Algoritmos e Programação I: Lista de Exercícios 09 - 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., e 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*.

Exercícios

1. (`telefonista.c`) [ftm] Você foi contratado como ajudante de telefonista da UFMS e agora, para cada nome que a telefonista diz, você precisa procurar em uma caderno com milhares de nomes e dizer para ela o telefone da pessoa procurada. Como felizmente você estudou Algoritmos e Programação I e aprendeu registros, você decide fazer um programa que dado um nome, imprime o telefone da pessoa procurada. Logo, dado uma lista com n nomes de pessoas e seus respectivos telefones que estão no caderno e após isso, outra lista com m nomes, dizer o telefone de todas as pessoas que estão nessa segunda lista.

   Entrada: A primeira linha contém um inteiro n indicando o número de pessoas, seguido do nome das pessoas e seus respectivos telefones, após isso, outro número inteiro m, indicando o número de nomes da outra lista, onde para cada nome dado, você deverá imprimir o telefone da pessoa. Obs: Veja que, quando você for ler o número telefônico, deve ler ele como uma string e não um número inteiro, caso contrário, poderá estourar o alcance do tipo inteiro.

   ```
   telefonista.in
   2
   Poca-Sombra 92030303
   Quase-Nada 99010203
   1
   Poca-Sombra
   ```

   Saída: A saída corresponde dos telefones dos m nomes dados.

   ```
   telefonista.sol
   92030303
   ```

2. (`biblioteca.c`) [etm] Após seu emprego como ajudante de telefonista da UFMS e sua implementação impecável do sistema telefônico, o LEDES tomou conhecimento do seu incrível software feito para o sistema telefônico da UFMS e convidou-o para um estágio. Sua primeira tarefa é o de desenvolver um sistema para a Biblioteca Central da UFMS. Agora, sua tarefa é, dado uma lista de n livros, onde cada livro tem como informação o nome de seu autor e a prateleira onde ele se encontra e outra lista, com m nomes de livros, dizer qual é o nome do autor do livro e em qual prateleira os livros que estão nessa segunda lista se encontram. Caso o livro que esteja na segunda lista, não esteja na primeira lista, ou seja, não exista na biblioteca, imprimir apenas -1.

   Entrada: A primeira linha contém um inteiro n indicando o número de livros (máximo de 100), seguido dos n nomes de livros (nomes formados por uma única palavra de no máximo 49 caracteres)/ nomes dos autores dos livros (nomes formados por uma única palavra de no máximo 49 caracteres)/ prateleiras onde os livros de

encontram, após isso, outro número inteiro m, indicando o número de livros da outra lista, onde para os m livros dados, você deverá imprimir o autor do livro, seguido da prateleira onde o livro se encontra.

```
biblioteca.in
2
Algoritmos Cormen 3
Quimica Feltre 1
3
Quimica
Historia
Biologia
```

Saída: A saída corresponde a impressão do nome dos autores do livro/ prateleiras onde os livros se encontram, ou -1, caso o livro não exista na biblioteca.

```
biblioteca.sol
Feltre 1
-1
-1
```

3. (`notas.c`) A avaliação da disciplina de Algoritmos e Programação I é composta de três provas e dois trabalhos. Projete um programa que leia um conjunto de alunos com as respectivas notas das provas e dos trabalhos. O programa deve ordenar os alunos pelo primeiro nome usando o método de ordenação por inserção (`insertionSort`). O programa deve imprimir a lista dos alunos ordenada pelo primeiro nome.

   Na implementação o programa deve utilizar a função:

```
void insertionSort(ESTUDANTE lista[], int ultimo);
```

   para fazer a ordenação dos nomes dos alunos.

   Entrada: A primeira linha contém um inteiro n equivalente ao número de alunos, seguido dos n alunos com suas respectivas notas.

```
notas.in
5
Karin Tuiuiu 75 91 89 89 90
Carlos Bigua 85 94 79 93 76
Turibio Garca 87 88 89 90 89
Maria Dourado 83 79 93 91 90
Bartolomeu Pintado 78 96 88 91 89
```

   Saída: A saída consiste em escrever a lista ordenada dos nomes dos alunos com as suas respectivas notas.

```
notas.sol
Bartolomeu   Pintado        78   96   88   91   89
Carlos       Bigua          85   94   79   93   76
Karin        Tuiuiu         75   91   89   89   90
Maria        Dourado        83   79   93   91   90
```

```
Turibio      Garca          87   88   89   90   89
```

