Universidade Federal de São Carlos
Bacharelado em Ciência da Computação
Estruturas de Dados I
Prof. Tiago A. Almeida <talmeida@ufscar.br>

EXERCÍCIOS – REVISÃO

- **Atenção:**

  1. **Arquivos:** o nome do arquivo referente ao código-fonte de cada exercício deverá seguir o seguinte padrão: **RA**<número do RA>**_ED1_L**<número da lista>**_EX**<número do exercício>.c. Exemplo: `RA123456_ED1_L01_EX01.c`;
  2. **E/S:** tanto a entrada quanto a saída de dados devem ser "secas", ou seja, não devem apresentar frases explicativas;
  3. **Identificadores de variáveis:** escolha nomes apropriados;
  4. **Documentação:** inclua cabeçalho, comentários e indentação no programa.

- **Exercícios:**

  1. Crie uma única função que retorne o valor máximo e o mínimo de um vetor, além de um apontador para o valor mediano. Use aritmética de ponteiros para percorrer o vetor.
  2. Implemente uma função que calcule a área da superfície e o volume de uma esfera de raio r. A função deverá obedecer o protótipo:

     ```
     void esfera (float r, float *area, float *volume);
     ```

     A área da superfície e o volume são dados, respectivamente por $4\pi r^2$ e $\frac{4}{3}\pi r^3$.
  3. Implemente uma função para resolver uma equação do segundo grau do tipo $ax^2+bx+c = 0$. Dados a, b e c, o programa deverá retornar os valores de x_1 e x_2.
  4. Implemente o método recursivo para encontrar o n-ésimo termo da função de Fibonacci. A rotina deverá retorna o valor de $F(n)$ e a quantidade de vezes que a recursão foi chamada.
  5. Crie a função do tipo `void TrocaTres(int a, int b, int c)` que retorna `a = c`, `b = b` e `c = a`.
  6. Crie a função `void OrdenaTres(int a, int b, int c)` que devolve em ordem crescente os valores contidos em `a`, `b` e `c`. Por exemplo, se `a = 5`, `b = 4` e `c = 3`, após a chamada a função OrdenaTres, `a = 3`, `b = 4` e `c = 5`.
  7. Escreva um programa que contenha a seguinte estrutura:

     ```
     typedef struct duracao{
        int horas;
     ```

```
   int minutos;
   int segundos;
} tpduracao;
```

O programa também deverá conter uma função com o seguinte protótipo:

```
int calc(tpduracao a, tpduracao b, tpduracao *total, tpduracao *dif);
```

A função deverá receber por valor a duração dos experimentos a e b (em segundos), e deverá calcular a soma da duração dos dois experimentos e a diferença de tempo dos mesmos, através dos parâmetros `total` e `dif`, recebidos por referência. O programa deverá retornar 1, por meio do uso do comando `return`, se o experimento a é mais curto ou de igual duração a b e 0 caso contrário.

O programa deverá receber os valores das durações dos testes e informar ao usuário as duas medidas calculadas. O programa também deverá informar qual é o teste mais curto.

8. Criar um registro `Aluno` e um registro `Materia` tal como dado abaixo:

```
typedef struct Aluno{
   int matricula;
   float vNotas[5]; // Armazena as 5 notas de um aluno ao longo de um ano.
   char nome[100];
} Aluno;

typedef struct Materia{
   Aluno V[MAX]; // Armazena a informação de MAX alunos
   float media[5]; // Armazena as 5 médias do ano.
   int nAlunos; // Número de alunos matriculados no curso.
} Materia;
```

   (a) Criar uma função `Aluno fillAluno(Aluno a1)` que preenche os campos de uma variável `a1` do tipo `Aluno` e retorna essa variável preenchida.

   (b) Criar uma função `Materia fillMateria(Materia m1, int numAlunos)` que preenche os campos de uma variável `m1` do tipo `Materia` realizando chamadas a função `fillAluno` em um número de vezes igual ao número contido na variável `nAlunos`. Nota: número máximo de aluno por matéria (`MAX`) é igual a 60.

   (c) Criar uma função `int mediaMateria(Materia m1)` que fornece a média de cada prova do ano para os alunos contidos na variável `m1` e assim preenche o campo `media` de `m1`.

   (d) Criar uma função `void mostraMateria(Materia m1)` que mostra as informações dos alunos contidas na variável `m1`.

   (e) Criar um programa que ilustra o funcionamento das funções anteriores. Para tanto, o programa deve possuir uma variável `pc1` do tipo `Materia` com 5 alunos. Depois, serão utilizadas as seguintes funções: `fillMateria` (que emprega `fillAluno`), `mediaMateria` e `mostraMateria`.

9. Escreva um programa que solicita ao usuário um vetor de notas (números reais) e imprime a média aritmética das notas.

   Observações:

   a. Apesar de não ser necessário utilize um vetor.
   b. O programa não deve limitar o tamanho do vetor.
   c. Não deve ocorrer desperdício de memória.
   d. Após ser utilizada a memória deve ser devolvida.

10. Escreva um programa que solicita ao usuário o RA (inteiro) e a média final (real) de todos os seus alunos e imprime todos os alunos que estão de SAC ($5 \leq nota < 6$).

    Observações:

    a. Utilize um vetor de registros (estruturas) para armazenar os dados dos alunos.
    b. O programa não deve limitar o número de alunos.
    c. Não deve ocorrer desperdício de memória.
    d. Após ser utilizada a memória deve ser devolvida.

11. Utilizando alocação dinâmica, crie um programa que leia n idades (`int`), armazene-as e depois mostre-as em ordem crescente. O valor de n será informado pelo usuário. Crie funções e procedimentos para realizar as tarefas.

12. Faça um programa que receba dois vetores: p de tamanho m e q de tamanho n. Primeiro, o usuário deverá informar o valor de m e os valores de cada elemento de p. Em seguida, o valor de n e os elementos de q. Finalmente, o programa deverá imprimir um novo vetor s no qual cada elemento corresponde a soma de $p[i] + q[i]$.

13. Crie um programa que leia um vetor de inteiros de tamanho n. Após cada valor lido, exiba o máximo, o mínimo e a média. Obs: use `calloc` para alocar o vetor e implemente uma função para cada operação.

    Exemplo:

```
3                            // tamanho de v
5                            // v[0]
max = 5 | min = 5 | med = 5  // saída 1
7                            // v[1]
max = 7 | min = 5 | med = 6  // saída 2
3                            // v[3]
max = 7 | min = 3 | med = 5  // saída 3
```

14. Escreva um programa que utiliza a estrutura abaixo para criar uma lista. Numa lista o campo `prox` recebe o endereço do próximo elemento da lista. Primeiro, solicite ao usuário o número de elementos da lista e crie a mesma preenchendo o campo `val` com a posição do elemento na lista. Depois imprima a lista.

```
typedef struct Elemento {
   int val;
```

```
   struct Elemento *prox;
} Elemento;
```

Exemplo de E/S:

```
 5           % Entrada:  Tamanho da lista
 1 2 3 4 5   % Saída:  Impressão da lista
```