a capacidade de memória de s2 o programa escreverá na ela uma mensagem de erro e não copiará os caracteres na string s2.

5) Escreva um programa que faça o cadastro dos funcionários de uma empresa com as seguintes informações para cada funcionário: nome, sexo, salário, código (matrícula), endereço e cargo.

   O programa deverá primeiramente ler o número de empregados que se deseja cadastrar (no máximo 100). Em seguida deverá ler os dados de cada funcionário.

   Finalmente o programa deverá calcular e escrever na tela o salário médio dos funcionários da empresa, o salário médio dos funcionários do sexo masculino e o salario médio dos funcionários do sexo feminino da empresa.

Neste programa você deverá definir um tipo composto (estrutura) com a diretiva **typedef** e utilizar um vetor de estruturas.

6) Modifique o programa anterior de modo que ele não mais defina um vetor com número fixo de estruturas, mas sim um vetor de ponteiros para a estrutura desejada e que faça a alocação de memória para os dados de cada funcionário usando a função **malloc**, à medida em que o usuário for entrando estes dados.

   Ao final do programa a função **free** deverá ser invocada de forma apropriada para liberar toda a memória alocada com a função malloc.

7) Modifique o programa anterior de modo que ele não mais defina um vetor com número fixo de estruturas, mas defina apenas um (somente um) ponteiro para o tipo da estrutura desejada e use a função **malloc** para fazer a alocação de toda memória necessária para os dados dos funcionários após ler o número de empregados que se deseja cadastrar.

   Ao final do programa a função **free** deverá ser invocada de forma apropriada para liberar a memória alocada com a função malloc.

   O acesso às estruturas correspondentes aos dados dos empregados poderá ser feito com notação de vetor ou de aritmética de ponteiros:
   Exemplo (onde p é um ponteiro para uma estrutura):

   p[2].nome = “Carlos Augusto” (notação de vetor)
   (p + 2)->nome=”Carlos Augusto” (notação de aritmética de ponteiros)

8) Faça um programa que leia, de dois arquivos, duas strings formadas apenas por letras (palavras) e espaços brancos. Cada string deverá ser lida de um arquivo. O programa deverá escrever num arquivo de saída uma lista das palavras que aparecem simultaneamente nas duas strings de entrada. Pode-se supor que em cada uma das strings não há palavras repetidas.

   O programa deve ler os seguintes dados via teclado: os nomes dos arquivos de entrada (que conterão as strings) e o nome do arquivo de saída. O programa deve verificar se os arquivos de entrada existem e retornar uma mensagem de erro caso um (ou os dois) não existam.

   O programa deverá também escrever na tela uma mensagem de erro caso não seja possivel criar o arquivo de saída ou, se este já existir, se não for possivel abri-lo apagando o seu conteúdo. Neste caso, o programa deverá escrever a lista de palavras de saída na tela.

   Se tudo ocorrer bem (sem erros), o programa deverá também escrever na tela uma mensagem de sucesso tal como: “Resultado escrito com sucesso no arquivo <NomedoArquivodeSaida>”.

9) Escreva um programa que procure uma palavra em um arquivo texto e a substitua por outra palavra. O arquivo de entrada deverá conter várias palavras ou frases. A palavra a ser procurada e a palavra substituta deverão ser lidas como entradas do programa via teclado. O nome do arquivo também deverá ser lido como dado de entrada via teclado. A palavra substituta deverá ser truncada caso tenha mais caracteres que a palavra a ser substituida.

   O programa deverá escrever na tela as mensagens de erro apropriadas (em caso de erro) e uma mensagem de sucesso (se não houver nenhum erro).

10) Reescreva o programa anterior considerando que a palavra substituta possa ter mais caracteres que a palavra a ser substituída e (a palavra substituta) deverá ser escrita em sua totalidade sem apagar o conteúdo já existente no arquivo (apagando somente a palavra a ser substituida).

   O programa deverá escrever na tela as mensagens de erro apropriadas (em caso de erro) e uma mensagem de sucesso (se não houver nenhum erro).