4. (seguinte.c) [fhvm]

   a. Escreva um programa que receba um número inteiro $k > 0$, que indica a quantidade de casos de teste; para cada caso de teste, receba uma data no formato `dd/mm/aaaa` e imprima a data que representa o dia seguinte no mesmo formato. Use as funções dos itens anteriores.

   b. O programa abaixo deve ser usado para resolver o problema:

```
// Programa: seguinte.c
// Programador:
// Data: 09/12/2010
// Este programa recebe um número inteiro k > 0, que indica a
// quantidade de casos de teste; para cada caso de teste, receba
// uma data no formato dd/mm/aaaa e imprime a data que representa
// o dia seguinte no mesmo formato. O programa usa as seguintes
// funções: bissexto (verifica se um ano é bissexto), diasMes
// (informa a quantidade de dias do mês), diaSeguinte (retorna
// o dia seguinte no formato dd/mm/aaaa).
// declaração das bibliotecas utilizadas
#include<stdio.h>
#include<stdbool.h>

// declaração de tipos
typedef struct {
    int dia;
    int mes;
    int ano;
} DATA;

// declaração das bibliotecas utilizadas
bool bissexto(DATA d);
int diasMes(DATA d);
DATA diaSeguinte(DATA d);

// início da função principal
int main(void) {
// declaração de variáveis locais
   int k;
   DATA hoje, amanha;

// Passo A. Leia a quantidade de datas
   scanf("%d", &k);
// Passo B. Repita k vezes
   while (k > 0) {
// Passo 1. Leia a data atual
      scanf("%d/%d/%d", &hoje.dia, &hoje.mes, &hoje.ano);
// Passo 2. Calcule a data do dia seguinte
      amanha = diaSeguinte(hoje);
```

```
// Passo 3. Imprima a data do dia seguinte no formato dd/dd/aaaa
      printf("%02d/%02d/%04d\n", amanha.dia, amanha.mes, amanha.ano);
// Passo B.1. Atualize o número de vezes
      k--;
   } // fim while

   return 0;
} // fim da função principal

// Função: bissexto
// Recebe um registro d contendo uma data no formato dd/mm/aaaa
// e devolve true se o ano é bissexto e false caso contrário.
// Um ano é bissexto se é divisível por 4 e não por 100 ou é divisível por
// 400.
bool bissexto(DATA d) {
// Passo b.1 verifique se o ano é bissexto
   if ((d.ano % 400 == 0) || (d.ano % 4 == 0 && d.ano % 100 != 0))
      return true;
   else
      return false;
} // fim da função bissexto

// Função: diaMes
// Recebe um registro d contendo uma data no formato dd/mm/aaaa
// e devolve a quantidade de dias que o mês possui
int diasMes(DATA d) {
// Passo dM.1. Calcule o número de dias do mês

} // fim da função diaMes

// Função: diaSeguinte
// Recebe um registro d contendo uma data no formato dd/mm/aaaa e
// devolve um registro contendo a data que representa o dia seguinte */
DATA diaSeguinte(DATA d) {
// Passo dS.1. Compute o dia seguinte

// Passo dS.2. Calcule o dia seguinte

// Passo dS.2.1. Calcule o mês seguinte (se for o caso)

// Passo dS.3. Calcule o ano seguinte (se for o caso)

   return d;
} // fim da função diaSeguinte
```

c. Escreva uma função com a seguinte interface:

```
int diasMes(DATA d)
```

que receba uma data no formato `dd/mm/aaaa` e devolva o número de dias do mês em questão.

Exemplo: se a função recebe a data `10/04/1992` então deve devolver 30;

se recebe a data `20/02/2004` então deve devolver 29.

d. Escreva uma função com a seguinte interface:

```
DATA diaSeguinte(DATA d)
```

que receba uma data no formato `dd/mm/aaaa` e devolva a data que representa o dia seguinte. Use as funções dos itens (a) e (b).

Exemplos de entrada:

```
seguinte.in
3
03/07/1995
31/12/2009
28/02/2000
```

Exemplos de saída:

```
seguinte.sol
04/07/1995
01/01/2010
29/02/2000
```

5. (`sucao.c`) [etm] Você, como todo acadêmico da nossa querida universidade, adora ir tomar aquele delicioso suco com seus amigos. Após chegar lá e pedir o seu suco, seu espírito empreendedor aflora e você decide observar por 1 dia qual suco é o suco mais vendido da universidade para futuramente, abrir um novo quiosque e faturar milhões.

   Seu trabalho agora é dado n sucos, dizer quantas vezes cada suco apareceu, imprimindo os sucos na ordem em que foi dada na entrada.

   Entrada: A primeira linha contém um inteiro n equivalente a quantidade de sucos, seguido dos n sucos que foram feitos no dia.

```
sucao.in
morango
laranja
mamao
laranja
uva
acerola
laranja
laranja
```

   Saída: A saída consiste em escrever o nome de cada suco, sem repetição, e o número de quantos sucos daquele tipo foram feitos no dia.

```
sucao.sol
morango 1
laranja 4
mamao 1
uva 1
acerola 1
```

6. Para cada um dos seguintes itens, projete uma estrutura de registro apropriada (`struct`) para as informações dadas, e então escreva declarações de tipos para os registros.

   a. As cartas em um maço de jogo de baralho.

   b. Tempo medido em horas, minutos e segundos.

   c. Os registros de uma lista telefônica.

   d. Descrição de um automóvel (Marca, Ano, Modelo, Tipo, Cor, Acessórios, Preço).

   e. Descrição de um livro na biblioteca (Autor, Título, Editora, etc.).

   f. Times de futebol no campeonato nacional (Time, grupo, número de partidas jogadas, etc.).

   g. Posição das pedras em um jogo de damas no tabuleiro.

7. Os arquivos de dados `estudante.txt`, `estoque.txt` e `usuarios.txt` estão descritos abaixo. Escreva declarações apropriadas de registros (`structs`) para descrever as informações nesses arquivos.

```
estudante.txt
10103 Johnson James L
Waupun WI 7345229 M ENGR 15 3.15
10110 Peters Andrew L
Lynden WA 3239550 M ARTS 63 2.05
10298 Psycho Prunella E
De Motte IN 5384609 F PSYC 120 2.99
...
```

   Nesse arquivo temos os seguinte campos:

   - Número do estudante: um número inteiro sem sinal com 5 dígitos.
   - Sobrenome do estudante: uma string com no máximo 15 caracteres.
   - Nome do estudante: uma string com no máximo 10 caracteres.
   - Inicial do nome do meio do estudante: um caractere.
   - Cidade do estudante: uma string com no máximo 15 caracteres.
   - Estado do Estudante: uma string com 2 caracteres.
   - Telefone: uma string com 7 caracteres.
   - Sexo: um caractere (M ou F)
   - Código do curso: uma string com 4 caracteres.
   - Total de créditos obtidos: um número unsigned int
   - Média Geral acumulada: um número float

```
estoque.txt
1011     20     54.95     15 Camera Fotagrafica Portatil
2013     30     28.99      5 Flash Eletronico
...
```

   Nesse arquivo temos os seguinte campos:

- Número do item: um número inteiro sem sinal com 4 dígitos.
- Número de peças em estoque: número inteiro sem sinal variando de 0 a 999.
- Preço Unitário: um valor float.
- Estoque mínimo: número inteiro sem sinal variando de 0 a 999.
- Nome do Item: uma string de no máximo 25 caracteres.

```
usuarios.txt
10101 Jose Martins MOE 750 380.81
10102 Isaac Souza LARGE 650 598.84
...
```

Nesse arquivo temos os seguinte campos:

- Número de Identificação: um número inteiro
- Nome do usuário: uma string com no máximo 30 caracteres na forma Primeiro Nome, Ultimo Nome.
- Senha: uma string com no máximo 5 caracteres.
- Limite de crédito: um valor float.
- Limite usado até o momento: um valor float

8. Para cada um dos seguintes itens, projete uma estrutura de registros (`struct`) usando `union` para as informações dadas, e então escreva declarações de tipos para os registros:

   a. Informação à respeito de uma pessoa: nome; data de aniversário; idade; sexo; RG; altura; peso; cor do cabelo; estado civil, se casado, número de filhos.

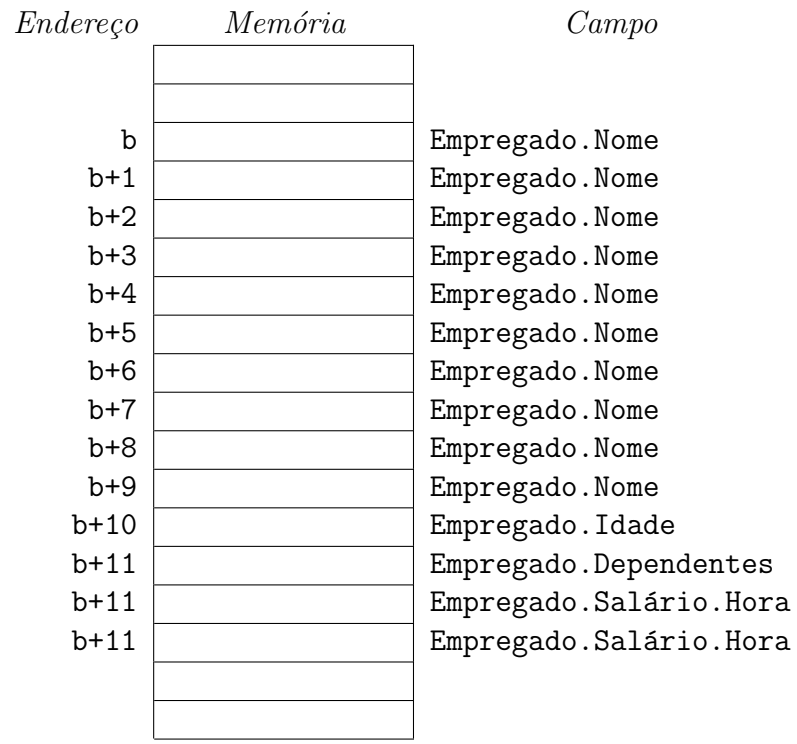
   b. Estatística do Tempo: data; cidade e estado; país; hora do dia; temperatura; pressão; condições do tempo (limpo, parcialmente nublado, nublado, chuvoso); se nublado, condições predominantes, nível e tipo de nuvens; para parcialmente nublado, porcentagem de cobertura das nuvens; para condição chuvosa, volume de precipitação da chuva; tamanho das pedras caso a chuva seja de granizo.

9. Da mesma forma que nos vetores, os campos de um registro podem ser armazenados em posições consecutivas de memória, uma translação de endereço é necessária para determinar a posição de um campo específico. Para ilustrar suponhamos que os números `unsigned int` são armazenados em 2 bytes, valores `float` em 4 bytes e cada `char` é armazenado em um byte. O registro da declaração

```
struct Empregado
       {
            char         Nome[20];
            unsigned int Idade;
            unsigned int Dependentes;
            float        SalárioHora;
        };
```

ocupa memória suficiente para armazenar uma string com 20 bytes, dois inteiros com 2 bytes cada, e um float com 4 bytes. Esse valor pode ser alterado em função dos slack bytes (fragmentação - No nosso exemplo supomos que não são necessário slack bytes). O formato deste objeto na memória é determinado pela opção do alinhamento

das palavras no compilador. Com esta opção off (default), um compilador reservará um bloco 28 bytes (isso varia de acordo com compilador e do tamanho da palavra do seu sistema operacional - para obter um resultado preciso, consulte os manuais dos produtos que vocês estiver usando) contíguos para armazenar esse registro. Vamos supor que uma palavra de memória é definida como tendo 2 bytes. Logo o compilador reservará um bloco de 14 palavras consecutivas de memória. Da mesma forma que para vetores, o endereço da primeira palavra neste bloco é denominada endereço base. Se o endereço base para `Empregado` é `b`, então o campo `Empregado.Nome` é armazenado nas 10 palavras consecutivas de memória iniciando em `b`, `Empregado.Idade` é armazenado na palavra `b+10`, `Empregado.Dependente` na palavra `b+11`, e `Empregado.SalárioHora` nas palavras `b+12` e `b+13`.



Assumindo esses requisitos de armazenamento para `unsigned int`, `float` e `char`, faça uma figura semelhante mostrando onde cada campo dos seguintes tipos de registro serão armazenados:

a.
```
struct Ponto {
            float x, y;
     };
```

b.
```
typedef char string[15];
typedef float Notas[5];
struct RegistroClasse {
        unsigned int NumEst;
        string       Primeiro, ÚltimoNome;
        char         Sexo;
        ListaDeNotas NotasProva;
     };
```

c.
```
typedef char string[15];
struct RegistroEstudante {
        unsigned int NumEst;
```

```
                string       PrimeiroNome, ÚltimoNome;
                struct       Notas {
                                float Listas, Provas, Exame;
                             };
                float        NotaFinal;
                char         Msg;
            };
```

d.

```
typedef char string[20];
struct RegistroEmpregado {
          string       Nome;
          unsigned int Idade,
                           Dependentes;
          char         CodEmpregado;
          union Codigo {
                    struct F {
                              char  CodDept;
                              float SalarioHora;
                             };
                    struct O {
                              float Salario;
                             };
                    struct S {
                              unsigned int Milhas;
                              float        TaxaComissao,
                                           BasePagamento;
                             };
               };
          };
```

10. Estenda o programa `calcreta.c` para encontrar:

    a. o ponto médio do segmento de reta ligando os dois pontos.

    b. a equação da reta perpendicular ao segmento de reta e passando pelo ponto médio do segmento.

11. A equação de uma reta com coeficiente angular m e passando pelo ponto P com coordenadas (x_1, x_2) é

$$y - y_1 = m(x - x_1)$$

    (a) Escreva uma descrição de um registro para uma reta, dados sua inclinação (coeficiente angular) e um ponto na reta.

    (b) Escreva um programa que leia a inclinação da reta e as coordenadas de um ponto na reta e então encontre a equação da reta.

    (c) Escreva um programa que leia o ponto e a inclinação para duas retas e determine se elas se interceptam ou são paralelas. Se elas se interceptam encontrar o ponto de interseção e determinar se elas são perpendiculares.

12. Escreva um programa que leia as horas no formato `hhmm, hh` variando de 0 a 23 e `mm` de 0 a 59 e determine o horário correspondente no formato usual, hora, minutos

e `AM/PM` ou leia as horas no formato usual e encontre a representação do tipo `hhmm`. Por exemplo, a entrada `0100` deve produzir como saída `1:00 AM`, e a entrada `3:45 PM` deve ter como saída `1545`.

13. Escreva uma declaração para as cartas de um baralho padrão de 52 cartas e 2 coringas. Então escreva um programa para manipular duas mãos com 10 cartas do baralho em cada uma. (Use o gerador de números aleatórios). Certifique-se de que a mesma carta não é manipulada mais de uma vez.

14. Escreva a declaração de um registro (usando `union`) para quatro figuras geométricas: círculo, quadrado, retângulo e triângulo. Para o círculo, o registro deve armazenar seu raio; para o quadrado o comprimento de um lado; para o retângulo, os comprimentos de dois lados adjacentes; e para o triângulo os comprimentos de seus três lados. Então escreva um programa que leia uma das letras `C` (círculo), `Q` (quadrado), `R` (retângulo), `T` (triângulo), e a quantidade (ou quantidades) apropriada(s) de valor(es) numérico(s) para a figura daquele tipo e então calcule a sua área. Por exemplo, a entrada `R 7.2 3.5` representa um retângulo com comprimento `7.2` e largura `3.5`, e `T 3 4 6.1` representa um triângulo tendo lados de comprimentos `3, 4` e `6.1`. (Para um triângulo a área pode ser encontrada através da utilização da fórmula de *Hero*: `área = pow(s*(s-a)*(s-b)*(s-c), 0.5)`, onde `a`, `b` e `c` são os comprimentos dos lados e `s` é metade do perímetro.

15. Escreva uma programa que leia os registros do `estoque.txt` e armazene-os em um vetor de registros, leia um número de estoque fornecido pelo usuário, e então efetue uma busca nesse vetor para um item tendo o número de estoque fornecido pelo usuário. Se o número for encontrado, o nome do item e estoque atual devem ser impressos; caso contrário, uma mensagem indicando que o item não foi encontrado deve ser impressa.

16. Escreva um programa que leia os registros do `estudantes.txt` e armazene-os em um vetor de registros, ordene-os tal que os MGA's fiquem em ordem decrescente, e então imprima os números dos estudantes, nomes, cursos e MGA's dos registros do vetor ordenado.

17. Um número racional é da forma $\frac{a}{b}$, onde a e b são inteiros com $b \neq 0$. Escreva um programa para efetuar operações aritméticas com números racionais, representando cada número racional como um registro com um campo numerador e um campo denominador. O programa deve ler e imprimir todos os números racionais no formato `a/b`, ou simplesmente `a` se o denominador for 1. Os seguintes exemplos ilustram o menu de comandos que o usuário pode usar:

Entrada	Saída	Comentários
`3/8 + 1/6`	`13/24`	a fração deve estar reduzida ao seus menores termos
`3/8 - 1/6`	`5/24`	a fração deve estar reduzida ao seus menores termos
`3/8 * 1/6`	`1/16`	a fração deve estar reduzida ao seus menores termos
`3/8 / 1/6`	`9/4`	a fração deve estar reduzida ao seus menores termos
`3/8 I`	`8/3`	Inverter a/b
`8/3 M`	`2 + 2/3`	Escrever a/b como uma fração mista
`6/8 R`	`3/4`	Simplificar a/b
`6/8 G`	`2`	Máximo divisor comum entre o numerador e o denominador
`1/6 L 3/8`	`24`	Mínimo múltiplo comum de a/b e c/d
`1/6 < 3/8`	`true`	
`1/6 <= 3/8`	`true`	
`1/6 > 3/8`	`false`	
`1/6 >= 3/8`	`false`	
`3/8 = 9/24`	`true`	
`2/3 X + 2 = 4/5`	`X = -9/5`	Solução da equação linear.